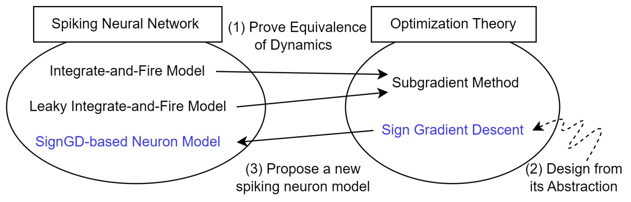
Sign Gradient Descent-based Neuronal Dynamics: ANN-to-SNN Conversion Beyond ReLU Network

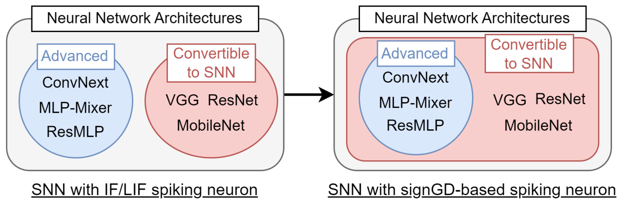
Hyunseok Oh [1] **Youngki Lee** [1]

Abstract

Spiking neural network (SNN) is studied in multidisciplinary domains to (i) enable order-of-magnitudes energy-efficient AI inference and (ii) computationally simulate neuroscientific mechanisms. The lack of discrete theory obstructs the practical application of SNN by limiting its performance and nonlinearity support. We present a new optimization-theoretic perspective of the discrete dynamics of spiking neurons. We prove that a discrete dynamical system of simple integrate-and-fire models approximates the subgradient method over unconstrained optimization problems. We practically extend our theory to introduce a novel sign gradient descent (signGD)-based neuronal dynamics that can (i) approximate diverse nonlinearities beyond ReLU and (ii) advance ANN-to-SNN conversion performance in low time steps. Experiments on large-scale datasets show that our technique achieves (i) state-of-the-art performance in ANN-to-SNN conversion and (ii) is the first to convert new DNN architectures, e.g., ConvNext, MLP-Mixer, and ResMLP. We publicly share our source code at www.github.com/snuhcs/snn_signgd .


Spiking Neural Network
(1) Prove Equivalence of Dynamics
Optimization Theory
Integrate-and-Fire Model
Leaky Integrate-and-Fire Model
SignGD-based Neuron Model
Subgradient Method
Sign Gradient Descent
(3) Propose a new spiking neuron model
(2) Design from its Abstraction

(a) Conceptual diagram of our technical contributions.



(b) Our signGD-based spiking neuron enables the conversion of neural networks that use nonlinear operators other than ReLU.

Figure 1. In this paper, we (i) mathematically connect the neuronal dynamics of integrate-and-fire models with the optimization dynamics of subgradient method, (ii) extend the theory to design a new spiking neuron model that can approximate arbitrary element-wise tensor operators, and (iii) use our neuron model to expand ANN-to-SNN conversion beyond ReLU networks (Fig. 1(b)).

1. Introduction

Understanding how a biological neuron processes information has been a milestone of both neuroscience and efficient artificial intelligence (Christensen et al., 2021; Furber et al., 2014b; Kasabov, 2014). Spiking neural network (SNN) is widely studied to get new insights on the information dynamics of the brain (Ghosh-Dastidar & Adeli, 2009; Furber et al., 2014b). SNN is a biologically plausible type of artificial neural network (ANN) in which its neuron models closely mimic neuroscientific mechanisms (Schuman et al., 2022). In detail, the spiking neuron, the central processing unit of SNN, (i) processes non-linearity with an internal dynamical system and (ii) communicates the information in the form of a spike train, a time series of short binary electrical pulses. These key characteristics of SNN are practically leveraged to develop extremely efficient AI algorithms (Schuman et al., 2022). For example, SNN inference is orders-of-magnitude more energy-efficient than the same-architecture DNNs, e.g., 35-560× less on VGG (Bu et al., 2022) and 280× less on YOLO (Kim et al., 2020). Discretization of spiking neuronal dynamics is thus pivotal to both (i) practically implement SNN for real-world applications and (ii) simulate brain mechanisms with SNN. However, the theoretical understanding of SNN remains unclear (Zhang & Zhou, 2022), especially regarding its discrete neuronal dynamics.

The lack of discrete theory obstructs the practical application of SNN by constraining its inference accuracy and nonlinearity support. Building a high-performance SNN is

[1]Department of Computer Science & Engineering, Seoul National University, Seoul, Republic of Korea. Correspondence to: Youngki Lee <youngki.lee@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

categorized into two groups: (i) training SNN from scratch (Reviewed in Appendix A.) and (ii) converting a pre-trained ANN model into SNN (Cao et al., 2015). A successful SNN training strategy, surrogate gradient methods (Neftci et al., 2019), consumes immense computational resources since they unfold a DNN backwards the entire time-steps (Li et al., 2021a). Yet, their best accuracy still falls behind DNNs with similar architecture, e.g., 4-6% on ResNet (Fang et al., 2021) and 6-10% on ViT (Zhou et al., 2022). On the other hand, ANN-to-SNN conversion techniques (Cao et al., 2015; Han & Roy, 2020) replace real-valued nonlinear operators of ANN with spiking neurons. However, the only known theoretical correspondence is between the clipped ReLU function and IF neuron (Rueckauer et al., 2017). This leads to three major limitations in prior conversion approaches. First, ReLU should be the only nonlinear operator in a target ANN. Second, an accurate SNN inference mandates data-dependent normalization or calibration techniques (Li et al., 2021a). Finally, low-latency techniques noticeably degrade the best accuracy of SNN by replacing ReLU with spike-aware functions (Bu et al., 2022; Jiang et al., 2023).

In this light, we provide a new optimization-theoretic perspective of the discrete neuronal dynamics that can (i) explain the underlying principle of neuronal dynamics and (ii) extend to design a new spiking neuron that can compute various nonlinearities. We first prove that a discrete dynamical system of simple integrate-and-fire models is equivalent to a subgradient method over an unconstrained optimization problem with its solution as a spike-coded nonlinear function value. SNN inference is thus a neuron-wise first-order optimization process to approximate real-valued activations. This framework provides a way to study discrete neuronal dynamics by expressing it as an equivalent optimization algorithm, i.e., its optimizer form, and performing convergence analysis to derive the neuron's asymptotic behavior.

Practically extending our theory, we present a novel signGD-based neuronal dynamics that can (i) approximate diverse nonlinearities beyond ReLU and (ii) achieve high accuracy in low time steps with converted SNNs. Specifically, we choose the sign gradient descent algorithm (signGD) (Bernstein et al., 2018) as the optimizer form of our neuron, replacing the subgradient method. A spiking neuron's key characteristic, spike-based communication, constrains the space of approximable nonlinear function for the subgradient-based neuronal dynamics. In contrast, in the case of signGD-based neuronal dynamics, a binary spike conveys only the sign of the gradient of the objective function, widening the space of approximable nonlinear functions. We generalize the learning rate schedule of the signGD-based optimizer form to formulate the new neuronal dynamics and the neural coding scheme. We empirically validate with experiments that our signGD-based neuron can approximate unary nonlinearities, e.g., ReLU, LeakyReLU, and GELU (Hendrycks & Gimpel, 2016), and n-ary nonlinearities, e.g., max pooling and layer normalization (Ba et al., 2016).

To empirically verify the effectiveness of our signGD-based neuronal dynamics, we convert high-performance DNNs to SNNs with our proposed neurons and evaluate their performance. Experimental results on large-scale ImageNet (Deng et al., 2009) and CIFAR (Krizhevsky et al., 2009) datasets show that our technique is (i) state-of-the-art in conversion techniques by precisely approximating the ANN performance in ≤ 64 time-steps, and (ii) first to convert complex DNN architectures, e.g., ConvNext (Liu et al., 2022b), MLP-Mixer (Tolstikhin et al., 2021), and ResMLP (Touvron et al., 2021). With our neuron, ImageNet top-1 accuracies of converted VGG16 and ResNet34 are $> 75\%$ in $T = 64$, outperforming runner-ups by 3%. ConvNext-B and RegNetX-3.2F reach $\approx 81\%$ in $T = 256$ for the first time.

