ACCELERATING BIOPHYSICAL NEURAL NETWORK SIMULATION WITH REGION OF INTEREST BASED APPROXIMATION

A Dissertation
Presented to
The Academic Faculty

by

Yun Long

In Partial Fulfillment
of the Requirements for the Degree
Master in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
April 2019

ACCELERATING BIOPHYSICAL NEURAL NETWORK SIMULATION WITH REGION OF INTEREST BASED APPROXIMATION

Approved by:

Dr. Saibal Mukhopadhyay, Advisor
School of ECE
Georgia Institute of Technology

Dr. Arijit Raychowdhury
School of ECE
Georgia Institute of Technology

Dr. Asif Islam Khan
School of ECE
Georgia Institute of Technology

Date Approved: [04/13, 2019]

ACKNOWLEDGEMENTS

First, I would like to especially thank my PhD advisor Dr. Saibal Mukhopadhyay, without his guidance and support I would not have the chance to work on these interesting topics, to learn how to do research. I also want to thank my PhD proposal committee members Dr. Arijit Raychowdhury and Dr. Asif Islam Khan, your suggestions and advises are very valuable and indispensable. I also would like to thank Georgia Tech and ECE department for providing such a wonderful place to study and work. I also want to say thank you to all my lab mates for the generous help during the past four years.

At last, I want to say thank you to my wife, who give me support and understanding all the time. I can't make it without you.

Many thanks to you all!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

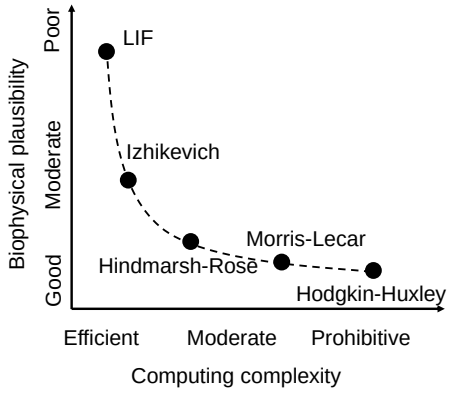
Modeling the dynamics of biophysical neural network (BNN) is essential to understand brain operation and design cognitive systems. Large-scale and biophysically plausible BNN modeling requires solving multiple-terms, coupled and non-linear differential equations, making simulation computationally complex and memory intensive. In this work, an adaptive simulation methodology is presented in which neurons in the region of interest (ROI) follow high biological accurate models while the other neurons follow computation friendly models. To enable ROI based approximation, we propose a generic template based computing algorithm which unifies the data structure and computing flow for various neuron models. We implement the algorithms on CPU, GPU and embedded platforms, showing 11x speedup with insignificant loss of biological details in the region of interest.

CHAPTER 1. INTRODUCTION

Biophysical neural network (BNN) modeling provides an avenue for exploring hypotheses about how human brain works and how to realize neuronal coding [1]. Moreover, it is a critical step towards developing cognitive system [2], artificial intelligence (AI) [3], and emerging computer architecture [4], etc. However, the BNN modeling is challenging as the dynamics of neurons and synapses are regulated by complex, coupled non-linear differential equations, making it computation intensive.

Moreover, for BNN simulation, there is a well-known tradeoff between the computing efficiency and the biology accuracy. Biologically accurate models always require more computation while less computing intensive models normally lack biophysical plausibility. For example, Hodgkin-Huxley model [5] presents high degree of biological credibility but is very computationally complex. Simplified mathematical models, such as leaky integrate-and-fire model (LIF model) and Izhikevich models [6], improve the computing efficiency but lack biology accuracy. Figure. 1 shows the tradeoff with several commonly used neuron models.

To accelerate BNN simulation, parallel computing frameworks such as CPU clusters and general-purpose graphics processing units (GPGPUs) are being actively explored [7, 8]. However, the performance is ultimately limited by the algorithm's ability to leverage parallel hardware and the memory bandwidth. There are efforts in developing specialized application-specific integrated circuit (ASIC) which provide significant improvements on performance and energy efficiency [4, 9]. However, the ASICs normally implement a single neuron model, which lack the flexibility to support different types of neuron



Models computation complexity

Models	# of equations and parameters	# of FLOPS [6]
LIF	2/5	5
Izhikevich	4/5	13
Hindmarsh-Rose	5/7	120
Morris-Lecar	5/12	600
Hodgkin-Huxley	10/7	1200

Figure 1. The tradeoff between computing efficiency and biophysical plausibility.

dynamics. Moreover, programming the ASIC also require specialized knowledge compared to standard CPU or GPU based platforms.

In this work, an adaptive simulation methodology is developed that maintains high biological accuracy with more complex neuron models (e.g. Hodgkin-Huxley model) in the region of interest (ROI), while simplifys the neuron models (e.g. LIF and Izhikevich model) elsewhere to improve the overall computation speed (different models complexity are shown in Figure 1). At the interested regions, the ROI based approximation retains all the biophysical information such as the sodium (Na^+) and potassium (K^+) ion channel conductance which is missing in LIF and Izhikevich models. In the proposed algorithm, the ROI can be statically defined as a specific region or determined dynamically during simulation based on factors such as spiking frequency. The key challenge in ROI based BNN simulation is the need to solve a system of hybrid neurons where the neuron models (i.e. sets of partial differential equations (PDEs)) change over space as well as time. To address the preceding challenge, a generic template based computing model is proposed, where both data and computing are stored/defined as templates. The proposed template

based processing algorithm provides a uniform data structure and computing flow for various neuron models. Therefore, models can be easily switched with no programming overhead.

The proposed ROI based computing algorithm is implemented on CPU, GPU and embedded system. Our baseline simulator with template based processing (without ROI approximation) provides accuracy and performance comparable to the state-of-the-art BNN simulators. The effectiveness of ROI based adaptive simulation is demonstrated through a visual cortex simulation. The experimental results measured from CPU, GPU, and embedded platforms demonstrate 11x speed-up on average with insignificant biological accuracy loss in the regions of interest.

CHAPTER 2. BACKGROUND

2.1 Dynamics of Biophysical Neural Networks

2.1.1 Dynamics of neuron models

As the basic computational element in BNN, the neuron operates in a "receive-integrate-send" mode. A neuron receives input signals from 10^3 - 10^4 neighborhood neurons via its *dendrites,* and, after integration, sends out the output signals to thousands of other neurons via its *axons*. The signal can be either electrical or chemical depending on the neurons and the connection types. The most critical variable in neuron dynamics is membrane potential which updates continuously based on the integration of input signals. Once the membrane potential reaches a pre-defined threshold, neuron fires and sends spikes down to its thousands of post-neurons through axons.

