Reservoir-computing-based, Biologically-inspired Artificial Neural Network for Modeling of a Single Machine Infinite Bus Power System

Jing Dai[1], Ganesh Kumar Venayagamoorthy[2], Ronald G. Harley[1], Steve M. Potter[3, 4]

[1]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA.
[2]Holcombe Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29634, USA
[3]Coulter Department of Biomedical Engineering at Georgia Institute of Technology and Emory University School of Medicine,
[4]Laboratory for Neuroengineering, Georgia Institute of Technology,Atlanta, GA, 30332, USA.
Emails: jingdai@gatech.edu, rharley@ece.gatech.edu, gkumar@ieee.org, & steve.potter@bme.gatech.edu

Abstract—**Inspired by living neuron networks (LNNs) in the brain, artificial neural networks (ANNs) have been broadly used in various applications as a computational intelligence tool. However, due to many fundamental differences between ANNs and LNNs, despite the mature training mechanisms for ANNs, it is often challenging to use LNNs as a computational intelligence tool. To bridge the gap between ANNs and LNNs, a novel type of artificial neural network, i.e. biologically-inspired artificial neural network (BIANN) is proposed in this paper. The BIANN, which is based on spiking neuron models of LNNs, processes information in a more "brain-like" fashion than conventional ANNs. A reservoir-computing-based training approach is also proposed for BIANNs to serve as a novel modeling and control tool for practical applications. The feasibility of the proposed BIANN is illustrated for the prediction of a synchronous generator's speed and terminal voltage signals in a single machine infinite bus electric power system setup. The proposed BIANN model is able to provide an accurate prediction for online monitoring of a generator.**

Keywords-spiking neural network; rate coding; biologically-inspired artificial neural network; reservoir computing; power system

I. INTRODUCTION

Artificial Neural Networks (ANNs) are well known for their function approximation capabilities and have been broadly used in nonlinear system identification and control problems. Various ANN architectures have been proposed [1-11] and some mature training mechanisms have been developed for these ANNs. However, most of them are only modeled loosely on the biological processes in a group of living neurons. These ANNs are usually based on computational units that apply an "activation function" with a continuous set of possible output values to a weighted sum (or polynomial) of the inputs. Common activation functions are the sigmoid function the linear saturated function. Some fundamental features such as neuronal coding, format of signal transmitted and synaptic plasticity in the Living Neural Network (LNN) were not taken into account when ANNs were designed. A number of spiking neuron models have been proposed for modeling the behavior of living neurons [12-19]. These spiking neuron models are designed to represent the various behaviors of a single neuron or a group of neurons with different computational features. These models resemble the neuronal system in the brain more closely than ANNs; therefore, they are broadly believed to have stronger computational potential than ANNs. Due to some fundamental differences between ANNs and spiking neuron networks, the well-developed training mechanisms for ANNs cannot be directly applied to spiking neuron networks. Therefore, the application of spiking neuron networks as a computational intelligence tool has been limited.

To bridge the gap between ANNs and LNNs, a new type of ANN, i.e. the Biologically-Inspired Artificial Neural Network (BIANN) is proposed in this paper. A BIANN processes information in a more "brain-like" fashion than the traditional ANN. The mean firing rate (MFR) coding and decoding method and the Izhikevich model [20] for spiking neurons are used in the BIANN architecture. An online training algorithm of the BIANN based on reservoir computing is proposed and validated. The proposed BIANN is illustrated in the online identification of a single machine infinite bus (SMIB) power system, to predict the five-step-ahead response of the system and to validate the feasibility of the online training algorithm of the BIANN.

In this paper, the structure of the BIANN is introduced in Section II. The background of the SMIB system is introduced in Section III. The proposed BIANN architecture and its training approach for modeling the SMIB system are demonstrated step by step in section IV. The results of using the BIANN for online modeling purposes are described in section V. It is shown that the BIANN is capable of providing accurate prediction of a generator's speed and terminal voltage signals in such a SMIB system for online monitoring and control of power systems.

II. BIOLOGICALLY-INSPIRED ARTIFICIAL NEURAL NETWORK

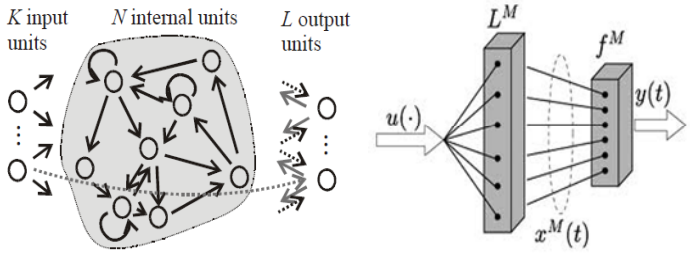
A. The Concept of Reservoir Computing

Two types of "reservoir computing", the Echo State Network (ESN) and the Liquid State Machine (LSM), were proposed by Jaeger [21]and Maass [22], independently in 2002. Despite different terminologies, they share a common

This work is supported by the National Science Foundation (NSF) of the United States, under grant # EFRI 0836017. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

978-1-4673-1490-9/12/$31.00 ©2012 IEEE

concept. The structure of the ESN is shown in Fig. 1(a). The dynamic reservoir in the middle layer consists of n units, each with a nonlinear activation function. The shaded arrows are optional connections and only the dotted arrows which represent synaptic weights are trained. The LSM structure is shown in Fig. 1(b). A function of time (time series) $u(\cdot)$ is injected as input into the liquid filter L^M, creating at time t the liquid state $x^M(t)$, which is transformed by a readout map f^M to generate an output $y(t)$.



(a) The ESN (b) The LSM

Figure 1. Two types of reservoir computing: the ESN[21] and the LSM[22].

