Neural Networks 22 (2009) 833–841

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2009 Special Issue

Comparison of a spiking neural network and an MLP for robust identification of generator dynamics in a multimachine power system

Cameron Johnson *, Ganesh Kumar Venayagamoorthy, Pinaki Mitra

Real-Time Power and Intelligence Systems Laboratory, Missouri University of Science and Technology, Rolla, MO 65401, USA

ARTICLE INFO

Article history:
Received 7 May 2009
Received in revised form 7 June 2009
Accepted 25 June 2009

Keywords:
MLP
Multimachine power system
Neuroidentification
Spiking neural network

ABSTRACT

The application of a spiking neural network (SNN) and a multi-layer perceptron (MLP) for online identification of generator dynamics in a multimachine power system are compared in this paper. An integrate-and-fire model of an SNN which communicates information via the inter-spike interval is applied. The neural network identifiers are used to predict the speed and terminal voltage deviations one time-step ahead of generators in a multimachine power system. The SNN is developed in two steps: (i) neuron centers determined by offline *k-means* clustering and (ii) output weights obtained by online training. The sensitivity of the SNN to the neuron centers determined in the first step is evaluated on generators of different ratings and parameters. Performances of the SNN and MLP are compared to evaluate robustness on the identification of generator dynamics under small and large disturbances, and to illustrate that SNNs are capable of learning nonlinear dynamics of complex systems.

1. Introduction

Online identification of generator speed and terminal voltage characteristics is very much essential for fast, intelligent and adaptive control in today's power system (Singh & Venayagamoorthy, 2002). Classical controllers for generators are generally designed based on linearized models obtained around some nominal operating point. But, in a real world power system, the environment continuously changes and a generator's dynamics also change since it is an integrated part of the power system (Park, Venayagamoorthy, & Harley, 2005). In these situations, the performance of classical controllers such as an automatic voltage regulator (AVR) generally degrades, and intelligent AVR designs are called for. A Neural Network (NN) is a very effective tool for designing these types of intelligent controllers. In order to take the correct control action in a dynamically changing environment, an NN based controller needs a neuroidentifier, which provides an estimation of the speed and terminal voltage characteristics of a generator from past values of speed and terminal voltage. The method of neuroidentification is also very effective for wide area monitoring and control (Venayagamoorthy, 2007) and finding dynamic equivalents of large power systems (Azmy, Erlich, & Sowa, 2004; Stankovic, Sarik, & Milosevic, 2003).

So far, different types of neural network architectures and their performances have been studied for the purpose of neuroidentification (Azmy et al., 2004; Park et al., 2005; Singh & Venayagamoorthy, 2002; Venayagamoorthy, 2007). This includes Multilayer Perceptrons (MLPs), Radial Basis Functions (RBFs), Recurrent Neural Networks (RNNs), and Echo-State Networks (ESNs). But with the advancement of neuroscience it has become clear that none of these networks actually represents the structure and function of biological neurons (Mishra, Yadav, Ray, & Kalra, 2007).

In a traditional neural network, a complete set of inputs is fed into the network, each neuron is allowed to activate once, and a single set of outputs is produced in a single time slice. There is no master clock; the network operates in time steps defined by these samples of inputs and their resulting outputs. The biological neuron (and any network based thereon), however, operates in continuous time. Inputs come in as a string of voltage spikes called "action potentials". A traditional artificial neuron uses weighted multipliers and simple summation to generate a net input value for an activation function, whose output is the neuron's output. Inputs and outputs are real-valued numbers. A biological neuron receives the action potentials, which drive up the voltage on its main body's membrane. The voltage on the main body (called the "soma") decays quickly, but if enough spikes arrive (usually from multiple neurons) in a short enough period of time, the biological neuron fires (Mishra et al., 2007).

