Event-Driven Simulation of the Tempotron Spiking Neuron

Bo Zhao[1], Qiang Yu[2], Ruoxi Ding[3], Shoushun Chen[3] and Huajin Tang[1]

[1]Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore
[2]Department of Electrical and Computer Engineering, National University of Singapore
[3]VIRTUS IC Design Centre of Excellence, School of EEE, Nanyang Technological University, Singapore

***Abstract*—This paper presents an event-driven method for the simulation of the tempotron spiking neuron. We propose an efficient dummy-spike-based approach to deal with the two-exponential decay postsynaptic potential kernel. Experimental results show that event-driven simulation runs much faster than conventional time-driven simulation. We also propose a simple but efficient method to solve a same-timestamp problem encountered in event-driven simulation of a simplified tempotron neuron which uses only single-exponential decay.**

I. Introduction

Spiking neural network (SNN), as the third generation of neural network models, provides more biological plausibility by incorporating spikes into its computation. Various models have been proposed in the literature to describe the dynamics of a single spiking neuron [1], such as Leaky Integrate-and-Fire (LIF), Hodgkin-Huxley model, and Izhikevich model. Among these models, LIF has the simplest structure and thus has been widely used. The tempotron [2] is just one of the recently proposed LIF spiking neurons. By using supervised spike timing-based learning [2], [3], the tempotron is able to efficiently discriminate different spatiotemporal spike patterns and thus can be used in categorization tasks [4].

Conventional simulation of the tempotron as well as other SNNs is based on a time-driven approach, in which the simulation is divided into fixed time steps. The whole network is updated for each time step. The size of the time step brings about a trade-off between the precision and the speed of the simulation. A small time step ($\leq 1ms$) is usually required for accurate simulations [5].

The alternative approach for simulating SNNs is the so-called event-driven simulation, in which the simulation clock is advanced by spikes themselves. Instead of evaluating the network at regular time intervals, event-driven simulation only computes a neuron's membrane potential whenever necessary, i.e. when there is an incoming or output spike at this neuron. This approach has been explored in networks of LIF spiking neurons with various decay functions, such as linear decay [6], single-exponential decay [7] and two-exponential decay [5]. The tempotron adopts the two-exponential decay in its postsynaptic potential (PSP) kernel. As will be illustrated in Section II, such a PSP kernel has a fast-rising and slow-decaying shape. The peak of the kernel does not happen at the time of input spikes. This property makes the corresponding event-driven simulation much more challenging than that of linear or single-exponential decay. We propose an efficient dummy-spike-based approach to deal with the two-exponential decay PSP kernel. Experimental results show that event-driven simulation runs much faster than time-driven simulation. We also discuss about a same-timestamp problem encountered in event-driven simulation of a simplified tempotron neuron (which uses only a single-exponential decay). A simple but efficient method is proposed to solve this problem.

The rest of this paper is organized as follows. Section II introduces the dynamics and learning rule of the tempotron spiking neuron. Section III and IV illustrate the conventional time-driven simulation and the proposed event-driven simulation of the tempotron, respectively. Section V discusses about the same-timestamp problem and its solution in event-driven simulation of the simplified tempotron. Section VI shows the experimental results and Section VII concludes this paper.

II. The Tempotron

The tempotron uses the LIF neuron model. A tempotron neuron receives spikes from all its afferent synapses. As shown in Fig. 1, each input spike initiates a PSP to the neuron. For an input spike received at time t_i, the normalized PSP kernel K is defined as:

$$K(t-t_i) = V_0 \times (\exp(\frac{-(t-t_i)}{\tau_m}) - \exp(\frac{-(t-t_i)}{\tau_s})) \quad (1)$$

where τ_m and τ_s denote decay time constants of membrane integration and synaptic currents. For simplicity, τ_s is set to be $\tau_m/4$. V_0 normalizes PSP so that the maximum value of the kernel is 1. Note that $t \geq t_i$ in Equation (1).

The accumulated PSP of the ith afferent synapse is thus:

$$A_i(t) = \sum\nolimits_{t_i<t} K(t-t_i) \quad (2)$$

where t_i is the spike timing from the ith afferent synapse.

The neuron's total membrane potential is the weighted summation of PSPs from all afferent synapses:

$$V(t) = \sum\nolimits_i \omega_i A_i(t) + V_{rest} \quad (3)$$

where ω_i is the weight of the ith afferent synapse. V_{rest} is the resting potential of the neuron.