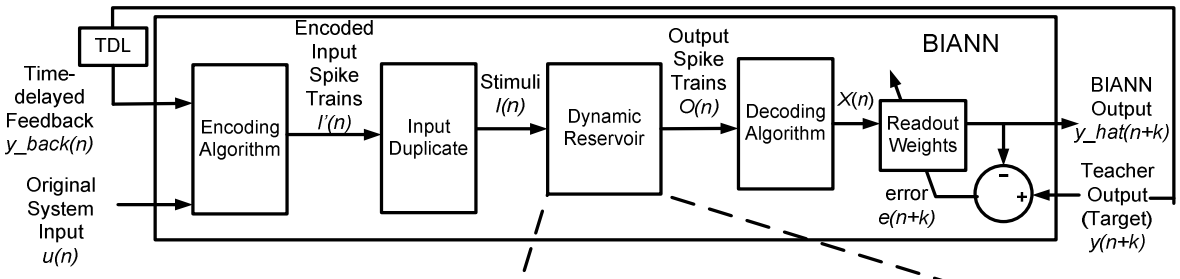
Typically, an input signal is fed into a large, random dynamical system (i.e. the "dynamic reservoir" or the "liquid") and the dynamics of the reservoir map the input to a higher dimension. Then a simple readout mechanism is trained to read the state of the reservoir and map it to the desired output. The main benefit is that the training is performed only at the readout stage, which requires much lower computational effort than conventional ANNs.

B. The Overall Structure of the BIANN

The overall structure of a BIANN is shown in Fig. 2. The original continuous input signal, for example, the terminal voltage of a generator in a power system, is first encoded into spike signals. To form a bigger stimulation to the dynamic reservoir, the encoded input spike signal is duplicated depending on the number of the input signals and the number of neurons in the dynamic reservoir. These encoded stimuli are then fed to a certain number of excitatory neurons in the spiking neuron network, which are randomly chosen as the input neurons. The dynamic reservoir consists of spiking neurons. After the calculation in the dynamic reservoir, the outputs of all the neurons in the dynamic reservoir are converted back into analog signals at each time step using a decoding method. Finally, the output of the BIANN is calculated based on the readout weights. The variables in the BIANN are demonstrated in Fig. 2 with their dimensions listed.

C. Neuronal Coding: Encoding and Decoding

Neurons in the brain communicate by short electrical pulses, the so-called action potentials or spikes. Since the goal of the BIANN research is to make it more biologically meaningful, all the neurons in the BIANN should be spiking neurons. To compute with spikes, proper encoding and decoding must first be developed. Encoding refers to transforming real-world signals to spike trains that can be used as stimuli to a neural network, while decoding refers to the processing of the spike trains coming out of the neural network and converting them back into meaningful real-world signals. A sequence of spikes may contain completely different information based on different coding schemes. A number of encoding methods, for example, rate coding, temporal coding and population coding have been proposed [23, 24] to convert real-world signals into spike trains that can



Signal	Dimension
u(n)	Ax1
y_back(n)	Bx1
I'(n)	(A+B)Mx1
I(n)	C(A+B)Mx1
O(n)	Nx1
X(n)	Nx1
y_hat(n+k)	Bx1
y(n+k)	Bx1
e(n+k)	Bx1

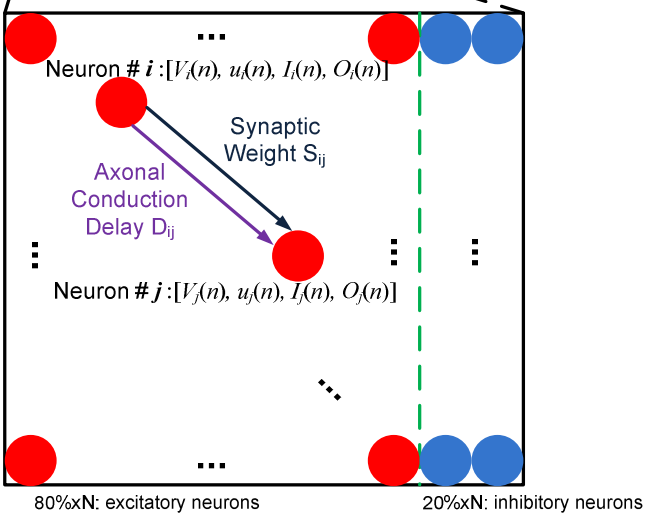


Figure 2. Overall Structure of the BIANN.

be recognized and processed by spiking neurons.

The encoding and decoding methods used in this paper are rate-coding-based methods, which is a type of temporal averaging method. The details of the encoding and decoding methods are presented in Section V.

D. The Izhikevich Model with Spike-timing-dependent Plasiticity (STDP) in the Dynamic Reservoir

Among all spiking models of living neurons, the Izhikevich model [20] has the capability of representing various different behaviors of a living neuron, without involving complex computational effort. In the Izhikevich model, the network consists of two types of neurons: namely excitatory neurons and inhibitory neurons. The ratio of excitatory to inhibitory cells is typically 4 to 1, as in the mammalian neocortex. The probability of a connection is typically 10% between any two neurons. Each excitatory neuron is randomly connected to 10% of the neurons in the entire neural network, and each inhibitory neuron is connected to excitatory neurons only. Each neuron in the network is described by the following simple spiking model:

$$\begin{cases} \frac{dV}{dt} = 0.04V^2 + 5V + 140 - u + I \\ \frac{du}{dt} = a(bV - u) \end{cases} \quad (1)$$

If $V \geq 30$ mV, then $\begin{cases} V = c \\ u = u + d \end{cases}$

The variable V represents the membrane potential of the neuron, and u represents a membrane recovery variable, which accounts for the activation of K^+ ionic currents and inactivation of Na^+ ionic currents, and it provides negative feedback to V. After the spike reaches its apex at +30 mV, the membrane voltage and the recovery variable are reset. The parameters are set differently for excitatory neurons and inhibitory neurons. Corresponding to cortical neurons, the parameters for excitatory neurons are set as [a, b, c, d] = [0.02, 0.2, -65, 8]; while for inhibitory neurons, [a, b, c, d] = [0.1, 0.2, -65, 2]. For the design of the BIANN, all these parameters are randomly altered in a ±10% range to generate a greater variety of neuron behaviors [19].