The artificial neuron designed to more closely model those found in biological systems is known as a spiking neuron, and a network based upon this type of neuron is referred to as Spiking Neural Network (SNN). First proposed in the Hodgekins/Huxley model in 1959 (Mishra et al., 2007), an artificial spiking neuron

* Corresponding author.
E-mail addresses: cameron.e.johnson@gmail.com (C. Johnson), gkumar@ieee.org (G.K. Venayagamoorthy), pm33d@mst.edu (P. Mitra).

0893-6080/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2009.06.033

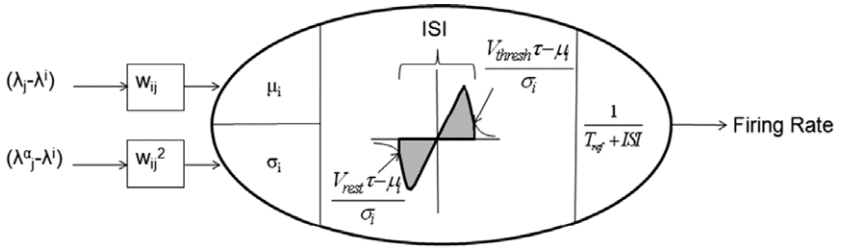

Fig. 1. A spiking neuron model used in the SNN. The inputs to the network are real-valued, and the firing rate serves as a real-valued output.

poses several problems for use: encoding/decoding of information, control of information flow, and computational complexity. Because of all of the variables that must be tracked in a faithful digital modeling of a biological neuron – most notably an internal clock that tracks how long it has been since the last spike, and constant monitoring of the current membrane potential at the current time – it can be far more computationally costly than a traditional neuron. The methods of encoding real values into spike trains include temporal encoding – the time between spikes (referred to as the Inter-Spike Interval, or *ISI*) or the time-to-next-spike (Iannella & Back, 1999; Rowcliffe & Feng, 2008) – and probability-based encoding. In Rowcliffe and Feng (2008), the model of the spiking neuron abstracts the spiking function by finding the mean and standard deviation of the voltage potential in the soma. It is this design of spiking neuron that is used successfully in Johnson, Venayagamoorthy, and Mitra (2009) as a neuroidentifier to estimate the speed and terminal voltage characteristics of a single generator in a multimachine power system. SNN is shown to perform quite well, promising to be an effective choice in use as a model for one-step-ahead prediction.

This paper extends upon the study in Johnson et al. (2009). Both an SNN and an MLP are fed newly generated data from two generators of different ratings and parameters in an IEEE 10-machine-39-bus New England power system. Using three past-time-step values of terminal voltage deviation and speed deviation, the SNN is trained to predict these values one-step-ahead. The performance of the SNN and the MLP are compared, and the SNN's robustness is shown by its ability to perform on more than one generator and its operating points. In addition, Johnson et al. (2009) and Rowcliffe and Feng (2008) discuss the similarities of the SNN design to that of an RBF; this paper examines the sensitivity of the SNN to its "centers" when identifying dynamics of generators of different ratings and parameters on a multimachine power system.

The rest of the paper is organized as follows: Section 2 presents the fundamentals of an SNN, Section 3 discusses the structure SNN based neuroidentifier, Section 4 shows typical and finally, the conclusion and scope of future work are presented in Section 5.

2. A spiking neuron model

The difficulty in encoding based directly on firing time, or any other temporal method, lies in the necessity not only of a master clock to keep track of when each spike is fired, when each spike is received, and the potential voltage in the soma of the artificial neuron at any given point in time, but in determining when a single complete set of inputs have been passed in, and when a single complete set of outputs have been received. The mean firing time encoding method eliminates this problem by abstracting the spike times, freeing the artificial SNN to operate in epochs just like a traditional network. This somewhat reduces the spiking neuron model to a complicated activation function, but the performance is still quite rich, with a great deal more flexibility and adaptability than in more traditional neuron models.

