Adapting unconstrained spiking neural networks to explore the effects of time discretization on network properties

Correlation between step size and accuracy for real world task

Zhejia Hu[1]
Supervisor(s): Nergis Tömen[1]**, Aurora Micheli**[1]
[1]EEMCS, Delft University of Technology, The Netherlands

Name of the student: Zhejia Hu
Final project course: CSE3000 Research Project
Thesis committee: Nergis Tömen, Aurora Micheli, Lilika Markatou

Abstract

Spiking Neural Networks (SNN) represent a distinct class of neural network models that incorporate an additional temporal dimension. Neurons within SNN operate according to the Leaky Integrate-and-Fire principle, governed by ordinary differential equations. Inter-layer neuronal communication occurs through spike propagation when membrane potentials reach a specified threshold. This unique mechanism renders conventional Artificial Neural Network (ANN) design principles and learning rules inapplicable to SNN. This study presents a theoretical investigation into the impact of time discretization on SNN performance in real-world tasks. We present a network architecture based on the Error Backpropagation Through Spikes model. This approach allows for the transformation of the original system of differential equations into a system of difference equations across multiple time steps. We evaluate our model's performance on the MNIST dataset and identify several phenomena that influence accuracy. Our analysis primarily focuses on gradient dynamics and spectral properties of the voltage signals. These findings contribute to a deeper understanding of SNN behavior and potential optimization strategies.

1 Introduction

Spiking Neural Networks (SNN) have emerged as a compelling biologically-inspired paradigm, offering a sophisticated alternative to traditional Artificial Neural Networks (ANN). These networks incorporate a temporal dimension, thereby enhancing their biological fidelity by modeling the time-dependent dynamics of neuronal activity [21]. In contrast to ANN, where neuronal output is a static function of aggregated inputs, SNN emulates the complex temporal behavior of biological neurons, allowing for the evolution of physical quantities over time, as exemplified by the Leaky Integrate-and-Fire (LIF) model [6].

However, the implementation of SNN in clock-driven simulations necessitates the discretization of time, a requirement that introduces non-trivial inaccuracies in modeling individual neuron dynamics. These inaccuracies stem from numerical approximations, such as the non-differentiable impulse function, and the inevitable information loss due to temporal down-sampling. While quantization techniques have been extensively studied in the context of ANN [10], the impact of time-discretization on SNN presents unique challenges that remain insufficiently explored in the literature. Consequently, the direct application of ANN quantization techniques to discretize timestamps in SNN appears to be a non-trivial and potentially suboptimal approach.

This research endeavor aims to address the following question: **How does the granularity of time discretization in the Error Backpropagation Through Spikes (BATS) model [1] correlate with performance accuracy on real-world tasks**? Our investigation adopts a dual approach, combining theoretical analysis and empirical experimentation. From a theoretical perspective, we consider neurons based on the LIF model as distinct mathematical objects, rigorously examining how discretization affects their properties. This analysis is motivated by the extensive body of research on the divergent behaviors of continuous and discrete mathematical constructs. Subsequently, we propose a network architecture that extends the BATS model to support configurable multi-timestamp operations. This architecture synergistically combines elements from Recurrent Neural Networks and Spiking Neural Networks. Finally, we conduct a series of experiments using this proposed architecture to empirically verify whether the theoretically predicted mathematical phenomena manifest in practical implementations.

2 Background

This section presents a comprehensive theoretical analysis of individual neurons and the spiking neural network as a whole. In Section 2.1, we derive the closed-form solution for the Leaky Integrate-and-Fire System of Ordinary Differential Equations, providing a foundation for subsequent analyses and network implementation. Section 2.2 examines the implications of discretizing the original model from a frequency domain perspective, while Section 2.3 focuses on the impact of discretization on numerical accuracy. Finally, Section 2.4 investigates the collective behavior of neurons within the network, a crucial aspect in the design of hidden layers where neuronal behavior tends to exhibit convergence. This multi-faceted theoretical approach aims to provide a robust framework for understanding the intricate dynamics of spiking neural networks under discrete-time conditions.

2.1 Leaky Integrate-And-Fire Model

The Leaky Integrate-and-Fire (LIF) model, as its name suggests, reflects the integration process of neuronal dynamics in a network [8]. This model is defined by a first-order linear differential equation over time, which describes the evolution of the *membrane potential*, denoted as u. The membrane potential dynamics are influenced by the input current, i, and are parameterized by the membrane resistance, R, and the membrane capacitance, C. These parameters collectively determine the membrane time constant, τ_m, which is the product of resistance and capacitance ($\tau_m \;:=\; R\,C$)

$$\tau_m \,\frac{\mathrm{d}u(t)}{\mathrm{d}t} \;+\; u(t) \;=\; R\, i(t) \tag{1}$$

The neuronal dynamics can be accurately modeled as a linear time-invariant (LTI) system, whose transfer function, $G(s)$, can be analytically derived by employing the Laplace transform, a powerful mathematical tool for analyzing LTI systems in the complex frequency domain, commonly referred to as the s-domain. By defining the Laplace transforms of the input and output variables as $I(s) = \mathcal{L}\left\{i(t)\right\}$ and $U(s) = \mathcal{L}\left\{u(t)\right\}$, respectively, the original differential equation governing the neuronal dynamics can be transformed into an algebraic equation in the s-domain. This transformation facilitates the analytical derivation of the transfer function, $G(s)$, which is defined as the ratio of the output transform, $U(s)$, to the input transform, $I(s)$, i.e., $G(s) = \frac{U(s)}{I(s)}$

$$G(s) \;=\; \frac{R}{\tau_m\, s + 1} \tag{2}$$

If the current i is constant over time, i.e., $i \;\equiv\; i_0$, the differential equation describing the membrane voltage dynamics possesses a closed-form solution. This solution illustrates that the membrane voltage $u(t)$ converges exponentially to a steady-state value u_∞, which is determined by the input current i_0 and the membrane resistance R, such that $u_\infty \;:=\; i_0\, R$. The convergence of $u(t)$ to u_∞ occurs at an exponential rate governed by the membrane time constant τ_m

$$\left(u(t) \;-\; u_\infty\right) \;=\; \left(u_0 \;-\; u_\infty\right)\,\exp\left(-\frac{t}{\tau_m}\right) \tag{3}$$

Where $u_0 \;:=\; u(0)$ is the initial membrane voltage at time $t \;=\; 0$. The term $\exp\left(-\frac{t}{\tau_m}\right)$ decays exponentially with time, and the rate of decay is determined by the membrane time constant τ_m. This implies that the convergence of $u(t)$ to u_∞ occurs in $O\left(\exp\left(-\frac{t}{\tau_m}\right)\right)$ time.