Assuming a time step of 1 ms, (1) can be rewritten in the following iterative form:

$$\begin{cases} V(n+1) = 0.04V^2(n) + 6V(n) + 140 - u(n) + I(n) \\ u(n+1) = u(n) + a[bV(n) - u(n)] \end{cases} \quad (2)$$

Each connection between two neurons (e.g. connection from neuron i to neuron j) has two properties: the synaptic weight S_{ij} and the conduction delay D_{ij}. The synaptic connections in the reservoir are modified according to the spike-timing-dependent plasticity (STDP) rule [25]. If a spike from an excitatory presynaptic neuron arrives at a postsynaptic neuron prior to its firing (possibly making the postsynaptic neuron fire), then the synapse is potentiated (strengthened). In contrast, if the spike arrives right after the postsynaptic neuron has fired, then the synapse is depressed (weakened) [26]. This STDP rule is illustrated in Fig.3. Define Δt_{ij} (n) as the time difference between the firing of postsynaptic neuron (neuron j) and the arrival of excitatory spikes from a presynaptic neuron (neuron i). Let ΔS_{ij} be the change of synaptic weight, then ΔS_{ij} can be calculated using (3):

$$\Delta S_{ij}(n) = \begin{cases} A_P * e^{-\frac{\Delta t_{ij}(n)}{\tau_P}} & if\ \Delta t_{ij}(n) \geq 0 \\ -A_N * e^{\frac{\Delta t_{ij}(n)}{\tau_N}} & if\ \Delta t_{ij}(n) < 0 \end{cases} \quad (3)$$

where A_P is the maximum magnitude of weight increase; A_N is the maximum magnitude of weight decrease; τ_P and τ_N are time constants. The recommended parameter setting is A_P = 0.1; A_N = 0.12; τ_P = τ_N =20 ms.

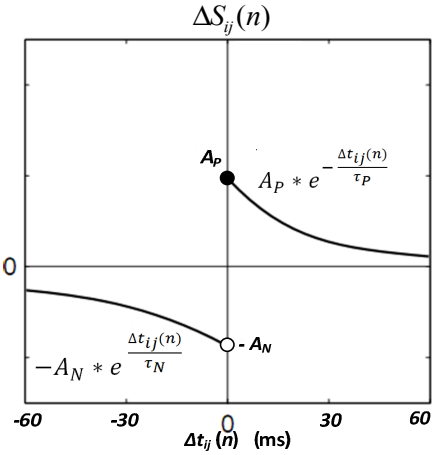


Figure 3. The STDP rule.

E. Obtain Meaningful Final Output via a Readout Layer

The readout layer of the BIANN is a set of trainable weights. Unlike the spontaneous, STDP-based weights update in the dynamic reservoir, these weights are adjusted using supervised learning. The detailed training algorithm of the readout weights will be given in Section IV. E. The final output of the BIANN is calculated using (4):

$$\hat{y}(n) = f[W^{out} \cdot x(n)] \quad (4)$$

where $\hat{y}(n)$ is the output of the BIANN at time step n, W^{out} is the output weight matrix and $x(n)$ is the decoded analog signal from all the neurons in the spiking neuron network, including the input neurons, as shown in Fig. 2. The function f can be a linear function, a sigmoid function, etc. In this paper, a linear function is used. The large, yet sparsely connected Izhikevich-Model-based spiking neural network in the middle is treated as a dynamic reservoir, or the liquid. According to both ESN and LSM theories, the input to the spiking network evokes rich dynamics in the dynamic reservoir or the liquid. When a target, i.e. the teacher output, is available, supervised learning can always be achieved by adjusting the weights in the readout layer.

III. Online Modeling of the Single Machine Infinite Bus System using the BIANN

The BIANN framework and the reservoir-based training algorithm proposed in this paper are tested in a prototype SMIB power system application of Fig. 4. A BIANN is trained to provide one-step-ahead and five-step-ahead predictions of the system output, which are the rotor speed and terminal voltage of the generator.

A. The SMIB System

The infinite bus in Fig. 4 is represented as an ideal voltage source, and the network is represented by algebraic equations, since ignoring the network dynamics is common practice in transient stability studies [27]; therefore, the only dynamics in the system are those of the generator, its automatic voltage regulator (AVR), and its turbine and governor.

Generator rotor speed and terminal voltage are critical quantities to monitor and control both real power and reactive power transmitted from the generator. The time-ahead predictions of these quantities are required by the controllers of rotor speed and terminal voltage which indirectly affect the real and reactive power.

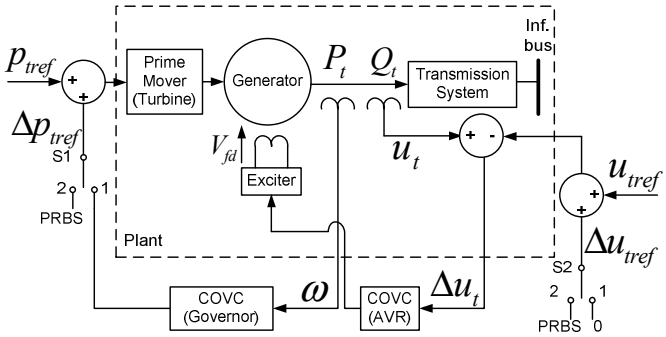


Figure 4. The Single Machine Infinite System.

B. Online Training and Testing Scheme of the BIANN Model for the SMIB System

The goal of the BIANN model is to learn the dynamics of the SMIB system using only input-output data collected a-priori to produce predictions of the terminal voltage and rotor speed. The online ***k***-step-ahead prediction scheme is shown in Fig. 5. The SMIB system dynamics are described by (5):

$$y(n+k) = f(u(n), y(n)) \quad (5)$$

where $(n) = [u_{t_{ref}}(n) + \Delta u_{t_{ref}}(n),\ p_{t_{ref}}(n) + \Delta\, p_{t_{ref}}(n)]$ and $y(n) = [\omega(n),\ u_t(n)]$. ω and u_t are the rotor speed and terminal voltage of the generator respectively. $u_{t_{ref}}$ and $p_{t_{ref}}$ are the terminal voltage and active output power references to the AVR and turbine governor, respectively.