If the neuron's potential is higher than a specified threshold, the neuron will fire an output spike. After firing, the neuron

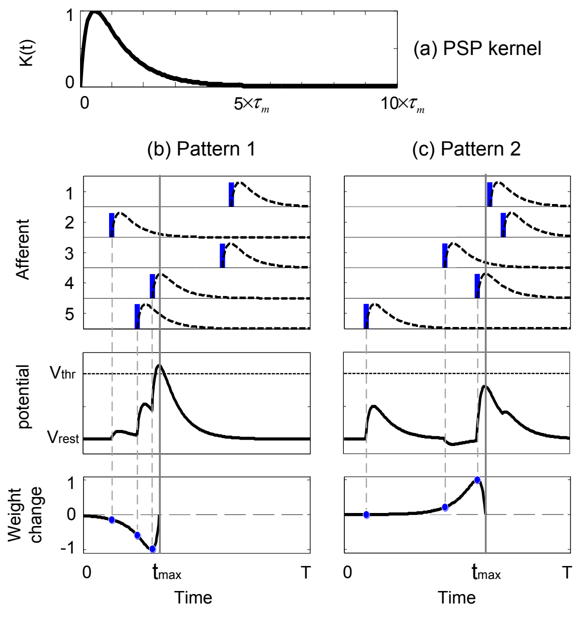

Fig. 1. The dynamics and learning rule of the tempotron spiking neuron. Input spikes are represented as vertical bars.

ignores all the following input spikes within the evaluation time window $[0, T]$. If the membrane potential fails to cross the threshold, then the neuron will not fire.

The tempotron learning rule aims to train the weights of afferent synapses so that the neuron can fire or not according to its target label. If there is an error of the neuron's firing status, the weights will then be modified in the following way: first, we find the peak potential V_{max} during the whole time window $[0, T]$ and label the corresponding timestamp as t_{max}; second, update the weights using gradient descent $\omega_i \leftarrow \omega_i + \Delta\omega_i$. The weight change at the ith afferent synapse is defined as:

$$\Delta\omega_i = \begin{cases} \lambda A_i(t_{\max}), & \text{if fail to fire} \\ -\lambda A_i(t_{\max}), & \text{if fire wrongly} \\ 0, & \text{no error} \end{cases} \quad (4)$$

where λ is the learning rate and $A_i(t_{max})$ is the accumulated PSP of the ith afferent synapse at time t_{max}.

III. Time-Driven Simulation of the Tempotron

In time-driven simulation, the time is advanced by a fixed time step (dt). At each time step, the neuron's membrane potential is computed and then compared with the threshold to check whether there is an output spike or not. The procedure of the time-driven simulation of a tempotron neuron is as follows.

For each time step $t = 0 : dt : T$,

1) Find spikes before current time.
2) Compute PSP for each such spike $K(t - t_i)$.
3) Compute PSP for each afferent $A_i(t)$.
4) Compute the membrane potential $V(t)$.
5) Find V_{max}, t_{max}, and $A_i(t_{max})$ on the fly.
6) Check whether the neuron fires or not.

If it is in the training process, after evaluating the whole time window, we need to compute $\Delta\omega_i$ and update the weight for each afferent synapse. The whole procedure will then be reiterated until the stop conditions are met, e.g. the neuron responds correctly (i.e. fires or not) to input spike pattern(s) or the specified epochs have been reached.

In time-driven simulation, the size of the time step affects the precision and the speed of the simulation. Fig. 2 shows the time-driven simulation of a tempotron neuron on a sample spatiotemporal spike pattern. The time window is set as $T = 200ms$, the membrane time constant is $\tau_m = 20ms$, and 500 afferent synapses are connected to the neuron with each spike per afferent. During the simulations, different time steps are used but the weights are kept the same. One can see that the potential curves vary for different time steps. The firing time and even the firing status can be changed by changing the time step. Accurate simulations usually require a small time step ($\leq 1ms$). However, a smaller time step also leads to a slower simulation speed.

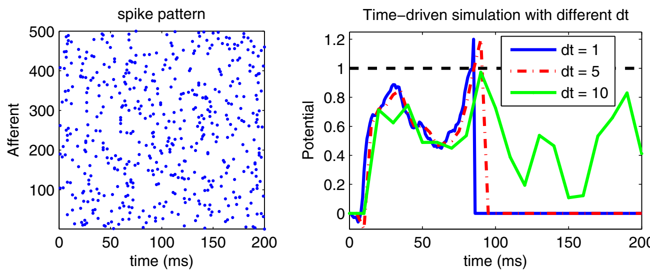

Fig. 2. Time-driven simulation with different time steps.