Numerous efforts have been made during past few decades to model the neuronal dynamics. One of the most famous is the Hodgkin-Huxley model (HH model) which was developed in 1952 [10]. The general phenomenon of the membrane potential described by HH model is shown in Figure 2(a). The HH model explores neuron dynamics and also behaviors of voltage-gated ion channels (e.g. sodium and potassium channels) using a set of coupled differential equations. However, HH model is computationally complex, particularly for simulating large BNNs. On the other hand, the Leaky Integrate-and-Fire model (LIF model) is much simpler, but lacks biological plausibility, shown in Figure 2(b). To better balance the biological plausibility and the computation efficiency, phenomenological spiking models are adopted for BNN simulation. For example,

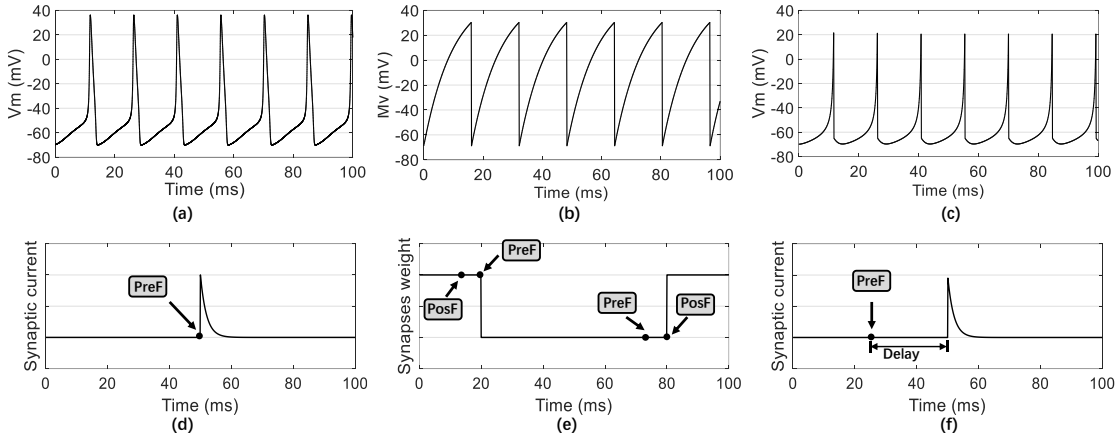


Figure 2. General phenomenon for neuron and synapse dynamics with different models. (a-c) show neuron membrane potential (Vm) trajectories with constant input current considering different neuron models; (d-f) show synaptic weight or synaptic current evolution considering different synapse dynamics. (a) Hodgkin-Huxley model; (b) Leaky Integrate-and-Fire model; (c) Izhikevich model. (d) Short Term Plasticity; (e) Long Term Plasticity (e.g. STDP); and (f) Delay type of synapse. PreF and PosF mean pre-synaptic and post-synaptic neuron fires, respectively.

Izhikevich model, shown in Figure 2(c), which has only two Ordinary Differential Equations (ODEs) for neuron dynamics, are much more computation friendly than HH model yet contains detailed neuronal behaviors that maintains biological plausibility [11]. Details of the aforementioned neuron models are presented in appendix.

2.1.2 Dynamics of synapse models

As the bridge between the axon of pre-neuron and the dendrite of post-neuron, synapse is the key element for both electrical and chemical signal propagation. There are

mainly two synapse models: the current based (CUBA) model where the transmitted current is proportional to synaptic weight and the conductance based (COBA) model where the synaptic current depends on the synaptic value as well as pre-neuron's membrane potential [12].

The synapse weight is also time dependent variable, and in most cases, regulated by differential equations. In this work, besides the fixed value synapse, we consider three different aspects of synapse dynamics. First, Short Term Plasticity (STP) in which the synapse weight can dynamically increase or decrease based on pre-neuron's activity but will quickly recovers to its original state with decades of milliseconds, as shown in Figure 2(d). Second, Long Term Potentiation (LTP) in which the synapse strength can permanently (at least in a much longer time period than STP) be changed. Following Hebbian theory: "Neurons that fire together, wire together" [13], our simulations utilize the Spike-Timing Dependent Plasticity (STDP) that synapse weight potentiates when its post-synaptic neuron fires right after its pre-synaptic neuron, and weight depresses when it post-synaptic neuron fires right before its pre-synaptic neuron. The STDP phenomenon is illustrated in Figure 2(e). Third, Synapse Delay in which the signal propagation get delayed when passing through synapses, shown in Figure 2(f). Details of synapse dynamics are presented in the appendix.

2.2 Performance Analysis for Different Models

Neuron models with detailed biophysical dynamics requires more computation while less computationally demanding models tend to be less biologically plausible. Besides neuron models, synapse dynamics is also very critical especially when the network has

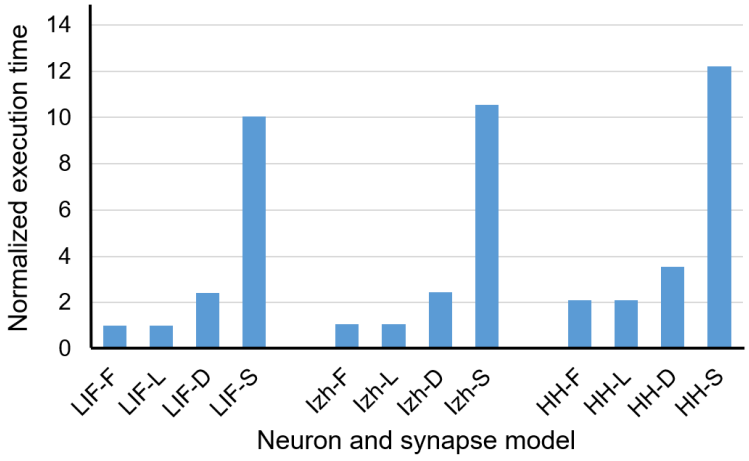


Figure 3. Computing complexity considering combinations of different neuron and synapse models. Neuron models include LIF, Izhikevich, and HH model; Synapse models include fixed (F), STP (S), LTP (L), and Delay (D).

very dense connections (many connections per neuron). Figure 3 shows the computation complexity considering different combinations of neuron and synapse models [6, 12, 14]. Neuron models includes LIF model, Izhikevich model, and HH model. Each neuron model is evaluated under different synapse dynamics including Fixed (F), STP (S), LTP (L), and Delay (D). For example, "Izh-F" means the measurement is based on a Izhikevich neuron which is connected to fixed type of synapses. The firing rate of all the pre-neurons are 20 Hz. Data are normalized according to the execution time with LIF neuron model and fixed value synaptic connections for better visualization. We observe that with different combination of neuron and synapse models, the computing speed could be very different.