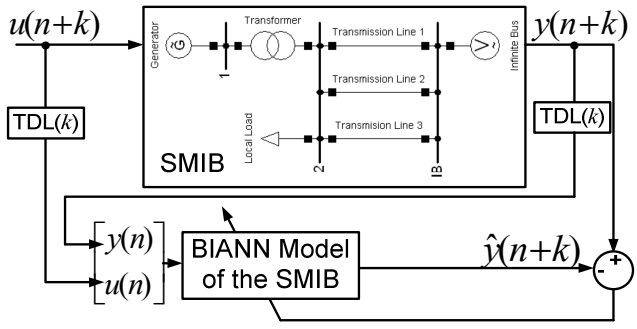


Figure 5. Online generator responses prediction scheme.

For this application, each step is 10 ms. At each step, the BIANN model is updated based on the previous inputs and outputs of the system and provides k-step-ahead prediction of the system outputs.

IV. Implementation of the BIANN Model for the SMIB System

The details of encoding, decoding, online training and testing algorithms of the BIANN model for the SMIB system are given in this section. The steady state values of u_{tref} and p_{tref} are 1.025 and 0.714 p.u. respectively.

A. Training Data

The SMIB system is simulated using PSCAD, which is a commercial power system simulation software package. Pseudo-Random Binary Signals (PRBS) Δu_{tref} and Δp_{tref} are added to the terminal voltage and the active power set point respectively, so that $u(n+k)$ at the input to Fig. 5 is equal to $[(1.025+\Delta u_{tref}),(0.714+\Delta p_{tref})]$.

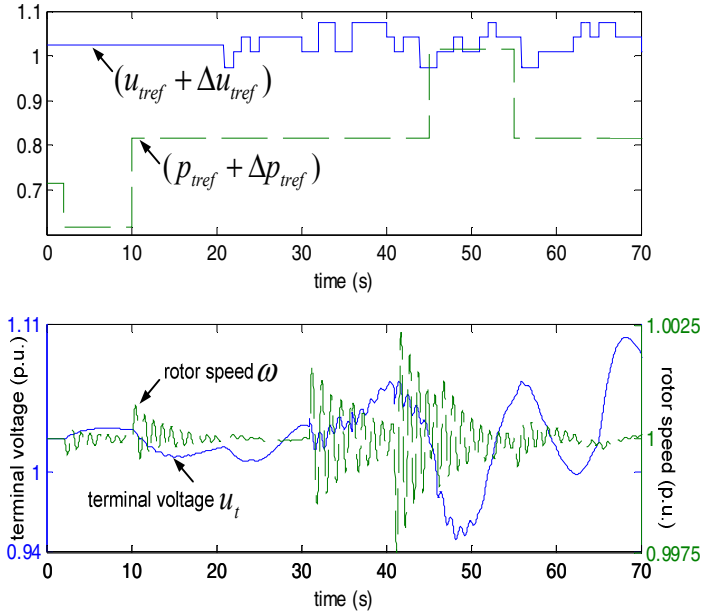


Figure 6. Training data: input with PRBS and output of the SMIB system.

This input is plotted in the top part of Fig. 6. The PRBS excites the dynamics of the SMIB system. The rotor speed ω and terminal voltage u_t of the generator exhibit small oscillations around their steady state values, as shown in the bottom part of Fig. 6. The input and output data of the SMIB block in Fig.5 are then sampled from Fig. 6 as the training data set of the BIANN. The sampling frequency of the training data is 100 Hz.

B. Encoding Process

At each time step n, the sampled input vector to the BIANN model of Fig. 5 has four variables, as shown in (6):

$$input_{1-4}(n) = [u(n), y(n)] =$$

$$[u_{t_{ref}}(n) + \Delta u_{t_{ref}}(n),\ p_{t_{ref}}(n) + \Delta\, p_{t_{ref}}(n), \omega(n),\ u_t(n)]\ , \quad (6)$$

The goal of the encoding process is to convert these continuous signals into spike trains, which is the form required to stimulate the spiking neurons in the BIANN. Each of the four variables is encoded into M parallel spike trains (M=10 in this paper), hence after the encoding process, the four continuous signals of Fig. 6 become forty spike sequences.

The input data are transformed as follows:

1) Scale each orginal signal to the range of [0.1 0.9] using (7);

$$input_{scaled}(n) = \frac{input(n) - \min(input(n))}{\max(input(n)) - \min(input(n))} \times 0.8 + 0.1 \quad (7)$$

2) At time step *n*, calculate the mean firing rate of all the *M* spike trains over the period (*n*-*T*+*1*) to (*n*-1) using (8);

$$past(n) = \frac{1}{T \cdot M} \sum_{i=1}^{M} \sum_{\tau=n-T+1}^{n-1} Fire_i(\tau) \quad (8)$$

Here, *T* is the time window for calculating the mean firing rate (*T*=20 ms) and $Fire_i(\tau)$ is the spiking signals.

3) Calculate the difference between *past*(*n*) and $input_{scaled}(n)$ using (9);

$$need(n) = T \cdot M \cdot [input_{scaled}(n) - past(n)] \quad (9)$$

4) At time step *n*, calculate how many spikes in total still need to be generated to form $input_{scaled}(n)$ using (10);

$$\sum_{i=1}^{M} Fire_i(t) = \max(\min(round[need(n)], M), 0) \quad (10)$$

The result from this step is an integer in the range of [0, *M*].

5) Assign the spikes that still need to be generated at time step *n* randomly to the M converting units.

An illustration of the encoding method is shown in Fig. 7. At each time step *n*, the value of the scaled signal $\text{input}_{\text{scaled}}(\text{n})$ is considered as the mean firing rate of all the *M* spike trains over the time period [(*n*-*T*+*1*), *n*]. Each original continuous signal is encoded into *M* spatiotemporal spike trains, which are then sent into M excitatory neurons in the BIANN as input, i.e. "*I*(*n*)" in (2).