The model introduced in Rowcliffe and Feng (2008) and expanded upon in this paper is used in a feedforward network, and is explained here. The inputs from the previous layer (whether other neurons or the external real-valued inputs to the SNN) are collected and aggregated by (1) and (2) to generate the mean μ_i and standard deviation σ_i of the membrane potential for neuron i over a given time-slice represented by a single sample of inputs and outputs.

$$\mu_i = \sum_{j=1}^{n} (\lambda_j - \lambda^i) w_{ij} (1 - r) \tag{1}$$

$$\sigma_i^2 = \left(\sum_{j=1}^{n} (\lambda_j^\alpha - \lambda^i) w_{ij}^2 \right) (1 + r) + \rho \sum_{j \neq h=1}^{n} \left(\lambda_j^{\frac{\alpha}{2}} - \lambda^i \right) \left(\lambda_h^{\frac{\alpha}{2}} - \lambda^i \right) w_{ij} w_{ih} (1 + r). \tag{2}$$

The jth input into the SNN is λ_j, α is a tuning constant greater than 0, and r is the ratio of excitatory to inhibitory inputs. The superscripted λ values are centers, used to offset a given neuron in much the way a radial basis function (RBF) works. The weights connecting neuron j to input i are given by w_{ij}, with ρ the correlation factor between neurons j and h (whose connections to input i are given by w_{ih}). This unique structure allows for an equal number of excitatory and inhibitory inputs (caused when $r = 1$) to not completely damp out the neuron's ability to generate meaningful output (Rowcliffe & Feng, 2008). The inputs, once converted to mean and standard deviations of voltage spikes, are used to generate the limits of integration that determine the *ISI* of the neuron, which is finally converted into a firing rate that serves as the real-numbered output, as shown in Fig. 1.

When $r = 1$, the output of the single neuron using two inputs and outputs is a paraboloid, as shown in Fig. 2. By varying r, the output range of the neuron can be altered significantly. However, this output structure resembles an RBF neuron with a Gaussian activation function closely enough that, for the experiments in this paper, the value for r is left at 1.

The actual output of the neuron is the "firing rate", which serves as the real value the neuron is to be calculating, with no further decoding necessary. To determine this firing rate, two more pieces of information are required: the *ISI* and the "refractory period" T_{ref}. After a biological neuron fires off a spike due to having its membrane's threshold potential exceeded, the membrane is actually at a lower potential than its resting potential. The refractory period is the amount of time it takes for this potential to rise back to the resting state, and, in this model, is a user-set parameter.

The *ISI* is given by (3), with $g(x)$ being Dawson's Integral (given in (4)). The relaxation period of the neuron – how long it takes for a spike's influence on the membrane potential to fade – is given by τ. V_{rest} and V_{thresh} are the resting and threshold potentials of the membrane, respectively.

$$ISI = \frac{2}{\tau} \int_{\frac{V_{rest}\tau - \mu_i}{\sigma_i}}^{\frac{V_{thresh}\tau - \mu_i}{\sigma_i}} g(x) \mathrm{d}x \tag{3}$$

$$g(x) = \mathrm{e}^{x^2} \int_0^x \mathrm{e}^{-u^2} \mathrm{d}u. \tag{4}$$

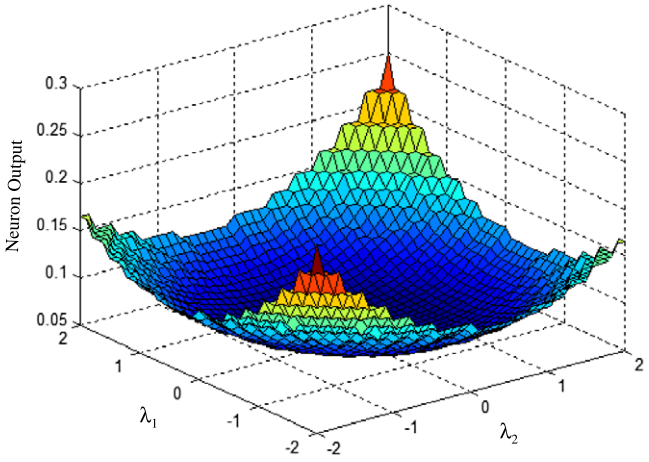

Fig. 2. Output of a spiking neuron with $r = 1$ and $\alpha = 2$, with λ_1 and λ_2 varying independently from −2 to 2 with a step size of 0.01.