2.3 Prior Works for BNN Simulation Tools Design

Various software-based simulators have been developed to expedite the simulation of BNN [8, 15-22]. The early stages of simulators design mainly utilized CPU and super-computing cluster for the best performance. For example, NEURON [15] and GENESIS [16] offer CPU version with both single node and cluster based computation mode. Later on, simulation tools with better user convenience were developed by implementing high-level language (e.g. Python) based user interface. For example: NEST [17], BRIAN [18], PCSIM [19], and PyNN [20]. NEST is a C++ based highly optimized tool which provides Python interface. BRIAN is a high flexible tool written in Python which supports both standard types of neuron models and user-defined models. PCSIM is similar with NEST which has C++ based computation kernel and Python programming language based user interface. Different from the above simulators, PyNN is a simulator-independent BNN computing platform. With PyNN API and Python, code can run without any modifications on different simulators that PyNN supports (currently NEURON, NEST, PCSIM, and BRIAN). Recent years, the GPU based simulator design has attracted lots of interests. BNN simulation running on an off-the-shelf GPU can achieve more than 10X speedup than the most advanced CPU. There are several public available GPU based simulators, such as CARLsim [8], NCS6 [21], and HRLSim [22]. CARLsim is a Izhikevich model based, highly-optimized, GPU-accelerated BNN simulator which is mainly used to accelerate BNN simulation with GPU but also provides non-GPU support. NCS6 is designed to run on clusters of multiple CPUs and GPUs. It supports both LIF neuron model and Izhikevich neuron model. HRLSim is motivated by the need to support the neuromorphic hardware

[23]. It is implemented on a cluster of GPU as an affordable and scalable tool for design, real-time simulation, and analysis for large-scale BNN.

2.4 Challenges of BNN Simulation Tools Design

Developing of BNN simulation tools faces two main challenges: *system flexibility* and *computing efficiency*. A simulation framework should be flexible enough to support multiple neuron/synapse dynamics. To address this, Object-oriented Programming (OoP) can be utilized to enhance the code reusability and system flexibility. In the OoP based design, neurons and synapses are instances of respective model classes. Parameters and variables for models are encapsulated as attributes and dynamics (ODEs/PDEs) are defined as methods inside objects. This approach results in flexibility but sacrifices the computing efficiency, which is a critical challenge for large-scale BNN simulation. Some BNN simulation tools implement a mixed configuration which has a high-level object-oriented interface for flexibility and user convenience, and a low-level procedural kernel utilizing the vectorization techniques for computing efficiency [17, 18, 20]. In most cases, the high-level user interface is presented with an interpreted language such as Python while the computing kernel are written by low level language such as C or C++. As shown in Figure 4, the high-level user interface (Python script) describes a BNN with 1000 LIF neurons; Neurons are divided into two groups and randomly connected together with 0.1 connection possibility. The corresponding data layout for the network is also shown here. The demonstration code is modified from the sample provided by BRIAN simulator [18].

Sample code:

```
P = NeuronGroup(1000, model='LIF',
          threshold=50, reset=-60);  // network size and neuron type
Pe = P.subgroup(800);  // excitory neuron
Pi  = P.subgroup(200);  // inhibitory neuron
Connect_random(Pe, P, 0.1, weight=1.62);  // connection for excitory neuron
Connect_random(Pi, P, 0.1, weight=-9);  // connection for inhibitory neuron
```

Data layout:

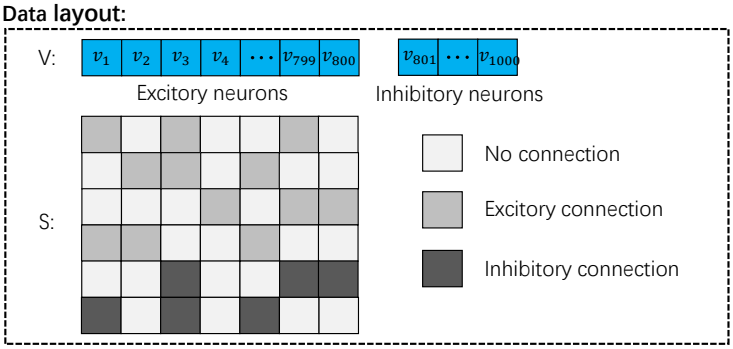


Figure 4. High-level object-oriented user interface and low-level data layout. Demon code is modified from the sample provided by BRIAN simulator.

CHAPTER 3. PROBLEM FORMULATION

3.1 The opportunities for ROI based BNN simulation

Large-scale cortical modeling is one of the most critical scientific challenges in 21st century. The cortical simulation facilitates better understanding of human brain, exploring hypotheses of neuroscience, and developing human-like artificial intelligence. The central nervous system (CNS) of human contains 10^{11} neurons and 10^{14} synapses, coupling together and regulated by complex neuronal dynamics, which is challenging for simulation even with the most powerful supercomputers [1, 4]. **On the other hand, it is well known that only a small portion of CNS are responsible to functions such as vision, sound, and motion control**. Modern imaging techniques such as PET (positron-emission tomography) and fMRI (functional magnetic resonance imaging) also indicate that neurons

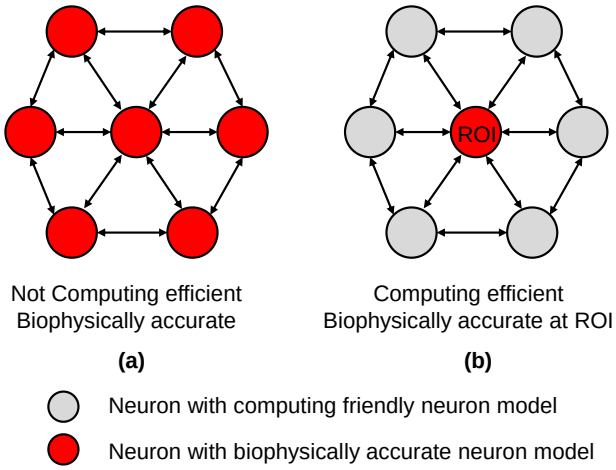


Figure 5. BNN simulation with (a) homogenous neuron models (all biophysical accurate model) and (b) ROI based heterogeneous models with biophysical accurate model at concerned region and simpler model for the others.

in different regions have varying activation levels. This inspires us to develop a ROI based approximation algorithm to accelerate cortical simulation where the highly-active or critical regions are modeled with biophysical accurate neuron models while the rest are modeled with simpler models (Figure. 5).

3.2 Challenges in ROI based BNN Simulation models

The ROI based trade-off in accuracy and computation speed is a well-known concept in scientific computation. For example, adaptive mesh refinement (AMR) algorithm imposes finer sub-grids (denser grids) at the regions that require higher resolution to achieve better accuracy. However, unlike the conventional ROI based computations that mainly focus on the data-level approximation, the proposed algorithm for ROI based BNN simulation is to change the underlying neuron dynamics being solved at each node (i.e. model-level approximation). To be more specific, rather than emulating the detailed neuronal dynamics with the same model for all the neuron groups (Figure 5(a)), we can simplify the simulation by using computing friendly model for the less important regions and biophysically accurate model for the critical regions (Figure 5(b)). Therefore, *we need to simulate a system of coupled differential equations where the equations are not only different at different nodes, but also might changes over time (ROI changes over time).* This is a unique approach for performance-accuracy trade-off in BNN simulation. A few of prior works implement heterogeneous simulation where different neuronal dynamics are modeled simultaneously [17], however, our approach is to dynamically change the model of the neurons, for example, depending on their spiking activity. Moreover, our objective is to accelerate the simulation while maintaining the biology accuracy of the interested region.