2. Related Works

ANN-to-SNN Conversion. Prior conversion techniques substitute the ReLU function of ANN with the IF neuron (Roy et al., 2019; Rueckauer et al., 2017). Its limitation is that converted SNNs require a huge number of time steps for high accuracy (Han & Roy, 2020), leading to large latency and energy consumption (Liu et al., 2022a). Hence, subsequent works sought to *accelerate* SNN inference, i.e., achieve higher accuracy in lower time steps. Data-dependent normalization (Diehl et al., 2015; Rueckauer et al., 2017; Wang et al., 2022), calibration (Li et al., 2021a), or neuron adaptation techniques (Hao et al., 2023) minimize the layer-wise empirical error between ANN and SNN activations (Deng & Gu, 2021). Temporal coding-based techniques (Park et al., 2020; Han & Roy, 2020) embed information in latency of few spikes. Studies in SNN-aware nonlinearities substitute ReLU of an ANN architecture with piecewise-continuous functions, e.g., QCFS (Bu et al., 2022), StepReLU (Wang et al., 2023a), SlipReLU (Jiang et al., 2023). Unlike these works, we theoretically explain integrate-and-fire models beyond IF neurons, propose novel neuronal dynamics to support diverse nonlinearities other than ReLU, and achieve the highest inference accuracy with our converted SNNs.

A few prior works customize neuronal dynamics to accelerate SNN or approximate different nonlinearities. (Liu et al., 2022a) replaces event-driven computation with layer-wise computation. Instead of binary spikes, ternary spikes of $\{-1, 0, 1\}$ are used to approximate LeakyReLU (Kim et al., 2020) or accelerate SNN (Wang et al., 2022). Time series of float instead of spikes are used to support max pooling (Li et al., 2022) or accelerate SNN (Jiang et al., 2023). Overall, these works sacrifice key characteristics of SNN, e.g., event-driven computation or binary spike train. In contrast, our signGD-based neuron computes in an event-driven manner with a binary spike train to support diverse nonlinearities.

Theoretical understandings of SNN. The theoretical basis of SNN is an unclear but widely investigated area (Zhang & Zhou, 2022). SNN can behave as a computational model of Turing machine (Maass, 1996a;b). Chaos theory analyzes singularities and asymptotic behavior of SNN (Cessac, 2008). Mancoo et al. shows that a continuous LIF network holistically solves quadratic programming with convex constraints as a gradient flow. Bifurcation theory shows that the LIF network is a bifurcation dynamical system highly sensitive to decay factor (Zhang et al., 2021). Prior theoretical results on continuous SNN applies inexactly to discrete-time dynamics due to discretization errors (Roy, 2010; Mancoo et al., 2020) accumulating through time (Niesen & Hall, 2004). In contrast, we show that discrete neuronal dynamics of integrate-and-fire models approximate a subgradient method. Furthermore, we practically extend our theory to achieve state-of-the-art performance on SNN inference and ANN-to-SNN conversion.

3. Preliminaries

Integrate-and-fire models are simplified phenomenological models of biological neuronal dynamics (Gerstner et al., 2014). It consists of two components: (i) a time-evolution of membrane potential (Integration) and (ii) a firing mechanism to create a spike (Thresholding). Its continuous neuronal dynamics is a differential equation with a thresholding criterion (See Appendix E.). For computational tractability, the general one-dimensional integrate-and-fire model is discretized as follows.

$$u_{pre}(t) = u(t-1) + f\big(u(t-1)\big) + \frac{R}{\tau_m} I(t) \qquad (1)$$

$$s(t) = \mathbb{H}(u_{pre}(t) - \theta_{th}) \qquad (2)$$

where time $t \in \mathbb{N}$, dynamics function $f(u) : \mathbb{R} \to \mathbb{R}$, pre-firing potential $u_{pre}(t)$, $s(t) \in \{0, 1\}$ a spike, post-firing potential $u(t)$, heaviside step function $\mathbb{H}$, influx current $I(t)$, threshold θ_{th}, membrane resistance R, and membrane constant τ_m. The potential u_{pre} resets to u after the spike $s(t)$ fires based on its pre-defined reset mechanism.

$$\begin{aligned} u(t) &= u_{pre}(t) - \theta_{th} s(t) \quad \text{(reset-by-subtraction)} \qquad (3)\\ u(t) &= u_{pre}(t)(1 - s(t)) \qquad \text{(reset-to-zero)} \end{aligned}$$

Discretized reset mechanisms are categorized into two: (i) *reset-to-zero* discards the leftover potential, and (ii) *reset-by-subtraction* (Han et al., 2020) retains the leftover potential after the reset. We focus on the reset-by-subtraction mechanism since it is easier to theoretically analyze and more performant in practical applications (Han & Roy, 2020).

Integrate-and-fire (IF) neuron has the simplest neuronal dynamics defined as $f(u) = 0, \tau_m = 1$ in equation (1).

$$u_{pre}(t) = u(t-1) + R\, I(t) \qquad (4)$$

Leaky-Integrate-and-fire(LIF) neuron introduces linear leakage $f(u) = -\frac{u - u_{rest}}{\tau_m}$ into the dynamics of equation (1).

$$u_{pre}(t) = u(t-1) - \frac{u(t-1) - u_{rest}}{\tau_m} + \frac{R}{\tau_m} I(t) \qquad (5)$$

Neural coding schemes interpret the information representation of SNN by encoding a real value into a spike train and vice versa. Rate coding decodes an activation value as a ratio of spike events over time steps, i.e., $y =$ (# of spikes)/(total # of time-steps). Importantly, it can be equally defined as a moving average of spikes,

$$y(t) = y(t-1) \cdot (t-1)/t + s(t) \cdot (1/t) \qquad (6)$$

Another coding scheme used in ANN-to-SNN conversion literatures is phase coding (Liu et al., 2022a; Li et al., 2021b; Kim et al., 2018). Phase coding encodes information in the phase of spikes, which correlates with internal oscillation rhythms (Guo et al., 2021b). Phase coding-based techniques in SNN assign the weight $W_i = (1/2)^i$ to the phase i, similar to binary digits (Kim et al., 2018). To simplify the theoretical analysis, we generalize the phase coding into an arbitrary base $\tau \in \mathbb{R}^+$ of weight $W_i = \frac{1}{\tau}\left(\frac{\tau-1}{\tau}\right)^{i-1}$ and an infinite-length period. We define it as exponential moving average (EMA) coding since its streaming update over a spike train can be defined as follows.

$$y(t) = y(t-1) \cdot (\tau - 1)/\tau + s(t) \cdot 1/\tau \qquad (7)$$

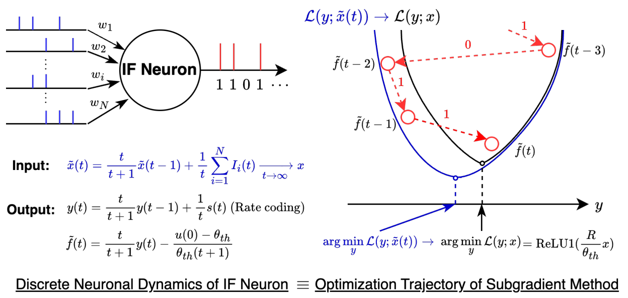

w_1
w_2
w_i
w_N
IF Neuron
1 1 0 1 ···
Input: $\tilde{x}(t) = \frac{t}{t+1}\tilde{x}(t-1) + \frac{1}{t}\sum_{i=1}^{N} I_i(t) \xrightarrow[t\to\infty]{} x$
Output: $y(t) = \frac{t}{t+1}y(t-1) + \frac{1}{t}s(t)$ (Rate coding)
$\tilde{f}(t) = \frac{t}{t+1}y(t) - \frac{u(0)-\theta_{th}}{\theta_{th}(t+1)}$
$\mathcal{L}(y;\tilde{x}(t)) \to \mathcal{L}(y;x)$
$\tilde{f}(t-3)$
$\tilde{f}(t-2)$
$\tilde{f}(t-1)$
$\tilde{f}(t)$
1
0
1
1
y
$\arg\min_y \mathcal{L}(y;\tilde{x}(t)) \to \arg\min_y \mathcal{L}(y;x) = \mathrm{ReLU1}(\frac{R}{\theta_{th}}x)$
Discrete Neuronal Dynamics of IF Neuron ≡ Optimization Trajectory of Subgradient Method

Figure 2. Mathematical equivalence of discrete neuronal dynamics of IF neuron (left) and subgradient method over an unconstrained convex optimization problem (right), described in Theorem 4.1.

4. Optimizer Model of Neuronal Dynamics

In this section, we show that neuronal dynamics of integrate-and-fire models is a first-order iterative optimization process approximating a spike-coded nonlinear function value.

4.1. Theoretical Analysis

We begin by showing that the IF neuron with rate-coded input behaves as a subgradient method with diminishing step sizes to approximate a clipped ReLU, as illustrated in Figure 2. Due to limited space, we list detailed proofs of all the following theorems in the Appendix H. We denote a ReLU clipped at $x = 1$ as ReLU1(x).