In the context of spiking neural networks, the dynamics of the incoming synaptic current for a neuron with indices (l, k) are governed by the weighted sum of the presynaptic spike trains impinging upon its dendritic tree. Specifically, the change in the synaptic current $\frac{\mathrm{d}i^{(l,k)}(t)}{\mathrm{d}t}$ at time t is determined by the dot product between the vector of learnable synaptic weights $\mathbf{w}^{(l,k)}$ and the vector of presynaptic spike trains $\mathbf{s}^{(l-1)}(t)$, which encapsulates the spiking activity of all neurons projecting onto the neuron (l, k) [16]

$$\tau_s\,\frac{\mathrm{d}i^{(l,k)}(t)}{\mathrm{d}t} \;+\; i^{(l,k)}(t) \;=\; \mathbf{w}^{(l,k)} \;\cdot\; \mathbf{s}^{(l-1)}(t) \tag{4}$$

When we decompose the dot product $\mathbf{w}^{(l,k)} \;\cdot\; \mathbf{s}^{(l-1)}(t) \;=\; \sum_n \mathbf{w}_n^{(l,k)}\;\delta\left(t \;-\; t_n\right)$ and apply the Laplace transform, it becomes apparent that the learned synaptic weights, $\mathbf{w}^{(l,k)}$, exert an influence on the phase distribution. This influence can be attributed to the characteristic of the delayed Laplace transform of the impulse function, which is expressed as $\mathcal{L}\left\{\delta\left(t \;-\; t_n\right)\right\} \;=\; \exp\left(-t_n\, s\right)$. Consequently, for the neuron with index (l, k), under the assumption that the neurons at the preceding layers have been activated, we can derive the closed-form solution of the current in the s-domain

$$I^{(l,k)}(s) \;=\; \frac{\tau_s\, i\,(0) \;+\; \sum_n \mathbf{w}_n^{(l,k)}\; \exp\,(-t_n\, s)}{\tau_s\, s \;+\; 1} \tag{5}$$

Substituting the input signal $I(s)$ into $U(s) \;=\; \frac{R\, I(s)}{\tau_m\; s\,+\,1}$, which characterizes the neuron's dynamics, yields:

$$U(s) \;=\; \frac{\tau_m\, u\,(0)}{\tau_m\, s \;+\; 1} \;+\; R\, \left[\frac{\tau_s\, i\,(0)}{(\tau_m\, s \;+\; 1)\;(\tau_s\, s \;+\; 1)} \;+\; \frac{\sum_n \mathbf{w}_n^{(l,k)}\; \exp(-t_n\, s)}{(\tau_m\, s \;+\; 1)\;(\tau_s\, s \;+\; 1)}\right]$$

Subsequently, applying the inverse Laplace transform facilitates the analysis of the neuron's temporal response $u(t)$ to the incoming spike train

$$u^{(l,k)}(t) \;=\; u\,(0)\; \exp\left(-\frac{t}{\tau_m}\right) \tag{6}$$

$$+\; R\, i\,(0)\; \frac{\tau_s}{\tau_m \;-\; \tau_s}\; \left(\exp\left(-\frac{t}{\tau_m}\right) \;-\; \exp\left(-\frac{t}{\tau_s}\right)\right) \tag{7}$$

$$+\; R \sum_n \left[\frac{\mathbf{w}_n^{(l,k)}}{\tau_m \;-\; \tau_s}\; \left(\exp\left(-\frac{t \;-\; t_n}{\tau_m}\right) \;-\; \exp\left(-\frac{t \;-\; t_n}{\tau_s}\right)\right)\right] \tag{8}$$

2.2 Time Discretization: Signal Processing Perspective

In SNN, each neuron can be modeled as a second-order continuous-time system. The input to this system is the spike train generated by all presynaptic neurons, denoted as $b\,(t)$, while the output is the membrane potential, represented as $u\,(t)$. This relationship is governed by a second-order differential equation, which characterizes the dynamic behavior of the neuron's membrane potential in response to incoming spikes

$$\frac{\mathrm{d}^2 u}{\mathrm{d}t^2} \;+\; \left(\frac{1}{\tau_s} \;+\; \frac{1}{\tau_m}\right)\, \frac{\mathrm{d}u}{\mathrm{d}t} \;+\; \frac{1}{\tau_m\, \tau_s}\, u \;=\; \frac{R}{\tau_m\, \tau_s}\, b \tag{9}$$

From the equation presented, the corresponding *damping ratio* ζ and the *undamped natural frequency* ω_n of the system can be calculated [18]

$$\begin{cases} \zeta \;=\; \frac{\tau_m \;+\; \tau_s}{2\,\sqrt{\tau_m\; \tau_s}} \\ \omega_n \;=\; \frac{1}{\sqrt{\tau_m\; \tau_s}} \end{cases} \tag{10}$$

When $\tau_m \;\neq\; \tau_s$, it can be observed that $\zeta \;>\; 1$. This indicates that the system is *overdamped* with two distinct real negative poles at $-\frac{1}{\tau_m}$ and $-\frac{1}{\tau_s}$. Furthermore, as the value of ζ increases, the system's decay to zero is prolonged. The frequency response $H\,(\mathrm{j}\Omega)$ can be subsequently evaluated, revealing that the system functions as a low-pass filter

$$H\,(\mathrm{j}\Omega) \;=\; \frac{R}{(\tau_m\, \mathrm{j}\Omega \;+\; 1)\;(\tau_s\, \mathrm{j}\Omega \;+\; 1)} \tag{11}$$

An alternative approach to discretizing the original ODE involves sampling and scaling the original voltage signal: $u\,[n] \;:=\; \Delta t\; u\,(n\; \Delta t)$, where the scaling factor preserves the signal's energy. This process employs a sampling frequency $\Omega_s \;:=\; \frac{2\,\pi}{\Delta t}$. Under the assumption that aliasing effects from high frequencies are negligible, this sampling procedure can be represented as a mapping from the s-domain to the z-domain via the transformation $z \;:=\; \exp\,(s\; \Delta t)$ [17]

$$H_d\,(z) \;=\; \frac{R\, \Delta t}{\tau_m \;-\; \tau_s}\; \left[\frac{1}{1 \;-\; \exp\left(-\frac{\Delta t}{\tau_m}\right)\; z^{-1}} \;-\; \frac{1}{1 \;-\; \exp\left(-\frac{\Delta t}{\tau_s}\right)\; z^{-1}}\right] \tag{12}$$

The region of convergence for this transformation is defined by $|z| \;>\; \exp\left(-\frac{\Delta t}{\tau_m}\right)$. This formulation facilitates the derivation of the Discrete-Time Fourier Transform of the discrete time signal

$u\,[n]$, noting that $z \,=\, \exp\left(s\,\Delta t\right) \,=\, \exp\left(\mathrm{j}\Omega\,\Delta t\right) \,=\, \exp\left(\mathrm{j}\omega\right)$. For $\omega \,\in\, [-\pi,\ \pi]$, the corresponding sampled continuous time frequency range is $\Omega \,\in\, \left[-\frac{\pi}{\Delta t},\ \frac{\pi}{\Delta t}\right]$. Frequencies beyond this range induce aliasing effects. To map the entire jΩ-axis onto the unit circle in the z-plane, the Bilinear Transformation can be employed. This transformation introduces a non-linear warping of the frequency axis, expressed as $\omega \,=\, 2\,\arctan\left(\frac{\Omega\,\Delta t}{2}\right)$ (in contrast to the linear relationship $\omega \,=\, \Omega\,\Delta t$), while preserving the system's causality [17].

