Developing Neural Networks Library in RSCAD for Real-Time Power System Simulation

Bipul Luitel, *Member, IEEE*, Ganesh Kumar Venayagamoorthy, *Senior Member, IEEE* and Gabriel Oliveira

***Abstract*—Real-time digital simulators (RTDS) are used for real-time simulation of power systems. As the complexity of the electric power grid and associated control increases in the future, modeling and simulation of the power network as well as the control becomes essential. This requirement will be even more prominent in the context of smart grid. As enabling technology, intelligent methods of monitoring and control that utilize computational intelligence techniques are expected to be an integral part of smart grids. Therefore integration of computational intelligence based tools in power system simulation tools is an important aspect of smart grid research. Most past and current applications of neural networks in power systems are carried out offline in non-real-time platforms. In this study, neural networks libraries are developed to run on RTDS. The neural networks component is then used to predict the speed deviation of a generator in a multi-machine power system. These neural networks components can be trained in real-time and hence can be useful tool for smart grid applications.**

***Index Terms*—Neural networks, Real-Time Power System Simulation, RSCAD, RTDS**

I. Introduction

Addition of renewable resources, distributed generation and bidirectional power flows in the smart grid will make it more complex than ever. Smart grid will have high uncertainty and variability owing to high penetration of intermittent energy sources. Therefore, it is necessary to have real-time simulation of power system phenomena, and related systems of communication and control under these changed scenarios in order to ensure stability, reliability, integrity and security of the electric power grid. Today, real-time digital simulators (RTDS) are widely used by industries and universities across the world [1], [2]. Real-time digital simulator (RTDS®) from RTDS Technologies, and its graphical user interface (GUI) based design platform, RSCAD, are especially designed for real-time power system simulations. RSCAD library provides a plethora of devices and components for designing small to large power system simulation cases and allows testing of various aspects of power system operation and control in real-time. However, reliable operation and actionable situational awareness of the smart grid is not possible without the use of computational intelligence (CI) methods [3]. Therefore, a lot of effort in the drive towards smart grid is also focused on the development of intelligence for wide area situational awareness, visualization, and decision making and control [4]. Many literature in the past decade have shown effectiveness of CI based techniques in power system applications [5]–[10]. Use of neural networks (NN) in power system applications has been an area of research for a long time [11]–[16] but has received even more interest recently [17]–[19] due to the necessity of intelligent methods of monitoring and control for handling the complexity of large networks such as the smart grid. Despite the increasing interest in the application of CI methods in power system, most of the past and current applications of neural networks in power systems are carried out offline in non-real-time platforms , hence are not suitable for real-time power system simulations. Although non-real-time simulations such as Simulink, as well as simulators from Opal-RT may be able to use neural networks toolbox available in MATLAB, the component library of RSCAD lacks it. Therefore, it is necessary to have the components that use CI techniques for power system applications in the real-time simulation of smart grids. Hence there is an active need for research and development in this area. Development of CI component libraries in RSCAD will enable testing of intelligent methods of monitoring and control, intelligent data mining techniques, intrusion detection and cyber security techniques, as well as many other futuristic applications that are only possible by the use of CI methods. The study presented in this paper is an initiative in this direction. Development of neural networks libraries in RSCAD is presented in the paper. The developed feedback neural network component is trained in real-time using the RTDS, and is used to predict the speed deviation of a generator in a multi-machine power system.

The remaining sections of this paper are arranged as follows: background in real-time power system simulation is provided in Section II. Implementation of neural network component in RSCAD is presented in Section III. Its application in power system simulation and results thus obtained are shown in Section IV. Conclusions are given in Section V.

II. Real-time power system simulation

Real-time simulation is a computer model of a physical system that can be executed at the same rate as the actual system. Real-time simulator is a combination of computer hardware and software that enables real-time simulation. Because of the deterministic times for simulation, real-time simulators are used in various engineering fields, and more so in the simulation of mission and time critical applications. Real-

Bipul Luitel and Ganesh Kumar Venayagamoorthy are with Real-Time Power and Intelligent Systems Laboratory, Clemson Universtiy, Clemson, SC 29634 USA, Contact: iambipul@ieee.org, gkumar@ieee.org

Gabriel Oliveira is a visiting student at Real-Time Power and Intelligent Systems Laboratory from Universidade Federal de Itajuba, Brazil. Contact: gbnoliveira@gmail.com

The funding provided by National Science Foundation, USA under the CAREER grant ECCS #1231820, #1232070 and EFRI #1238097 is gratefully acknowledged.

978-1-4673-6002-9/13/$31.00 ©2013 IEEE

time digital simulators are a set of hardware platforms and accompanying software suite for simulation of power systems [20]. These allow for the simulation of power system and control components to evaluate system performance under various operating conditions, and to test devices and control systems. Many researchers over the past decade have used RTDS as a power system test bed for real-time power system simulation studies [21], [22].

As the interest in power system research has shifted towards the smart grid, RTDS®is also being used in simulation of distributed power grids [23], for the study of HVDC grid [24], and several other smart grid applications. Synchronized phasor measurement technology is a key element in smart grids. Wide deployment of phasor measurement units (PMUs) is taking place in the electric power grids all over the world. Therefore, RTDS®has been used in several real-time simulation studies on integration and testing of PMUs [25]–[27], and for data mining of PMU data and cyber-attacks [28]. Thus, RTDS®has proven to be an important tool for successful deployment of smart grid technologies.