Theorem 4.1. *Dynamical system of IF neuron (Eq. 2,3, 4) with rate-coded input $\tilde{x}(t) = \frac{1}{t}\sum_{i=1}^{t} I(i)$ and output $y(t) = \frac{1}{t}\sum_{i=1}^{t} s(i)$ is equivalent to the subgradient method over an optimization problem $\min_{y\in\mathbb{R}} \mathcal{L}(y;x)$, approximated with $x \leftarrow \tilde{x}(t+1)$ as,*

$$\tilde{f}(t) = \tilde{f}(t-1) - \frac{1}{t+1}\cdot \tilde{g}\big(\tilde{f}(t-1);\tilde{x}(t)\big) \tag{8}$$

$$\mathcal{L}(y;x) = h\big(\frac{R}{\theta_{th}}x - y\big) + \frac{1}{2}y^2 \quad \textit{(objective fn.)}$$
$$\tilde{f}(t) = \frac{t}{t+1}y(t) - \frac{u(0)-\theta_{th}}{\theta_{th}(t+1)} \; \textit{(t-th approx.)} \tag{9}$$

where $\tilde{g}(y;x)$ is a subgradient of $\mathcal{L}(y;x)$, $h(x) = ReLU(x)$. Solution of the problem is ReLU1$(\frac{R}{\theta_{th}}x)$. (Proof at H.1)

We now demonstrate that our framework can also predict the behavior of unknown combinations of neuronal dynamics and neural coding, beyond the known combination. We show that the LIF neuron with EMA-coded input behaves as a subgradient method with a constant step size.

Theorem 4.2. *Dynamical system of LIF neuron (Eq. 2,3, 5) with EMA-coded input $\tilde{x}(t)$ and output $y(t)$ (Eq. 7), if $\tau_m = \tau$, is equivalent to subgradient method over an optimization problem $\min_{y\in\mathbb{R}} \mathcal{L}(y;x)$ with $x \leftarrow \tilde{x}(t+1)$,*

$$\tilde{f}(t+1) = \tilde{f}(t) - \frac{1}{\tau}\cdot \tilde{g}\big(\tilde{f}(t);\tilde{x}(t+1)\big)$$
$$\mathcal{L}(y;x) = \frac{1}{2}y^2 + \frac{u_{rest}}{\theta_{th}(\tau-1)}y + h\left(\frac{Rx-\theta_{th}}{\theta_{th}(\tau-1)}x - y\right)$$
$$\tilde{f}(t) = y(t) - \frac{1}{\tau\theta_{th}}(\frac{\tau-1}{\tau})^t u(0) - \frac{u_{rest}\sum_{i=0}^{t}(\frac{\tau-1}{\tau})^{t-i}}{\theta_{th}\tau(\tau-1)}$$

where $\tilde{g}(y;x)$ is a subgradient of $\mathcal{L}(y;x)$, $h(x) =$ ReLU(x). If $u_{rest} = 0$, the solution to the problem is ReLU1$(\frac{R}{\theta_{th}(\tau-1)}x - \frac{1}{\tau-1})$. (Proof at H.2)

4.1.1. Convergence Analysis

Our framework provides a way to understand the asymptotic behavior of spiking neurons by theoretically analyzing its corresponding *optimizer form*. To show this, we conduct a convergence analysis on the optimizer form of an IF neuron to reveal its asymptotic properties. Note that the spike-based information representation discrepates the subgradient estimate $\tilde{g}(y;\tilde{x}(t))$ and the true subgradient $\tilde{g}(y;x)$. Thus, the convergence property of spike train input determines the convergence property of the neuron output.

Theorem 4.3. *Let $f^*(x) = \arg\min_{y\in\mathbb{R}} \mathcal{L}(y;x)$ be the minimizer of $\mathcal{L}(y;x) =$ ReLU$(x-y) + \frac{1}{2}y^2$ over a true input $x\in\mathbb{R}$, and its t-th approximation $\tilde{f}(t)$ be defined as equation 8 and rate-coded input $\tilde{x}(t) = \frac{1}{t}\sum_{i=1}^{t} I(i)$. Denote $h(i) = \big(1 - \sqrt{\frac{i-1}{i+1}}\big)$. If $\|\tilde{f}(t)\| < M \in \mathbb{R}^+$, then the error $\|\tilde{f}(t) - f^*(x)\|$ is upper bounded as follows. (Proof at H.3)*

$$\|\tilde{f}(t) - f^*(x)\|^2 \le \frac{\|\tilde{f}(0) - f^*(x)\|^2}{t+1} + \underbrace{\frac{M+1}{t+1}\sum_{i=1}^{t} h(i)}_{\textit{Nondifferentiability Error}} + \underbrace{\frac{4}{t+1}\sum_{i=1}^{t}\min(\|x-\tilde{x}(i)\|,1)}_{\textit{Input Error}} \tag{10}$$

An assumption $\|\tilde{f}(t)\| < M$ over the optimization trajectory is rarely violated since $\tilde{f}(t)$ is attracted towards the bounded value ReLU1$\big(\tilde{x}(t+1)\big)$, and $\tilde{x}(t) \xrightarrow{t\to\infty} x$. Furthermore, the upper bound can be tighter since the nondifferentiability error term $h(i)$ is non-zero only when the i-th update crosses the singularity $\tilde{x}(t+1)$. We now derive the convergence of neuron output in diverse input setups.

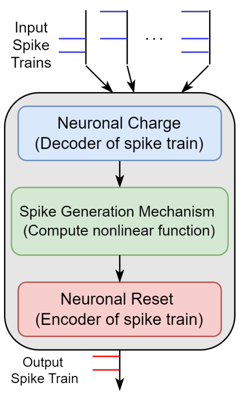
Corollary 4.4. *Let $\|\tilde{f}(t)\| < M \in \mathbb{R}^+$, then (Proof at H.6)*

- *(Exact Input) If $\tilde{x}(t) = x$, then $\|\tilde{f}(t) - f^*\| \to 0$ as $t\to\infty$.*
- *(Deterministic Input) If $\|\tilde{x}(t) - x\| = \mathcal{O}(\frac{1}{t})$, then $\|\tilde{f}(t) - f^*\| \to 0$ as $t\to\infty$.*
- *(Stochastic Input) If $\mathbb{E}[\|\tilde{x}(t) - x\|] = \mathcal{O}(\frac{1}{t})$, then $\mathbb{E}[\|\tilde{f}(t) - f^*\|] \to 0$ as $t\to\infty$.*

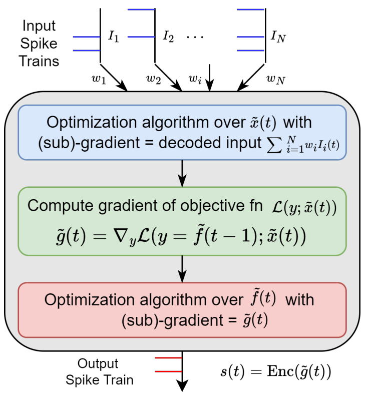
Empirical Validation. To empirically validate our theoretical findings, we conduct toy experiments on a single neuron and compare the output of a neuron and its optimizer form. We use spikingjelly's implementation (Fang et al., 2023) of spiking neurons and Poisson encoding. Figure 11 in Appendix shows that the time-evolution of IF neuron output transformed with Equation 9 exactly equals its optimizer form in theorem 4.1. As Corollary 4.4 states, the decoded output $\tilde{f}(t)$ of IF neuron converges to ReLU1$(\frac{R}{\theta_{th}}x)$ as the input spike train $\tilde{x}(t)$ converges to x. We also verify in Figure 12 that the time-evolution of LIF neuron output matches the output of its subgradient method-based optimizer form in theorem 4.2. Both results show that the asymptotic behavior of spiking neuron outputs follows the known convergence properties of the subgradient method (Boyd et al., 2003). Since the IF neuron with rate-coded input has a non-summable diminishing step size schedule $\eta(t) = \frac{1}{t+1}$ for its optimizer form, the output converges. In contrast, the output only converges up to an error bound since the LIF neuron has a constant step size for its optimizer form.

