Real Time Interfacing of Spiking Neural Networks in SpineCreator

Sharad Chitlangia
Department of Electrical and Electronics Engineering
BITS Pilani K K Birla Goa Campus
f20170472@goa.bits-pilani.ac.in

Basabdatta Sen Bhattacharya
Department of Computer Science and Engineering
BITS Pilani K K Birla Goa Campus
basab@goa.bits-pilani.ac.in

***Index Terms*—Spiking Neural Networks, Real-time Interfacing, SpineCreator, Network Server, Conductance neurons, DVS E-retina, Izhikevich**

I. Introduction

With the rise of Spiking Neural Network research and their proven advantages over Artificial Neural Networks like being more biologically plausible, researchers have started to experiment with their real-world applications. Artificial Neural Networks essentially act as complex function approximators. They are used from editing photographs in Cameras to predicting really high stake algorithmic stock market trading coefficients. Although Spiking Neural Networks are yet to beat Artificial Neural Networks in every set of tasks, their applications in places where it's possible is an exciting opportunity and research area to explore. There already has been work showing the advantages of Spiking Neural Network software and hardware showing benefits like very low response time [1]. To this end, we explore the real-time interfacing of an up and coming software in Spiking Neural Network literature SpineCreator [2]. SpineCreator allows for easy building of Neural Network models through a GUI interface. Parameters, State Variables and constants as well as their differential equations and experiments can be easily specified and setup. We test and show results of SpineCreator for its external interfacing capabilities with different sets of test populations and keep our focus on its real-time interfacing capabilities. We create external servers to make the data available on through the help of sample codes in SpineCreator.

II. Background

A. Artificial Neural Networks and their real world applications

Artificial Neural Networks are a particular set of Machine Learning models which postulate the neuron structure to be a simple function approximator of the form

$$o = f(W * x + b) \tag{1}$$

Which learns its weights W using the famous backpropagation rule. Although this seems to be a very model-based approach, ANNs are proven to be effective in solving problems in almost every industry ranging from Medical Imaging models to Stock market trading to Weather prediction, etc. Researchers have also made custom models suit specific purposes based on top of Artificial Neural Networks. For example Graph Neural Networks for Protein Interface prediction [3], Convolutional Neural Networks for Image Models [4], Recurrent Networks for Text and Speech based data [5].

B. Spiking Neural Networks and their real-world applications

Recent work in showing the real-time interfacing advantages of Spiking Neural Network like [1] has demonstrated the advantages of Spiking Neural Networks. They are more power-efficient as well as allow for faster decision making compared to artificial neural networks. Artificial Neural Networks can be made faster with the use of Specialised hardware such as GPU, TPU [6] but these also are very power-hungry reducing run-time of systems with limited power supply. The response time of SNNs is almost close to zero which for example allows for faster manoeuvring in drones.

C. SpineCreator

SpineCreator is a GUI based software to build Neural Network Models. It has multiple panes - network editor, experiment editor, component editor and also two panes to observe the network and the final results (plots using OpenGL). Network editor allows a user to add components e.g. neural bodies, synaptic connections, spike sources. It also allows for editing constant parameters that are defined in the component editor. The component editor allows editing equations, adding state variables, constant parameters, etc. Finally, the experiment editor allows for editing the inputs, outputs, etc.

D. Izhikevich based Neuron model

Reference [7]. The model is represented by two state variables $u(t)$ and $v(t)$. $v(t)$ represent the membrane potential and $u(t)$ is the recovery variable. It only depends on four parameters a, b, c, d.

$$\frac{dv(t)}{dt} = 0.04v^2 + 5v(t) + 140 - u(t) + I_{\text{dc}} - I_{\text{syn}}(t) \tag{2}$$