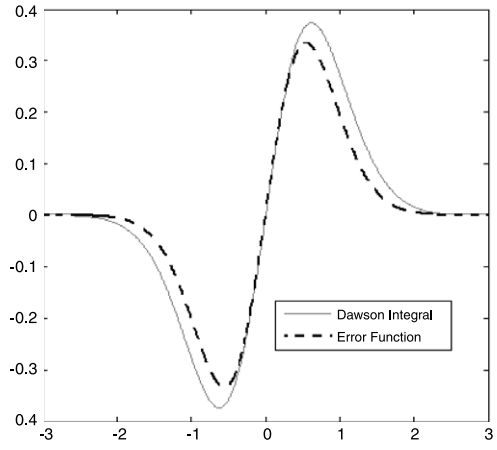

Fig. 3. Error function (bold line) and the Dawson Integral (light line).

Dawson's integral, however, is not a direct analytical calculation. It must be found individually for each given point x, integrating over dummy variable u, which is most easily done with the Maclaurin series expansion. Because of this, it is impossible to analytically perform the integral necessary in (4), and so the Matlab code used to run the experiments in this paper used a Riemann Sum to approximate the integral. Further, because Matlab does not have a built-in Dawson's Integral function, it was found to improve performance speed to use the error function $erf(x)$, given in (5), instead. This has the same shape, with only a slightly different amplitude, as Dawson's Integral, and worked quite well as a substitute. Fig. 3 shows the comparison of the two.

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t} dt. \tag{5}$$

Finally, the output of a spiking neuron (the firing rate) is calculated by using the ISI as determined in (3), and is given by (6).

$$f_i(\lambda) = \frac{1}{T_{ref} + ISI}. \tag{6}$$

Spiking neurons constructed in this manner are used as the hidden layer neurons in a feedforward network. The input layer consists of linear neurons that provide direct inputs to the network, and the output layer consists of a number of linear weighted-sum neurons equal to the number of desired outputs. It is worth noting that only the weights between the hidden layer and the output layer (output weights) are trained in this kind of network. The weights between the input layer and hidden layer (input weights) are set, in this experiment, to a constant value of 0.5. The reduction in the number of weights that need to learn greatly simplifies training, and had no deleterious effect on the performance of the network.

Table 1
Operating points of the generators on the IEEE 39 machine power system.

	G2	G3	G4	G5	G6	G7	G8	G9	G10
P (p.u.)	0.86	0.85	0.90	0.83	0.81	0.85	0.82	0.72	0.83
Q (p.u)	0.32	0.26	0.17	0.27	0.25	0.16	0.01	0.02	0.47
V (p.u)	0.9992	0.984	0.9977	1.013	1.052	1.067	1.028	1.027	1.049

In contrast with the MLP, which relies on backpropagation to learn the weights (both input and output), training of the SNN is conducted only on the output weights, and is accomplished via simple gradient-descent learning much like RBF network training (Park et al., 2005) by calculating the error between the desired output and the actual output as shown in (7). In the work done here, the dynamics being identified are the terminal voltage deviation $\Delta\hat{V}_t(k)$ and the speed deviation $\Delta\hat{\omega}(k)$. The input weights are all set to 0.5.

$$Error = \begin{cases} \Delta V_t(k) - \Delta\hat{V}_t(k) \\ \Delta\omega(k) - \Delta\hat{\omega}(k). \end{cases} \tag{7}$$

The change in the output weight (from hidden neuron i to output neuron l) at instant k is given in (8) and (9). The learning rate is denoted by lr.

