Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12-17, 2007

System Identification for the Hodgkin-Huxley Model using Artificial Neural Networks

Manish Saggar, Tekin Meriçli, Sari Andoni, Risto Miikkulainen

Abstract— **A single biological neuron is able to perform complex computations that are highly nonlinear in nature, adaptive, and superior to the perceptron model. A neuron is essentially a nonlinear dynamical system. Its state depends on the interactions among its previous states, its intrinsic properties, and the synaptic input it receives. These factors are included in Hodgkin-Huxley (HH) model, which describes the ionic mechanisms involved in the generation of an action potential. This paper proposes training of an artificial neural network to identify and model the physiological properties of a biological neuron, and mimic its input-output mapping. An HH simulator was implemented to generate the training data. The proposed model was able to mimic and predict the dynamic behavior of the HH simulator under novel stimulation conditions; hence, it can be used to extract the dynamics (in vivo or in vitro) of a neuron without any prior knowledge of its physiology. Such a model can in turn be used as a tool for controlling a neuron in order to study its dynamics for further analysis.**

I. Introduction

In the past 50 years, much has been learned about the behavior of biological neurons. In particular, the model developed by Alan Hodgkin and Andrew Huxley [2] made it possible for the first time to understand how neurons spike. Ion channels with complex voltage-gated properties were brought together into a mathematical model that explained how action potentials are generated. This model is still the foundation for most models of biological neurons today, although it is computationally expensive to build and simulate. In particular, the model includes many properties that need to be set by making various assumptions about the physiology of a neuron.

The question therefore arises; would it be possible to construct a black box that would capture the input-output relationship of a neuron without worrying about its physiological composition? If so, it would be possible to do online modeling that can provide the necessary tools for capturing the dynamical state of a biological neuron, simulate its output for further analysis, and may provide a more powerful dynamic clamp and online control.

ANNs have been successfully used for system identification in nonlinear domains [6, 7], as well as controllers for nonlinear dynamic plants [6]. In systems theory, multi-layer networks represent static nonlinear maps, while recurrent networks represent nonlinear dynamic feedback systems. Multi-layer networks have indeed proven useful in pattern recognition tasks [8-10], while recurrent networks have been used extensively in learning time sequences and modeling associative memories [11-14]. Further, given unbounded number of hidden neurons, the feedforward ANNs can approximate the behavior of an arbitrary continuous or "otherwise reasonable" function within arbitrary accuracy ε on a compact domain (this result is called universal approximation theorem [20]). However, there is no such result for time series prediction. Moreover, to our knowledge, ANNs have not been used to model the temporal behavior of a single neuron, although it is a task for which they are well suited.

The main objective of this paper is to find out whether it is possible to learn the nonlinear dynamics of a neuron, using ANNs with restricted number of building blocks, without any prior knowledge about its structural properties. The resulting model can be used as a tool for physiologists to perform more accurate experiments with biological neurons; such as determining how they will respond to given inputs, making it possible to design interactions that bring about the desired behaviors in the neuron.

Section II reviews the background on modeling biological neurons, including the Hodgkin-Huxley model, as well as the NARX and LRN architectures used in the experiments. Section III presents the experiments and the corresponding results. Finally, Section IV discusses future applications of the model.

II. Background

This section gives a brief introduction to modeling the behavior of neurons, specifically the HH model. It also reviews prior work on identifying nonlinear dynamical systems with various ANN architectures.

A. Modeling a biological neuron

Conventional methods of modeling a biological neuron require knowledge of its spatial structure and biophysical

Manish Saggar is with the Department of Computer Science at the University of Texas at Austin, Austin TX 78712 USA (e-mail: mishu@cs.utexas.edu).

Tekin Meriçli is with the Department of Computer Science at the University of Texas at Austin, Austin TX 78712 USA (e-mail: tmericli@cs.utexas.edu).