$$\frac{du(t)}{dt} = a(bv(t) - u(t)) \tag{3}$$

$$If \;\; v(t) > 30, then \tag{4}$$

$$v(t) \leftarrow c; u(t) \leftarrow u(t) + d \tag{5}$$

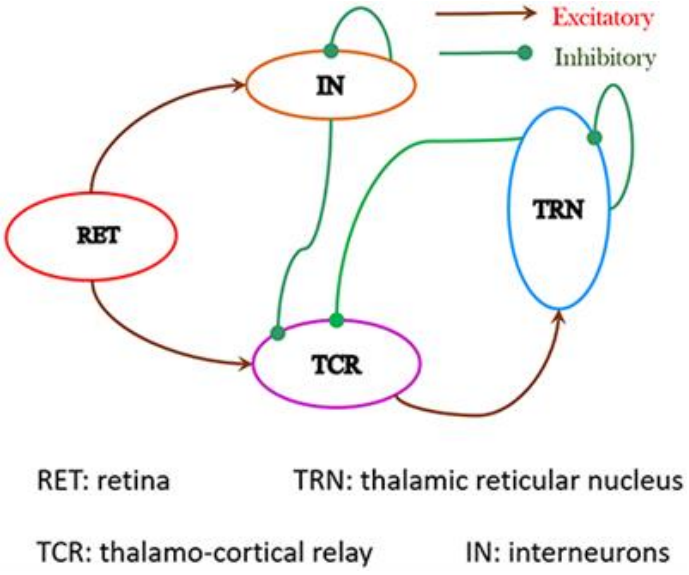

Fig. 1: The LGN model as used in [8]

Fig. 2: The DVS E-retina sensor

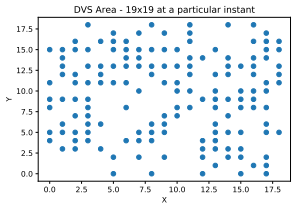

Fig. 3: Spiking at an instant in DVS

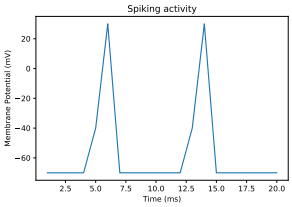

Fig. 4: Spiking Activity v s timesteps

E. Lateral Geniculate Nucleus

The LGN (in Fig. 1) is a body of neurons which helps in relaying and processing information from the optic nerve to the occipital lobe. It lies in the thalamic nerve and is one of the major components of the visual pathway. In particular, we take a look at the ventral LGN model. We formulate the LGN as formulated in [8] we three groups of neuron bodies - TCR (Thalamo Coritical Relay Cells), IN (Thalamic interneurons) and TRN (Thalamic Reticular Neurons). The TCR and IN cells help in relaying information from the retinal axons to the visual cortex. The TRN primarily plays a role of feedback from the visual cortex to the TCR and IN cells which comprise of a majority of excitatory synapses. In the experiments mentioned in this paper, we only test the TCR cells.

F. Dynamic Vision Sensor E-retina

Dynamic Vision Sensor E-retina [9] is very much like the human retina which transmits local pixel level changes at the time of occurrence. Its perfect for real time monitoring with Spiking Neuron Models unlike traditional Artificial Neural Network models. It transmits visual spikes. A higher rate of spikes would be equivalent to higher current in the retinal axon. Current is what we feed into the Izhikevich model based neurons. The DVS plays a similar role as the retinal cells rods and cones of transmitting visual light into electrical signals which can be processed by neuronal bodies. The electrical signals that are passed through the optic nerve are similarly passed into the model on SpineCreator through a network server that is created through sample codes available in SpineCreator's repository. Fig. 2 shows how the sensor looks like. Fig. 3 shows how a particular instant the DVS might be spiking. Fig. 4 shows the spiking a particular sensor in the DVS Camera and how it may have been spiking. For a complete description, the DVS can be thought of to be capturing a 3-dimensional recording. The front 2D face of which is represented by the Fig. 3 and the z-axis looks like Fig. 4 for every sensor in the DVS area

III. Related Work

Spiking Neural Networks have shown a lot of promise in their real time interfacing applications with customized hardware because of their low latency, low computational overhead and low power design. Researchers have even shown their applications on edge devices, on which even Artificial Neural Networks have shown very less promise due to their high computational load. [1] showed that with customized neuromorphic hardware on a drone using Mixed - Analog and Digital Signal design it's possible to navigate a drone with Dynamic Vision Sensor based inputs with a very fast reaction time. [10] showed that using simple biologically inspired dynamics using Spiking Neural Networks one can design a control algorithm so as to control a prosthetic arm using Brain Computer Interface. Their design was tested with a rhesus monkey with the control algorithm using a 2000 Spiking Neural Network in Matlab. Similarly, using the help of a Neural Engineering framework [11] showed that using a