2.3 Time Discretization: Numerical Methods Perspective

The process of time discretization involves the sampling of the original equations, specifically equations (1) and (4), at discrete time instances denoted by $\{t_k\}$. This is achieved by employing difference equations to approximate the original differential equations, which are integral to our neuron models in the spiking neural network. In this study, we introduce a range of widely-used discretization techniques.

Forward Euler Method is a first-order explicit integration scheme with a local truncation error of $\mathcal{O}\left(\Delta^2 t\right)$: $\frac{\mathrm{d}f(t_k)}{\mathrm{d}t} \,\approx\, \frac{f(t_k\,+\,\Delta t)-f(t_k)}{\Delta t}$. The corresponding z-transform is $\frac{z\,-\,1}{\Delta t}$. Considering the state of an arbitrary neuron in the network as a vector comprising voltage and current, the application of the Forward Euler Method yields the following state transition function

$$\begin{bmatrix} u\,(t_k+\Delta t) \\ i\,(t_k+\Delta t) \end{bmatrix} \,=\, \begin{bmatrix} 1\,-\,\frac{\Delta t}{\tau_m} & \frac{R\,\Delta t}{\tau_m} \\ 0 & 1\,-\,\frac{\Delta t}{\tau_s} \end{bmatrix} \begin{bmatrix} u\,(t_k) \\ i\,(t_k) \end{bmatrix} \,+\, \frac{\Delta t}{\tau_s} \begin{bmatrix} 0 \\ \mathbf{w}\,\cdot\,\mathbf{s}\,(t_k) \end{bmatrix} \tag{13}$$

To ensure the stability of the numerical integration scheme as the time step t_k approaches infinity, the step size Δt must adhere to the following condition: $\Delta t \,<\, 2\,\min\left\{\tau_m,\ \tau_s\right\}$.

Backward Euler Method is a first-order implicit integration scheme with the same truncation error of $\mathcal{O}\left(\Delta^2 t\right)$: $\frac{\mathrm{d}f(t_k)}{\mathrm{d}t} \approx \frac{f(t_k)-f(t_k-\Delta t)}{\Delta t}$ (corresponding z-transform is $\frac{z\,-\,1}{z\,\Delta t}$). but inherently numerical stable. For the neuron system, it could be derived that

$$\begin{bmatrix} u\,(t_k) \\ i\,(t_k) \end{bmatrix} \,=\, \begin{bmatrix} 1\,+\,\frac{\Delta t}{\tau_m} & -\frac{R\,\Delta t}{\tau_m} \\ 0 & 1\,+\,\frac{\Delta t}{\tau_s} \end{bmatrix}^{-1} \left(\begin{bmatrix} u\,(t_k-\Delta t) \\ i\,(t_k-\Delta t) \end{bmatrix} \,+\, \frac{\Delta t}{\tau_s} \begin{bmatrix} 0 \\ \mathbf{w}\,\cdot\,\mathbf{s}\,(t_k) \end{bmatrix} \right) \tag{14}$$

Before calculating the global truncation error of the Euler method for this system of ODEs, we first calculate the Lipschitz constant L that satisfies the following condition for any two states $\mathbf{n}_1\,(t) \,:=\, \left[u_1\,(t) \quad i_1\,(t)\right]^{\mathrm{T}}$ and $\mathbf{n}_2\,(t) \,:=\, \left[u_2\,(t) \quad i_2\,(t)\right]^{\mathrm{T}}$ [2] (define $\Delta u \,:=\, u_1\,-\,u_2$ and $\Delta i \,:=\, i_1\,-\,i_2$)

$$\left\|\dot{\mathbf{n}}_1\,-\,\dot{\mathbf{n}}_2\right\|_2 \,=\, \sqrt{\frac{1}{\tau_m^2}\,\left[\Delta u\,-\,R\,\Delta i\right]^2\,+\,\frac{1}{\tau_s^2}\,\Delta^2 i} \tag{15}$$

$$\leq \sqrt{\frac{1\,+\,R}{\tau_m^2}\,\Delta^2 u\,+\,\left(\frac{R^2\,+\,R}{\tau_m^2}\,+\,\frac{1}{\tau_s^2}\right)\,\Delta^2 i} \tag{16}$$

$$\leq \sqrt{\max\left\{\frac{1\,+\,R}{\tau_m^2},\ \frac{R^2\,+\,R}{\tau_m^2}\,+\,\frac{1}{\tau_s^2}\right\}}\,\left\|\,\mathbf{n}_1\,-\,\mathbf{n}_2\right\|_2 \tag{17}$$

Moreover, the exponential function $\exp\left(-t\right)$ is bounded over the domain $t \,\in\, [0,\,+\infty)$. Given that the sum of multiple bounded functions remains bounded, it follows that there exists a finite real number M such that $|u''(t)| \,\leq\, M$. This property has significant implications for the numerical analysis of the system. Specifically, the global truncation error of Euler's method at time t_k is constrained by an upper bound of $\frac{M\,\Delta t}{2\,L}\,\left(\exp\left(L\,t_k\right)\,-\,1\right)$, where Δt denotes the step size, and L represents the Lipschitz constant of the derivative $u'\,(t)$.

2.4 Population Activity of Identical Neurons in Layer

The population dynamics and state statistics of neurons within a single layer of a spiking neural network, where neurons share identical parameters (τ_m, τ_s, and R), can be rigorously evaluated [8]. A key aspect of this analysis is the evolution equation governing the probability density function of the neurons' internal states, specifically voltage and current. Given an input spike train following a Poisson Process, the probability density function $p(t,\ u,\ i)$ can be modeled using the Fokker-Planck equation [4]. To derive the corresponding Delayed Partial Differential Equation, we begin by expressing the probability density function in terms of expectation: $p(t,\ v,\ i) = \mathbb{E}[\delta(u - U(t))\,\delta(i - I(t))]$, where $(U,\ I)$ represents the two-dimensional random variable describing a neuron's internal state. We then evaluate $p(t+\Delta t,\ u,\ i)$ using conditional expectation at time t. Given that the incoming spike train follows a Poisson Distribution, the probability of a spike arriving within the time interval $[t,\ t+\Delta t)$ is $\lambda(t)\,\Delta t$ [7]. This probabilistic framework allows us to formulate the conditional expectation as follows

$$\mathbb{E}\left[\delta(u - U(t+\Delta t))\,\delta(i - I(t+\Delta t)) \mid U(t), I(t)\right] \tag{18}$$

$$= (1 - \lambda(t)\,\Delta t)\ \delta\left(u - \left[U(t) + \left(-\frac{1}{\tau_m}U(t) + \frac{R}{\tau_m}I(t)\right)\Delta t\right]\right)\delta\left(i - \left[I(t) - \frac{1}{\tau_s}I(t)\,\Delta t\right]\right) \tag{19}$$

$$+ \lambda(t)\,\Delta t\,\delta(u - U(t))\,\delta\left(i - \left[I(t) + \frac{w}{\tau_s}\right]\right) \tag{20}$$