$$\Delta w_{li}(k) = lr \times Error_l(k) \times f_i(\lambda) \tag{8}$$

$$w_{li}(k+1) = w_{li}(k) + \Delta w_{li}(k). \tag{9}$$

3. SNN based neuroidentifier

A neural network provides a solution for modeling complex, fast-changing multiple-input/multiple-output (MIMO) systems. SNN-based neuroidentifiers are used to learn the dynamics of generators G7 and G10 on the IEEE 39 bus system, as shown in Fig. 4. The real and reactive power outputs (P and Q, respectively) and terminal voltage of each generator in Fig. 4 is shown in Table 1 in per-unit (p.u.). G1 is an infinite bus, and is not listed in the table.

A neuroidentifier is used to predict the terminal voltage deviation and speed deviation of generators G10, connected to bus 30, and G7, connected to bus 36, based on past values at instants $(k-1), (k-2), (k-3)$. A standard feedforward MLP network is also trained as a neuroidentifier to do the same. MLP networks are well-established archetypes, and thus serve as a good base line to which to compare the SNN. The MLP is trained using the backpropagation algorithm, whereas the SNN output weights are determined using a simple gradient-descent algorithm (Fig. 5). The SNN centers λ^i (in (1)) are selected using the offline method of k-means clustering.

The multimachine power system shown in Fig. 4 is modeled on a Real Time Digital Simulator (RTDS), with the generators at the operating points shown in Table 1. The objective of the neuroidentifier in this paper is to estimate the speed and terminal voltage of generators G10 and G7 at kth instant ($\hat{V}_t(k)$ and $\hat{\omega}(k)$) based on the deviation in the speed and terminal voltage at the $(k-1), (k-2), (k-3)$ instances: $\Delta V_t(k-1)$, $\Delta V_t(k-2)$, $\Delta V_t(k-3)$, and $\Delta\omega\,(k-1)$, $\Delta\omega\,(k-1)$, $\Delta\omega\,(k-1)$. The neuroidentifier can then provide information to a controller to respond adaptively to changing environmental conditions by predicting the deviations due to given inputs and compensating for them preemptively. The parameters associated with the SNN are given in Table 2 and

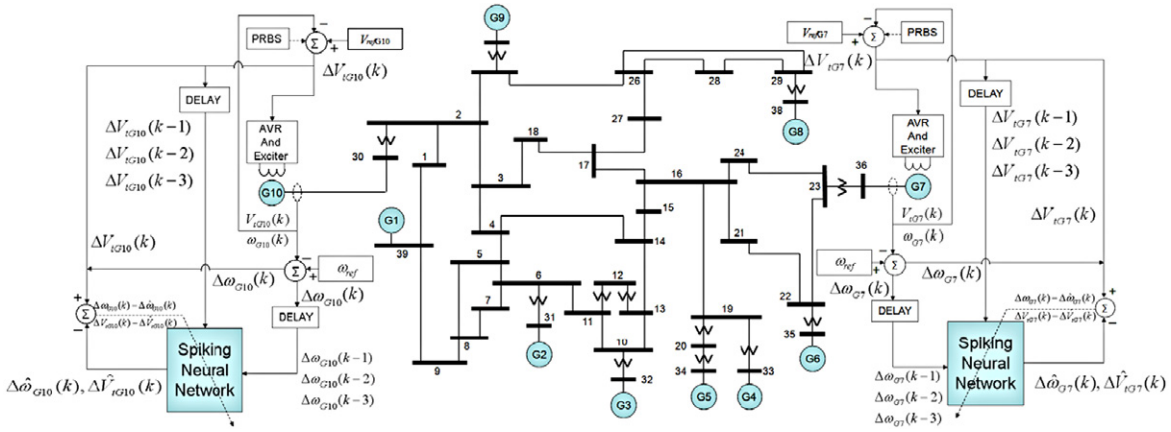

Fig. 4. Schematic of the IEEE 10 generator 39 bus system with SNN-based neuroidentifiers on generators G7 and G10.