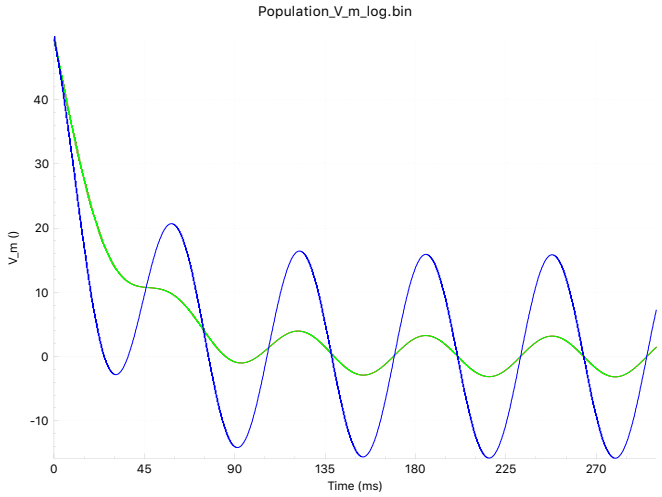

Fig. 5: Output V_m of the two neurons corresponding to V_in in baseline

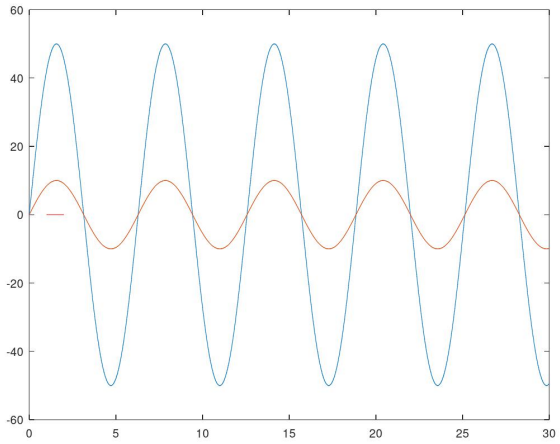

Fig. 6: Input V_m to the two neurons in baseline.

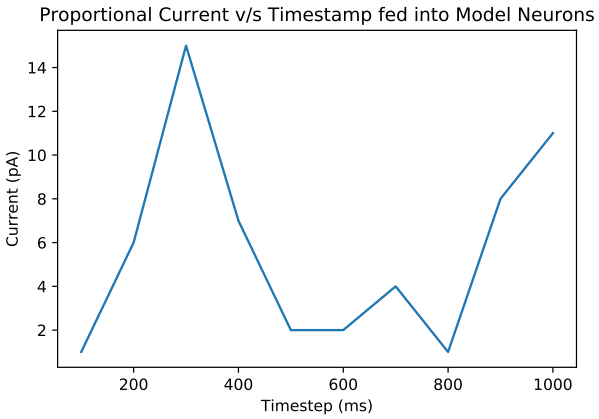

Fig. 7: Example of Analog current converted through Eq. 8

Kalman filter based approach for transforming Spiking Neural Networks for usage in Brain Machine Interface. Reference [12] explored learning dynamics of computer vision using event based sensors using a synaptic plasticity based rule. Their complete experiments were performed on SpiNNaker [13]. Reference [14] made use of Resistive Random Access Memory for implementation of low latency spikes of a spiking neural network followed by brain computer interface to for neural prosthetic application. Their online learning algorithm is inspired by Spike Timing Dependent plasticity. Reference [15] showed a novel Spiking Neural Network based Classification algorithm implemented on a GP-GPU platform. As their learning algorithm they use NormAD approximate gradient descent based supervised learning algorithm.

IV. Methods

In this section we describe our methods to create the external server, pre-processing of the data and the neural networks on which we test the methodology.

A. Network Server

The network server is created by sample codes available in SpineCreator in Octave. The complete communication with the network server can be divided into five steps:-

1) Starting the local Network server
2) Add data to the context object. Also define cleanup objects.
3) Query the current state of the network server. This tells if the network server has been contacted by an external source for data.
4) Get data from the local network server, if the external source sending in data is sending any.
5) Stop the local network server.

Configuring SpineCreator to receive external input from this local server is fairly simply and just requires setting the external input option in the inputs section in the Experiment tab. The port numbers should be carefully matched while setting up the experiment in the Experiments tab and while running the sample scripts because if the port numbers will not match there will be no 'handshake'. The exact definitions of handshake and data sending are defined in a file named protocol.txt in SpineCreator's repository.