We perform first-order Taylor expansion the resetting state transition with regards to u and i at point $U(t)$ and $I(t)$

$$\delta\left(u - \left[U(t) + \left(-\frac{1}{\tau_m}U(t) + \frac{R}{\tau_m}I(t)\right)\Delta t\right]\right) \tag{21}$$

$$= \delta(u - U(t)) + \frac{\partial}{\partial u}\left[\delta(u - U(t))\left(\frac{1}{\tau_m}U(t) - \frac{R}{\tau_m}I(t)\right)\Delta t\right] \tag{22}$$

$$\delta\left(i - \left[I(t) - \frac{1}{\tau_s}I(t)\,\Delta t\right]\right) = \delta(i - I(t)) + \frac{\partial}{\partial i}\left[\delta(i - I(t))\ \frac{1}{\tau_s}I(t)\,\Delta t\right] \tag{23}$$

By using the fact on Dirac Delta function $x\,\delta(x-a) = a\,\delta(x-a)$, and omitting the terms with $O\left(\Delta^2 t\right)$, we have

$$\mathbb{E}\left[\delta(u - U(t+\Delta t))\,\delta(i - I(t+\Delta t)) \mid U(t), I(t)\right] \tag{24}$$

$$= \delta(u - U(t))\,\delta(i - I(t)) \tag{25}$$

$$+ \frac{\partial}{\partial u}\left[\delta(u - U(t))\,\delta(i - I(t))\left(\frac{u}{\tau_m} - \frac{Ri}{\tau_s}\right)\Delta t\right] \tag{26}$$

$$+ \frac{\partial}{\partial i}\left[\delta(u - U(t))\,\delta(i - I(t))\ \frac{i}{\tau_s}\Delta t\right] \tag{27}$$

$$+ \lambda(t)\,\Delta t\left[\delta(u - U(t))\,\delta\left(i - \frac{w}{\tau_s} - I(t)\right) - \delta(u - U(t))\,\delta(i - I(t))\right] \tag{28}$$

Taking the unconditional expectation on both sides yields the probability density function at various states in discretised form with step size Δt

$$p(t+\Delta t, u, i) = p(t, u, i) \tag{29}$$

$$+ \frac{\partial}{\partial u}\left[p(t, u, i)\left(\frac{u}{\tau_m} - \frac{Ri}{\tau_s}\right)\Delta t\right] \tag{30}$$

$$+ \frac{\partial}{\partial i}\left[p(t, u, i)\ \frac{i}{\tau_s}\Delta t\right] \tag{31}$$

$$+ \lambda(t)\,\Delta t\left(p\left(t, u, i - \frac{w}{\tau_s}\right) - p(t, u, i)\right) \tag{32}$$

Diving Δt by both sides and taking the limit $\lim_{\Delta t \to 0}$ which brings about $\frac{\partial}{\partial t}p(t, u, i) = \lim_{\Delta t \to 0} \frac{p(t+\Delta t, u, i)\, -\, p(t, u, i)}{\Delta t}$

$$\frac{\partial}{\partial t}p(t, u, i) = \frac{\partial}{\partial u}\left[p(t, u, i)\left(\frac{u}{\tau_m} - \frac{Ri}{\tau_s}\right)\right] + \frac{\partial}{\partial i}\left[p(t, u, i)\ \frac{i}{\tau_s}\right] \tag{33}$$

$$+ \lambda(t)\left(p\left(t, u, i - \frac{w}{\tau_s}\right) - p(t, u, i)\right) \tag{34}$$

Diffusion Approximation [9] could be used to eliminate the delayed term and obtain the final Fokker-Planck equation. We first perform second-order Taylor expansion on (34)

$$\lambda(t)\left(p\left(t,u,i-\frac{w}{\tau_s}\right)-p(t,u,i)\right) \;=\; \lambda(t)\ \left(-\frac{w}{\tau_s}\frac{\partial}{\partial i}p(t,u,i)+\frac{1}{2}\frac{w^2}{\tau_s^2}\frac{\partial^2}{\partial i^2}p(t,u,i)\right) \quad (35)$$

Let $p := p(t,u,i)$ denote the probability density function of the system's state. We derive the corresponding Fokker-Planck equation, which characterizes the temporal evolution of this probability density. Through this analysis, we identify the drift coefficient $\lambda(t)w$ and the diffusion coefficient $\frac{\lambda(t)w^2}{2\tau_s}$ associated with the stochastic current variable $I(t)$ [3]. This formulation leads to the conclusion that the membrane potential $U(t)$ and the synaptic current $I(t)$ jointly form a bivariate Markov process [20]

$$\frac{\partial}{\partial t}p \;=\; \frac{\partial}{\partial u}\left[\left(\frac{u}{\tau_m}-\frac{Ri}{\tau_s}\right)p\right] \;+\; \frac{\partial}{\partial i}\left[\frac{i-\lambda(t)w}{\tau_s}p+\frac{1}{2}\frac{\lambda(t)w^2}{\tau_s^2}\frac{\partial}{\partial i}p\right] \quad (36)$$

3 Methods and Model Architecture

The architecture of our model builds upon the foundational BATS algorithm, incorporating two significant enhancements:

1. The integration of a time-discretization feature facilitates the temporal modeling of spiking neural networks.
2. The inherent constraints of the BATS algorithm are relaxed, allowing for more flexible and biologically accurate depictions of neural dynamics.

3.1 Time Discretization and Forward Pass

Consider a neuron, denoted as (l,k), within a spiking neural network (SNN). The state of this neuron comprises its membrane potential (voltage) and current, represented by the vector $\mathbf{n}^{(l,k)} := \begin{bmatrix}u^{(l,k)} & i^{(l,k)}\end{bmatrix}^{\mathrm{T}}$. Given a time step size of Δt and a time range $[t_{\text{pre}},\ t_{\text{post}})$, the number of steps to be simulated is $\left\lfloor \frac{t_{\text{post}}-t_{\text{pre}}}{\Delta t}\right\rfloor$. Instead of continuously solving quadratic equations until numerical errors accumulate or the solution falls outside the range $[t_{\text{pre}},\ t_{\text{post}})$, as the BATS algorithm does, spike times are enumerated for each discrete time step in SNN. A spike is emitted when the neuron's membrane potential reaches a predefined threshold value. The Backward Euler Discretization method is employed in the implementation, as it can be interpreted as a multi-layer Recurrent Neural Network (RNN), a dynamic system that maps input sequences to output sequences, with current and voltage serving as the two hidden layers. SNN can be interpreted as a specialization of RNN, where the input and output sequences consist solely of binary values (0 and 1) . However, there are two key distinctions that further differentiate SNN from RNN:

1. *Parameter Complexity*: SNN typically requires fewer trainable parameters compared to RNN. In RNN, three types of parameters are trainable: how the input sequence affects the hidden states, how the state transitions from the current timestamp to the next, and how the hidden states generate the output sequence [23]. However, in SNN, only the influence of incoming spiking trains on the hidden states needs to be trained. The state transition is captured by the Leaky Integrate-and-Fire (LIF) ordinary differential equation (ODE), and the output is generated based on the rule that the membrane potential surpasses the threshold.