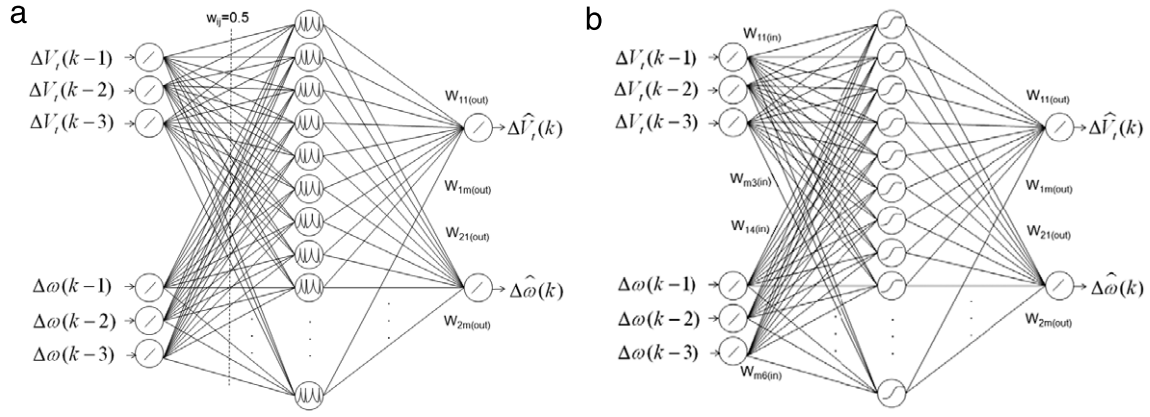

Fig. 5. (a) Diagram of feedforward neural network with spiking neurons in the hidden layer. All input neurons from input j to hidden neuron i are set to 0.5, and are not trained. (b) Diagram of feedforward MLP neural network with sigmoidal activation functions. Input weights are trained using the backpropagation algorithm. Output weights are trained via gradient descent.

Table 2
Parameters for SNN neuroidentifier.

Spiking neural network parameter	Spiking neural network parameter value
Number of inputs	9
Number of outputs	2
Number of neurons	20
α	2
ρ	1
τ	rand
r	1
Learning gain (η)	0.01
T_{ref}	0.5
V_{rest}	0
V_{thresh}	1

explained in great detail in Rowcliffe and Feng (2008). Briefly, however, α is a tuning factor, ρ is the interconnectivity of neural links, and τ is a factor reflecting how quickly the soma of the simulated neuron drains charge. The values for each of these parameters in this paper were 2, 1, and randomly chosen on the interval [0, 1], respectively. The parameters not taken from Rowcliffe and Feng (2008) were either dictated by the form of the problem or found with some trial-and-error in order to optimize just enough to get good results. Optimizing these parameters is beyond the scope of this paper and is left for later work. The MLP parameters are as much the same as possible to those shown for the SNN in Table 2 in order to best compare their performance, though the MLP uses a sigmoidal activation function (10) in its hidden layer neurons, and thus lacks the various parameters relevant to the SNN activation function. The input and output weights of the MLP are trained via a backpropagation algorithm more fully detailed in Park et al. (2005), which contains a thorough comparison of MLP and RBF performance on similar power system dynamics.

$$f(\lambda_i) = \frac{1}{1 + e^{-w_{ij}\lambda_i}}. \tag{10}$$

4. Results and discussion

The three main things studied here are the performance of the SNN in comparison to the MLP on forced training of the system dynamics via PRBS, a comparison of the SNN to the MLP on natural training of the system dynamics with both one- and three-phase-to-ground faults, and an experiment testing the robustness of the SNN when used on different generators with different operating points and parameters. That last experiment specifically explores the sensitivity of the SNN to the choice of neuron centers by utilizing three distinct sets of centers for the SNN: one set is chosen using data taken from generator G7, one set is chosen using data taken from generator G10, and one set is chosen using the combined data sets.