Real-time digital simulators also enable hardware-in-the-loop based simulations. These are useful for implementing parts of the physical system in simulation and still operate as a complete physical system. Such advanced features allow fast integration of new technologies into the smart grid. Real-time simulation also allows testing of various operating conditions and phenomena that cannot be realistically implemented on the physical system. Several products are commercially available as real-time digital simulators for power systems.

III. Development of Neural Networks Library in RSCAD

Neural network consists of neurons in two or more layers connected to each other by synapses having some synaptic weights. There are mainly three types of NN architectures: viz. feedforward NN or the multilayer perceptron (MLP), feedback or recurrent NN (RNN), and the cellular NN. In each of these NN architectures, the neurons implement some transformation (linear or non-linear) on the input. The process of adjusting the weights of the NN in order to map the output of the neural network with the inputs in order to match the desired output is known is training. Although several techniques for training of NN exist, gradient decent based techniques such as backpropagation is most common. In this method, the gradient of the error with respect to the weights is used to adjust the weights. Each weight adjustment is known as epoch and can be done at each sample (in case of online learning) or after accumulating the change in the weights over a period of time (in case of batch learning). The internal structure of an NN is shown in Fig. 1. The output of a NN is calculated from its inputs using (1) and (2). In case of an MLP, $\vec{I}$ is a vector of inputs where as it is a vector consisting of inputs as well as feedback elements ($\vec{H}$) in case of an RNN. Vectors $inW\vec{eights}$ and $outW\vec{eights}$ are the input and the output weight matrices, respectively.

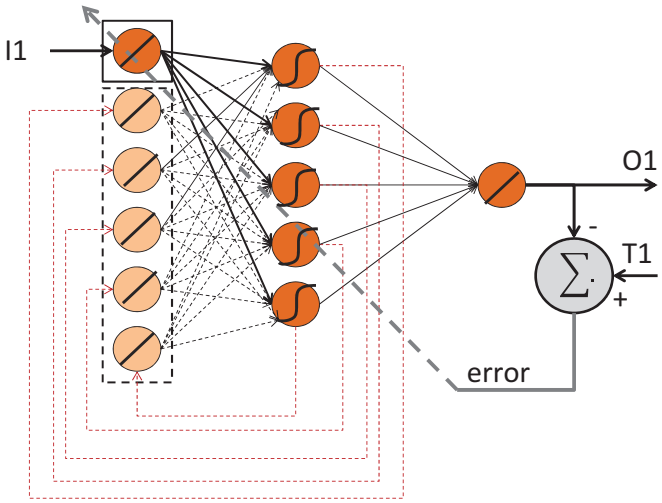

Fig. 1. Internal structure of an MLP (solid lines) and RNN (dashed lines) input and five context neurons, five hidden neurons and one output neurons. The error between the output and the target is used to adjust the weights of the RNN using backpropagation.

$$\vec{H} = \frac{1}{1 + e^{-(inW\vec{eights} \times \vec{I})}} \tag{1}$$

$$\vec{O} = \vec{H} \times outW\vec{eights} \tag{2}$$

RSCAD is the GUI for system development and visualization [29]. It provides library of power system and control components and an interface for connection of these components for simulation on the real-time hardware. It also provides an interface called the 'CBuilder' for users to build their own components that can be added to the RSCAD library and used in real-time simulation. In this study, neural networks library consisting of components for different architectures and training modes are developed. The set of libraries consists of the components that provide the following NN architectures for training in online and batch mode each:

1) Feedforward NN
2) Elman Recurrent NN
3) Jordan Recurrent NN
4) Simultaneous Recurrent NN

The NN in RSCAD is developed as a three-layered architecture with input and output neurons having a linear activation function and the hidden neurons having a sigmoid activation function. The component allows selection of different types of architectures (MLP, RNN, SRN) and training modes (online, batch) using backpropagation learning method [30]. One sample of input data is fed into the RNN and an output is generated at each step of the RTDS®operation. The part of the program associated with the NN component operates on the input, calculates an output based on the weights and transformation from the hidden layer, and compares it with the provided target. The error based on the output and the target is then used for updating the weights of the neural network using backpropagation method. This process continues until

the simulation is stopped. At the end of the simulation, the trained weights can be captured and used for testing and implementation purposes. An external switch is provided in each component that allows to switch the operation mode from training to testing.

Fig. 2 shows the RTDS®, and the 'Draft' and 'Runtime' interface of RSCAD used in this study. The 'Draft' shows the built in libraries (Power System, Controls, Generator Controls, Protection & Automation, Small_dt) under "Master" library. The neural network library developed in this study is shown under the "User" library (rtpisnn2012). The NN block is designed as a GUI with variable number of inputs up to 10, a variable number of outputs up to five, and equal number of target inputs for training it. There are two aspects of developing a component in RSCAD - the graphics and the code. Variable graphical structure of the component block based on the input parameters is achieved by having conditional statements in the component definition file. The CBuilder interface uses slightly customized form of C programming language for developing the code associated with the component. Parameters defined for the component appear on the component property window and are accessible in the associated C program for function implementation. Figs. 3 and 4 show the properties windows of a recurrent NN when trained online and in batch, respectively. In the testing mode, the component also allows loading of external weights. The different parameters shown in the property window are explained below:

- $inNum$: Number of neurons in the input layer
- $hidNum$: Number of neurons in the hidden layer
- $outNum$: Number of neurons in the output layer
- lg: Learning rate
- mg: Momentum gain
- $samplerate$: Rate at which the signal is sampled before feeding into the NN
- $epochs$: Number of times each sample is presented to the NN
- $batchsize$: Number of samples in each batch before the weights are updated
- $batchloops$: Number of times each batch is presented to the NN
- $type$: Type of NN architecture - recurrent (RNN) vs. simultaneous recurrent (SRN)
- $recurrences$: Number of internal recurrences for SRN

The RTDS®runs at a frequency of 20 KHz. Therefore, the requirement for real-time operation is that all the components in a simulation case should complete within a period of 50 μs. Therefore, the maximum number of neurons that can be implemented on one neural network component were empirically determined to be 10 in the input layer, 20 in the hidden layer, and five in the output layer, such that all the computations can be completed within the time step. Fig. 5 shows different steps for implementing a NN component in RTDS®and their timing requirements. The complete implementation of the NN library in RSCAD using CBuilder is explained in the pseudocode.

Algorithm 1 MAIN: NN COMPONENT IN RSCAD

```
INITIALIZE
inWeights[inNum][hidNum]
outWeights[hidNum][outNum]
deltaInWeights[inNum][hidNum]
deltaOutWeights[hidNum][outNum]
count = 0
epochs = 0
batchCount = 0
srnLoop = 0
if (TRAIN/TEST is FALSE) then
   load weights
end if
if (TRAIN/TEST is TRUE) then
   initialize random weights
end if
repeat
   At each time step:
   if (count%samplerate) then
      read input
      read target
      read NN_TYPE
      read TRAIN/TEST
      read MODE
   end if
   if (TRAIN/TEST is TRUE) then
      output = TRAIN() : Algorithm (2)
   else
      output = TEST() : Algorithm (3)
   end if
   write output
   count ← count + 1
until STOPPED
```

IV. Application and Results

Neural networks components developed in RSCAD are tested for predicting the speed deviation of a generator in a multi-machine power system simulated in RTDS®. Feed-forward and recurrent architectures are trained using online as well as batch mode. The experiments are also repeated for different values of learning rate and momentum gain in order to make a comparison of performance between the architectures, the learning methods and the impact of variables parameters in NN learning. These multiple case studies carried out also demonstrate the useful features made available in the developed NN libraries, and how these will be important in real-time power system simulation. The MLP is trained using time delayed values and hence consists of four neurons in the input, five in the hidden and one in the output layer. Similarly, the Elman RNN consists of two neurons in the input, five in the hidden and one in the output layer. The input to the NN is sampled at 100 Hz (i.e. every 200 samples as indicated by $samplerate$ parameter in the components' properties window for the simulation time step of 50μs). During batch training,