In this work, we target on three neuron models: LIF model, Izhikevich model, and Hodgkin-Huxley model. We focus on the ROI approximation for neuron dynamics and assume synaptic weights are fixed. In the future work, we will implement the ROI approximation for synaptic dynamics.

CHAPTER 4. ALGORITHMS

4.1 Template based computing

We propose a template based computing algorithm where variables (e.g. neuron membrane potential) and synaptic weights are stored in *data templates*, neuron dynamics are defined in *computing templates*. The template based processing allows us to configure and simulate a BNN where all the data and computing are represented in a unified form. Therefore, ROI based adaptive BNN simulations consisting of different neuron models can be easily realized by simply utilizing different data and computing templates.

An example is used to show how the template based processing works. As shown in Figure 6(a), the BNN is utilized for visual cortex simulation, which contains 10^3 Izhikevich neurons and the synapse connectivity is 0.1 (i.e. each neuron has 10^2 synapses). Figure 6(b) shows the corresponding data templates including V, U and synaptic weights. Storing data into templates provides a compact data structure and enhance the computing efficiency for element-wise functions, especially for parallel computing platforms such as GPU [8].

The concept of computing template is similar with the computational graph in popular deep learning frameworks such as Tensorflow [11]. A computing template consists of a series of matrix operations. Each operation in a computing template takes two or more data templates (vectors and matrix) as inputs. The computing templates are pre-define based on the differential equations in the neuron models. As shown in Figure 6(c), the computing templates are defined based on the differential equations of Izhikevich model in Figure 6(a).

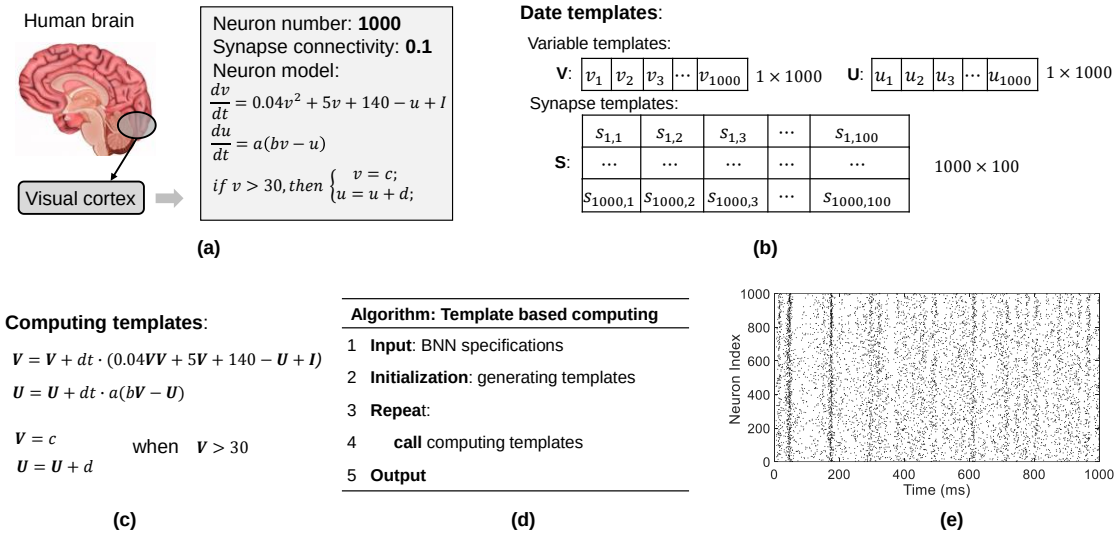


Figure 6. Template based processing. (a) BNN specifications. (b) Data templates. (c) Computing templates. (d) Algorithm for template based processing (no ROI). (e) Output results as a raster plot.

During computing, templates are first generated based on network specifications. Then the computing templates are called repeatly to perform matrix operations. The whole algorithm is shown in Figure 6(d). Meanwhile, neuron activities such as membrane potential and spiking frequency can be recorded for post analyses. As shown in Figure 6(e), we plot the recorded spikes (i.e. raster plot) for neurons in the network.

4.2 ROI based approximation

Benefiting from the template based computing, the neuron models can be switched easily by calling different computing templates, enabling ROI based approximation. As mentioned earlier, the ROI can be statistically defined by identifying a spatial region where higher accuracy is desired during the simulation. Alternatively, the ROI can be dynamically

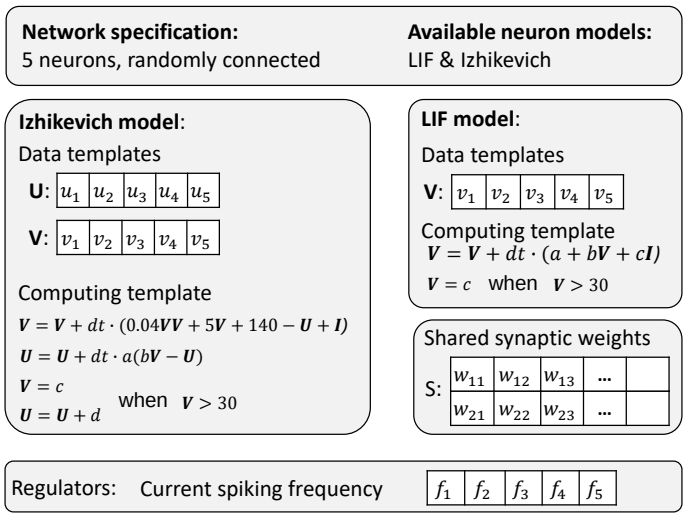


Figure 7. A simple network with 5 neurons. Neurons can switch models dynamically between LIF and Izhikevich model according to the spiking frequency.

defined using certain criteria. For example, the information theory of a rate-based network suggests that the more a neuron fires, the more information it propagates [12]. Hence, the spiking frequency can be a criterion to keep the 'busy' neurons in biological plausible model (defined as ROI).

Figure 7 illustrates the proposed algorithm considering 5 randomly connected neurons, and assuming the neuron models can switch between LIF and Izhikevich models (in practice, we consider model switching between the LIF and the Hodgkin-Huxley models). We first initialize data and computing templates for both models. Then we create a *regulator vector* which contains the spiking frequency for each neuron. During inference, system first accesses the regulator vector and determines which model the neuron should follow, which data templates and computing templates should be called. For example, if we define the model switching threshold to be 20 Hz, the corresponding neuron will follow