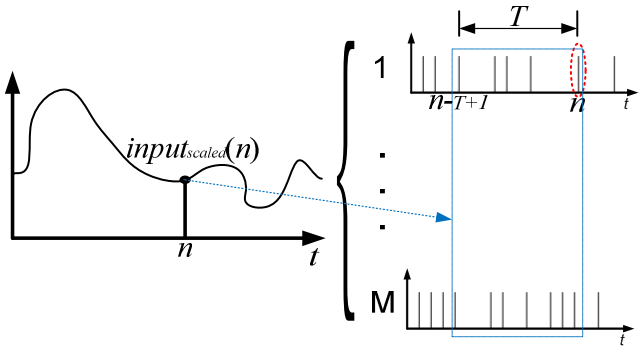


Figure 7. Illustration of the encoding method.

This mean-firing-rate-based encoding method is reversible, in other words, the original signal can be accurately recovered from the spikes using a reverse algorithm. As an example, the encoding result of the third element in the input vector, which is the rotor speed of the generator $\omega(n)$, is shown in Fig. 8, and Fig. 9 shows that the original signal can be recovered from these spikes.

C. Maturing of the Reservoir

Due to the neuronal latency plasticity of spiking neural networks, it has been observed that although initialized randomly, using STDP rules, the synaptic weights in the spiking neural network will eventually distribute in two ranges: 0-10% and 90%-100% of the weight limits, given a certain input. In this paper, the network is run for a certain amount of time until more than 40% of all the synaptic weights fall into these two ranges respectively, as shown in Fig.10. When the spiking neural network is matured, it is used as the initial reservoir network for training.

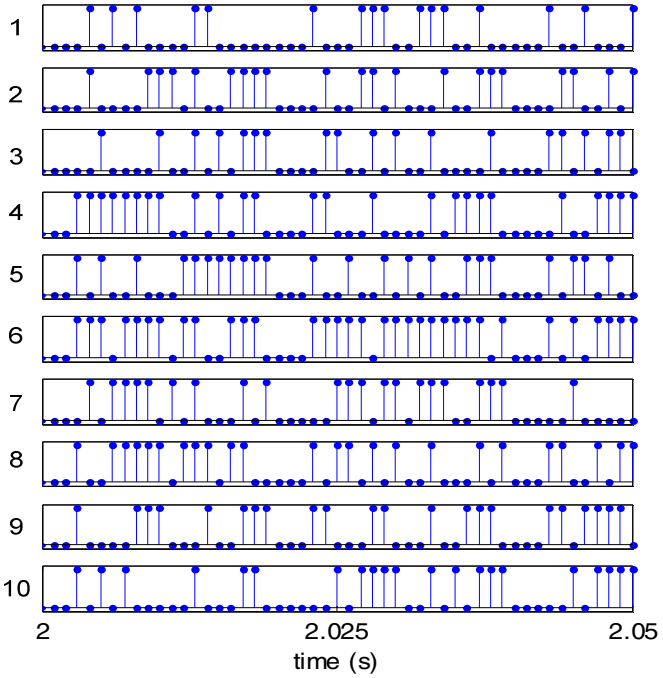


Figure 8. One of the encoded inputs: the rotor speed ω.

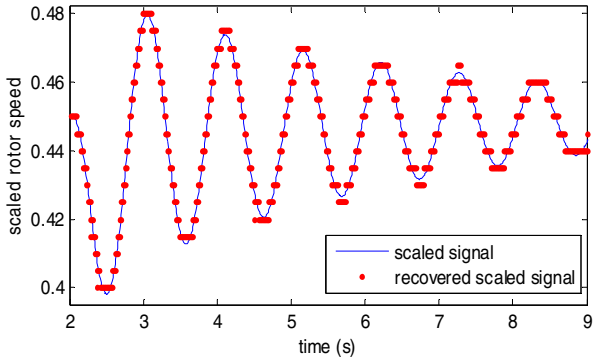


Figure 9. Reversibility of the encoding algorithm to recover signal from spike data in Fig. 8.

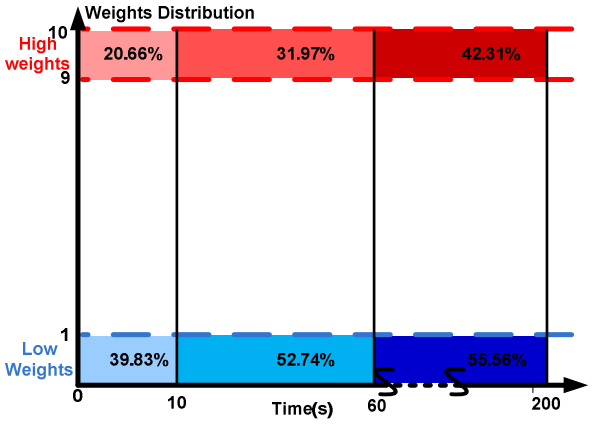


Figure 10. The Izhikevich Spiking Neuron Model.

D. Decoding Method

The computation in the dynamic reservoir strictly follows the Izhikevich spiking neuron model described in Section II. D. When a neuron fires at time step n, the output of this neuron at this moment is marked as $Fire_i(n)$=1, otherwise, $Fire_i(n)$=0.

The decoding method is the well known mean firing rate method [28]. At time step n, the output of the i^{th} neuron in the BIANN is decoded from spiking signals $Fire_i$ to continuous signals using (11):

$$output_i(n) = \frac{1}{T}\sum_{\tau=n-T+1}^{n} Fire_i(\tau) \quad (11)$$

Again T=20 ms is the averaging window. $output_i(n)$ is not the final output of the BIANN, since an additional readout layer with trainable weights is still needed to extract useful information from the dynamic reservoir.

E. Training Algorithm of the Readout Weights

Based on the algorithm proposed in [21], the training and testing of the BIANN model for the SMIB system consists of the following steps:

1) Use the matured network obtained from Section IV.C as the initial network;

2) Start feeding the encoded input data into the BIANN, and let the input evoke rich dynamics in the dynamic reservoir. The internal weights in the dynamic reservoir are adjusted following the STDP rule described in Section II. D. This is different from the original ESN and LSM theories. In both theories, the internal weight matrix of the dynamic reservoir or the liquid is fixed as soon as it has been generated. It is believed that the input history is memorized and stored in the states (values) of each internal neuron in the dynamic reservoir. However, in the case of the BIANN proposed in this paper, the internal dynamic reservior is always changing according to the STDP rule. The "memory " of the BIANN is affected by two factors: the states of each neuron and the weights in both the dynamic reservoir and the readout layer.