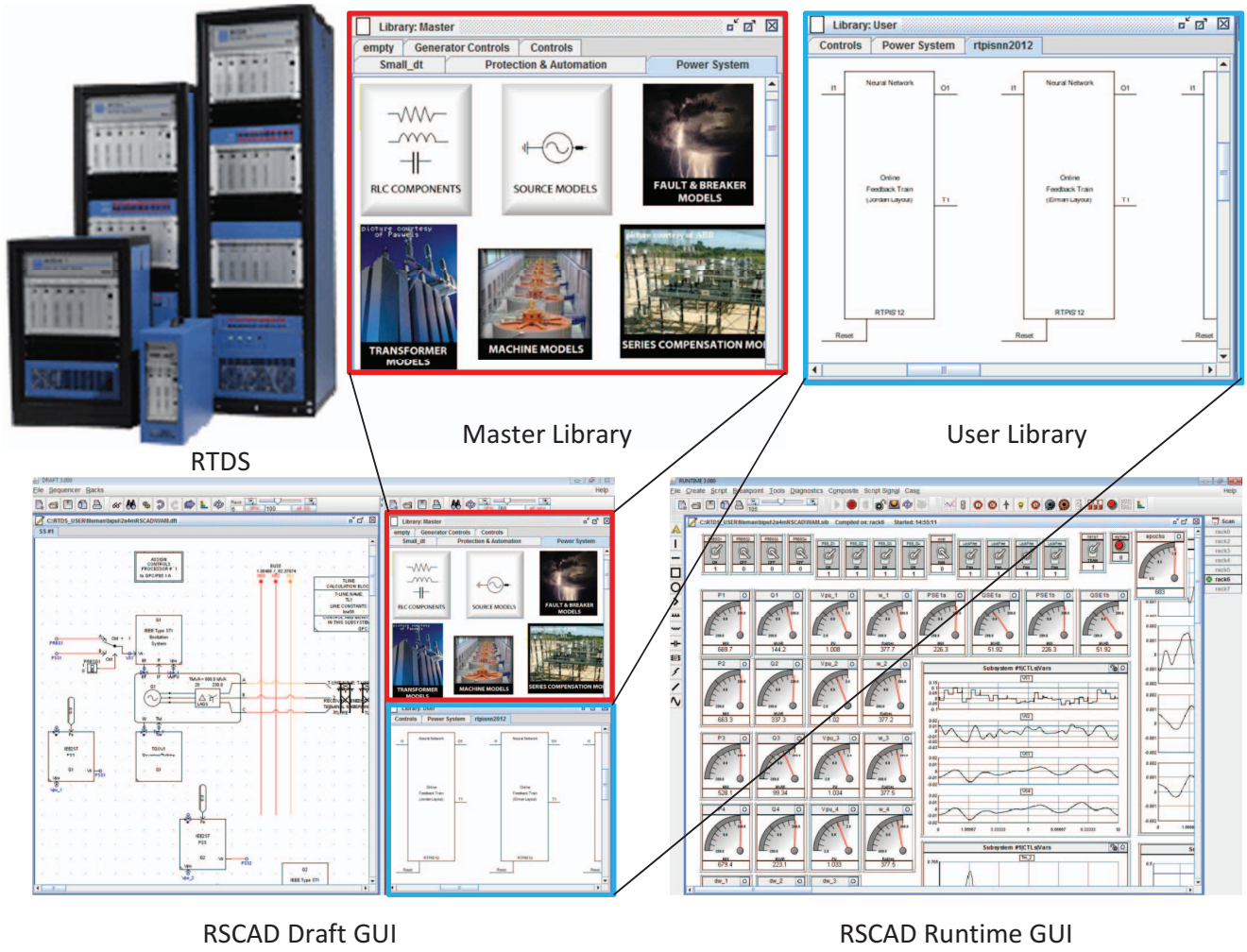

Fig. 2. RTDS® and its RSCAD GUI showing Draft and Runtime Window. The Draft shows the default RSCAD libraries and the developed NN library.

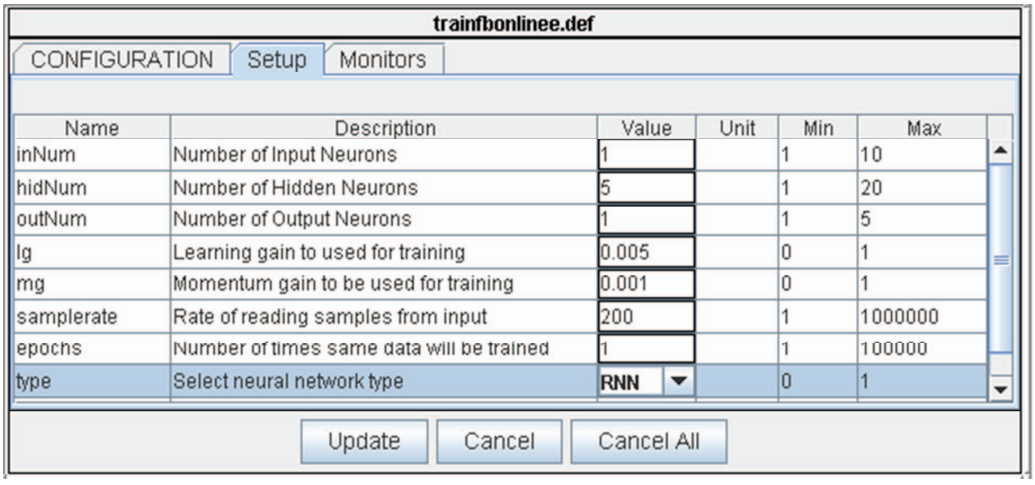

Fig. 3. Parameters associated with the neural network architecture and learning method can be defined by the user using the component's properties window. The figures shows the window for a recurrent architecture set up for online mode.

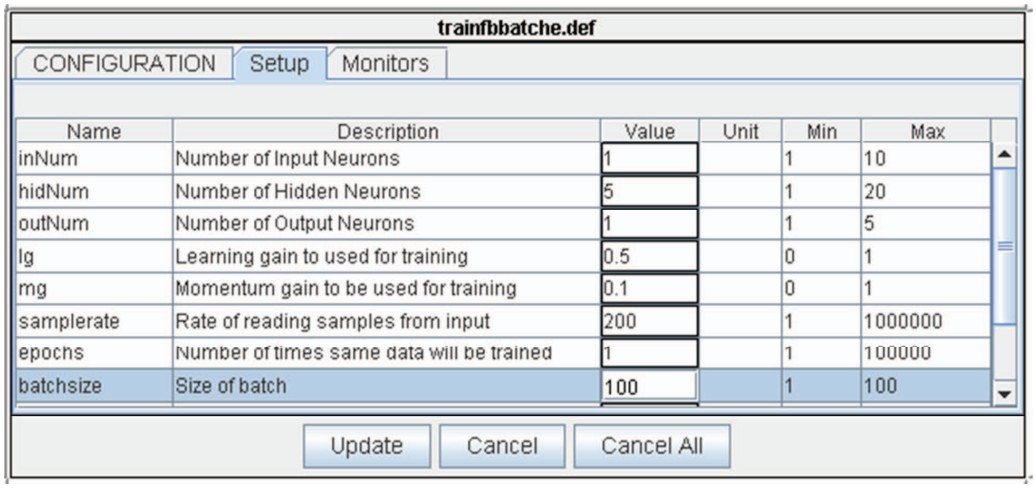

Fig. 4. Parameters associated with the neural network architecture and learning method can be defined by the user using the component's properties window. The figure shows the window for a recurrent architecture set up for batch mode.