IV. Proposed Event-Driven Simulation of the Tempotron

In event-driven simulation, there is no fixed time step, instead the time "jumps" from the current-event time to the next-event time. The neuron's potential is updated whenever an input spike comes in. The total computation of a tempotron neuron is linear with respect to the number of input spikes.

The PSP kernel of the tempotron is the difference of two exponential decay functions. As can be seen in Equation (1) and Fig. 3, it is the difference between a slower exponential decay (with a time constant of $\tau_m = 20ms$) and a faster exponential decay (with a time constant of $\tau_s = \tau_m/4$):

$$K_1(\Delta t) = V_0 \times \exp(-\Delta t/\tau_m) \quad (5)$$

$$K_2(\Delta t) = V_0 \times \exp(-\Delta t/\tau_s) \quad (6)$$

where Δt is the time difference between the current time and a past spike time, V_0 is a normalization coefficient that makes the peak value of the final PSP kernel $K = K_1 - K_2$ to be 1.

The total potential of the tempotron neuron is the weighted summation of the PSP kernels from all input spikes. Here the PSP kernel is the difference of two exponential decays. For better illustration, let us first look at the case of a single-exponential decay PSP kernel.

A. Simplified Tempotron using Single-Exponential Decay

The tempotron can be simplified when using a single-exponential decay K_1 as its PSP kernel. Note that the weighted summation of exponentially decaying PSPs is equivalent to

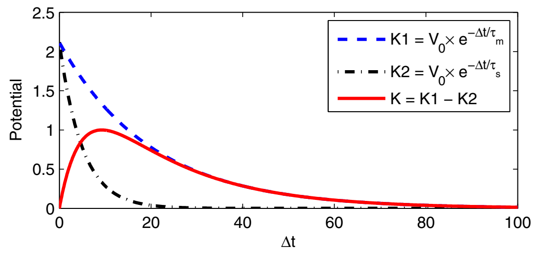

Fig. 3. PSP kernel of the tempotron neuron.

implementing an exponential decay directly on the neuron's membrane potential. Therefore, the event-driven simulation procedure of a simplified tempotron neuron is as follows.

For each input spike:

1) Calculate the time difference Δt between the current time t (i.e. the timestamp of the input spike) and last update time t_{last}. Note that we also need to overwrite last update time using the current time.
2) Refresh the neuron's membrane potential by implementing the exponential decay on it using the equation
$$V_1 \leftarrow V_1 \times \exp(-\Delta t/\tau_m) \tag{7}$$
where V_1 denotes the membrane potential. Note that $\exp(-\Delta t/\tau_m)$ can be pre-calculated and stored in a lookup table (LUT) to speed up the simulation.
3) Add $\omega_i \times V_0$ to the neuron's membrane potential.
$$V_1 \leftarrow V_1 + \omega_i \times V_0 \tag{8}$$
where ω_i is the weight associated with the afferent synapse from which the input spike comes. V_0 is the normalization factor in Equation (5). The increment of V_1 is caused by the input spike.
4) Find V_{max} and corresponding t_{max} on the fly. Check whether the neuron fires an output spike or not by comparing V_1 with the threshold. If it fires, V_1 is reset to 0, and the following spikes are shunted.

After the last input spike is processed, the simulation of one epoch finishes. If it is in the training process and there is an error of the neuron's firing status, the weights need to be changed according to the following procedure:

- Find spikes before t_{max}.
- Compute the PSP for each such spike: $K_1(t_{max} - t_i)$.
- Compute the PSP for each afferent:
$$A1_i(t_{max}) = \sum\nolimits_{t_i<t_{max}} K_1(t_{max} - t_i) \tag{9}$$
- Update weights using Eq. (4) by replacing A_i with $A1_i$.

B. The Tempotron using Two-Exponential Decay

Getting back to the original tempotron which uses two-exponential decay as its PSP kernel. Let t and t_{last} denote the current time and last update time, respectively. Let V_1 and V_2 denote the membrane potential that is corresponding to PSP kernel K_1 and K_2, respectively. The procedure of the event-driven simulation of the tempotron neuron (with two-exponential decay) is as follows.