B. Establishing a Baseline

To establish a baseline we run the sample example given in SpineCreator's repository. The sample defines a neural body of two LIF neurons with a state variable V_m and τ_{m}.

$$\frac{dV_m}{dt} = \frac{-(V_m + V_i n)}{\tau_{\mathrm{m}}} \tag{6}$$

Here, V_in is the external input voltage to the two LIF neurons. The parameter τ_{m} is set to 30ms. Two input sine waves of frequency with different amplitudes are passed in. The input and output are shown in Fig. 6 and Fig. 5 respectively.

Fig. 8: Rasterplot of 80 TCR conductance based neurons responding to current in Fig. 7

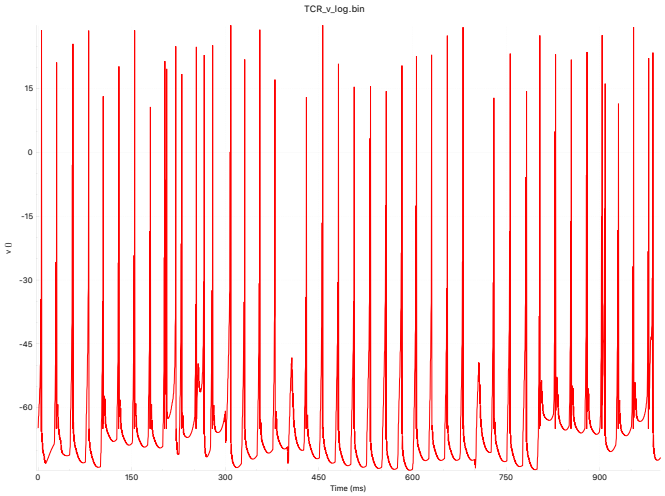

Fig. 9: Response Voltage log of index 0 neuron to current in Fig. 7

C. Conductance based Neurons

Based on the baseline from sample codes in SpineCreator, we test a conductance based model similar to the model postualted in [7] with the equations described in Related Work subsection D. We use the same sample inputs as used in the baseline. To our surprise, we find no response from SpineCreator whatsoever. The experiment seems to be stuck at 0% progress and results in an error. So we instead feed in through the *Time Varying Input*. We take special care in changing no values from the external recorded DVS Input. So we convert the DVS Input spikes into an analog current using the following formula:

$$I \propto \frac{Numberofspikes}{WindowLength} \tag{7}$$

$$I = k\frac{Numberofspikes}{WindowLength} \tag{8}$$

For simplicity purposes, we assume k=1. A sample input is show in Fig. 7. The parameters and equations for the model are as follows:

$$\frac{dv(t)}{dt} = 0.04v(t)^2 + 5v(t) + 140 - u(t) + I_{\text{dc}} + T_{\text{syn}} \tag{9}$$

$$\frac{du(t)}{dt} = a(bv(t) - u(t)) \tag{10}$$

$$I_{\text{syn}} = I_{\text{ampa}} + I_{\text{gaba}} \tag{11}$$

On condition: $v(t) > V_{\text{peak}}$

$$v(t) := c \tag{12}$$

$$u(t) := u(t) + d \tag{13}$$

We monitor the output $v(t$, which is a state variable here. I_{ampa} and I_{gaba} are input currents here. For simplicity purposes, I_{gaba} is given a constant value of $1pA$ but I_{ampa} is given value from the Analog current formula defined in Eq. 8 with k=1. We assume V_{peak} to be 30mV. Table I defines the values of the other parameters.

Parameter	Value
a	0.02
b	0.2
c	-65
d	6
V_{peak}	30
I_{dc}	5
$v_{initial\ value}$	-65
$u_{initial\ value}$	-13

TABLE I: Parameters of Conductance based neurons experiment

V. Discussion

A. Baseline

For the baselines, through the external server we often see that the total experiment run time is often 10x-100x than the experiment times. For e.g. if the experiment is run for 0.3s, the total run-time of the experiment would be somewhere in the range of 3seconds - 30 seconds. This may vary from system to system. If the external input is replaced with a constant input to the network, the experiment run-time would be the same as the experiment simulation time.