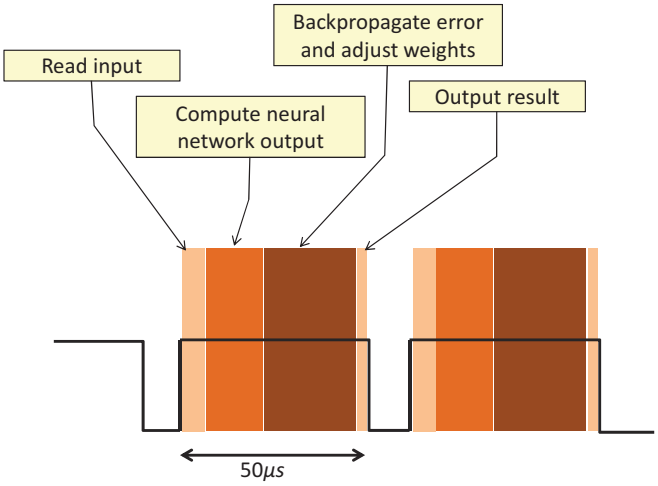

Fig. 5. Figure shows that the internal operations of RTDS need to be completed within one time step for it to execute in real-time. The width of the bands in figure is for visualization only and time taken by individual operations may be different.

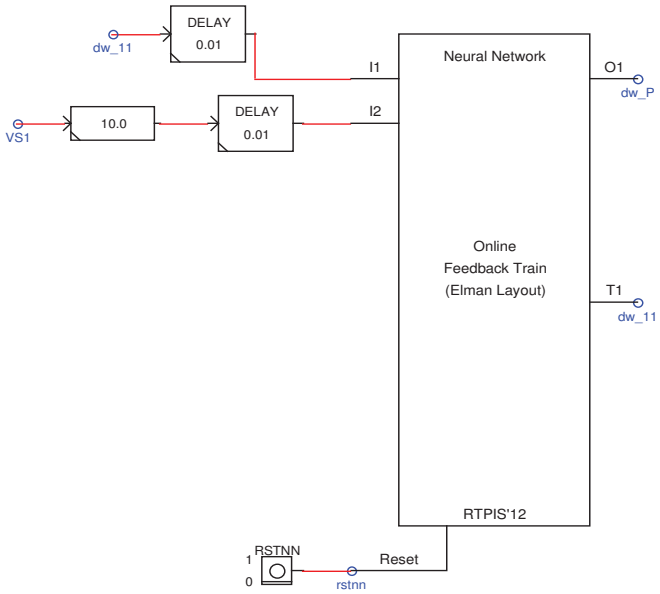

Fig. 6. A RSCAD draft case uses the developed RNN component for learning the speed deviation of a generator in a multi-machine power system.

a *batchsize* of 100 samples is used. Fig. 6 shows the NN block set up for predicting the speed deviations of a generator using an RNN. Fig. 7 shows the two-area four-machine power system used in this study.

The multi-machine power system simulated in RTDS® is perturbed using a pseudo-random binary signal. The speed deviation of the generator, and the change in reference voltage at the excitation system as a result of the perturbation are used as inputs to the NN component. After training the NN for 10,000, 50,000 and 100,000 samples, it is tested by stopping the training, thus freezing the weights while the signals from system continue to feed into the NN. This is considered as Case I. The trained NN is then also tested on a different operating point by simulating a 10-cycle three-phase short circuit fault on one of the buses of the power system. This is

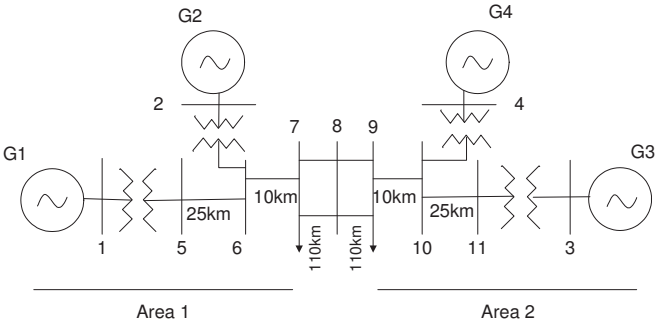

Fig. 7. The power system used in this study consists of four generators spread across two areas. The case studies referred in the paper are carried out on generator G1 of this system.

Algorithm 2 FUNCTION: TRAIN()

$output \leftarrow$ TEST() : Algorithm (3)
$error \leftarrow target - output$
BACKPROP($error$) : Algorithm (4)
if (MODE is $batch$) **then**
  if ($batchCount = batchsize$) **then**
    $inWeights \leftarrow inWeights + deltaInWeights$
    $outWeights \leftarrow outWeights + deltaOutWeights$
    $batchCount \leftarrow 0$
    $epochs \leftarrow epochs + 1$
  else
    accumulate $deltaInWeights$
    accumulate $deltaOutWeights$
    $batchCount \leftarrow batchCount + 1$
  end if
else
  $inWeights \leftarrow inWeights + deltaInWeights$
  $outWeights \leftarrow outWeights + deltaOutWeights$
  $epochs \leftarrow epochs + 1$
end if

return $output$

Algorithm 3 FUNCTION: TEST()

if (NN_TYPE is MLP) **then**
  create input vector $\vec{I}$ from $input$
end if
if (NN_TYPE is $Elman$) **then**
  create input vector $\vec{I}$ from $input$ and feedback from the hidden layer neurons
end if
if (NN_TYPE is $Jordan$) **then**
  create input vector $\vec{I}$ from $input$ and feedback from the output layer neurons
end if
if (NN_TYPE is SRN) **then**
  while ($srnLoop < recurrences$) **do**
    calculate output of hidden neurons using (1)
  end while
end if
calculate $output$ using (2)

return $output$

Algorithm 4 FUNCTION: BACKPROP($error$)