4.2. Interpretation

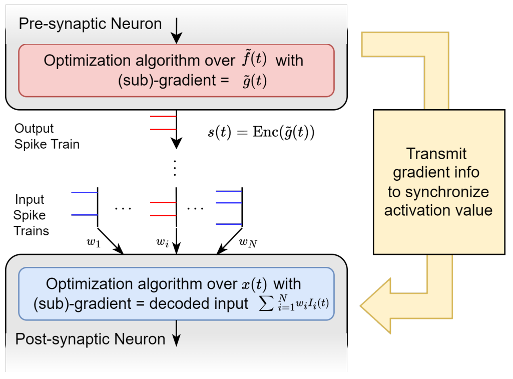
Our framework provides an optimization-theoretic interpretation of SNN's key characteristics in four-fold. First, neuronal dynamics, i.e., a dynamical system of spiking neurons, can be analyzed using its corresponding *optimizer form*. Second, in an end-to-end SNN inference, each spiking


Input Spike Trains
Neuronal Charge (Decoder of spike train)
Spike Generation Mechanism (Compute nonlinear function)
Neuronal Reset (Encoder of spike train)
Output Spike Train

(a) Sequential stages of neuronal dynamics


Input Spike Trains
I_1 I_2 I_N
w_1 w_2 w_i w_N
Optimization algorithm over $\tilde{x}(t)$ with (sub)-gradient = decoded input $\sum_{i=1}^{N} w_i I_i(t)$
Compute gradient of objective fn $\mathcal{L}(y; \tilde{x}(t))$
$\tilde{g}(t) = \nabla_y \mathcal{L}(y = \tilde{f}(t-1); \tilde{x}(t))$
Optimization algorithm over $\tilde{f}(t)$ with (sub)-gradient = $\tilde{g}(t)$
Output Spike Train
$s(t) = \mathrm{Enc}(\tilde{g}(t))$

(b) Optimization-theoretic interpretation of dynamics stages in Fig. 3(a)


Pre-synaptic Neuron
Optimization algorithm over $\tilde{f}(t)$ with (sub)-gradient = $\tilde{g}(t)$
Output Spike Train
$s(t) = \mathrm{Enc}(\tilde{g}(t))$
Input Spike Trains
w_1 w_i w_N
Optimization algorithm over $x(t)$ with (sub)-gradient = decoded input $\sum_{i=1}^{N} w_i I_i(t)$
Post-synaptic Neuron
Transmit gradient info to synchronize activation value

(c) A spike train carries iterative updates (gradients) of a presynaptic neuron's output value to postsynaptic neurons, thereby synchronizing an activation value.

Figure 3. Our interpretation of SNN's computational characteristics: neuronal dynamics (3(a)-3(b)), spike train (3(c)) .

neuron computes a first-order iterative algorithm to approximate a nonlinear function value. The integration phase spike-decodes pre-synaptic currents to approximate an input activation, and the thresholding phase estimates a subgradient from the approximated input. Third, the role of the spike train is to deliver gradient information of a neuron's iterative updates to post-synaptic neurons. Finally, the neural coding scheme and neuronal dynamics jointly determine the learning rate schedule of its optimizer form.

5. SignGD-based Neuronal Dynamics

Spiking neurons, with their optimizer form as the subgradient method, are difficult to approximate arbitrary nonlinear functions. Our framework interprets that a spike train delivers (sub)-gradient information. In detail, a binary spike transfers an activation update step between spiking neurons, and the update step is gradient information $\nabla_y \mathcal{L}(y;x)$ of a certain objective function $\mathcal{L}(y;x)$. However, binarity of a spike $s(t)$ constrains the space of (sub)-gradient $\nabla_y \mathcal{L}(y;x)$ to a bivariate function of $s(t)$ and y, thereby limiting the space of objective function $\mathcal{L}(y;x)$. For example, in Theorem 4.1, $\mathcal{L}(y;x)$ is a form of (ReLU + regularizer) since the subgradient $\nabla_y \mathcal{L}(y;x) = y - s(t) = y - \mathbb{H}(\frac{R}{\theta_{th}}\tilde{x}(t) - y)$. To support arbitrary nonlinear functions, the binarity of a spike should not constrain the space of objective functions. At the same time, the spike-based computational characteristic should be preserved. This implies that the dynamics of the optimizer form of a new spiking neuron should be different from the subgradient method.

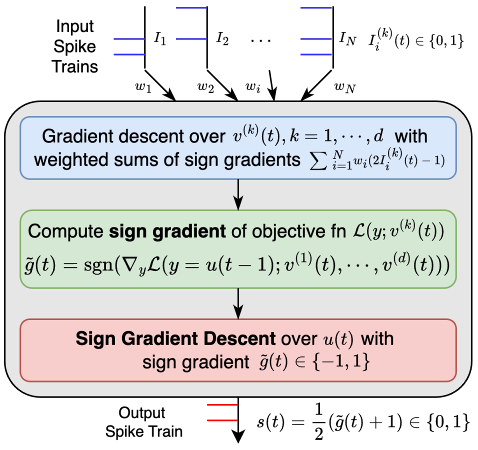

Input Spike Trains
I_1 I_2 I_N $I_i^{(k)}(t) \in \{0,1\}$
w_1 w_2 w_i w_N
Gradient descent over $v^{(k)}(t), k = 1, \cdots, d$ with weighted sums of sign gradients $\sum_{i=1}^{N} w_i (2I_i^{(k)}(t) - 1)$
Compute **sign gradient** of objective fn $\mathcal{L}(y; v^{(k)}(t))$
$\tilde{g}(t) = \mathrm{sgn}(\nabla_y \mathcal{L}(y = u(t-1); v^{(1)}(t), \cdots, v^{(d)}(t)))$
Sign Gradient Descent over $u(t)$ with sign gradient $\tilde{g}(t) \in \{-1, 1\}$
Output Spike Train
$s(t) = \frac{1}{2}(\tilde{g}(t) + 1) \in \{0,1\}$

Figure 4. sign gradient descent(signGD)-based neuronal dynamics, a design optimization-theoretically extended from Fig. 3(b).

We thus present a new sign gradient descent (signGD) - based neuronal dynamics that can (i) easily approximate a larger space of nonlinear functions and (ii) accelerate SNN inference. Our key approaches are two-fold: First, we apply sign gradient descent (signGD) (Bernstein et al., 2018) instead of the subgradient method to design neuronal dynamics. Second, we generalize the learning rate schedule of the optimizer form of neuronal dynamics.

signGD (Bernstein et al., 2018) is a distributed optimization method that cuts down the network communication cost of gradient tensor. Each node transmits only the sign of the gradient instead of the exact (or compressed) gradient to the parameter server. We apply signGD to design new neuronal dynamics and a coding scheme. It preserves the SNN's key characteristic of binary spike-based communication since the optimizer form has a binary update of $\{-1, 1\}$, which easily transforms into a $\{0, 1\}$ spike. Thus, given arbitrary objective function, a binary spike can represent an activation update step, which is the *sign* of the (sub)-gradient.

The learning rate schedule is important for the convergence speed of signGD since the learning rate solely determines the update size. It is difficult to determine optimal step sizes for SNNs, which perform neuron-wise convex optimization, different from a single convex objective. For instance, schedules with fast decay make approximation faster for neurons in frontal SNN layers but slower for lateral SNN layers, since neurons in lateral SNN layers receive accurate gradients in later time steps. Thus, learning rate scheduling should consider the approximation speed of entire SNN neurons end-to-end. We hence generalize the learning rate schedule of the optimizer form of signGD-based neuron and empirically search local optimum. Specifically, we define our signGD-based neuron and coding scheme as follows.

Definition 5.1. (Signed schedule coding) Let a spike train $s(t) \in \{0,1\}$, a step size schedule $\eta(t) \in \mathbb{R}^+$ for $t \in \mathbb{N}$ and $y(0) = 0$. Signed schedule coding with $\eta(t)$ decodes an activation $y(t)$ from $s(t)$ as

$$y(t) = y(t-1) - \eta(t)\big(2 \cdot s(t) - 1\big) \tag{11}$$

This coding scheme interprets a binary spike $s(t) \in \{0,1\}$ as a sign information $\{-1,1\}$ with a simple mathematical formula $2s(t) - 1$. We list three spike encoding algorithms for our signed schedule coding scheme in Appendix D.

Definition 5.2. (SignGD-based neuronal dynamics) Let a smooth objective function $\mathcal{L}(y; x_1, \cdots, x_d) : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$, positive coefficients $\alpha_i(t) \in \mathbb{R}^+$, $\beta_i(t) \in \mathbb{R}^+$, and step size schedule $\eta(t) \in \mathbb{R}^+$ for $i = 1, 2$ and $t \in \mathbb{N}$. Suppose an influx current $I^{(k)}(t)$ of k-th operand in time t is a sum of weighted spike trains, i.e., for real weights W_i,

$$I^{(k)}(t) = \sum_{j=1}^{N} W_j I_j^{(k)}(t), \quad I_j^{(k)} : \mathbb{N} \to \{0,1\} \tag{12}$$

We denote $W = \sum_{i=1}^{N} W_i$. Then the signGD-based neuronal dynamics over two internal variables $u(t) \in \mathbb{R}$ and $v(t) = \big(v^{(1)}(t), \cdots, v^{(k)}(t), \cdots, v^{(d)}(t)\big) \in \mathbb{R}^d$ is

$$v^{(k)}(t+1) = \alpha_1(t)v^{(k)}(t) - \alpha_2(t+1)(2I^{(k)}(t+1) - W)$$

$$s(t) = \mathbb{H}\bigg(\nabla_y \mathcal{L}\Big(\frac{\eta(t-1)}{\beta_2(t-1)}u(t); \frac{\eta(t)}{\alpha_2(t)}v(t)\Big)\bigg) \tag{13}$$

$$u(t+1) = \beta_1(t)u(t) - \beta_2(t)(2s(t) - 1) \tag{14}$$

The dynamics coefficients $\alpha_i(t)$ and $\beta_i(t)$ with $i = 1, 2$ are time-scheduled, and remains constant irrespective of the input current $I^{(k)}(t)$ or any internal variables $u(t)$ or $v^{(k)}(t)$. Hence, we can pre-compute and load these values up to a specific time-step T. The term W translates a weighted sum of $\{0,1\}$ spikes into a weighted sum of sign gradients $\{-1,1\}$. In practice, W is pre-computed in three steps: (i) stimulate all neurons to spike for a single step and record the influx current I^+, (ii) depress all neurons to not spike for a single step and record the influx current $I-$, (iii) use I^+ and I^- to compute W neuron-wise. Figure 4 illustrates the optimization-theoretic abstraction of our spiking neuronal dynamics. Below, we show that our signGD-based neuronal dynamics is equivalent to the signGD algorithm.