4.1. Forced training

In forced training, a Pseudo Random Binary Signal (PRBS), bounded within the range of ±15% of the normal excitation voltage signal, is applied at the input to the excitation system of the generator whose dynamics are being learned. This method

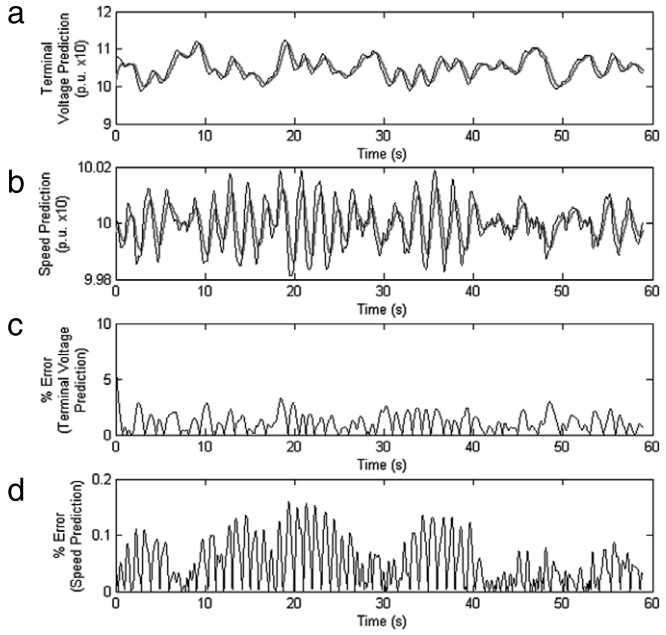

Fig. 6. G7 forced training of the MLP. The signals and their estimations are scaled up by a factor of 10 to allow the neuroidentifier a palatable scale. (a) Estimation of terminal voltage (10 × p.u.); (b) estimation of speed (10 × p.u.); (c) %-error between estimated and actual terminal voltages; (d) %-error between estimated and actual speeds. The thin, black lines in (a) and (b) are target values, and the thick, gray lines are the SNN's estimation.

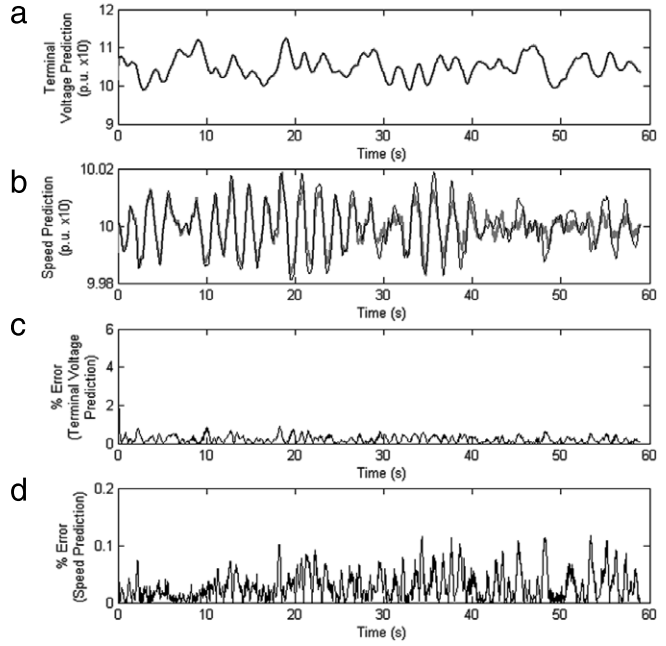

Fig. 7. G7 forced training of an SNN using its own centers. The signals and their estimations are scaled up by a factor of 10 to allow the neuroidentifier a palatable scale. (a) Estimation of terminal voltage (10 × p.u.); (b) estimation of speed (10 × p.u.); (c) %-error between estimated and actual terminal voltages; (d) %-error between estimated and actual speeds. The thin, black lines in (a) and (b) are target values, and the thick, gray lines are the SNN's estimation.

