Spiking Neural Networks on Serial CPU

Marc DeCarlo, Nagarajan Kandasamy

***Abstract*—The goal is to bring the event-driven paradigm of a Spiking Neural Network (SNN) to embed it on a serial CPU with minimal energy and performance loss compared to a neuromorphic hardware approach.**

I. Introduction

THIS paper addresses the challenge of maintaining decay accuracy in event-driven spiking neural network simulations. Traditional synchronous approaches struggle with computational inefficiency, while naive asynchronous methods often sacrifice accuracy due to unaccounted delays in neuron updates. This work presents a method to achieve high accuracy while preserving the event-driven nature of simulations, critical for applications requiring real-time responses.

The method ensures that each neuron's state is updated accurately by considering its last spike time and its interspike activity, leveraging an event queue for spike handling and dynamic decay calculations.

II. Neuron Dynamics

The dynamics of a Leaky Integrate-and-Fire (LIF) neuron are given by:

$$\frac{dV(t)}{dt} = \begin{cases} \frac{-[V(t)-V_{\text{rest}}]+R_m I(t)}{\tau_m} & \text{if } V(t) < V_{\text{th}} \\ 0 & \text{if } V(t) \geq V_{\text{th}} \end{cases} \quad (1)$$

Here, $V(t)$ is the membrane potential, V_{rest} is the resting potential, R_m is the membrane resistance, $I(t)$ is the input current, τ_m is the membrane time constant, and V_{th} is the firing threshold.

A. Key Challenges

In an event-driven framework, accurately maintaining the decay function requires:

- Efficient calculation of decay using Δt (time elapsed since the last update).
- A mechanism to service neurons waiting for update due to computational delays without affecting accuracy.

III. Problem Overview

The problem involves simulating a spiking neural network in an event-driven manner, balancing: - **Accuracy**: Ensuring that the decay and spike timings are computed close to event-driven equivalence. - **Efficiency**: Minimizing unnecessary computations for inactive neurons and monitoring of the system.

Revised January 7, 2025

This paper proposes a hybrid approach that combines event queues with dynamic state recalculation for idle neurons. This method accurately updates neurons based on the exact elapsed time Δt, derived as follows:

A. Total Simulation Time and Δt

The **total simulation time** (T_{total}) is determined by:

- Input duration: T_{input} (e.g., 100 ms for a presented stimulus).
- Post-stimulus observation: T_{post} (e.g., 20 ms).

$$T_{\text{total}} = T_{\text{input}} + T_{\text{post}}$$

The **delta time** (Δt) is the granularity of updates and is chosen based on:

- Decay time constant τ: $\Delta t \leq \frac{\tau}{10}$. (random rule needs to be experimentally derived)

B. Keeping Track of Simulation Time

Simulation time is tracked using a global clock, T_{sim}, which advances in discrete steps based on events. Key components include:

- **Event Queue:** Each event in the queue is timestamped, ensuring chronological processing.
- **Time Increment:** After processing an event, T_{sim} is updated to the timestamp of the next event.

Neurons that remain idle between events use T_{sim} to calculate the elapsed time (Δt) since their last update, ensuring accurate decay calculations.

C. Dynamic Δt Adjustment

Dynamic adjustment of Δt improves efficiency by tailoring the update frequency to the activity level of neurons:

- **Active Neurons:** For neurons involved in frequent spiking or events, Δt is minimized to maintain high temporal resolution.
- **Idle Neurons:** For neurons with no incoming spikes, Δt is extended, reducing computational overhead.

This adjustment is governed by:

$$\Delta t = \min(\Delta t_{\text{max}}, T_{\text{next}} - T_{\text{last}}),$$

where Δt_{max} is a predefined upper limit, T_{next} is the time of the next event, and T_{last} is the neuron's last update time.

D. Accounting for Runtime and Neuron Update Delays

In an event-driven system, the runtime per update is influenced by the number of active events and their interdependencies. The computational efficiency is optimized by:

1. **Dynamic Delta Time Adjustment:** Adjusting Δt based on network activity ensures a balance between computational load and accuracy. 2. **Latency Management:** Each neuron's Δt is recalculated based on its last update time and the current simulation time, ensuring accurate decay computations even with delayed updates.

IV. Implementation Outline

A. Initialization

- Set simulation parameters: T_{total}, Δt, neuron properties (V_{rest}, V_{th}, τ_m).
- Initialize event queue and neuron states.

B. Event-Driven Simulation Loop

1) **Process Events:**
   - Dequeue next event.
   - Update neuron's membrane potential $V(t)$ using:

$$V(t) = V_{\text{rest}} + (V_{\text{prev}} - V_{\text{rest}})e^{-\Delta t/\tau_m}$$

2) **Handle Spikes:**
   - If $V(t) \geq V_{\text{th}}$, emit a spike and reset $V(t)$ to V_{rest}.
   - Enqueue connected neurons with spike propagation delay.
3) **Update Idle Neurons:**
   - Recalculate decay for neurons without recent spikes based on elapsed Δt.

Acknowledgment

Thank you

References

[1]