Sari Andoni is with the Institute of Neuroscience at the University of Texas at Austin, Austin TX 78712 USA (e-mail: andoni@mail.utexas.edu).

Risto Miikkulainen is with the Department of Computer Science at the University of Texas at Austin, Austin TX 78712 USA (e-mail: risto@cs.utexas.edu).

properties of its membrane. One common approach is to divide its structure into small connected compartments each having similar electrical properties.

Compartmental modeling has been widely used in simulating neurons and is the basis of numerous modeling packages such as Neuron [18] and GENESIS [19]. Based on their levels of detail, various compartmental models have been proposed, the most common of which is the HH model [2].

The HH model simulates the biophysical properties of a neuron, including the ionic mechanism involved in generating an action potential. The initial experiments by Hodgkin and Huxley were conducted using the giant axon of a squid to explain the ionic mechanisms underlying the initiation and propagation of action potential [2]. The model has since played a seminal role in biophysics and neuronal modeling. It describes the kinetics of sodium (Na^+) and potassium (K^+) channels involved in generating the action potential. The model can be described by the following four nonlinear ordinary differential equations:

$$Cm\frac{\partial V}{\partial t} = -g_L(V - E_L) - g_{Na}m^3h(V - E_{Na}) - g_K n^4(V - E_K)$$

$$\frac{\partial m}{\partial t} = \alpha_m(V)(1-m) - \beta_m(V)m$$

$$\frac{\partial h}{\partial t} = \alpha_h(V)(1-h) - \beta_h(V)h$$

$$\frac{\partial n}{\partial t} = \alpha_n(V)(1-n) - \beta_n(V)n$$

where n, m, h describe the gating particles involved in the activation of K^+ channels and the activation and inactivation of Na^+ channels, respectively. The functions α and β are voltage-dependent rate functions for each gating particle, E_{ion} and g_{ion} are the equilibrium potential and conductance of each ion, respectively, L is the leak current, C is the capacitance of the membrane, and I_e is the input current injected through the recording electrode.

It is important to note that the above equations describe a nonlinear dynamical system that can exhibit complex behavior including periodic and chaotic firing as well as a variety of different bifurcations depending on its parameters and the input (I_e) [17].

B. System identification using neural networks

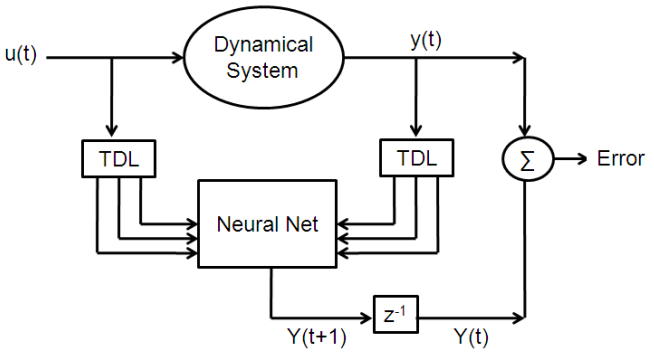
Two artificial neural network architectures are commonly used for system identification: the nonlinear autoregressive network with exogenous inputs (NARX) and the layer-recurrent network (LRN).

NARX [3, 4] is a dynamic network with feedback connections enclosing several layers of the network. The NARX model is based on the linear ARX model, which is commonly used in time-series modeling. The NARX model can be defined as:

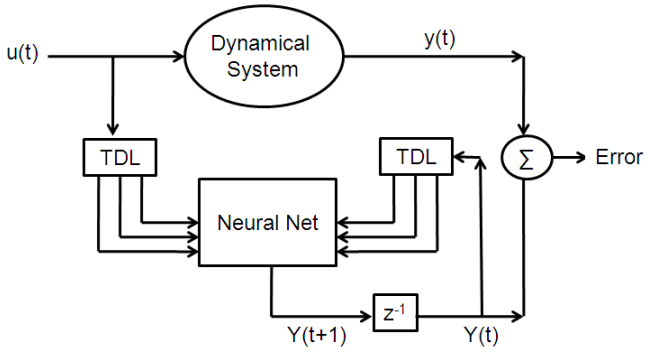
$$y(t) = f(y(t-1), y(t-2), ..., y(t-n_y), u(t-1), u(t-2), ..., u(t-n_u)),$$

where the next value of the output signal, *y(t),* is regressed on previous values of the output signal and previous values of the input signal, *u(t)* [3, 4]. Previous values of both input and output can be fed to the network using tapped-delay lines.