2. *Temporal Dynamics*: RNN typically models states in a time-discretized manner, or in cases of extremely large step numbers, can be viewed as sampling from a learned ODE [5]. In contrast, SNN represents a sampling of the original LIF dynamical system. This sampling process inherently accounts for the effects of the chosen time step. Consequently, different step sizes yield distinct representations of the original equation, each capturing the system's dynamics with varying fidelity (see Figure 1).

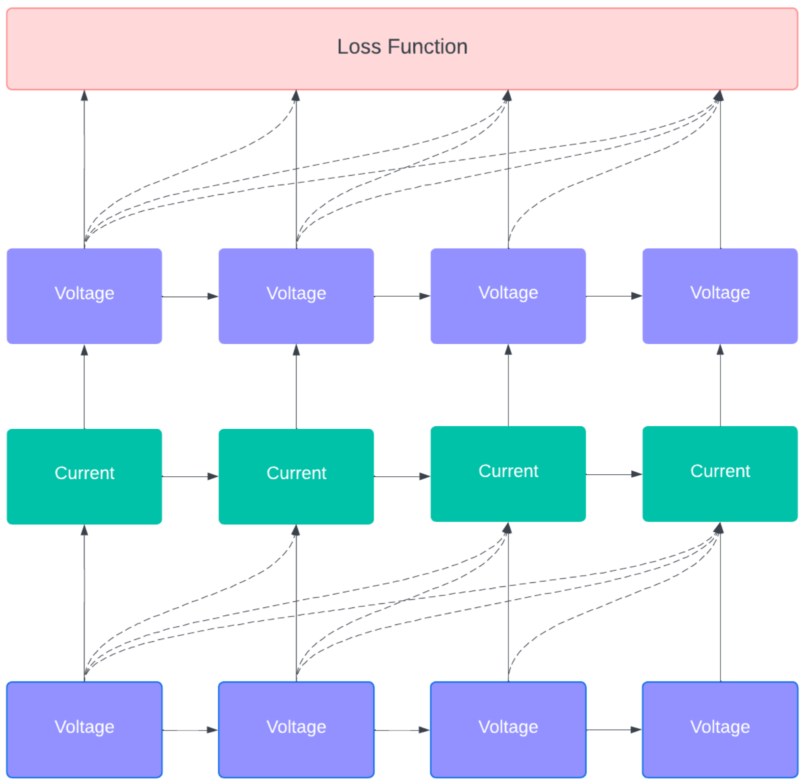


Figure 1: Network Architecture

3.2 Backward Pass

3.2.1 Backpropagation Methods

Training Spiking Neural Network with Backpropagation rules suffer the non-differentiation of the Dirac Delta function, which prevents the estimate of the gradient on how change of membrane potential affects the spikes: $\frac{\partial \delta(u \geq \vartheta)}{\partial u}$. The common approach to tackle the problem is to use the *surrogate gradient* [16], for instance, the SLAYER model claims the similarity between the derivative of spike function and probability density function for firing the spike [22]. In this project, we adopt an event-driven learning approach, propagating gradient information to previous layers through spike time. We utilize a time-based gradient method to estimate how changes in spiking times affect the neuron's state . Specifically, let $x(t_1)$ be a state variable at time t_1 and t_2 be a spike time. The time-based gradient method requires accurate estimations of $\frac{\partial x(t_1)}{\partial t_2}$ and $\frac{\partial t_2}{\partial x(t_1)}$. This establishes the broader impact chain, indicating how separate state variables $x\left(t_1\right)$ and $y\left(t_2\right)$ influence each other, denoted as $\frac{\partial x(t_1)}{\partial y(t_2)}$, and how the spike times influence each other, denoted as $\frac{\partial t_1}{\partial t_2}$. In the BATS algorithm, the closed-form solution for spike time can be explicitly calculated, rendering the inclusion of state variables in the chain rule unnecessary . Thus, only $\frac{\partial t_1}{\partial t_2}$ is needed to propagate errors. However, after time discretization, the explicit solution for spike time is no longer calculable, necessitating the estimation of how spike time affects different state variables.

The recurrence formula between error functions differs within each layer of neurons and between different layers. For the current layer l, assume the errors on $\frac{\partial E}{\partial u^{(l)}}\left[t_k\right]$ are already evaluated. Our task is to evaluate the errors on current in the same layer $\frac{\partial E}{\partial i^{(l)}}\left[t_k\right]$ and the errors on voltage in the previous layer $\frac{\partial E}{\partial u^{(l-1)}}\left[t_k\right]$. The recurrent dynamics within a layer of a Spiking Neural Network can be explicitly formulated using the Backpropagation Through Time (BPTT) algorithm [26]

$$\frac{\partial L}{\partial i^{(l)}\left[t_k\right]} = \frac{\partial L}{\partial u^{(l)}\left[t_k\right]} \frac{\partial u^{(l)}\left[t_k\right]}{\partial i^{(l)}\left[t_k\right]} + \frac{\partial L}{\partial i^{(l)}\left[t_k + \Delta t\right]} \frac{\partial i^{(l)}\left[t_k + \Delta t\right]}{\partial i^{(l)}[t_k]} \tag{37}$$

The error backpropagation between different layers is nontrivial to compute. The term $\frac{\partial i^{(l)}[t_m]}{\partial u^{(l-1)}[t_k]}$ denotes the partial derivative of the current at time t_m in layer $l+1$ with respect to the membrane potential at time t_k in layer l. To maintain causality, it is imperative that $t_m \geq t_k$. This derivative can be accurately estimated and computed using dynamic programming techniques [27], which account for both inter-neuron and intra-neuron dependencies. Here, t_l is defined as the immediate spike time following t_k

$$\frac{\partial i^{(l)}\left[t_m\right]}{\partial u^{(l-1)}\left[t_k\right]} = \frac{\partial i^{(l)}\left[t_m\right]}{\partial t_k} \frac{\partial t_k}{\partial u^{(l-1)}\left[t_k\right]} + \frac{\partial i^{(l)}\left[t_m\right]}{\partial u^{(l-1)}\left[t_l\right]} \frac{\partial u^{(l-1)}\left[t_l\right]}{\partial t_k} \frac{\partial t_k}{\partial u^{(l-1)}\left[t_k\right]} \tag{38}$$