Theorem 5.3. *Let an input activation $\tilde{x}(t)$ be signed schedule coded over an N weighted input spike trains, i.e.,*

$$\tilde{x}^{(k)}(t) = \tilde{x}^{(k)}(t-1) - \sum_{i=1}^{N} W_i \Big(\eta(t)(2I_i^{(k)}(t) - 1)\Big)$$

Output $\tilde{f}(t) = \tilde{f}(t-1) - \eta(t)\big(2 \cdot s(t) - 1\big)$ is signed schedule coded with $s(t)$ of signGD-based neuronal dynamics.
If α_1, α_2, β_1, β_2 and η satisfies $\eta(1) = \alpha_2(1) = \beta_2(1)$ and

$$\frac{\eta(t)}{\eta(t-1)} = \frac{\beta_1(t)\beta_2(t)}{\beta_2(t-1)} = \frac{\alpha_2(t)}{\alpha_1(t-1)\alpha_2(t-1)} \tag{15}$$

Then the dynamical system of $\tilde{f}(t)$ is equivalent to the sign gradient descent method formulated as, (Proof at H.7)

$$\tilde{f}(t) = \tilde{f}(t-1) - \eta(t) sgn(\nabla_y \mathcal{L}(\tilde{f}(t-1); \tilde{x}(t))) \tag{16}$$

We now demonstrate that our neuron can support spike train-based evaluation of novel nonlinear functions.

5.1. Single-operand Nonlinearities

We first approximate single-operand nonlinear functions: ReLU, LeakyReLU, and GELU. We utilize a simple objective function $\mathcal{L}(y; x) = \frac{1}{2}\|y - f(x)\|^2$, which is smooth and convex over y. Note that the choice of nonlinear function only affects the firing mechanism (Eq. 13) of our signGD-based neuronal dynamics. Also, we approximate the exact ReLU instead of the clipped ReLU1. Figure 8 in Appendix shows toy experiments validating that our signGD-based neuron approximates the unary nonlinearities accurately, and the learning rate schedule affects the convergence speed.

Corollary 5.4. *SignGD-based neuronal dynamics (Def. 5.2) satisfying Eq. 15 and $\eta(t) = \alpha_2(t) = \beta_2(t)$ is equivalent to signGD (Eq. 16) if $s(t)$ (Eq. 13) satisfies, (Proof at H.8)*

- ***(ReLU)*** *$\mathcal{L}(y; x) = \frac{1}{2}\|y - ReLU(x)\|^2$ and $s(t) = \mathbb{H}\big(v(t)\big)\mathbb{H}\big(u(t) - \beta_1(t)v(t)\big) + \mathbb{H}\big(-v(t)\big)\mathbb{H}\big(u(t)\big)$*
- ***(Sigmoid approximation of GELU)*** *(Hendrycks & Gimpel, 2016)) $\mathcal{L}(y; x) = \frac{1}{2}\|y - \frac{x}{1+e^{-1.702x}}\|^2$ and $s(t) = \mathbb{H}\big((1 + (e^{-1.702})^{v(t)})u(t) - \beta_1(t)v(t)\big)$*
- ***(LeakyReLU)*** *$\mathcal{L}(y; x) = \frac{1}{2}\|y - LeakyReLU(x, \delta)\|^2$, where δ is the negative slope, and $s(t) = \mathbb{H}(v(t))\mathbb{H}\big(u(t) - \beta_1(t)v(t)\big) + \mathbb{H}(-v(t))\mathbb{H}\big(u(t) - \delta\beta_1(t)v(t)\big)$*

5.2. Multi-operand Nonlinearities

In this section, we approximate two tensor operators, max pooling and layer normalization (Ba et al., 2016), with binary-input signGD-based neuronal dynamics ($d = 2$). We also empirically validate the multi-operand nonlinearity approximation with our neuron in Appendix C and Figure 9.

5.2.1. Max Pooling

The max pooling operator downsamples a feature map with a maximum value for each patch in CNN (He et al., 2015; Simonyan & Zisserman, 2014; Tan & Le, 2019). Max pooling approximation in SNN has been a long-standing problem

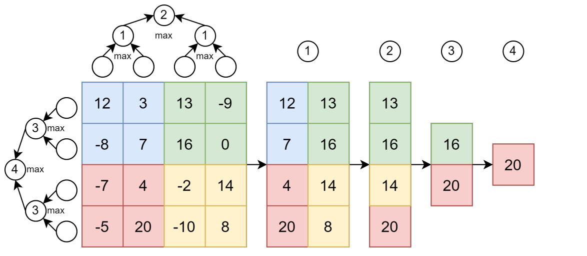


Figure 5. An example of decomposing max pooling of 4x4 window with binary-input maximum operators.

(Gaurav et al., 2022) since it should anticipate the maximum of spike trains ahead of time. A common strategy is to replace it with average pooling (Sengupta et al., 2018; Li et al., 2021a), which degrades the performance (Rueckauer et al., 2017). Other works compute the instantaneous maximum instead of exact maximum (Gaurav et al., 2022; Guo et al., 2019), use TTFS coding (Stanojevic et al., 2023), or sacrifice SNN's characteristics, e.g., spike-based (Li et al., 2022) or event-driven computation (Lu & Xu, 2022).

Our signGD-based neuron can approximate the exact maximum in an event-driven manner, thereby supporting the max pooling operation. If a neuron can evaluate the maximum over two input spike trains, then we can construct a binary computational tree over row and column dimension of the pooling kernel, as in Figure 5. If a kernel size is $K_r \times K_c$, the tree depth of max neurons is $\lceil \log_2 K_r \rceil \times \lceil \log_2 K_r \rceil$. We approximate maximum function over two activations with a binary-input signGD-based neuron defined as follows.

Corollary 5.5. *SignGD (Eq. 16) with $\mathcal{L}(y; x_1, x_2) = \frac{1}{2}\|y - \max(x_1, x_2)\|^2$ is equivalent to signGD-based neuronal dynamics (Def. 5.2) satisfying Eq. 15, $\alpha_2(t) = \beta_2(t)$,*

$$s(t) = \mathbb{H}(v^{(1)}(t) - v^{(2)}(t))(u(t) - \beta_1(t)v^{(1)}(t)) + \mathbb{H}(v^{(2)}(t) - v^{(1)}(t))(u(t) - \beta_1(t)v^{(2)}(t))$$

5.2.2. Layer Normalization

Normalization techniques stabilize and accelerate the training of DNN models (Huang et al., 2023). Batch normalization (Ioffe & Szegedy, 2015) is supported for SNN inference since batch statistics are fixed in the inference phase; hence, it becomes an affine operator (Li et al., 2021a). It is not the case for layer normalization (Ba et al., 2016), an operator employed at high-performance DNNs, e.g., Transformer (Vaswani et al., 2017), ConvNext (Liu et al., 2022b), and MLP-Mixer (Tolstikhin et al., 2021). Layer normalization in SNN should evaluate data instance-wise channel statistics across multiple spike trains ahead of time. This requires event-driven spike-based computation of intricate nonlinearities, e.g., variance and inverse square root, which was infeasible with prior spike neurons.

To approximate layer normalization with our signGD-based neuron, we decompose the operator into a computational graph of linear operators and two nonlinearities, as in Figure 6. Linear operators can be implemented in SNN or be fused to other linear layers. Two key nonlinearities are square function $h_1(x) = x^2$ and multiply-inverse-sqrt function $h_2(x_1, x_2) = \frac{x_1}{\sqrt{x2}}$. We derive signGD-based neurons that can approximate these nonlinearities as follows.