NARX has been used in several tasks including one-step time series prediction [3, 4], nonlinear filtering, and nonlinear dynamic system control [6]. Figure 1(a) and Figure 1(b) illustrate two possible configurations of a NARX network; series-parallel and parallel, respectively. In series-parallel configuration the input and the output of the system are both fed to the network. In contrast in the parallel configuration, only the input of the system is connected to the network and the output of the network is fed back to itself.



(a)



(b)

Figure-1. Two versions of the NARX network architecture. (a) Series-parallel architecture. The actual output (y) of the system to be modeled is used as input, instead of feeding back the network's estimates of the output (Y). (b) Parallel architecture. The output of the network (Y) is fed back, making the network a recurrent system. Such a network can predict the entire series; however, it is harder to train.

In series-parallel configuration the output of the system itself is available during training, which makes the input to the network more accurate. Also, the resulting network has feed-forward architecture; hence static back-propagation can be used for training. The disadvantage of the series-parallel configuration is that the network predicts only the next time step and needs to be transformed to a parallel configuration in order to predict the complete behavior of the system. Parallel configuration is usually preferred in system identification because once trained it can predict the behavior of the system for the entire period. However, the

parallel configuration also has a disadvantage; it requires recurrent back-propagation, which is a computationally expensive operation.

The second common system identification architecture is the Layer-Recurrent Network (LRN). The LRN is a simplified version of the Simple Recurrent Network (SRN) network [5]. It has a recurrent connection in each of its hidden neurons; an LRN with two hidden layers is illustrated in Figure 2. These recurrent connections store the values from the previous time step, which can be used to identify the input-output mapping for the future. An LRN commonly has one hidden layer with a feedback connection from the output of the hidden layer to its input. This recurrent connection allows LRNs to both detect and generate time-varying patterns. The experiments in this paper are based on a two-hidden-layers architecture, which was found to be more effective in practice. Because the recurrent connections only store a copy of the activation, they do not need to be trained. Therefore, regular back-propagation can be used to train the LRN.

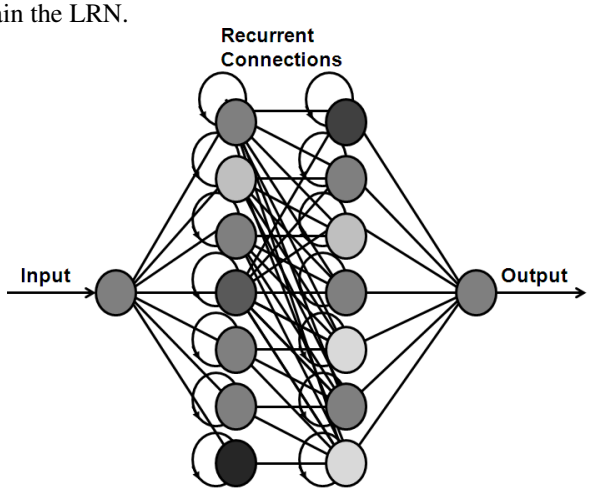


Figure-2. An LRN with two hidden layers. Each hidden layer has a recurrent connection that acts as a memory to store pattern information. The network can therefore be trained efficiently with regular back-propagation.

III. Experiments

A simulator was implemented based on the HH equations to test the hypothesis that ANNs can learn the behavior of a HH model. This simulator was used to generate training data for the neural network. Standard parameter values for HH equations used in our implementation are provided in Table1.