For each input spike:

1) Compute time difference $\Delta t \leftarrow t - t_{last}$ and $t_{last} \leftarrow t$
2) Compute V_1 exp decay: $V_1 \leftarrow V_1 \times \exp(-\Delta t/\tau_m)$
3) Compute V_1 increment: $V_1 \leftarrow V_1 + \omega_i \times V_0$
4) Compute V_2 exp decay: $V_2 \leftarrow V_2 \times \exp(-\Delta t/\tau_s)$
5) Compute V_2 increment: $V_2 \leftarrow V_2 + \omega_i \times V_0$
6) Compute the final potential: $V = V_1 - V_2$
7) Find V_{max} and corresponding t_{max} on the fly. Check whether the neuron fires or not. If it fires, reset V_1 and V_2, and shunts all the following spikes.

Note that the peak of the final PSP kernel K does not happen at the time of input spike; instead it has a delay from the input spike timing. This can be seen from Fig. 3. The PSP peak does not happen at $\Delta t = 0$. We found that it roughly happens at $\Delta t \approx 0.462 \times \tau_m$ (for $\tau_s = \tau_m/4$). To get the correct output (firing or not), we need to check the firing status at the PSP peak time in addition to the input spike time. This can be easily done by generating a dummy spike for each input spike. The dummy spike has a timestamp of $0.462 \times \tau_m + t$, where t denotes the input spike timing. For the dummy spike, the computation does not involve the increment of V_1 and V_2, i.e. step 3) and 5).

During training, the weight change procedure is as follows.

- Find spikes before t_{max}.
- Compute the PSP for each such spike: $K_1(\Delta t_{max})$ and $K_2(\Delta t_{max})$, where $\Delta t_{max} = t_{max} - t_i$.
- Compute the PSP for each afferent:
$$A_i(t_{\max}) = \sum_{t_i<t_{\max}} [K_1(\Delta t_{max}) - K_2(\Delta t_{max})] \tag{10}$$
- Update weights using Eq. (4).

Note that both $\exp(-\Delta t/\tau_m)$ and $\exp(-\Delta t/\tau_s)$ can be pre-calculated and stored in two separate LUTs to speed up the simulation.

V. Same-Timestamp Problem in Simplified Tempotron

The two-exponential decay PSP kernel used in the tempotron neuron models the transmission delay from a presynaptic spike to the postsynaptic potential. Considering the transmission delay makes the neuron model more biologically plausible, however, it also results in more computation. In many practical engineering problems, single-exponential decay is widely used in order to simplify the neuron model and reduce the computation. For the simplified tempotron with single-exponential decay PSP kernel, its event-driven simulation has been illustrated in Section IV-A; this section will discuss about a potential problem caused by input spikes of the same timestamps in its event-driven simulation.

As shown in Fig. 4(a), a simplified tempotron neuron receives three input spikes (ABC), among which spike B and spike C have the same timestamp. The synaptic weights associated with these three spikes are 1, 1, and -0.5, respectively. The middle figure shows the time-driven simulation with a small time step. The neuron does not fire for these input spikes.

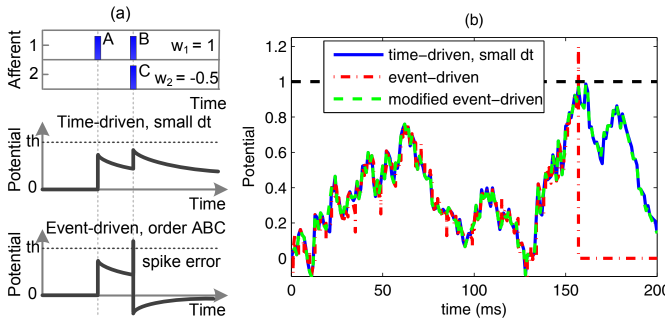

Fig. 4. Event-driven simulation using a single-exponential decay.

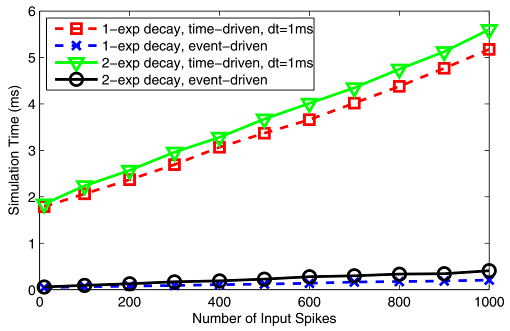

Fig. 5. Speed comparison of event-driven and time-driven simulation.

However, in the event-driven simulation (lower figure), the neuron fires an error spike. This is because that spikes are processed one by one in the event-driven simulation. Spike B is processed before spike C. The potential crosses the threshold after incorporating the sole impact of spike B. However, the correct potential at that timing should be the combination of the impact of spike B and C.