Because changing $u^{(l-1)}\left[t_k\right]$ would affect $\left\{i^{(l)}\left[t_m\right]\right\}_{m=k}^{T}$. Therefore, we find the recurrence relation between $\frac{\partial L}{\partial u^{(l-1)}[t_k]}$ and $\frac{\partial L}{\partial i^{(l)}[t_m]}$

$$\frac{\partial L}{\partial u^{(l-1)}\left[t_k\right]} = \sum_{m=k}^{T} \frac{\partial L}{\partial i^{(l)}\left[t_m\right]} \frac{\partial i^{(l)}\left[t_m\right]}{\partial u^{(l-1)}\left[t_k\right]} + \frac{\partial L}{\partial u^{(l-1)}\left[t_k + \Delta t\right]} \frac{\partial u^{(l-1)}\left[t_k + \Delta t\right]}{\partial u^{(l-1)}\left[t_k\right]} \tag{39}$$

There are two primary approaches to estimating the gradients with respect to weights ($\frac{\partial L}{\partial w}$). Continuous Estimator yields data-dependent gradient estimates. The gradient values are influenced by the specific input data and the continuous dynamics of the network. On the other hand, Discrete Estimator produces fixed gradient estimates for a given discretization scheme

$$\frac{\partial L}{\partial w^{(l-1,l)}} = \begin{cases} \frac{\partial L}{\partial u^{(l)}} \frac{\partial u^{(l)}}{\partial w^{(l-1,l)}} & \text{(continuous)} \\ \frac{\partial L}{\partial i^{(l)}} \frac{\partial i^{(l)}}{\partial w^{(l-1,l)}} & \text{(discrete)} \end{cases} \tag{40}$$

4 Experimental Setup and Results

This section outlines our experimental setup and delineates the methodologies employed in our investigation. We focus on two key experiments that significantly impact the accuracy and training dynamics of our Spiking Neural Network model. First, we evaluate various optimization methods to elucidate the properties of gradients in our SNN. Second, we conduct a comprehensive spectral analysis of the output signals from each layer of the network. This examination reveals the frequency domain characteristics of signal propagation through the SNN.

4.1 Dataset and Setup

The dataset employed for evaluating the algorithm is the MNIST handwritten digit dataset [13], which includes $60,000$ training samples and $10,000$ testing samples. Each sample is a single-channel image with dimensions of 28×28 pixels. The experimental setup utilizes a batch size of 50 samples to address space complexity, and the training process spans 4 epochs. This configuration reveals that our algorithm is less competitive compared to ANN with an equivalent number of layers and weights shape. To discretize the ODE, the simulation is performed over 50 time steps, again due to space complexity constraints. Consequently, each test simulates the ODE over various time ranges. The Poisson input spikes are generated using a rate encoding scheme, with the encoding parameter determined by pixel intensity. Due to implementation time constraints, we exclusively constructed Fully Connected Layers, forming all layers from previous to current ones, without incorporating a traditional ANN model as seen in [27]. Kaiming Initialization [11] was used for parameter initialization in each layer, given that the ReLU component in ANNs shares similarities and can be bidirectionally mapped with the IF neuron in SNN [14]. The shared hyper-parameters are $(\tau_m,\ \tau_s,\ R) = (0.8,\ 0.6,\ 1.2)$ and $(v_{\text{thres}},\ v_{\text{reset}}) = (1.5,\ 1)$. Hyperparameter optimization was deliberately excluded from the scope of this study. The results are shown in figure 4

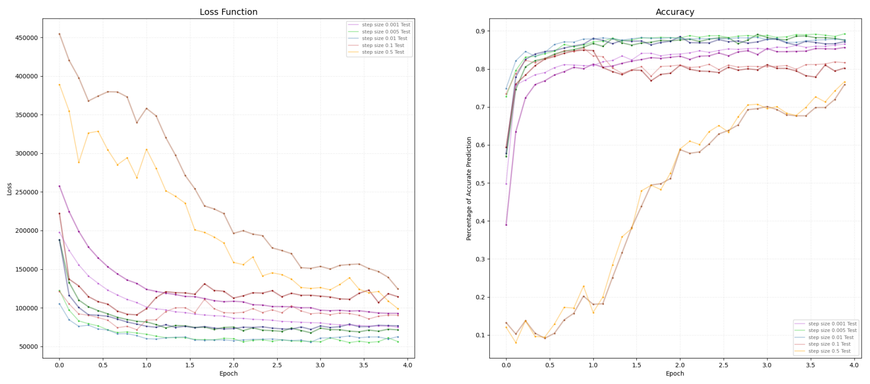
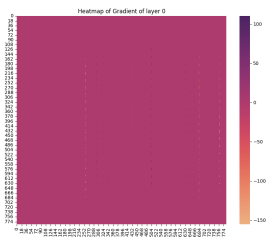


Figure 2: Accuracy and Losses for different step sizes

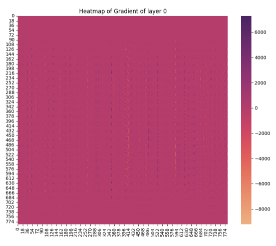
4.2 Practical Issues: Gradient and Optimization

Due to the similarity between SNN and RNN, analogous difficulties in training RNN has emerged when training SNN, primarily concerning the issues of exploding and vanishing gradients [23, 19]. In SNN, gradient explosion is caused by the term $\left(\frac{\mathrm{d}u}{\mathrm{d}t}\right)^{-1}$, which tends towards infinity when both voltage and current approach zero. This phenomenon further affects the estimation of $\frac{\partial i^{(l)}[t_m]}{\partial u^{(l-1)}[t_k]}$, impacting the derivative with respect to state variables in earlier timestamps and deeper layers from the output layer, by inducing greater outliers in the gradient density. This issue can be mitigated through gradient clipping [19], a technique also employed in the original implementation [27]: we clamp the intermediate gradient absolute value to 1×10^4.

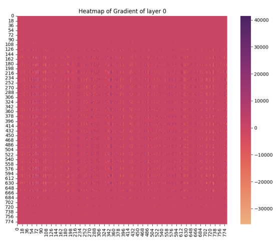
Layer 0 Weight Gradient



(a) Step size 0.001

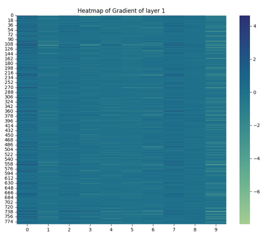


(b) Step size 0.01

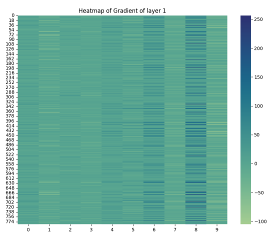


(c) Step size 0.1

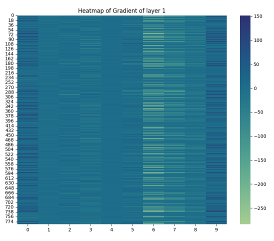
Layer 1 Weight Gradient



(d) Step size 0.001



(e) Step size 0.01



(f) Step size 0.1

Figure 3: Weight Gradients of the Initial Backpropagation

The Adam (Adaptive Moment Estimation) optimizer [12] is employed for training the neural network, leveraging a combination of first-order gradient-based optimization and second-moment estimation. Moment methods accelerate convergence by utilizing an autoregressive approach, which combines the accumulated velocity direction with the current calculated gradient direction to update parameters [25]. Motivated by research suggesting that second-order (curvature) information may enhance RNN training [15], we explored the use of the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm as an alternative optimizer. However, this approach exhibited unstable behavior, manifesting undefined outcomes even when repeatedly fitting a single batch.