The four dynamic properties that a HH simulator should depict in the output, given a current step, are described as follows.

Threshold: The membrane potential is required to exceed a certain value to fire an action potential. This limiting value is called the threshold as shown in Figure 3.

***Periodic firing*:** Multiple spikes are fired by a neuron when its membrane potential is sustained at a voltage higher than the threshold for some duration. This phenomenon is called periodic firing, and is illustrated in Figure 4.

Parameter	Value
E_K	-77 mV
E_{Na}	+50 mV
E_{Leak}	-57.4 mV
g_{Leak}	3 µS
g_K	360 µS
g_{Na}	1200 µS
C_m	1 nanoFarad
Area	0.1 mm2

Table-1. Standard parameter values for the Hodgkin-Huxley Simulator.

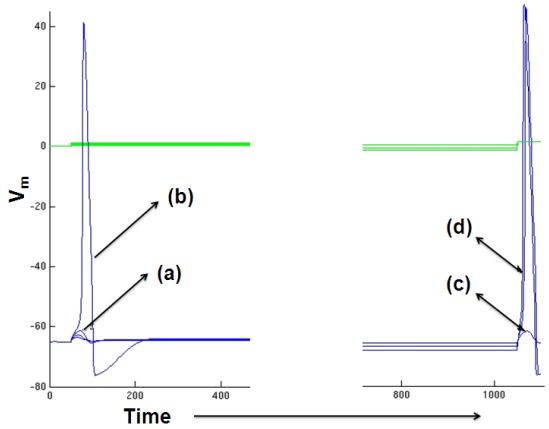


Figure-3. (a) HHS sub-threshold output given positive step current, followed by set of positive step currents ultimately triggering an action potential (b). (c) HHS output depicting sub-threshold response to a negative step current followed by an anode break action potential (d).

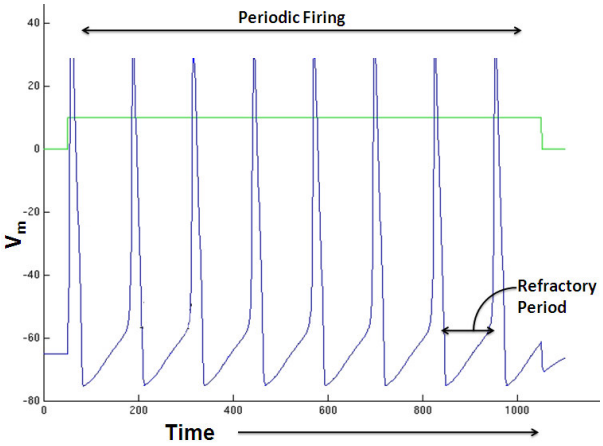


Figure-4. HHS output depicting the refractory period and periodic firing due to sustained input current.

***Refractory period*:** A brief period of time, typically one millisecond, following the action potential during which the nerve does not respond to a second stimulus. The refractory period is illustrated in Figure 4.

***Anode break*:** An anode break action potential occurs at the trailing edge of a negative current step input. It is a result

of the deactivation of the Na^+ channels during the negative input and their sudden activation at the end of the stimulus. Figure 3 illustrates this phenomenon.

The rest of this section explains implementation details for each of the three ANN architectures and the corresponding results. The parameter values for those architectures are provided in Table 2.

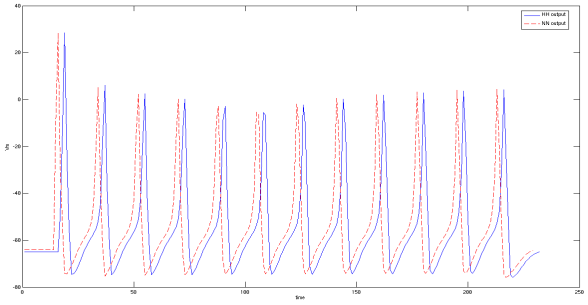
First we tested the series-parallel NARX network. One of the inputs for the network was a current step, *u(t)*, for a 232 *ms* duration. The delay for this input was set to 4. The second input was the output from the HH model, *v(t)*, for the same duration and delay. This network was trained separately on negative and positive current values. The output of the system after training is shown in Figure 5.