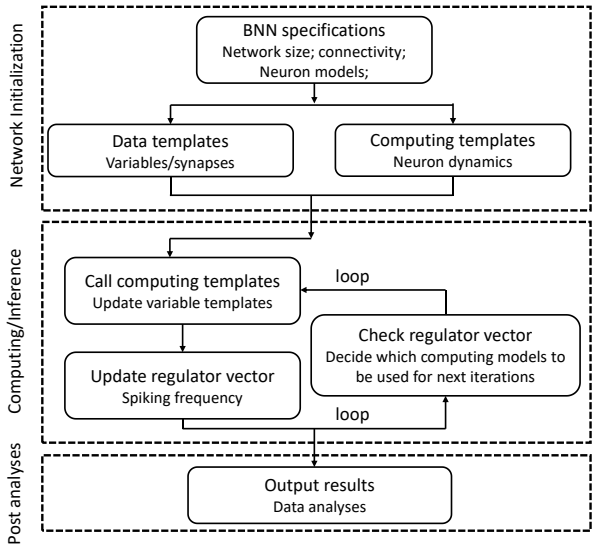


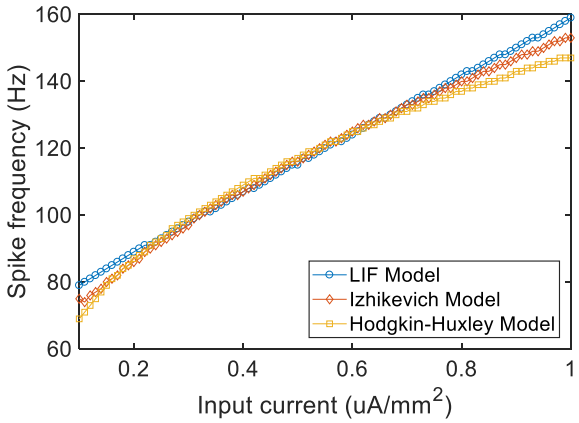
Figure 8. Computation flow for ROI based adaptive simulation algorithm.

Izhikevich model once the spiking frequency is higher than 20 Hz, otherwise it will follow LIF model.

Figure 8 illustrates the computation flow for ROI based adaptive simulation. First, during network initialization, data and computing templates for different neuron models are created. Second, during computation/inference, computing templates are called and neuron spiking frequency are updated with a given time window. After each iteration, program checks the regulator vector and determine which model the neuron should follow.

4.3 Parameter tuning

Even though the internal neuronal dynamics are different for different neuron models, ROI based approximation requires that the output (represented with spikes) of a neuron should remain the same after changing models. To acheve this goal, we carefully tune the parameters in computing friendly models (i.e. LIF model and Izhikevich model) with the



Tuned Parameters:

LIF model:

$a = -6.77$ $b = -0.099$

$c = 0.315$

Izhikevich model:

$a = 5.0$ $b = -0.273$

$c = -56.0$ $d = 2.0$

Figure 9. Neurons with different models generate similar spiking frequency after parameter tuning.

Hodgkin-Huxley model as a baseline, to match the spiking frequency for different input current. Figure 9 shows the tuned results. The spike frequencies match with each other very well in the explored region (The input current scope for Hodgkin-Huxley neuron model is 0.1 to 1, unit is uA/mm^2).

CHAPTER 5. EXPERIMENTAL RESULTS

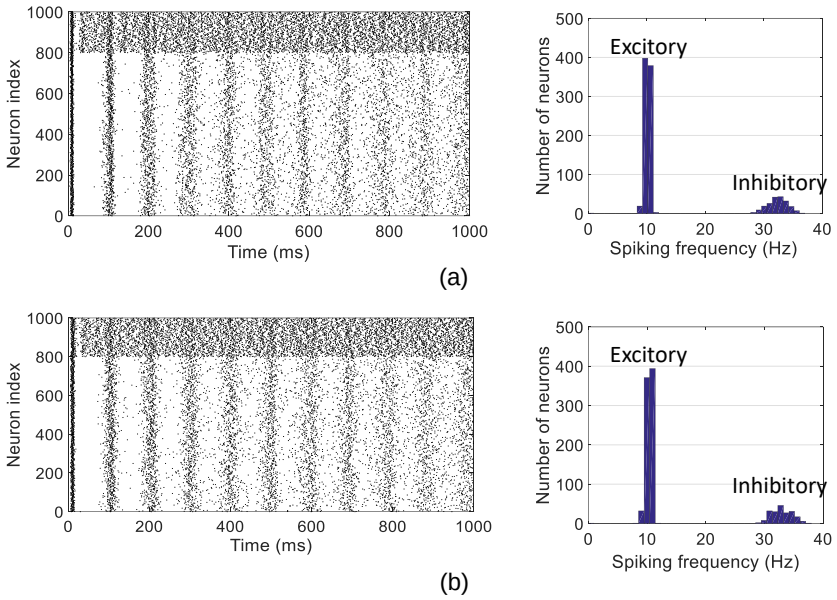
We implement the proposed BNN simulator on three platforms: CPU, GPU, and embedded system, targeting three different application environments. For CPU implementation, we use MATLAB since it is highly optimized for matrix-vector operation which is the main type of computing for BNN simulation. For GPU implementation, we utilize NVIDIA CUDA programming language. For embedded system, we use the popular Raspberry Pi, a single-board micro-computer which promotes Python as the main programming language. Table I summarizes the parameters for these platforms.

5.1 Performance analysis without ROI

First, we compare the accuracy of the proposed BNN simulator (without ROI) with state-of-the-art GPU based BNN simulator, CARLsim [8]. Figure 10 (a, b) shows the raster plots and spiking frequency distributions measured from CARLsim and our work. The network contains 10^3 Izhikevich neurons and 10^4 synapses, excitatory and inhibitory neurons in a 4:1 ratio [1]. Neurons receive uniformly distributed external input current and spikes from the pre-neurons. We observe good match for the spiking pattern and spiking

Table 1. Parameters for CPU, GPU, and Embedded system.

Parameters	CPU	GPU	Embedded system
Name	Intel i7-7700k	NVIDIA GTX 1080TI	Raspberry PI 3 (ARM Cortex-A53)
RAM	32 GB (DDR4)	11 GB (GDDR5X)	1 GB (DDR3)
Maximum power	75 W	250 W	4.8 W
Maximum throughput	320 GFLOPS	11.3 TFLOPS	3.62 GFLOPS
Programming tools	MATLAB	CUDA	Python



Simulator	Brian	Nest	Our work	CARLsim	Our work
Speed (10^3 iterations)	1.89s	1.25s	0.46s	0.26s	0.18s
Platform/model	CPU/LIF			GPU/Izhikevich	

Figure 10. Correctness verification and computing speed comparison. Raster plot and spiking frequency distribution for (a) our work and (b) CARLsim. Insert table lists the speed for different simulators. For CPU and GPU implementation, LIF and Izhikevich model are considered, respectively. Data are measured for BNN containing 10^3 neurons and 10^4 synapses.

frequency distribution. We further compare the computing speed of our simulator with existing simulators: Brian [13], Nest [10], and CARLsim [8]. The first two simulators are CPU based and CARLsim implements GPU version. To ensure the best performance, Data for these simulators are measured from the sample codes provided by the tools developer. As shown in the insert table of Figure 10, our simulator provides the state-of-the-art

performance compared with prior CPU and GPU based implementation (3.4x and 1.4x speedup than other CPU and GPU based simulators, respectively).