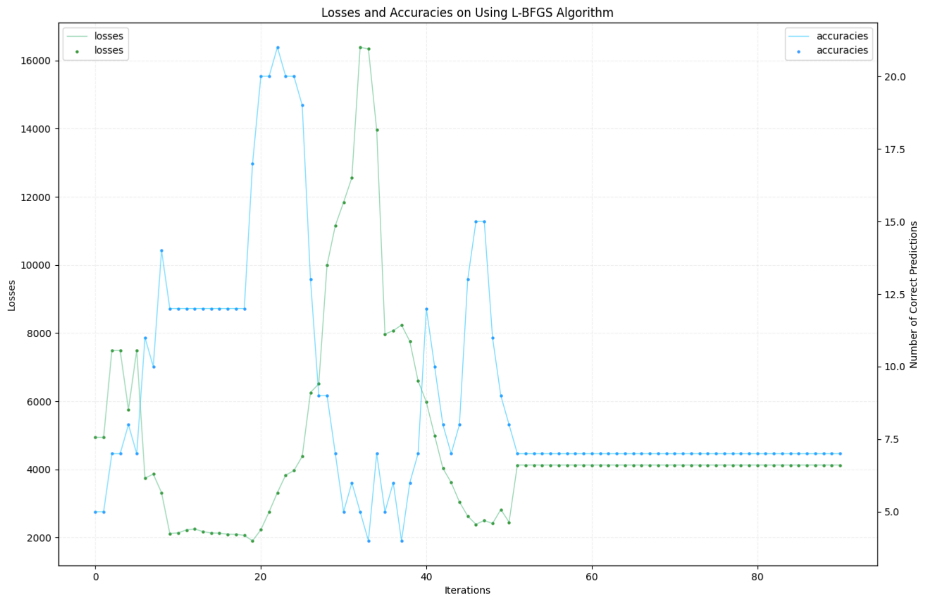
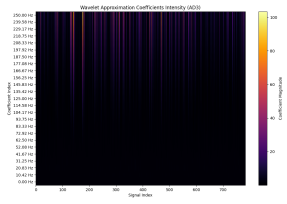


Figure 4: Accuracy and Loss Trajectories Using the L-BFGS Optimizer. The plot shows the evolution of accuracy and loss over training iterations. Note that iterations differ from epochs; the L-BFGS algorithm requires multiple iterations per epoch to approximate Hessian matrix and update parameters.

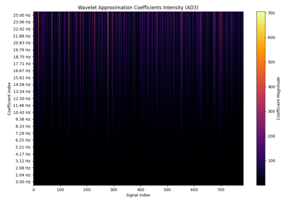
4.3 Spectral Analysis of Hidden Layer

We apply Wavelet transformation to the batch-averaged voltage signals of the hidden layer to conduct spectral analysis. This technique enables the decomposition of the signal into distinct frequency components while preserving temporal information. The abscissa represents individual neuron indices, while the ordinate depicts voltage values averaged across a single batch.

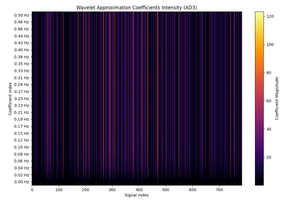
Approximate Coefficients



(a) Step size 0.001

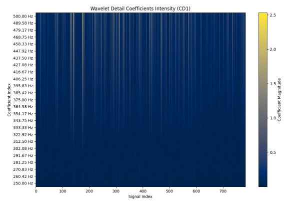


(b) Step size 0.01

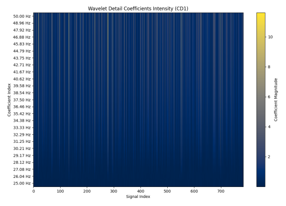


(c) Step size 0.5

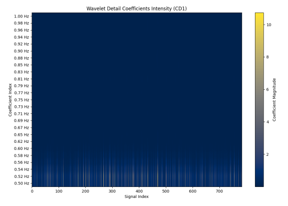
Detailed Coefficients



(d) Step size 0.001



(e) Step size 0.01



(f) Step size 0.5

Figure 5: Spectral Analysis of the Hidden Layer Current Output

5 Discussion

The network architecture we have designed, based on the BATS model but incorporating RNN principles to account for time discretization, presents several notable limitations:

1. Inter-layer error backpropagation exhibits a non-monotonic relationship with respect to temporal proximity. Specifically, it does not hold that $\frac{\partial i^{(l+1)}[t_k]}{\partial u^{(l)}[t_m]} < \frac{\partial i^{(l+1)}[t_k]}{\partial u^{(l)}[t_n]}$ for $t_m < t_n$, due to the influence of the term $\left(\frac{\partial u^{(l)}[t_n]}{\partial t_n}\right)^{-1}$. The implications of this property are ambiguous: while intuitively a temporally proximate spike change should have a greater impact on the current, this characteristic may mitigate the vanishing gradient problem commonly observed in RNN as training progresses through time [24]. In the original implementation, with only 5 simulated steps, it may be feasible to implement a temporal cutoff, setting $\frac{\partial i^{(l+1)}[t_k]}{\partial u^{(l)}[t_m]} \equiv 0$ when $t_k - t_m$ exceeds a predetermined threshold.
2. The algorithm exhibits substantial memory complexity: $O(BT^2N^2)$, where T denotes the number of time steps and N represents the number of neurons. This constrains the model's applicability to scenarios requiring extensive temporal simulation or involving consecutive layers with high neuronal density.

Based on our experimental observations, we can draw the following conclusions:

1. Time step size significantly influences the training process of Spiking Neural Networks. Smaller step sizes generally yield superior training and testing accuracies, particularly in the early stages of training where they facilitate rapid loss reduction across batches. However, this relationship is not monotonic, as other factors come into play when the step size becomes excessively small: 1) When the step size is very small, only a narrow range of the original ordinary differential equation is simulated, potentially discarding broader dynamical behaviors crucial to the network's function. 2) As the temporal discretization step size approaches zero, numerical issues become increasingly pronounced in the simulation of Spiking Neural Networks. A key manifestation of this phenomenon is observed in the behavior of the partial derivative $\lim_{\Delta t \to 0} \frac{\partial i[t+\Delta t]}{\partial i[t]} = 1$.