forces all of the harmonic modes of the generator to become active, at least briefly, providing a strong overview of the dynamic properties of the system to the neuroidentifier. This property makes it desirable for the initial training of the neuroidentifier, and also provides a good basis for testing the overall performance of a neuroidentifier under varying parameters. In particular, forced

Table 3
Mean and standard deviations of G7 terminal voltage deviations errors for different numbers of hidden neurons taken over 30 runs.

Number of neurons	SNN, specialized centers		SNN, generalized centers		MLP	
	Mean error	Std. dev. of error	Mean error	Std. dev. of error	Mean error	Std. dev. of error
5	0.5538	0.4724	0.2376	0.3773	1.1357	0.8041
10	0.4066	0.3550	0.1356	0.2730	1.1360	0.8046
15	0.3036	0.2908	0.0978	0.2315	1.1357	0.8043
20	0.2542	0.2592	0.0900	0.2226	1.1360	0.8046
30	0.2159	0.2354	0.0770	0.208	1.1358	0.8046

Table 4
Mean and standard deviations of G7 speed deviations errors for different numbers of hidden neurons taken over 30 runs.

Number of neurons	SNN, specialized centers		SNN, generalized centers		MLP	
	Mean error	Std. dev. of error	Mean error	Std. dev. of error	Mean error	Std. dev. of error
5	0.0118	0.0183	0.0126	0.0204	0.0214	0.0319
10	0.0082	0.0155	0.0081	0.0150	0.0194	0.0286
15	0.0073	0.0151	0.0065	0.0128	0.0178	0.0262
20	0.0069	0.0130	0.0061	0.0142	0.0162	0.0242
30	0.0064	0.0134	0.0053	0.0118	0.0136	0.0206

training is used here to compare the performance of the SNN and MLP with differing numbers of neurons in the hidden layer. Tables 3 and 4 show the average %-error, as calculated in (11), during online forced training data taken from G7 for several different numbers of hidden neurons in both the SNN and the MLP. Only G7 is shown in these tables to save space; the trends on G10 are similar.

$$\%error = \left| \frac{actual - estimated}{actual} \right| \times 100\%. \tag{11}$$

Because the initial weights, before training, are randomly determined, the mean training and testing errors are the average taken over 30 trial runs, and the standard deviations for training and testing are shown, as well. Table 3 compares these errors for terminal voltage deviation prediction for generator G7 between the results from the SNN with specialized centers determined solely from the data taken from generator G7 itself, the SNN using generalized centers taken from data collected from both generators, and the MLP. Table 4 compares these same values for the speed deviation. In all cases, %-error is calculated as shown in (11).

In all cases, more neurons lead to lower average errors and more consistency of results during online training. However, the more hidden neurons are in the network, the longer it takes to execute each epoch. The results shown in Figs. 6–13 and 15–18 used 20 neurons in the hidden layer, as more than that caused dramatic enough slowdown as to interfere with real-time prediction. The limiting factor when the neuroidentifier is implemented in hardware for real-world use will be the real time requirement. Twenty neurons are found to balance the speed of calculation with good performance. A quick glance at Tables 3 and 4, comparing the MLP's consistency and precision to those of the SNN, reveals that the MLP has a higher average error than does either SNN, though its consistency is greater (as shown by the standard deviations). The units of all values in these tables are given in per-unit (p.u.).

Forced training is performed with data points sampled over 60 s at a sample rate of 50 Hz. The results of forced training of the MLP on generator G7 is shown in Fig. 6, and on the SNN in Fig. 7. Figs. 8 and 9 show the results of forced training on both kinds of neuroidentifier for generator G10. With both generators, the SNN demonstrates much more faithful identification than does