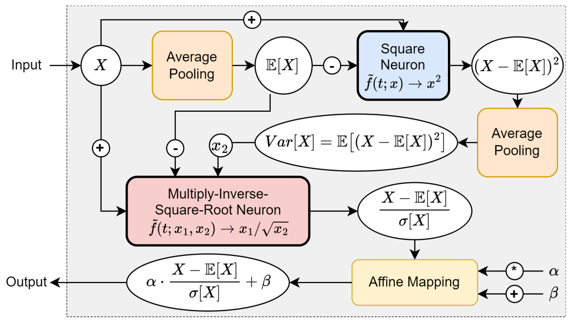

Input
X
Average Pooling
$\mathbb{E}[X]$
Square Neuron $\tilde{f}(t;x) \to x^2$
$(X - \mathbb{E}[X])^2$
Average Pooling
$Var[X] = \mathbb{E}[(X-\mathbb{E}[X])^2]$
x_2
Multiply-Inverse-Square-Root Neuron $\tilde{f}(t;x_1,x_2) \to x_1/\sqrt{x_2}$
$\frac{X-\mathbb{E}[X]}{\sigma[X]}$
Affine Mapping
α
β
Output
$\alpha \cdot \frac{X-\mathbb{E}[X]}{\sigma[X]} + \beta$

Figure 6. Decomposing layer normalization (Ba et al., 2016) into operators that can be approximated with signGD-based spiking neurons (Corollary 5.6, 5.7) and affine connections.

Corollary 5.6. *SignGD (Eq. 16) with $\mathcal{L}(y; x) = \frac{1}{2}\|y - x^2\|^2$ is equivalent to the signGD-based neuronal dynamics (Def. 5.2) satisfying Eq. 15, $\eta(t) = \beta_2(t) = \alpha_2(t)$ and $s(t) = \mathbb{H}\big(u(t) - \beta_1(t)v(t)^2\big)$ (Proof at H.10)*

Corollary 5.7. *SignGD (Eq. 16) with $\mathcal{L}(y; x_1, x_2) = \|y - \frac{x_1}{\sqrt{x_2}}\|^2$ is equivalent to the signGD-based neuron (Def. 5.2) satisfying Eq. 15, $\eta(t) = \alpha_2(t) = \beta_2(t)$, (Proof at H.11)*

$$s(t) = \mathbb{H}\big(u(t)\big)\mathbb{H}\big(v^{(1)}(t)\big)\mathbb{H}\big(v^{(2)}(t)u(t)^2 - v^{(1)}(t)^2\big) + \mathbb{H}\big(-u(t)\big)\mathbb{H}\big(-v^{(1)}(t)\big)\mathbb{H}\big(v^{(1)}(t)^2 - v^{(2)}(t)u(t)^2\big) + \mathbb{H}(u(t))\mathbb{H}(-v^{(1)}(t))$$

6. Evaluations

We demonstrate the practical effectiveness of our signGD-based neuronal dynamics in four-fold. First, we validate our support for diverse nonlinearities by converting new DNN architectures. Second, we compare the accuracy of converted ANNs with existing conversion techniques. Third, we verify our design choices through ablation studies. Finally, we visualize the effect of our technique on SNN inference speed. We also conduct an energy consumption analysis of our technique in Appendix G. For a fair comparison, we generalize the learning rate schedule of the subgradient method-based neuron, which is formulated in Appendix I. We detail our conversion technique and its spikingjelly (Fang et al., 2023) implementation in Appendix F.

6.1. Diversifying DNN Architecture Support

To verify that our signGD-based neuronal dynamics can approximate diverse nonlinearities, we convert large-scale DNN architectures that prior works fail to convert. We

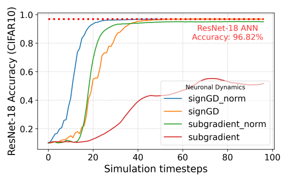
Table 1. Comparing ANN-to-SNN conversion performance on ImageNet (Deng et al., 2009) models. *Exact Arch* means the trained ANN architecture is identical to its original paper, without any customization or tailoring. *No Spike-aware Activation Func* means ReLU functions in ANN architecture are not replaced with spike-aware functions before training, e.g., QCFS (Bu et al., 2022), SlipReLU (Jiang et al., 2023). For our signGD-based neuron, we use $\eta(t) = \frac{5.0}{t+1}$ for ConvNext and MLP-Mixer, and $\eta(t) = 0.15 \cdot 0.965^t$ for the others. Results of RTS (Deng & Gu, 2021) are from (Li et al., 2021a).

Methods	Exact Arch.	No Spike-aware Activation Func.	ANN Acc.	Simulation time-steps T = 32	T = 64	T = 128	T = 256
ResNet-34 (He et al., 2015) ImageNet							
TSC (Han & Roy, 2020)	✗	✓	70.20	-	-	-	55.65
RTS (Deng & Gu, 2021)	✗	✗	75.66	0.09	0.12	3.19	47.11
SNNC-AP (Li et al., 2021a)	✗	✓	75.66	64.54	71.12	73.45	74.61
QCFS (Bu et al., 2022)	✗	✗	74.32	**69.37**	72.35	73.15	73.37
SlipReLU (Jiang et al., 2023)	✗	✗	75.08	66.61	72.71	74.01	-
SRP (Hao et al., 2023)	✗	✗	74.32	68.40	68.61	-	-
Ours (with Max Pooling)	✓	✓	73.30	58.09	72.38	73.31	73.29
Ours (without Max Pooling)	✗	✓	75.65	59.85	**75.34**	**75.67**	**75.65**
VGG-16 (Simonyan & Zisserman, 2014) ImageNet							
TSC (Han & Roy, 2020)	✗	✓	73.49	-	-	-	69.71
RTS (Deng & Gu, 2021)	✗	✗	75.36	0.114	0.118	0.122	1.81
SNNC-AP (Li et al., 2021a)	✗	✓	75.36	63.64	70.69	73.32	74.23
SNM (Wang et al., 2022)	✗	✓	73.18	64.78	71.50	72.86	-
QCFS (Bu et al., 2022)	✗	✗	74.29	68.47	72.85	73.97	74.22
SlipReLU (Jiang et al., 2023)	✗	✗	71.99	67.48	71.25	72.02	-
SRP (Hao et al., 2023)	✗	✗	74.29	**69.35**	69.43	-	-
Ours (with Max Pooling)	✓	✓	73.36	38.08	67.04	71.33	71.50
Ours (without Max Pooling)	✗	✓	75.35	69.16	**75.32**	**75.31**	**75.34**
RegNetX (Radosavovic et al., 2020) ImageNet							
RTS (Deng & Gu, 2021)	✗	✗	80.02	0.218	3.542	48.60	71.22
SNNC-AP (Li et al., 2021a)	✗	✓	80.02	**55.70**	70.96	75.78	77.50
Ours (RegNetX-3.2GF)	✓	✓	81.19	26.85	**77.74**	**80.93**	**80.99**
New DNN architectures converted with Our signGD-based neuron							
ResMLP-S24 (Touvron et al., 2021)	✓	✓	80.76	**72.94**	**76.91**	**77.99**	78.04
ConvNext-B (Liu et al., 2022b)	✓	✓	84.06	0.11	5.07	72.60	**81.07**
MLP-Mixer-B32 (Tolstikhin et al., 2021)	✓	✓	76.59	0.11	0.35	50.06	72.97

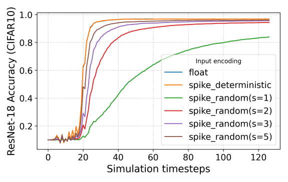
experiment with the latest non-transformer architectures: MLP-Mixer (Tolstikhin et al., 2021), ResMLP (Touvron et al., 2021), ConvNext (Liu et al., 2022b), ResNet34, and VGG16 with max pooling. Note that GELU conversion should be supported for ResMLP, MLP-Mixer, and ConvNext. Layer normalization should be converted for MLP-Mixer and ConvNext. Table 1 shows that our signGD-based neuron enables conversion of these architectures for the first time. Converted ResNet and VGG with max pooling layer achieves $> 70\%$ accuracy in $T \geq 128$. MLP-Mixer and ConvNext take comparably more time-steps slow due to the layer normalization, as expected in Appendix C. ResMLP approximates its ANN accuracy the fastest among all DNN models, achieving 72% in $T = 32$ and 78% in $T = 128$.