B. Conductance based neurons

Interestingly we see no response whatsoever through the network server. This could potentially be due to the double differential nature of the Conductance based networks. So we instead feed in the network with Time varying input defined according to the formula defined in Equation 8.

As expected we see that the TCR neurons respond exactly as expected. The spiking activity is higher when the input current is higher. It can also be observed through the voltage

plots that the frequency of an individual neuron firing is also higher when the current is higher. It should be noted that the depicted voltage response, rasterplot and the analog current is only one of the multiple runs performed to confirm the behavior.

VI. Conclusion and Future Work

We test the real-time interfacing capabilities of SpineCreator and show that whereas it's possible to manually feed in values, going by the network server approach there is sometimes a bit of a 'lag' in communication. We show the results if the bug is resolved and confirm the hypothesized behavior of Conductance based Izhikevich Neurons to DVS E-retina Inputs by a novel method of converting the input spikes into Analog current.

The next steps would be to test on the complete LGN model and resolution of the network server bug that currently limits the complete real time interfacing of the model in SpineCreator.

Acknowledgment

I would like to thank my supervisor, **Dr. Basabdatta Sen Bhattacharya** for their valuable time, patience and expertise during this project. Even though mentoring so many students, there was never a time where I was denied time. In fact, I was even told that unplanned meetings would be ok. I would also like to thank the SpineCreator team especially Sebastian James for the invaluable documentation provided. Even though SpineCreator is a software meant to be for Spiking Neural Network research, I actually got myself learning a few networking concepts here and there! I would also like to thank Aditya Ahuja for helping me figure our SpineCreator a bit initially and getting me upto speed with it as we started with the project a bit late (almost half the semester in).

References

[1] Moritz B. Milde, Hermann Blum, Alexander Dietmüller, Dora Sumislawska, Jörg Conradt, Giacomo Indiveri, and Yulia Sandamirskaya. Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system. *Frontiers in Neurorobotics*, 11:28, 2017.

[2] A. J. Cope, P. Richmond, S. S. James, K. Gurney, and D. J. Allerton. Spinecreator: a graphical user interface for the creation of layered neural models. *Neuroinformatics*, 15(1):25–40, Jan 2017.

[3] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6533–6542, USA, 2017. Curran Associates Inc.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[6] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. *SIGARCH Comput. Archit. News*, 45(2):1–12, June 2017.

[7] E. M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572, Nov 2003.

[8] Basabdatta Sen-Bhattacharya, Teresa Serrano-Gotarredona, Lorinc Balassa, Akash Bhattacharya, Alan B. Stokes, Andrew Rowley, Indar Sugiarto, and Steve Furber. A spiking neural network model of the lateral geniculate nucleus on the spinnaker machine. *Frontiers in Neuroscience*, 11:454, 2017.

[9] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck. Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output. *Proceedings of the IEEE*, 102(10):1470–1484, Oct 2014.

[10] Julie Dethier, Paul Nuyujukian, Chris Eliasmith, Terrence C. Stewart, Shauki A. Elasaad, Krishna V Shenoy, and Kwabena A Boahen. A brain-machine interface operating with a real-time spiking neural network control algorithm. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2213–2221. Curran Associates, Inc., 2011.

[11] Julie Dethier, Paul Nuyujukian, Stephen I Ryu, Krishna V Shenoy, and Kwabena Boahen. Design and validation of a real-time spiking-neural-network decoder for brain–machine interfaces. *Journal of Neural Engineering*, 10(3):036008, apr 2013.

[12] Michael Hopkins, Garibaldi García, Petruț Bogdan, and Steve Furber. Spiking neural networks for computer vision. *Interface Focus*, 8:20180007, 08 2018.

[13] Javier Navaridas, Mikel Luján, Luis A. Plana, Steve Temple, and Steve B. Furber. Spinnaker. *Parallel Comput.*, 45(C):49–66, June 2015.

[14] Thilo Werner, Elisa Vianello, Olivier Bichler, Daniele Garbin, Daniel Cattaert, Blaise Yvert, Barbara De Salvo, and Luca Perniola. Spiking neural networks based on oxram synapses for real-time unsupervised spike sorting. *Frontiers in Neuroscience*, 10:474, 2016.

[15] Shruti R. Kulkarni, John M. Alexiades, and Bipin Rajendran. Learning and real-time classification of hand-written digits with spiking neural networks, 2017.