Parameter	**Value**		
	NARX		Layer Recurrent
	Series-Parallel	Parallel	
Input units	2	1	1
Hidden layers	1	1	2
Hidden neurons	40	10	{10,20}
Transfer function (hidden)	tan-sigmoid	tan-sigmoid	{tan-sigmoid, tan-sigmoid}
Initial weights	Random	Random	Random
Output neurons	1	1	1
Transfer function (output)	Pure-linear	Pure-linear	Pure-linear
Training algorithm	Backprop	Conjugate-Gradient	Conjugate-Gradient
Temporal delay input	4	10	N/A
Temporal delay output	4	10	N/A

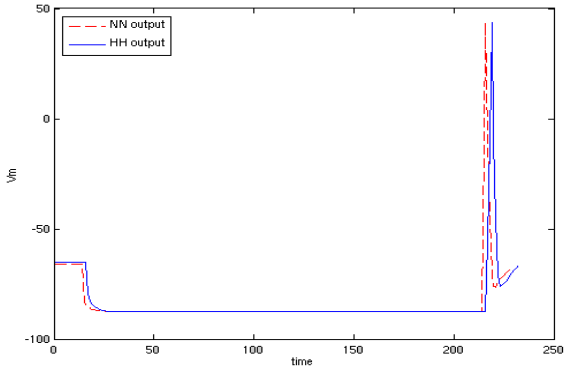
Table-2. Parameter values for the series-parallel and parallel NARX networks and LRN networks.

The series-parallel architecture was able to learn the properties of the HH membrane; i.e. the threshold, periodic firing, refractory period, and anode break for any given time step. This architecture has a great potential in control applications [3, 4]; hence, the trained network can be used to control the dynamical system (i.e. the HH model), and make it behave according to the user requirements. It can potentially be used as a tool for dynamic clamping.

However, the series-parallel NARX network was not able to extend these predictions beyond the current time step. The reason was that while the output of the HH model was always presented as one of the inputs to the network during training, such information was not available during testing. Furthermore, this network did not generalize to novel stimuli.



(a)



(b)

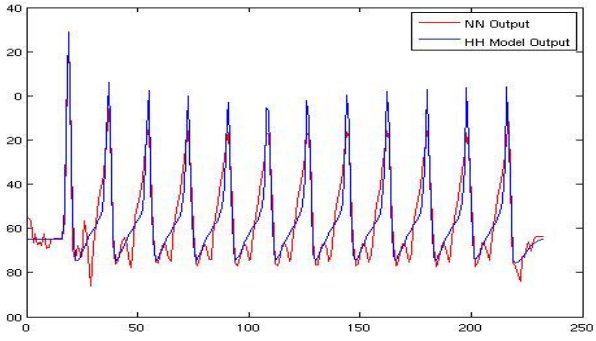
Figure-5. (a) Output of the series-parallel NARX network trained on positive step current. (b) Output of the series-parallel NARX network trained on negative step current.

The parallel version of NARX was tested in the second experiment. This network took as its input, a current step, *u(t)*, for 232 *ms*, as well as its own output with a temporal delay. In other words, this network is recurrent, which made it more powerful than the series-parallel architecture. Its output was the voltage signal, *v(t)*, for the same amount of time as the input. The temporal delays for both inputs were 10, as shown in Table 2.