3) Collect 10 seconds of decoded output from the dynamic reservoir and save it in a matrix called $X(n)$ (see Fig. 2). $X(n) = [output(1), output(2), \dots, output(n)]$ is a 1000-by-10000-dimensional matrix since all 1000 neurons (including 40 input neurons) in the dynamic reservior are connected to the final output of the BIANN via the readout layer and the Izhikevich model runs at a frequency of 1000 Hz;

4) k steps later from the current moment n, when the actual output $y(n+k)$ from the SMIB system becomes available, also put 10 seconds of y in a matrix called $Teacher(n)$. $Teacher(n) = [y(1+k), y(2+k), \dots, y(n+k)]$ is a 2-by-10000-dimensional matrix since there are two variables that need to be predicted;

5) Calculate the readout weight matrix $W^{out}(n)$ using a psuedoinverse algorithm as in (12);

$$W^{out}(n) = Teacher(n) \cdot psuedoinverse(X(n)) \quad (12)$$

In this paper, $W^{out}(n)$ is 2-by-1000-dimensional.

The training of the BIANN is realized by updating the readout weights given a teacher signal. Meanwhile, the weights in the dynamic reservoir are spontaneously updated based on the STDP rules described in section II.D. The two types of updates are of different nature. The STDP update simulates the biological process in the brain that adjusts the strength of connections between neurons. The readout weight updates belong to the category of supervised learning. The goal of the training of the readout weights is to build the relation between rich patterns produced by the dynamic reservoir and the teacher outputs.

F. Prediction with the BIANN

After the five steps described in the section above, the BIANN is trained and ready for use. The weights in the readout layer are now fixed, but the weights in the dynamic reservoir are still updating according to the STDP rule.

To capture any possible system change for modeling purposes, a moving window technique is used during the training and prediction of the BIANN. As shown in Fig.11, the network is first trained with 10 seconds of data and then used for prediction for the following 5 seconds. The window shifts 5 seconds to the right after a complete cycle of training and prediction has been completed.

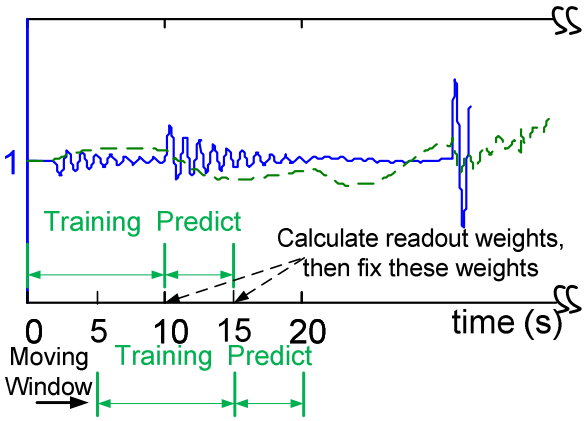


Figure 11. Moving window training and prediction scheme.

The final output of the BIANN, i.e. the k-step-ahead predicted rotor speed and terminal voltage of the generator are calculated using (13):

$$\hat{y}(n+k) = [\hat{\omega}(n+k), \hat{u}_t(n+k)] = W^{out}(n) \cdot output_i(n) \quad (13)$$

V. Results and Discussion

A. One-step-ahead Prediction Result

A BIANN model is first trained to predict the generator speed and terminal voltage under the operating conditions described in Section IV.A. The BIANN model is trained using the moving-window approach as described in Section IV.E. The online prediction results of the BIANN model are shown in Fig. 12. Fig. 12(a) shows the prediction performance over a 40 second time span while Fig. 12(b) shows the prediction results over a 5 second time span. These results show that the BIANN model sufficiently accurately predicts both rotor speed and generator terminal voltage.