5.2 Performance analysis with ROI approximation

We first evaluate the proposed ROI approximation computing algorithm with a randomly connected BNN containing 10^4 neurons (all excitatory) and 10^5 synapses. At the beginning, all neurons follow the Hodgkin-Huxley model. The spiking frequency distribution is plotted in Figure 11(a). Figure 11(b) shows the running time with different model switching threshold (high activity neurons are in Hodgkin-Huxley model, low activity neurons are in LIF model). Data are normalized with the running time when all the neurons are switched to LIF model. The insert figures in Figure 11(b) show the neuron activities with black representing Hodgkin-Huxley and white representing LIF, respectively. Simulation also indicates that the spiking frequency distribution after implementing ROI approximation is

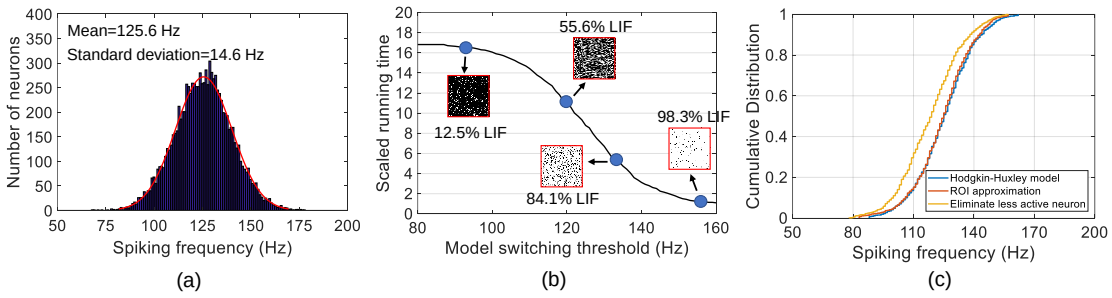


Figure 11. (a) Spiking frequency distribution for all Hodgkin-Huxley scenario. (b) Normalized running time for ROI approximation with different threshold, inserts are the neuron model map with black for Hodgkin-Huxley and white for LIF. (c) Cumulative distribution function (CDF) for all Hodgkin-Huxley, ROI approximation, and eliminating the less active neurons.

almost identical to the Hodgkin-Huxley baseline (blue and red lines in Figure 11(c)). However, if we eliminate the less active neuron (disconnect them from the networks), the distribution changes a lot (yellow line in Figure 11(c)). We conclude that even though the internal dynamics of the unconcerned region is less important, we cannot remove them as they are still critical for signal propagation.

5.3 Accelerate visual cortex modeling with ROI approximation

With the proposed computing algorithms, we implement the visual cortex modeling based on the theories of color vision: Young-Helmholz theory and opponent-process theory [14, 15]. Young-Helmholz theory states that there are three types of cone photoreceptors that are sensitive to short-wavelength (blue light), medium-wavelength (green light), and long-wavelength (red light), respectively. The opponent-process theory states that the cone photoreceptors are linked together to form three opposing color pairs: blue/yellow, red/green, and black/white. Activation of one member of the pair inhibits the other. Figure 12(a) shows the structure of BNN for visual cortex simulation. The neurons in the first layer are based on Young-Helmholz theory and are divided into three groups, sensitive to blue light, green light, and red light, respectively. The second layer contains four groups of opponent-process theory based neurons. The third layer is the output layer, similar with the first layer, sensitive to different colors. The connectivity and neuron numbers are specified in the figure. Here we assume the output layer as ROI region.

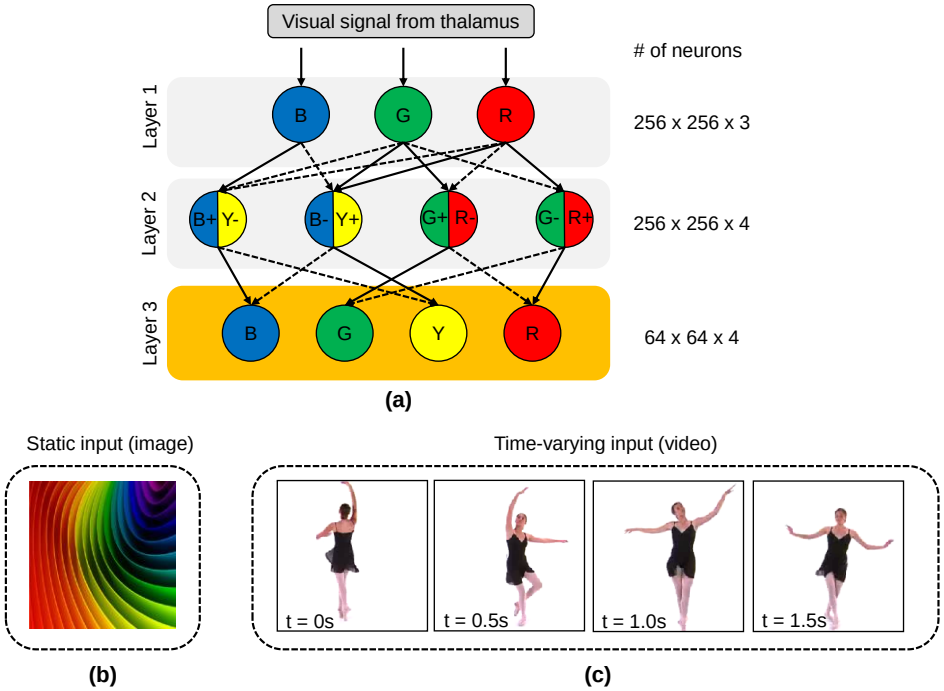


Figure 12. (a) Visual cortex model considering Young-Helmholz theory and opponent-process theory. Solid lines represent synaptic connections from excitatory pre-neurons. Dash lines represent synaptic connections from inhibitory pre-neurons. (b) Static input with an image. (c) Time-varying input with video, here we show 4 frames.

We run simulation with two types of input signals: static input using an image and time-varying input using a video, as shown in Figure 12(b, c). We plot the spiking frequency map for the last group of neurons (sensitive to red color) in the third layer considering the BNN with LIF neuron, Izhikevich neuron, Hodgkin-Huxley neuron, and ROI approximation (Hodgkin-Huxley for ROI and LIF for the rest). The results are shown in Figure 13. Neurons with highest activity are white and neurons with lowest activity are black. Note that we flip the light intensity for the video input to intensify the dancer. All

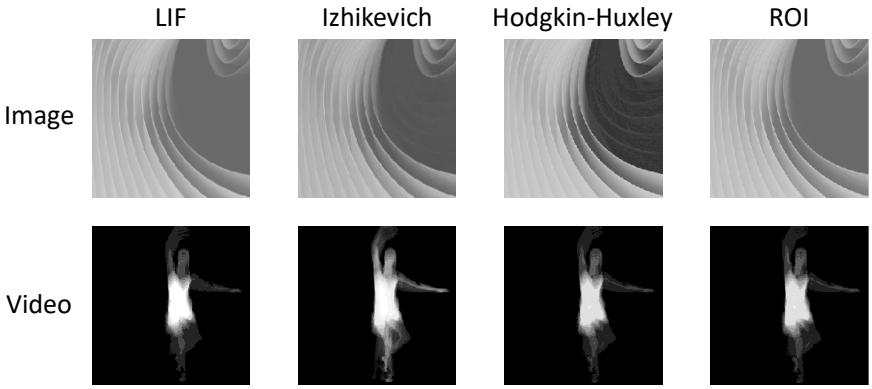


Figure 14. Spiking frequency map for (a) static input with an image; and (b) time-varying input with a video (the spiking frequency is sampled at t=0.5s).