6.2. Comparison with Prior Works

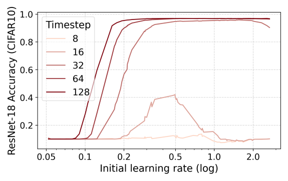
To show the practicality of our signGD-based neuron, we convert DNN to SNN with our neuron and compare its performance with prior techniques. We use ImageNet (Deng et al., 2009) and CIFAR (Krizhevsky et al., 2009) datasets. For ImageNet models, we use pretrained weights of (Li et al., 2021a) for a fair comparison. Table 1 shows that our converted SNNs accurately approximate the ANN performance on $T \geq 64$, achieving the best performance among all conversion techniques. In $T \leq 32$, spike-aware ANN activation works (Bu et al., 2022; Jiang et al., 2023; Hao et al., 2023) achieve higher accuracy but significantly sacrifices accuracy in later time-steps $T \geq 64$ with $1.97 - 5.71\%$. Such an accurate SNN acceleration is possible since our neuron approximates the true ReLU instead of clipped ReLU and does not fine-tune ANN activations for early convergence. Also, for CIFAR models, Tables 4 and 5 show that our converted SNN outperforms all the existing works in $T \geq 64$, and also in $T \leq 32$ except for spike-aware ANN activation techniques. Our SNN performance approaches the ANN in $T \geq 32$ with a small gap of $< 1\%$. Note that prior studies used specialized techniques for low time-steps $T \leq 32$, e.g., data-driven calibration (Li et al., 2021a) or spike-aware activation functions (Bu et al., 2022; Jiang et al., 2023). We do not apply these methods to isolate and directly compare the effect of neuron models, and minimize the influence of specific conversion techniques.



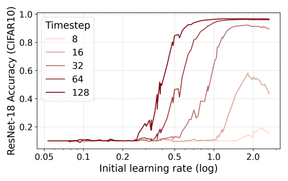
(a) Effect of neuronal dynamics and normalization. Exponential schedule $\eta(t) = 0.95 * (0.15)^t$.



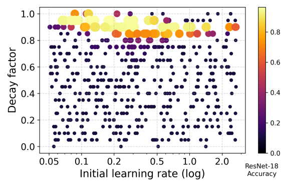
(b) Effect of input encoding through timestep. SignGD-based neuron with $\eta(t) = \frac{1}{t+1}$.



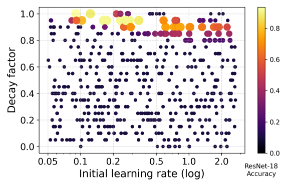
(c) Effect of LR on signGD-based neuron, inverse schedule.



(d) Effect of LR on subgradient-based neuron, inverse schedule.



(e) Effect of LR and decay factor on signGD-based neuron, exponential schedule.



(f) Effect of LR and decay factor on subgradient-based neuron, exponential schedule.

Figure 7. Ablation studies on hyper-parameters of signGD-based (Def. 5.2) and subgradient-based neuronal dynamics (Def. I.2). SNN inference accuracy is measured with ResNet-18 model (Fang et al., 2021) on CIFAR-10 dataset (Krizhevsky et al., 2009).

6.3. Ablation Study

Effect of Neuronal Dynamics. We first validate the acceleration effect of signGD-based dynamics in ANN-to-SNN conversion. In Figure 7(a), we test two variables: the choice of neuronal dynamics and the normalization on ReLU layer. Results show that signGD-based neuron approximates noticeably faster than the subgradient-based neuron. Although the normalization helps, it is auxiliary for our signGD-based neuron, different from subgradient-based neuron. The normalization is critical for the subgradient-based neuron since it approximates the clipped ReLU, not the true ReLU.

Input Encoding Scheme. Figure 7(b) compares the effect of three input encoding schemes (See Appendix D.) on the SNN accuracy. To observe the impact of stochasticity, we also vary the parameter s of the stochastic encoding. Results show that the performance gap is marginal between float and deterministic encoding, but it is noticeable with the stochastic encoding. Marginal performance drop is desirable with deterministic encoding, since it is easier and more efficient to hardware-implement than float encoding.

Case study: Inverse schedule. To generalize the acceleration capability of our signGD-based dynamics, we fix the learning rate schedule and compare the converted SNN performance of signGD-based and subgradient-based neurons. As a case study, we first test an inverse LR schedule. In Figure 7(c)-7(d), we measure the time-evolution of inference accuracy varying the initial learning rate (X-axis). Figures show that compared to subgradient-based neuron, our signGD-based neuron provides (i) faster convergence, in a same initial learning rate, of converted SNN's inference accuarcy towards ANN, and (ii) wider coverage of initial learning rate that leads to high-performance SNN.

Case study: Exponential schedule. We also compare two neuronal dynamics in the case of an exponential schedule. We evaluate the joint effect of two hyper-parameters, the initial learning rate and the decay factor. Similar to the inverse schedule case, Figures 7(e)-7(f) show that signGD-based neurons have a wider coverage of hyper-parameter combinations that achieve high inference accuracy. Also, the best accuracy is $> 4\%$ higher with signGD-based neuron; The best accuracy in time-step $T = 32$ is 91.4% for subgradient-based neuron at $\eta(t) = 0.163 \times (0.95)^t$ and 96.2% for signGD-based neuron at $\eta(t) = 0.177 \times (0.95)^t$.

6.4. Visualization of Layer-wise Optimization Flow

To qualitatively demonstrate the effect of our neuronal dynamics, Figure 13 in the Appendix shows the time-evolution of layer-wise ANN-SNN activation error. The figure shows that our signGD-based neuron (i) reduces the layer-wise conversion error more rapidly and (ii) accumulates less conversion error through layers, compared to the subgradient-based neuron. In each layer, the error of signGD-based neurons converges faster than the subgradient-based neurons with the same learning rate schedule. In Figure 13(b) and 13(d), ANN-to-SNN conversion with subgradient-based neurons accumulates error through layers. In contrast, the time-evolution of layer-wise error in signGD-based neurons is relatively uniform through layers.

7. Conclusion

This paper constructs a novel optimization-theoretic framework on the discrete dynamical system of spiking neurons. Based on the framework, we develop new signGD-based neuronal dynamics that approximate various nonlinearities beyond ReLU, e.g., GELU, LeakyReLU, max pooling, and layer normalization. Using our neuron, we (i) successfully convert new high-performance DNN architectures for the first time, e.g., ConvNeXt, MLP-Mixer, and ResMLP, and (ii) achieve state-of-the-art accuracy of converted ResNet34 and VGG16 in low time steps $T = 64$. We list our discussions and future works in the Appendix B.

Acknowledgements

This work was partly supported by National Research Foundation of Korea (NRF) grant (No. 2022R1A2C3008495) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant [NO.2021-0-01343-004, Artificial Intelligence Graduate School Program (Seoul National University)] funded by the Korean government (MSIT). We thank Minjae Kim for valuable feedback on writing the manuscript.

Impact Statement

This paper presents a work that aims to broaden the applicability of technical advances in machine learning and biologically inspired artificial intelligence. A key advantage of SNN, which is the resource efficiency, would benefit a wide range of real-world AI applications. Resource-constrained mobile and IoT applications would especially benefit from this work by enabling high-performance intelligent inference with low-power pervasive sensing. Such an efficient AI inference can be widespread for socially underprivileged people who cannot afford large energy bills of embedded AI devices, e.g., immigrant parents getting AI assistance to nurture the linguistic skills of an under-developed child (Kwon et al., 2022), or an AI chatbot to help the isolated elderly cope with their loneliness (Valtolina & Hu, 2021). Another important societal advantage of energy-efficient AI is that it leads to environmental sustainability. The super-linear growth of AI techniques also scaled its ecological impact as an energy footprint (Wu et al., 2022). Energy footprint directly leads to carbon emissions, i.e., carbon footprints, negatively impacting the atmosphere. SNN, with its orders-of-magnitude energy efficiency, can make DNN more environmentally sustainable while preserving the model's performance with this paper's approach.

Our theoretical findings may contribute to understanding the information dynamics of biological neuron models with refractory periods. Optimistically, this would foster neuroscientific studies in the human brain to develop diverse accessibility applications for the neurologically disabled, e.g., brain-computer interface for the motion-disabled (Gao et al., 2003), or a thought-controlled device to restore hand grasps of a paralyzed hand (Pfurtscheller et al., 2003). As a potential ethically negative impact, scientific advances in the overall neuroscience domain may enable decoding electronic signals of the human brain in the far future, thereby reading a person's mind. However, developing such a dangerous application is clearly out of the scope of this paper.

References

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G.-J., et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):3625, 2020.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Boyd, S., Xiao, L., and Mutapcic, A. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004(01), 2003.

Brette, R. and Gerstner, W. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of neurophysiology*, 94(5):3637–3642, 2005.

Bu, T., Fang, W., Ding, J., DAI, P., Yu, Z., and Huang, T. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2022.

Cao, Y., Chen, Y., and Khosla, D. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113: 54–66, 2015.

Cessac, B. A discrete time neural network model with spiking neurons: rigorous results on the spontaneous dynamics. *Journal of Mathematical Biology*, 56:311–345, 2008.

Chizhov, A. V. and Graham, L. J. A strategy for mapping biophysical to abstract neuronal network models applied to primary visual cortex. *PLoS Computational Biology*, 17(8):e1009007, 2021.