$deltaInWeights \leftarrow \frac{\delta(error)}{\delta(inWeights)}$
$deltaOutWeights \leftarrow \frac{\delta(error)}{\delta(outWeights)}$

TABLE I
COMPARISON OF MEAN SQUARED ERRORS (MSES) OBTAINED DURING TEST CASES.

	Case I: MSE ($\times$ 10^{-3})					
	$lg = 0.5, mg = 0.1$			$lg = 0.005, mg = 0.001$		
Training Samples	10k	50k	100k	10k	50k	100k
MLP Online	27.8	24.5	0.64	2	0.826	0.736
RNN Online	45.9	5.6	5.6	10.3	1.3	**0.303**
MLP Batch	55.2	0.55	2	195.3	87.4	43
RNN Batch	25.4	**0.51**	1.7	128.7	74.4	42.2
	Case II: MSE ($\times$ 10^{-3})					
MLP Online	182.9	94.3	21.7	48.3	31.9	28.2
RNN Online	249.8	94.3	28.3	80.2	9.9	**4.1**
MLP Batch	41.5	13.6	**12.1**	185.3	145.2	141.5
RNN Batch	59.7	24.7	17.3	251.8	144.6	138.1

considered as Case II. The fault results in speed deviations of the generators that was not seen by the NN during the training. The results of these two test outputs are then analyzed. In this study, the mean squared error (MSE) between the actual signal (dw) and the predicted output of NN (dw_P) obtained during testing is used as the performance index for measuring the learning in NN. The MSEs thus obtained for different case studies are listed in Table I. The result shows that both MLP and RNN learn better with smaller learning rate when trained online, and with higher learning rate when trained in batch with a size of 100 samples. The results also show that MLP leads to better predictions when trained in batch while RNN leads to better predictions when trained online. RNN trained online with a small learning rate had the best predictions among all combinations in both cases. The best results obtained by RNN compared with the actual signal for Case I are shown in Fig. 8. These results are obtained when trained online using a small learning rate. The best results obtained by MLP compared with the actual signal for Case I are shown in Fig. 9. These results are obtained when trained in batch using a high learning rate. Similarly, the best results obtained by RNN and MLP for Case II are shown in Figs. 10 and 11. Fig. 12 shows the absolute errors between the actual and the predicted signals. It needs to be noted that the testing data fed to the NN is different for each different combinations of learning rate and the number of epochs (number of samples). It is also important to note that these results may be application specific and will require further investigation in focused studies to be considered conclusive of the observations made in the paper.

Development of NN libraries in RSCAD allows use of these components along with power system simulation as shown in these applications. This means, intelligent neurocontrol methods, and methods of estimation and prediction can be integrated in power system simulation. These components are connected 'online' with the system in simulation. This allows them to be tested on different operating conditions while the system is running. Therefore, they more closely represent the actual implementation when these systems are deployed in the field. Therefore, inclusion of CI components in the library of simulation toolboxes will help faster integration of intelligent

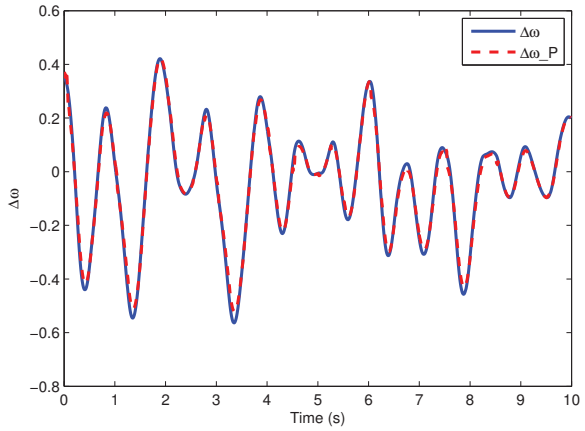

Fig. 8. The results show the ability of an online trained RNN to predict the speed deviations of a generator during testing on the same operating point.

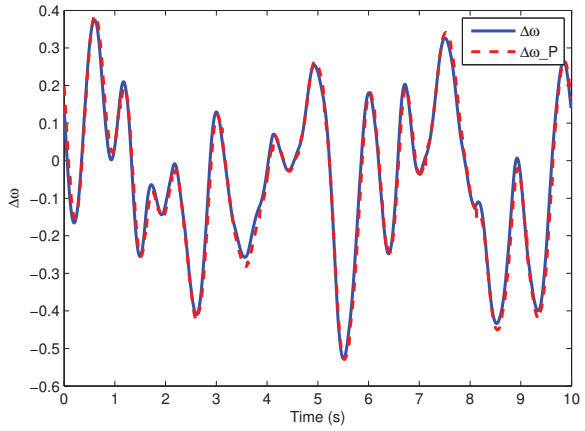

Fig. 9. The results show the ability of a batch trained MLP to predict the speed deviations of a generator during testing on the same operating point.