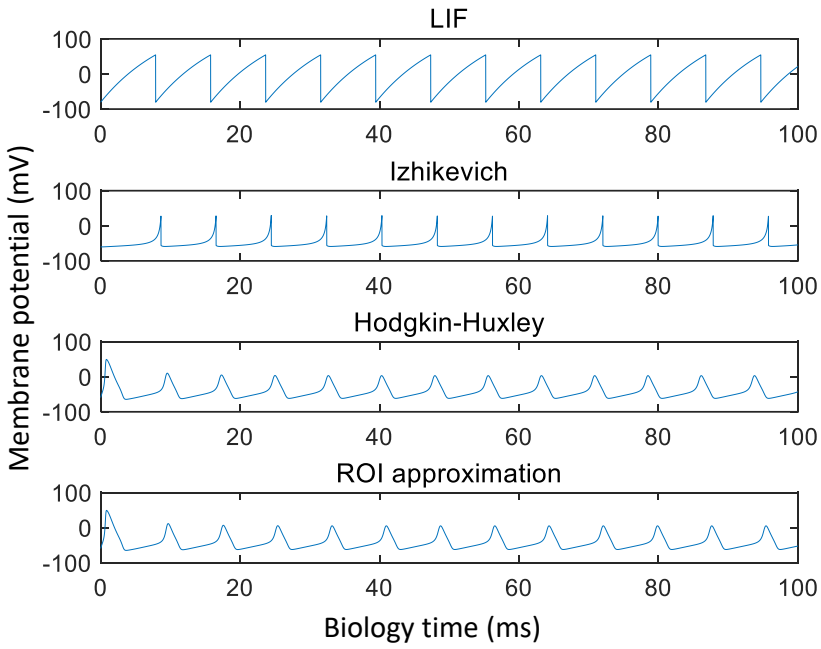


Figure 13. Membrane potential for the neuron in the same location with different neuron models

neuron models and ROI approximation can seasonably represent the color sensitivity. However, the dynamics of individual neuron are very different. We randomly select a

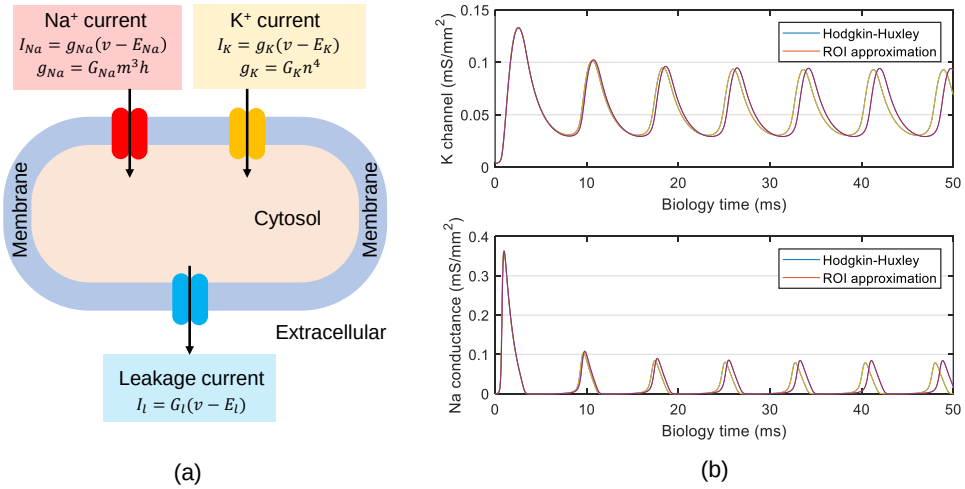


Figure 15. (a) Voltage-gated ion channel and leakage channel for a neuron cell based on Hodgkin-Huxley model. (b) Ion channels conductance simulated from Hodgkin-Huxley and ROI approximation.

neuron in the ROI (3rd layer) and plot the membrane potential (Figure 14). We observe that the ROI based approach matches well with the results from pure Hodgkin-Huxley baseline.

Moreover, the proposed ROI based approximation fully maintains the biology details such as ion channel (Figure 15) conductance in the interested regions. We plot the Na^+ and K^+ channel conductance for Hodgkin-Huxley based simulation and ROI approximation in Figure 15. We observe a phase delay between Hodgkin-Huxley and ROI approximation which is caused by the minor models output mismatch after the parameter tuning. The phase delay can be eliminated by a time shifting.

Finally, we evaluate the computing speed of the proposed algorithms running on different platforms with image and video as BNN input, shown in Figure 16. We observe

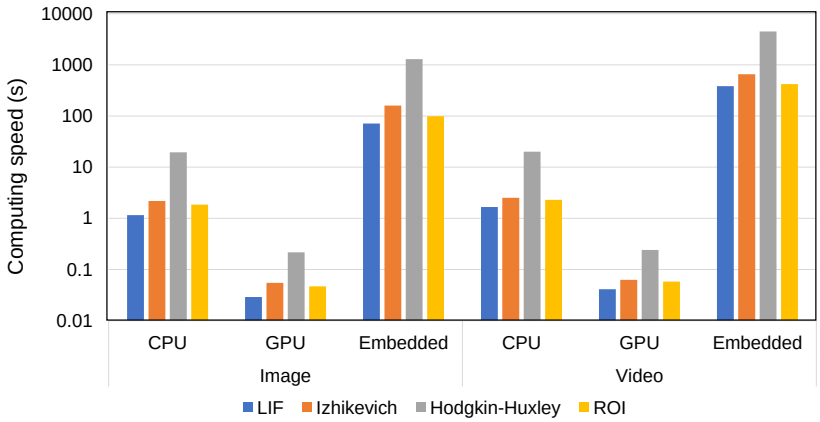


Figure 16. Running time for visual cortex simulation. The time step is 0.01 ms and we run 20 ms (1000 iterations) biology time.

that ROI based approximation can significantly enhance the computing efficiency with an average speed up of 11x over the pure Hodgkin-Huxley baseline. Therefore, in comparison to prior simulators, the template based ROI approximation computing algorithm achieves equivalent speedup of 37x and 16x for CPU (Nest and Brian) and GPU (CARLsim), respectively.

CHAPTER 6. CONCLUSION

This work presents an adaptive BNN simulation methodology that maintains high biological accuracy with more complex neuron models in the ROI while simplifies the neuron models elsewhere to improve the computation speed. A template based computing approach is presented to enable the simulation of heterogeneous BNN with dynamically varying neuron model. The proposed template based computing coupled with ROI approximation demonstrates more than one order of magnitude speedup over existing BNN simulators.