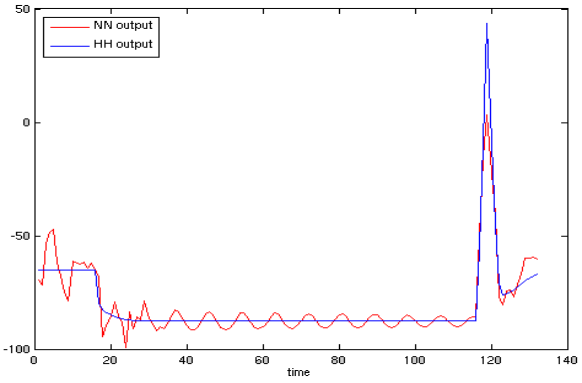
The network was trained on various current steps ranging from -12 to 28 *nA*, consistent with the operational limits for a neuron. Like the series-parallel architecture, the parallel network learned the dynamic properties of HH simulator and was also unable to generalize to novel stimuli (i.e. the testing data). However, unlike the series-parallel architecture, it was able to predict the complete series for training data. Figure 6(a) and Figure 6(b) show its output for positive and negative current steps, respectively.

The LRN was tested in the third experiment. The recurrence in its hidden layers provides explicit memory storage and makes it one of most powerful architectures for time series prediction. In the experiments, this feature also found to help generalize the behavior into novel stimuli.

The LRN was trained only on a single current step of +10 *nA* for a duration of 232 *ms*. It was able to learn all four characteristics of HH simulator from only this single training instance. Figure 7(a) shows the output of the trained network when tested with the training data itself. For comparison, Figure 7(b) shows the output of another LRN, which was specifically trained on -12 *nA* current step for a duration of 232 *ms*, and tested on the same data.



(a)



(b)

Figure-6. (a) Output of the parallel NARX network trained on positive step current. (b) Output of the parallel NARX network trained on negative step current.

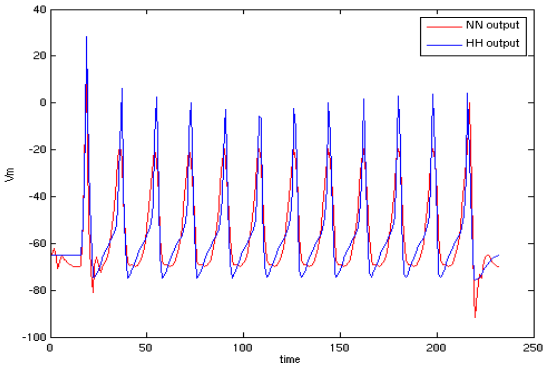
Interestingly, the LRNs generalized very well to novel inputs. For example, when the network from Figure 7(a) trained on a positive input, was tested with the input of Figure 7(b), a negative one, it predicted the anode break and the action potential accurately. This behavior is illustrated in Figure 8. It can easily be seen that although the output of the network was not a perfect match with the actual output, the network was able to capture the overall properties of negative inputs, such as anode break and an action potential afterwards. This result shows how powerful recurrent networks can be in this task.

Furthermore, when tested with a long-term prediction, this network again outperformed other network architectures. It was able to predict the output for an extra 800 time steps for a positive step signal, although it was trained once only for duration of 232 *ms*. This result is shown in Figure 9. All these tests show that the LRN learned the four characteristic properties of the HH model and was able to generalize successfully to previously unseen data.

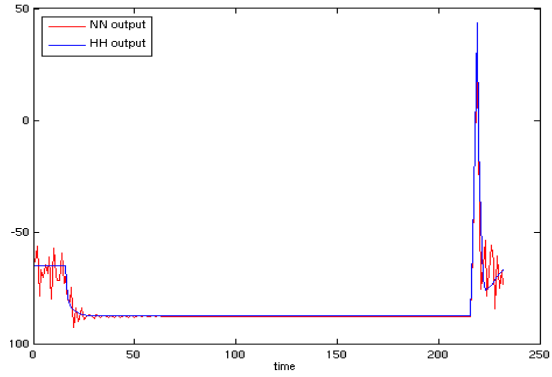
These properties make the LRNs suitable for the tasks of capturing the nonlinear dynamics of a neuron and interacting with the neurons to bring about the desired behaviors.