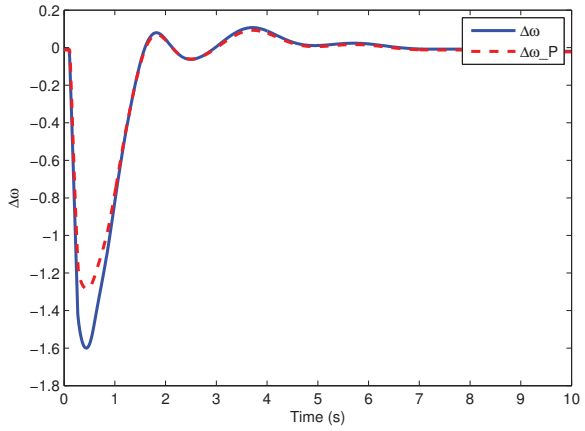

Fig. 10. Test results show the ability of an online trained RNN to predict the speed deviations of a generator resulting from a 3-phase 10-cycle short circuit fault.

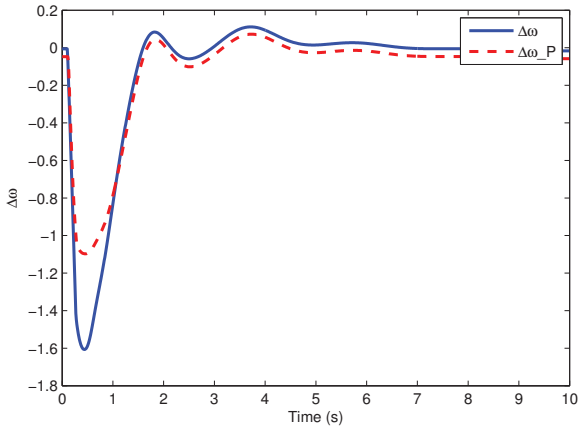

Fig. 11. Test results show the ability of a batch trained MLP to predict the speed deviations of a generator resulting from a 3-phase 10-cycle short circuit fault.

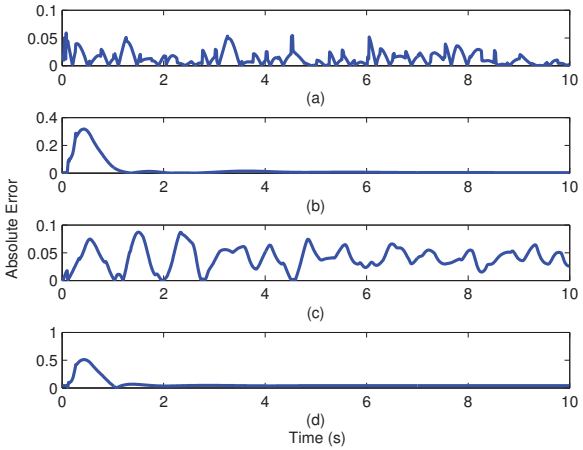

Fig. 12. Absolute error between the actual and the predicted signals during testing for online trained RNN in (a) Case I and (b) Case II, and batch trained MLP in (c) Case I and (d) Case II.

monitoring and control methods in smart grids. Apart from that, they also act as great learning tools for CI applications in power systems.

V. Conclusion

Most past and current applications of neural networks in power systems are done offline in non-real-time platforms. In this paper, neural network components are developed for a real-time hardware platform using RSCAD. The components are used to show application in real-time simulation of power systems. As the power system community is moving towards a more powerful, reliable and intelligent electric power grid, the components of computational intelligence based technology become important. Therefore, it is necessary to have such components in real-time simulation studies related to smart grids. The development of neural networks library for simulation

on RTDS®is discussed and their utility is illustrated through a simple application. This study leaves many open avenues for future research related to addition of CI components in real-time power system simulation. It also opens new areas of research related to real-time learning in neural networks and how that is different from offline learning. These will be future work in this area.

REFERENCES

[1] RTDS, "Clients of RTDS technologies," http://www.rtds.com/clients-rep/clients.html,Last accessed: Nov 21, 2012.

[2] OPAL-RT, "Clients of Opal-RT technologies," http://www.opal-rt.com/Customers,Last accessed: Nov 21, 2012.

[3] G. K. Venayagamoorthy, "Future grid will not be controllable without thinking machines," *IEEE Smart Grid Newsletter*, Nov. 2011.

[4] P. Werbos, "Putting more brain-like intelligence into the electric power grid: What we need and how to do it," in *International Joint Conference on Neural Networks*, 2009, pp. 3356–3359.

[5] S. Yang and L. Yin, "Application of intelligent control based on neural networks in power system," in *IEEE Conference on Software Engineering and Service Science (ICSESS)*, July 2011, pp. 348 –351.

[6] W. Chen, Q. Gong, C. Yin, T. Wang, and J. Yao, "An Elman neural network application on dynamic equivalents of power system," in *International Conference on Electrical and Control Engineering (ICECE)*, June 2010, pp. 376 –379.

[7] H. Li and L. Yin, "Control of turbine and excitation in power system based on neural networks," in *International Conference on Electrical and Control Engineering*. IEEE, 2010, pp. 3485–3488.