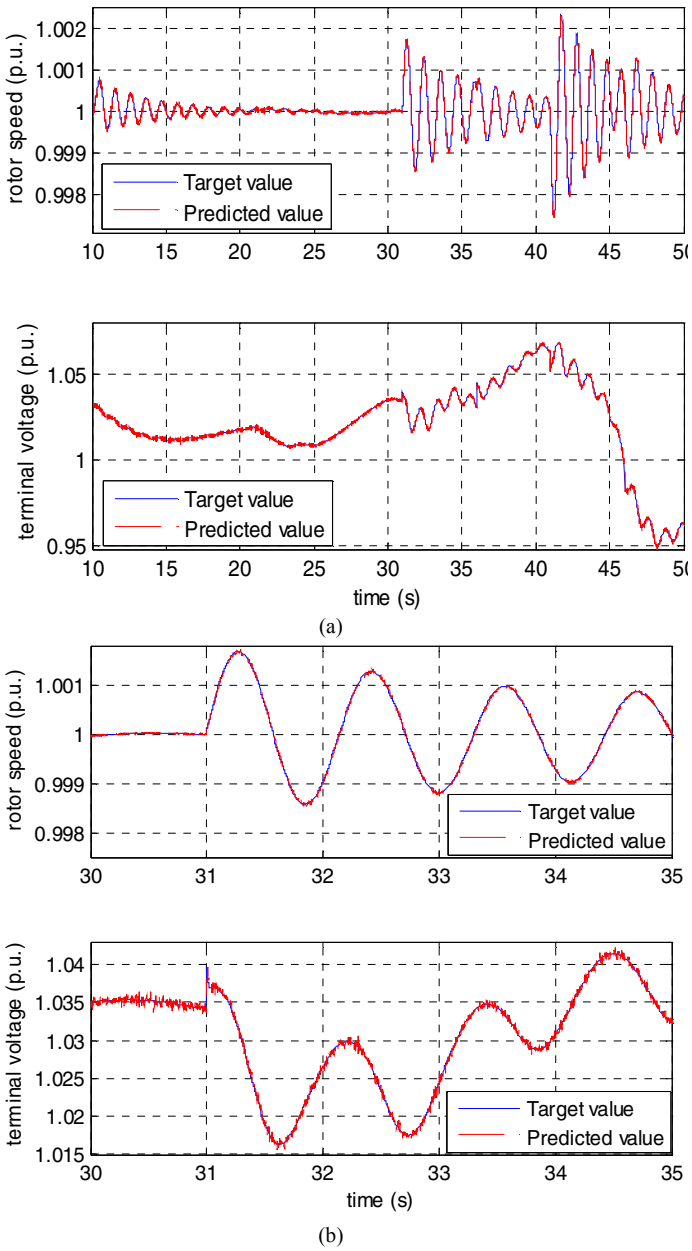


Figure 12. One-step-ahead prediction of generator speed and terminal voltage using BIANN model.

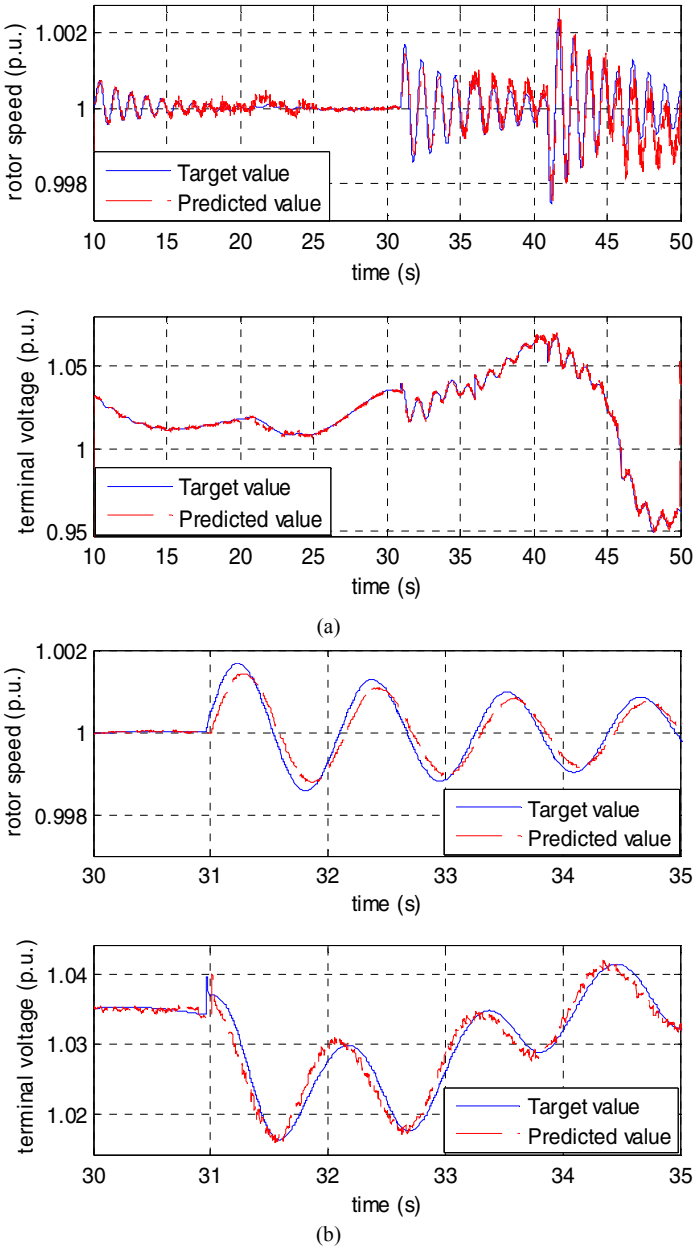


Figure 13. Five-step-ahead prediction of generator speed and terminal voltage using BIANN model.

B. Five-step-ahead Prediction Result

As discussed in Section III.A, it is not only important to accurately predict the generator rotor speed and the terminal voltage, but also critical to predict these quantities in a multi-step-ahead fashion so that these predictions can more effectively assist the monitoring and operation of the power system. Therefore, a similar exercise is conducted for the BIANN model to provide five-step-ahead prediction of the generator rotor speed and terminal voltage. The five-step-ahead prediction performance of the BIANN is illustrated in Fig. 13. The comparison between Fig. 12 and Fig. 13 shows that the five-step-ahead predictions are less accurate than the one-step-ahead predictions, which is expected since less relevant information is used for five-step-ahead prediction.

C. Prediction Accuracy Versus Number of Steps Ahead

The mean absolute relative errors (MAREs) between 10 and 50 seconds of the generator rotor speed and terminal voltage predictions, using BIANN models in five-step-ahead fashion, are defined by (14) and compared in Fig. 14.

$$MARE = \frac{1}{40\times100}\sum_{n=1000}^{5000}\left|\frac{y(n)-\hat{y}(n)}{y(n)}\right| \times 100\% \qquad (14)$$

As expected the prediction error increases with more-step-ahead predictions. However, in practical applications, the selection of the prediction steps is an important tradeoff between prediction accuracy and the time allowed by the prediction for predictive operation and control in the power system. This tradeoff selection is even more important for wide-area monitoring and control of large-scale power systems due to the communication and operational delays.

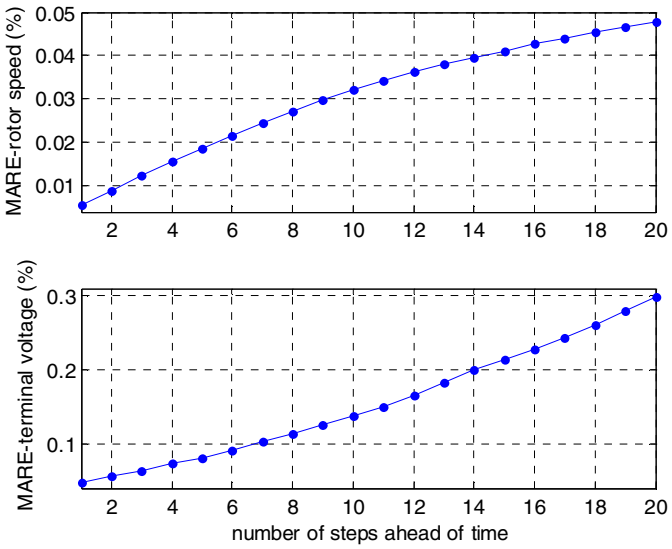


Figure 14. MAREs of generator speed and terminal voltage predictions using BIANN models versus prediction steps.