IV. DISCUSSION & FUTURE WORK

Although it has a microscopic structure, a single biological neuron can perform complex computations that are nonlinear and adaptive in nature. These characteristics make it superior to any simple artificial neuron model, at least in terms of complexity. In order to understand such complexity, HH models have been developed. However, such a model sometimes requires years of manual work to match the output of a biological neuron. This paper shows that it is possible to train an ANN to perform the input-output mapping of a biological membrane simulator from examples, providing a practical way to construct black-box models of biological neurons in future.



(a)



(b)

Figure-7. (a) Output of the LRN trained on positive step current. (b) Output of the LRN trained on negative step current; the network was able to predict the anode break.

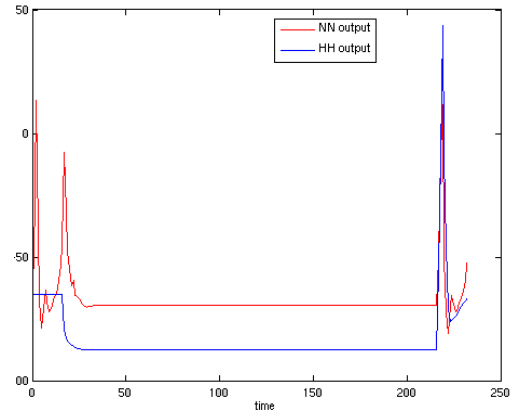


Figure 8. Output of the LRN trained on positive step current, predicting previously unseen negative stimuli.

The results obtained in the experiments indicate that it is possible to learn the basic characteristics of the HH model. Among the three architectures of time-series prediction, LRN stands out as the best, since it was not only able to learn the characteristics of HH equations, but was also able to generalize to novel stimuli. The power of LRN lies in the recurrent connections. The multiple recurrences provided enough memory storage so that the general characteristics of HH model can be learned based on a single example of the input-output mapping.

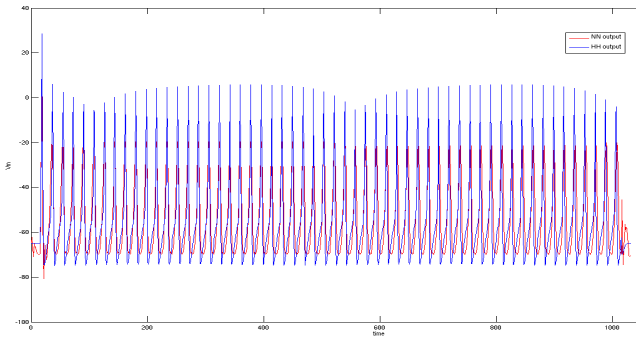


Figure-9. Output of the LRN, trained on positive step current, predicting long term future values.

The most obvious direction of future work is to use this approach to model a biological neuron in vitro. It should be possible to account for the dynamical response of the neuron online without any prior knowledge of its spatial structure, or the ionic currents involved in generating its membrane potential.

If successful, such a model can then be used to design sophisticated biological experiments. For instance, it should be possible to determine how to utilize a dynamic clamp [16] to introduce an artificial membrane or particular synaptic conductances into the biological neuron. The ANN model predicts the future output of the neuron, which means that complex patterns of input current can be calculated for use with the dynamic clamp to produce the desired neural behavior. In the future, it may even be possible to build hybrid circuits of artificial and biological neurons. Such a system would allow physiologists to control the neuron online without having to e.g. block particular ion channels. Such experiments would in turn contribute to developing a detailed understanding of how biological neurons operate and process information.