In order to solve the problem caused by same timestamps, we modified the step 4) of the procedure in Section IV-A. The firing check is only performed for last (past) spike whenever the time different between current-spike time and last-spike time is larger than 0. In this way, the firing check will be skipped for spike B and the error spike can thus be avoided. Fig. 4(b) shows the simulation of a simplified tempotron neuron on a sample spatiotemporal spike pattern. The event-driven simulation before modification causes an error spike, whereas the modified version provides a result that is consistent with the small-time-step time-driven simulation.

VI. Experimental Results

To compare the speed of event-driven and time-driven simulation, we evaluated a tempotron neuron using some randomly generated spatiotemporal spike patterns. The time window of the tempotron is set as $T = 200ms$, the membrane time constant is $\tau_m = 20ms$. Both single-exponential and two-exponential decay have been evaluated. We also tried different number of input spikes. The simulation running time per spike pattern is shown in Fig. 5. One can see that event-driven simulation runs much faster than time-driven simulation for both single-exponential and two-exponential decay.

We further evaluated a network of tempotron neurons on the MNIST dataset [8]. There are 10 categories of handwritten digits (0-9) in MNIST. We use the HMAX algorithm [9] to extract features from MNIST images. Each image is represented by 1000 features. The features are then converted into spike timings by a reverse linear mapping. The larger the feature, the smaller the spike timing. The generated feature spike patterns are then fed to a network of tempotron neurons for classification. We used 10 groups of tempotron neurons, with one group for each category. Each group has 5 neurons. The classification decision is made by a majority voting scheme: one just needs to check which group has the largest number of fired neurons [10]. We selected 500 images from the MNIST dataset and randomly split them into a training set (80%) and a test set. The tempotron network was trained for 10 epochs. The experimental results are listed in Table I. As we can see, event-driven simulations provide similar accuracy as time-driven approaches, however, the running time of event-driven simulation is only around 2% of that of time-driven simulation in this evaluation.

TABLE I
Evaluation of the Tempotron Neurons on MNIST Dataset

Methods	Total Time (s)		Accuracy (%)	
	Train	Test	Train	Test
Time-driven, 1-exp decay	1423	35.6	100	83.9 ± 1.66
Time-driven, 2-exp decay	1729	43.2	100	84.1 ± 1.20
Event-driven, 1-exp decay	19.6	0.46	100	85.4 ± 2.39
Event-driven, 2-exp decay	38.8	0.89	100	84.2 ± 1.69

VII. Conclusion

This paper presents an event-driven method for the tempotron simulation. An efficient dummy-spike-based approach is proposed to deal with the two-exponential decay PSP kernel. We also solved the same-timestamp problem in the case of single-exponential decay. Compared to time-driven simulation, event-driven simulation provides a much faster speed without sacrificing the classification performance.

References

[1] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, 1st ed. Cambridge University Press, Aug. 2002.
[2] R. Gutig and H. Sompolinsky, "The tempotron: a neuron that learns spike timing-based decisions," *Nature Neurosci.*, 2006.
[3] Q. Yu, H. Tang, K. C. Tan, and H. Li, "Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns," *PLoS ONE*, vol. 8, no. 11, p. e78318, Nov 2013.
[4] B. Zhao, Q. Yu, H. Yu, S. Chen, and H. Tang, "A bio-inspired feedforward system for categorization of AER motion events," in *IEEE BioCAS*, Oct 2013, pp. 9–12.
[5] M. D'Haene *et al.*, "Accelerating event based simulation for multi-synapse spiking neural networks," in *ICANN*, 2006, pp. 760–769.
[6] L. A. Camunas-Mesa *et al.*, "An event-driven multi-kernel convolution processor module for event-driven vision sensors." *IEEE J. Solid-State Circuits*, vol. 47, no. 2, pp. 504–517, 2012.
[7] E. Ros *et al.*, "Event-driven simulation scheme for spiking neural networks using lookup tables to characterize neuronal dynamics," *Neural Comput.*, vol. 18, no. 12, pp. 2959–2993, Dec 2006.
[8] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.
[9] T. Serre *et al.*, "Robust object recognition with cortex-like mechanisms," *IEEE TPAMI*, vol. 29, no. 3, pp. 411–426, 2007.
[10] Q. Yu, H. Tang, K. C. Tan, and H. Li, "Rapid feedforward computation by temporal encoding and learning with spiking neurons," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 24, pp. 1539–1552, Oct. 2013.