2. The SNN presents significant challenges compared to traditional RNN. Two key issues stand out: 1) The exploding gradient phenomenon, which becomes more pronounced with increased time step sizes. This issue can lead to unstable training and poor convergence. 2) The potential for second-order information to provide misleading optimization directions. This problem is partly exacerbated by the exploding gradients issue, as large gradient values can distort the curvature information used in second-order optimization methods. Momentum-based autoregression techniques, such as those employed in adaptive optimization algorithms like Adam [12], may offer a partial solution to these challenges. These methods implicitly penalize large variances in gradients across different training iterations.
3. Spectral analysis of voltage signals reveals a significant correlation between discretization step size and the magnitude of frequency components. Smaller step sizes result in higher magnitudes of high-frequency components in the voltage signals, while larger step sizes lead to increased magnitudes in low-frequency components. This empirical observation aligns with theoretical predictions that increasing the step size attenuates amplitudes in the high-frequency region of the discrete signal spectrum. The findings suggest that temporal discretization granularity directly influences the spectral characteristics of signals propagating through the network. However, the relationship between the frequency composition of signals generated in the hidden layers of the network and overall model performance (such as accuracy) or other behaviors remains understudied in the machine learning community. This knowledge gap presents a challenge in directly relating our spectral analysis results to the broader conclusions about network performance and behavior.
4. The comparative behavior between the Discrete Time Markov process ($U\left[t_k\right],\ I\left[t_k\right]$) and its theoretical continuous-time counterpart remains unexplored in this study due to time constraints.

6 Conclusions and Future Work

This study presents a theoretical and empirical analysis of the impact of time discretization on Spiking Neural Network accuracy. Our investigation encompasses multiple perspectives, revealing that the choice of time step size significantly influences various aspects of model training and, consequently, model accuracy. Key factors affected include signal propagation dynamics and numerical precision of computations. The following items outline directions for future research

1. Enhancement of the network architecture and its evaluation on a broader range of datasets to establish robust performance benchmarks. Our current model has been tested exclusively on the MNIST dataset, and its performance does not yet rival that of traditional Artificial Neural Networks.
2. Optimization of computational efficiency, addressing both time and memory complexity. The current quadratic scaling with respect to time steps and neuron count imposes significant constraints on the implementation of popular network architectures. For instance, even for the relatively small-scale MNIST dataset, training requires approximately 2 hours per epoch, rendering more extensive tasks computationally prohibitive.
3. Development of advanced analytical tools for characterizing the statistical properties of generated signals: While our current study employs spectral analysis, there remains a vast array of unexplored statistical features, particularly those related to the properties discussed in Section 2.4. These warrant further investigation to deepen our understanding of Spiking Neural Network dynamics. However, due to time constraints and the limited number of discrete timestamps available, we have not yet devised a methodology to measure these statistics comprehensively in both continuous and discrete settings.

7 Responsible Research

The responsible research aspects of our study are manifested in the following three dimensions:

1. Ethical Considerations: Our research primarily involves mathematical derivations, network implementation, and experimental procedures. As such, it does not directly implicate significant ethical concerns typically associated with human or animal subjects, data privacy, or environmental impact. Nevertheless, we remain cognizant of the broader ethical implications of advancing AI technologies and their potential societal impacts.
2. Methodological Rigor and Reproducibility: We have taken substantial measures to ensure the correctness and verifiability of our work:
    - Formula Derivation: Key steps in our mathematical derivations are explicitly delineated in the paper, facilitating independent verification by the examiners.
    - Code Implementation: Due to the performance requirements necessitating the use of PyTorch's parallel execution capabilities, much of our production code is vectorized. This approach, while efficient, deviates from conventional programming paradigms and may obscure readability. To mitigate potential implementation errors and enhance reproducibility, we have developed parallel serial implementations of critical components. These serve as a basis for random testing against the vectorized code, substantially reducing the likelihood of implementation errors in the original, optimized codebase.
3. Natural Language Processing (NLP) models, specifically those based on the Generative Pre-trained Transformer (GPT) architecture, are primarily utilized to enhance the grammatical accuracy and linguistic authenticity of academic papers. The core ideas and substantive content are conceived and composed entirely by the author without reliance on artificial intelligence systems. However, AI-powered tools may occasionally be consulted to verify specific technical details, ensuring precision in the presentation of complex concepts.

References

[1] Florian Bacho and Dominique Chu. Exploring tradeoffs in spiking neural networks, 2023.

[2] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. The Prindle, Weber and Schmidt Series in Mathematics. PWS-Kent Publishing Company, Boston, fourth edition, 1989.

[3] María J. Cáceres, José A. Carrillo, and Louis Tao. A numerical solver for a nonlinear fokker-planck equation representation of neuronal network dynamics. *J. Comput. Phys.*, 230(4):1084â1099, feb 2011.

[4] David Cai, Louis Tao, Aaditya Rangan, and David McLaughlin. Kinetic theory for neuronal network dynamics. *Communications in mathematical sciences*, 4:97–127, 01 2006.

[5] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *CoRR*, abs/1806.07366, 2018.

[6] Jason K. Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning, 2023.

[7] R.G. Gallager. *Stochastic Processes: Theory for Applications*. Stochastic Processes: Theory for Applications. Cambridge University Press, 2013.

[8] Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014.

[9] Peter W. Glynn. Chapter 4 diffusion approximations. In *Stochastic Models*, volume 2 of *Handbooks in Operations Research and Management Science*, pages 145–198. Elsevier, 1990.

[10] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. volume 86, pages 2278–2324, 1998.

[14] Sen Lu and Abhronil Sengupta. Exploring the connection between binary and spiking neural networks. *CoRR*, abs/2002.10064, 2020.

[15] James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 1033â1040, Madison, WI, USA, 2011. Omnipress.

[16] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *CoRR*, abs/1901.09948, 2019.

[17] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-Time Signal Processing*. Prentice-hall Englewood Cliffs, second edition, 1999.

[18] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid Nawab. *Signals & systems (2nd ed.)*. Prentice-Hall, Inc., USA, 1996.

[19] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[20] A Renart, N Brunel, and Xiao-Jing Wang. *Mean-field theory of recurrent cortical networks: Working memory circuits with irregularly spiking neurons*, pages 432–490. CRC Press, 2003.

[21] Kaushik Roy, Akhilesh R. Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575:607 – 617, 2019.

[22] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[23] Ilya Sutskever. *Training recurrent neural networks*. PhD thesis, CAN, 2013. AAINS22066.

[24] Ilya Sutskever and Geoffrey E. Hinton. Temporal-kernel recurrent neural networks. *Neural Networks*, 23(2):239–243, 2010.

[25] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[26] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[27] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *CoRR*, abs/2002.10085, 2020.