VI. CONSLUSIONS

Artificial neural networks have been used in various applications as a computational intelligence tool. Although inspired by living neuronal network in the brain, ANNs have many fundamental differences compared to LNNs, such as information transmission format, neuronal plasticity, etc. Due to these fundamental differences, the training mechanisms for ANNs cannot be extended to LNNs. As a result, it has been challenging to use LNNs as computational intelligence tools.

LNNs and their mathematical models, e.g., spiking neuron models, are broadly believed to have huge computational capability. To bridge the gap between ANNs and LNNs, a novel type of artificial neural network, i.e. biologically-inspired artificial neural network (BIANN) is proposed in this paper. The BIANN, which is based on the spiking neuron models of LNNs, processes information in a more "brain-like" fashion than conventional ANNs. A reservoir-computing-based training approach is also proposed for the BIANN to serve as a novel modeling and control tool for practical applications.

The generator rotor speed and terminal voltage are critical quantities for monitoring and operating power systems. The BIANN provides one-step-ahead and five-step-ahead predictions of these quantities for online monitoring of the generator in a single machine infinite bus power system. The prediction accuracy degrades as steps-ahead increases.

REFERENCES

[1] J. E. Dayhoff, *Neural Network Architecture: an Introduction*. New York, NY: Van Nostrand Reinhold Co., 1990, ISBN: 0-442-20744-1.

[2] M. W. Gardner and S. R. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric Environment,* vol. 32, pp. 2627-2636, 1998.

[3] J. Park, "Universal Approximation Using Radial-Basis-Function Networks," *Neural Computation,* vol. 3, pp. 246-257, 1991.

[4] D. F. Specht, "A General Regression Neural Network," *IEEE Trans. on Neural Networks,* vol. 2, pp. 568-576, 1991.

[5] T. Kohonen, *Learning vector quantization. In: M.A. Arbib, editor, The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995, ISBN: 0-262--01148-4.

[6] D. F. Specht, "Probabilistic Neural Networks," *Neural Networks,* vol. 3, pp. 109-118, 1990.

[7] J. L. Elman, "Finding Structure in Time," *Cognitive Science,* vol. 14, pp. 179-211, 1990.

[8] H. Jaeger, "The "Echo State" Approach to Analysing and Training Recurrent Neural Networks," GMD Report 148 - German National Research Institute for Computer Science2001.

[9] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the USA,* vol. 79, pp. 2554-2558, 1982.

[10] G. A. Carpenter and S. Grossberg, *Adaptive Resonance Theory, In Michael A. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks (2nd Ed.)*. Cambridge, MA: MIT Press, 2003, ISBN: 0-262-01197-2.

[11] K. Gopalsamy and X.-Z. He, "Delay-independent stability in bidirectional associative memory networks," *Neural Networks, IEEE Transactions on,* vol. 5, pp. 998-1002, 1994.

[12] C. Koch and I. Segev, *Methods in Neuronal Modeling (2nd Ed.)*. Cambridge, MA: Massachusetts Institute of Technology, 1998, ISBN: 0-262-11231-0.

[13] E. M. Izhikevich, "Which Model to Use for Cortical Spiking Neurons?," *IEEE Transactions on Neural Networks,* vol. 15, pp. 1063-1070, 2004.

[14] G. D. Smith, *et al.*, "Fourier analysis of sinusoidally driven thalamocortical realy neurons and minimal integrate-and-fire-or-burst model," *Journal of Neurophysiol.,* vol. 83, pp. 588-610, 2000.

[15] E. M. Izhikevich, "Resonate-and-fire neurons," *Neural Networks,* vol. 14, pp. 883-894, 2001.

[16] P. E. Latham, *et al.*, "Intrinsic dynamics in neuronal networks. I. Theory," *Journal of Neurophysiol.,* vol. 83, pp. 808-827, 2000.

[17] R. FitzHugh, "Impulses and physiological states in models of nerve membrane," *Biophys. J.,* vol. 1, pp. 445-466, 1961.

[18] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and application to conduction and excitation in nerve," *J. Physiol.,* vol. 117, pp. 500–544, 1954.

[19] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Networks,* vol. 14, pp. 1569–1572, 2003.

[20] E. M. Izhikevich, "Polychronization: Computation with Spikes," *Neural Computation,* vol. 18, pp. 245-282, 2006.

[21] H. Jaeger, " Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach. GMD Report 159," German National Research Center for Information Technology, 2002.

[22] W. Maass, *et al.*, "Real-Time ComputingWithout Stable States:A New Framework for Neural Computation Based on Perturbations," *Neural Computation,* vol. 14, pp. 2531-2560, 2002.

[23] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single neurons, Populations, Plasticity*. New York: Cambridge University Press, 2002.

[24] C. Johnson, *et al.*, "A reversibility analysis of encoding methods for spiking neural networks," *Proceedings of the 2011 International Joint Conference on Neural Newtorks(IJCNN)*, San Jose, CA, July 31-Aug 5, 2011, pp. 1802-1809.

[25] N. Caporale and D. Yang, "Spike-Timing–Dependent Plasticity: A Hebbian Learning Rule," *Annual Review of Neuroscience,* vol. 31, pp. 25-46, 2008.

[26] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured Hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.,* vol. 18, pp. 10464-10472, 1998.

[27] P. Kundur, *Power system stability and control* New York: McGraw-Hill, 1994, ISBN:978-0070359581.

[28] W. Gerstner, "Neural codes: Firing rates and beyond," in *Proceedings of the National Academy of Sciences of the United States of America*, 1997, pp. 12740-12741.