V. CONCLUSION

This paper shows that ANNs can learn to behave like the Hodgkin-Huxley model of a biological membrane. In the future it should be possible to apply this approach to modeling biological neurons in vitro. The main advantage of this approach is that it does not require any prior knowledge of the physiological properties of the neuron. After training is completed, the neural process is encoded within the weights of the ANN used to model the neuron. Several ANN architectures were tested in this task, with the recurrency in the LRN architecture proving to be the best. The results show that the Hodgkin-Huxley model can be perfectly learned by ANNs. Online modeling using ANNs can provide the necessary tools for capturing the dynamical state of a biological neuron, simulate its output for further analysis, and may provide a more powerful dynamic clamp and online control. Such mechanisms should prove valuable in understanding the behavior of biological neurons in the future.

ACKNOWLEDGMENT

This research was supported in part by NSF under grant EIA-0303.

REFERENCES

[1] F. Rosenblatt, "The Perceptron: a Probabalistic Model for Information Storage and Organization in the Brain," *Psychological Review* 65:386-408, 1958.

[2] Hodgkin, A., and Huxley, A. "A quantitative description of membrane current and its application to conduction and excitation in nerve". *J. Physiol.*, 117:500544, 1952.

[3] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27, Mar. 1990.

[4] S. Chen, S. A. Billings, and P. M. Grant, "Nonlinear system identification using neural networks," *Int. J. Contr.*, vol. 51, no. 6, pp. 1191-1214, 1990.

[5] J. L. Elman, "Finding structure in time," *Cognitive Science*, Vol. 14, pp. 179-211, 1990.

[6] Simon Haykin, "Neural Networks: A Comprehensive Foundation," *2nd Edition*, Prentice Hall, 1999.

[7] A.U. Levin and K.S. Narendra. "Identification of Nonlinear Dynamical Systems Using Neural Networks". *Neural Systems for Control*, Chapter 6, pp129-160.

[8] D. J. Burr, "Experiments on Neural Net Recognition of Spoken and Written Text," *IEEE Trans. On Acoustic, Speech and Signal Proc.*, 36, no. 7, 1162-1168, 1988.

[9] R. P. Gorman and T. J. Sejnowski, "Learned classification of sonar targets using a massively parallel network", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, no. 7, pp. 1135 – 1140, 1998.

[10] T.J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce English text", *Complex Systems* 1, 145-168 (1987).

[11] B. Widrow, R. G. Winter, R. A.Baxter, "Layered neural nets for pattern recognition," *IEEE Trans Acoustics Speech and Signal Processing*, ASSP-36(7):1109-1118, 1988.

[12] J. J, Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci., U.S.*, vol. 79, pp. 2554-2558, 1982.

[13] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biolog. Cybern.*, vol. 52, pp. 141-152, 1985.

[14] D.W. Tank and J.J. Hopfield, "Simple neural optimization networks: an A/D converter, signal decision network and a linear programming circuit", *IEEE Trans. Circ. Syst.*, 33, pp. 533-541, 1986.

[15] H. Rauch and T. Winarske, "Neural networks for routing communication traffic," *IEEE Control Syst. Mag.*, vol. 8, no. 2, pp. 26-31, 1988.

[16] A. A. Sharp, M. B. O'Neil, L. F. Abbott, and E. Marder, "Dynamic clamp: computer-generated conductances in real neurons," *J. Neurophysiol.* 3, 992–995, 1993.

[17] H. Fukai, S. Doi, T. Nomura, and S. Sato, "Hopf bifurcations in multiple-parameter space of the Hodgkin-Huxley equations I: Global organization of bistable periodic solutions," *Biological Cybernetics, 82*, 215-222, 2000.

[18] M. L. Hines and N. T. Carnevale, "The NEURON simulation environment.," *Neural Computation*, 9:1179-1209, 1997.

[19] J. M. Bower, D. Beeman, and M. Hucka, "The GENESIS Simulation System," *The Handbook of Brain Theory and Neural Networks*, Second edition, Cambridge, MA, MIT Press, 475-478, 2003.

[20] Tikk, D., Koczy, L. T., & Gedeon, T. D. (2003). "A survey on universal approximation and its limits in soft computing techniques". *International Journal of Approximate Reasoning*, 33(2), 185–202.