APPENDIX A. NEURON MODELS

For each neuron model, here we present its differential equations and corresponding parameters and variables.

A.1 LIF neuron model

$$\frac{dv(t)}{dt} = -\alpha v(t) + \beta I(t)$$

$$if\ v > 30,\ then\ v = -65$$

α and β are the parameters and $v(t)$ is the variable.

A.2 Izhikevich neuron model

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I$$

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

$$if\ v > 30,\ then \begin{cases} v = c; \\ u = u + d; \end{cases}$$

a, b, c, d are the parameters and v, u are the variables.

A.3 Hodgkin-Huxley neuron model

$$\frac{dv}{dt} = \frac{1}{C_m} \cdot [I - G_{Na} m^3 h(v - E_{Na}) - G_K n^4 (v - E_k) - G_l (v - E_l)]$$

$$\frac{dn}{dt} = \alpha_n(v)(1 - n) - \beta_n(v)n$$

$$\frac{dm}{dt} = \alpha_m(V)(1 - m) - \beta_m(V)m$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h) - \beta_h(V)h$$

$$\alpha_n = \frac{0.01(v + 50)}{1 - e^{-0.1(v+50)}} \qquad \beta_h = \frac{1}{1 + e^{-0.1(v+30)}}$$

$$\alpha_m = \frac{0.1(v+35)}{1-e^{-0.1(V+35)}} \qquad \beta_n = 0.125e^{-0.125(v+60)}$$

APPENDIX B. SYNAPS MODELS

B.1 CUBA and COBA synapse model

Current based (CUBA) synapse model:

$$\tau\frac{dV}{dt} = (V_{rest} - V) + g_{ex}(E_{ex} - V_{rest}) + g_{inh}(E_{inh} - V_{rest})$$

Conductance based (COBA) synapse model:

$$\tau\frac{dV}{dt} = (V_{rest} - V) + g_{ex}(E_{ex} - V) + g_{inh}(E_{inh} - V)$$

Here V_{rest} is the resting membrane potential, E_{ex} and E_{inh} are the reversal potential for excitatory and inhibitory neurons, respectively. g_{ex} and g_{inh} are the synaptic weight. τ is the time constant.

B.2 STP synapse model

When a neuron fires, its post-synapses are increased:

$$g_{ex} = g_{ex} + \Delta g_{ex}$$

$$g_{inh} = g_{inh} + \Delta g_{inh}$$

Otherwise, the value for synapse follows dynamics:

$$\tau_{ex}\frac{dg_{ex}}{dt} = -g_{ex}$$

$$\tau_{inh}\frac{dg_{inh}}{dt} = -g_{inh}$$

Here g_{ex} and g_{inh} are the synaptic weights for excitatory and inhibitory pre-neurons, respectively. Δg_{ex} and Δg_{inh} are the corresponding perturbation value of synaptic weight. τ_{ex} and τ_{inh} are the time constant, a typical value is 10ms.

B.3 LTP synapse model

Long Term Potentiation is modeled with a simplified STDP rules.

$$\Delta g_{ex/inh} = W_{ij}$$

$$\Delta W_{ij} = A_+ \, exp\left(-\frac{x_{ij}}{\tau_+}\right) \quad for \quad x_{ij} > 0$$

$$\Delta W_{ij} = A_{-}\, exp\left(-\frac{x_{ij}}{\tau_{-}}\right) \quad for \quad x_{ij} < 0$$

Here W_{ij} is the synaptic weight from neuron i to neuron j. $A_{+}, A_{-}, \tau_{+}, \tau_{-}$ are the fitting parameters. x_{ij} is the relative time window between pre-synaptic spike arrivals and post-synaptic spikes.

REFERENCES

[1] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag: cortical simulations with 109 neurons, 1013 synapses," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009, pp. 1-12.

[2] G. Indiveri, E. Chicca, and R. J. Douglas, "Artificial Cognitive Systems: From VLSI Networks of Spiking Neurons to Neuromorphic Cognition," *Cognitive Computation*, journal article vol. 1, no. 2, pp. 119-127, June 01 2009.

[3] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, p. 99, 2015.

[4] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668-673, 2014.

[5] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500-544, 1952.

[6] E. M. Izhikevich, "Which model to use for cortical spiking neurons?," *IEEE transactions on neural networks*, vol. 15, no. 5, pp. 1063-1070, 2004.

[7] R. Brette *et al.*, "Simulation of networks of spiking neurons: a review of tools and strategies," *Journal of computational neuroscience*, vol. 23, no. 3, pp. 349-398, 2007.

[8] M. Beyeler, K. D. Carlson, T.-S. Chou, N. Dutt, and J. L. Krichmar, "CARLsim 3: A user-friendly and highly optimized library for the creation of neurobiologically detailed spiking neural networks," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1-8: IEEE.

[9] B. V. Benjamin *et al.*, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699-716, 2014.

[10] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.

[11] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569-1572, 2003.

[12] T. P. Vogels and L. F. Abbott, "Signal propagation and logic gating in networks of integrate-and-fire neurons," *Journal of neuroscience*, vol. 25, no. 46, pp. 10786-10795, 2005.

[13] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[14] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag: cortical simulations with 109 neurons, 1013 synapses," in *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on*, 2009, pp. 1-12: IEEE.

[15] M. L. Hines and N. T. Carnevale, "The NEURON simulation environment," *Neural computation*, vol. 9, no. 6, pp. 1179-1209, 1997.

[16] M. A. Wilson, U. S. Bhalla, J. D. Uhley, and J. M. Bower, "GENESIS: A system for simulating neural networks," 1988.

[17] M.-O. Gewaltig and M. Diesmann, "NEST (neural simulation tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

[18] D. Goodman and R. Brette, "Brian: a simulator for spiking neural networks in Python," 2008.

[19] D. Pecevski, T. Natschläger, and K. Schuch, "PCSIM: a parallel simulation environment for neural circuits fully integrated with Python," ed, 2009.

[20] A. Davison *et al.*, "PyNN: a common interface for neuronal network simulators," 2009.

[21] R. V. Hoang, D. Tanna, L. C. Jayet Bray, S. M. Dascalu, and F. C. Harris Jr, "A novel CPU/GPU simulation environment for large-scale biologically realistic neural modeling," *Frontiers in neuroinformatics*, vol. 7, p. 19, 2013.

[22] K. Minkovich, C. M. Thibeault, M. J. O'Brien, A. Nogin, Y. Cho, and N. Srinivasa, "HRLSim: a high performance spiking neural network simulator for GPGPU clusters," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 2, pp. 316-331, 2014.

[23] N. Srinivasa and J. M. Cruz-Albrecht, "Neuromorphic adaptive plastic scalable electronics: analog learning systems," *IEEE pulse*, vol. 3, no. 1, pp. 51-56, 2012.