[8] S. Ray, G. Venayagamoorthy, B. Chaudhuri, and R. Majumder, "Comparison of Adaptive Critic-Based and Classical Wide-Area Controllers for Power Systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 38, no. 4, pp. 1002–1007, 2008.

[9] G. Venayagamoorthy, "Dynamic optimization of a multimachine power system with a FACTS device using identification and control ObjectNets," in *IEEE Industry Applications Conference*, vol. 4, Oct. 2004, pp. 2643 – 2650.

[10] V. S. S. Vankayala and N. D. Rao, "Artificial neural networks and their applications to power systems–a bibliographical survey," *Electric Power Systems Research*, vol. 28, no. 1, pp. 67 – 79, 1993.

[11] Y. Zhang, O. Malik, and G. Chen, "Artificial neural network power system stabilizers in multi-machine power system environment," *IEEE Transactions on Energy Conversion*, vol. 10, no. 1, pp. 147 –155, Mar 1995.

[12] Y.-M. Park, M.-S. Choi, and K. Lee, "A neural network-based power system stabilizer using power flow characteristics," *IEEE Transactions on Energy Conversion*, vol. 11, no. 2, pp. 435 –441, Jun 1996.

[13] M. Bostanci, J. Koplowitz, and C. Taylor, "Identification of power system load dynamics using artificial neural networks," *IEEE Transactions on Power Systems*, vol. 12, no. 4, pp. 1468 –1473, Nov 1997.

[14] R. Aggarwal and Y. Song, "Artificial neural networks in power systems. I. General introduction to neural computing," *Power Engineering Journal*, vol. 11, no. 3, pp. 129 –134, June 1997.

[15] ——, "Artificial neural networks in power systems. II. Types of artificial neural networks," *Power Engineering Journal*, vol. 12, no. 1, pp. 41 –47, Feb. 1998.

[16] ——, "Artificial neural networks in power systems. III. Examples of applications in power systems," *Power Engineering Journal*, vol. 12, no. 6, pp. 279 –287, Dec. 1998.

[17] M. Reaz, F. Choong, M. Sulaiman, F. Mohd-Yasin, and M. Kamada, "Expert system for power quality disturbance classifier," *IEEE Transactions on Power Delivery*, vol. 22, no. 3, pp. 1979 –1988, July 2007.

[18] V. Ferreira and A. Alves da Silva, "Toward estimating autonomous neural network-based electric load forecasters," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1554 –1562, Nov. 2007.

[19] P. Mandal, T. Senjyu, N. Urasaki, T. Funabashi, and A. Srivastava, "A novel approach to forecast electricity price for PJM using neural network and similar days method," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 2058 –2065, Nov. 2007.

[20] R. Kuffel, J. Giesbrecht, T. Maguire, R. Wierckx, and P. McLaren, "RTDS - a fully digital power system simulator operating in real time," in *IEEE Communications, Power, and Computing Conference*, vol. 2, 1995, pp. 300–305.

[21] I.-Y. Chung, W. Liu, K. Schoder, and D. A. Cartes, "Integration of a bi-directional DC-DC converter model into a real-time system simulation of a shipboard medium voltage DC system," *Electric Power Systems Research*, vol. 81, no. 4, pp. 1051 – 1059, 2011.

[22] P. Tatcho, Y. Zhou, H. Li, and L. Liu, "A real time digital test bed for a smart grid using RTDS," in *Symposium on Power Electronics for Distributed Generation Systems*, June 2010, pp. 658 –661.

[23] K. Ravikumar, N. Schulz, and A. Srivastava, "Distributed simulation of power systems using real-time digital simulator," in *IEEE/PES Power Systems Conference and Exposition*, March 2009, pp. 1 –6.

[24] P. Mitra, K. Vinothkumar, and L. Zhang, "Dynamic performance study of a HVDC grid using real-time digital simulator," in *Complexity in Engineering (COMPENG), 2012*, June 2012, pp. 1 –6.

[25] C. Liu, Z. H. Rather, N. Stearn, Z. Chen, C. L. Bak, and P. Thogersen, "Practical testing and performance analysis of phasor measurement unit using real time digital simulator (RTDS)," in *IEEE International Energy Conference and Exhibition (ENERGYCON)*, Sept. 2012, pp. 408 –414.

[26] H. Chen, B. Bhargava, F. Habibi-Ashrafi, J. Park, and J. Castaneda, "Integration of RTDS with EPG synchrophasor applications for visualization and analysis of simulation scenarios at Southern California Edison," in *North American Power Symposium*, Sept. 2012, pp. 1 –5.

[27] D. S. Ouellette, M. D. Desjardine, and R. Kuffel, "Using a real time digital simulator with phasor measurement unit technology," in *11th International Conference onDevelopments in Power Systems Protection*, April 2012, pp. 1 –6.

[28] U. Adhikari, T. H. Morris, N. Dahal, S. Pan, R. L. King, N. H. Younan, and V. Madani, "Development of power system test bed for data mining of synchrophasors data, cyber-attack and relay testing in RTDS," in *IEEE Power and Energy Society General Meeting*, July 2012, pp. 1 –7.

[29] RTDS, "Real time digital simulator tutorial manual (RSCAD version)," RTDS Technologies, Nov. 2010.

[30] P. Werbos, *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*. Wiley-Interscience, 1994, vol. 1.