Christensen, D. V., Dittmann, R., Linares-Barranco, B., Sebastian, A., Gallo, M. L., Redaelli, A., Slesazeck, S., Mikolajick, T., Spiga, S., Menzel, S., Valov, I. I., Milano, G., Ricciardi, C., Liang, S.-J., Miao, F., Lanza, M., Quill, T. J., Keene, S. T., Salleo, A., Grollier, J., Markovi'c, D., Mizrahi, A., Yao, P., Yang, J. J., Indiveri, G., Strachan, J. P., Datta, S., Vianello, E., Valentian, A., Feldmann, J., Li, X., Pernice, W. H. P., Bhaskaran, H., Furber, S. B., Neftci, E. O., Scherr, F., Maass, W., Ramaswamy, S., Tapson, J. C., Panda, P., Kim, Y., Tanaka, G., Thorpe, S. J., Bartolozzi, C., Cleland, T. A., Posch, C., Liu, S.-C., Panuccio, G., Mahmud, M., Mazumder, A. N., Hosseini, M. D. M., Mohsenin, T., Donati, E., Tolu, S., Galeazzi, R., Christensen, M. E., Holm, S., Ielmini, D., and Pryds, N. H. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2, 2021.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Deng, S.-W. and Gu, S. Optimal conversion of conventional artificial neural networks to spiking neural networks. *ArXiv*, abs/2103.00476, 2021.

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International joint conference on neural networks (IJCNN)*, pp. 1–8. ieee, 2015.

Dong, Y., Zhao, D., Li, Y., and Zeng, Y. An unsupervised stdp-based spiking neural network inspired by biologically plausible learning rules and connections. *Neural networks : the official journal of the International Neural Network Society*, 165:799–808, 2022.

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., and Tian, Y. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021.

Fang, W., Chen, Y., Ding, J., Yu, Z., Masquelier, T., Chen, D., Huang, L., Zhou, H., Li, G., and Tian, Y. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023. doi: 10.1126/sciadv.adi1480. URL https://www.science.org/doi/abs/10.1126/sciadv.adi1480.

Fang, X., Duan, S., and Wang, L. Memristive izhikevich spiking neuron model and its application in oscillatory associative memory. *Frontiers in Neuroscience*, 16:885322, 2022.

Fourcaud-Trocmé, N., Hansel, D., Van Vreeswijk, C., and Brunel, N. How spike generation mechanisms determine the neuronal response to fluctuating inputs. *Journal of neuroscience*, 23(37):11628–11640, 2003.

Frenkel, C. and Indiveri, G. Reckon: A 28nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 65, pp. 1–3. IEEE, 2022.

Furber, S. B., Galluppi, F., Temple, S., and Plana, L. A. The spinnaker project. *Proceedings of the IEEE*, 102(5): 652–665, 2014a.

Furber, S. B., Galluppi, F., Temple, S., and Plana, L. A. The spinnaker project. *Proceedings of the IEEE*, 102(5): 652–665, 2014b. doi: 10.1109/JPROC.2014.2304638.

Gao, X., Xu, D., Cheng, M., and Gao, S. A bci-based environmental controller for the motion-disabled. *IEEE Transactions on neural systems and rehabilitation engineering*, 11(2):137–140, 2003.

Gaurav, R., Tripp, B., and Narayan, A. Spiking approximations of the maxpooling operation in deep snns. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2022.

Gerstner, W., Kistler, W. M., Naud, R., and Paninski, L. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Ghosh-Dastidar, S. and Adeli, H. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.

Glanzman, D. L. Habituation in aplysia: the cheshire cat of neurobiology. *Neurobiology of learning and memory*, 92 (2):147–154, 2009.

Guo, S., Wang, L., Chen, B., and Dou, Q. An overhead-free max-pooling method for snn. *IEEE Embedded Systems Letters*, 12(1):21–24, 2019.

Guo, T., Pan, K., Sun, B., Wei, L., Yan, Y., Zhou, Y., and Wu, Y. Adjustable leaky-integrate-and-fire neurons based on memristor-coupled capacitors. *Materials Today Advances*, 12:100192, 2021a.

Guo, W., Fouda, M. E., Eltawil, A. M., and Salama, K. N. Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems. *Frontiers in Neuroscience*, 15:638474, 2021b.

Han, B. and Roy, K. Deep spiking neural network: Energy efficiency through time based coding. In *European Conference on Computer Vision*, pp. 388–404. Springer, 2020.

Han, B., Srinivasan, G., and Roy, K. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. *2020*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13555–13564, 2020.

Hao, Z., Bu, T., Ding, J., Huang, T., and Yu, Z. Reducing ann-snn conversion error through residual membrane potential. In *AAAI Conference on Artificial Intelligence*, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Hinton, G. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Hodgkin, A. L. and Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117 (4):500, 1952.

Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pp. 10–14. IEEE, 2014.

Huang, L., Qin, J., Zhou, Y., Zhu, F., Liu, L., and Shao, L. Normalization techniques in training dnns: Methodology, analysis and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Izhikevich, E. M. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

Jiang, H., Anumasa, S., Masi, G. D., Xiong, H., and Gu, B. A unified optimization framework of ann-snn conversion: Towards optimal mapping from activation values to firing rates. In *International Conference on Machine Learning*, 2023.

Kasabov, N. K. Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*, 52:62–76, 2014.

Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., and Masquelier, T. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018.

Kim, J., Kim, H., Huh, S., Lee, J., and Choi, K. Deep neural networks with weighted spikes. *Neurocomputing*, 311: 373–386, 2018.

Kim, S., Park, S., Na, B., and Yoon, S. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 11270–11277, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kundu, S., Pedram, M., and Beerel, P. A. Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5209–5218, 2021.

Kwon, T., Jeong, M., Ko, E.-S., and Lee, Y. Captivate! contextual language guidance for parent–child interaction. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pp. 1–17, 2022.

Li, C., Ma, L., and Furber, S. Quantization framework for fast spiking neural networks. *Frontiers in Neuroscience*, 16:918793, 2022.

Li, Y., Deng, S., Dong, X., Gong, R., and Gu, S. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International conference on machine learning*, pp. 6316–6325. PMLR, 2021a.

Li, Y., Zeng, Y., and Zhao, D. Bsnn: Towards faster and better conversion of artificial neural networks to spiking neural networks with bistable neurons. *Frontiers in Neuroscience*, 16, 2021b.

Liu, F., Zhao, W., Chen, Y., Wang, Z., and Jiang, L. Spike-converter: An efficient conversion framework zipping the gap between artificial neural networks and spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 1692–1701, 2022a.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022b.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Lu, S. and Xu, F. Linear leaky-integrate-and-fire neuron model based spiking neural networks and its mapping relationship to deep neural networks. *Frontiers in neuroscience*, 16:857513, 2022.

Maass, W. Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8: 1–40, 1996a.

Maass, W. Networks of spiking neurons: The third generation of neural network models. *Electron. Colloquium Comput. Complex.*, TR96, 1996b.

maintainers, T. and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016.

Mancoo, A., Keemink, S., and Machens, C. K. Understanding spiking networks through convex optimization. *Advances in Neural Information Processing Systems*, 33: 8824–8835, 2020.

Mayr, C., Hoeppner, S., and Furber, S. Spinnaker 2: A 10 million core processor system for brain simulation and machine learning. *arXiv preprint arXiv:1911.02385*, 2019.

Neftci, E. O., Mostafa, H., and Zenke, F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36:51–63, 2019.

Niesen, J. and Hall, T. *On the global error of discretization methods for ordinary differential equations*. PhD thesis, University of Cambridge, 2004.

Park, S., Kim, S., Na, B., and Yoon, S. T2fsnn: Deep spiking neural networks with time-to-first-spike coding. *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2020.

Pfurtscheller, G., Müller, G. R., Pfurtscheller, J., Gerner, H. J., and Rupp, R. 'thought'–control of functional electrical stimulation to restore hand grasp in a patient with tetraplegia. *Neuroscience letters*, 351(1):33–36, 2003.

Radosavovic, I., Kosaraju, R. P., Girshick, R. B., He, K., and Dollár, P. Designing network design spaces. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10425–10433, 2020.

Rhodes, O., Peres, L., Rowley, A. G., Gait, A., Plana, L. A., Brenninkmeijer, C., and Furber, S. B. Real-time cortical simulation on neuromorphic hardware. *Philosophical Transactions of the Royal Society A*, 378(2164):20190160, 2020.

Roy, C. Review of discretization error estimators in scientific computing. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, pp. 126, 2010.

Roy, K., Jaiswal, A., and Panda, P. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

Rueckauer, B., Lungu, I.-A., Hu, Y., and Pfeiffer, M. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052*, 2016.

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11, 2017.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Schuman, C. D., Kulkarni, S. R., Parsa, M., Mitchell, J. P., Date, P., and Kay, B. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2:10 – 19, 2022.

Sengupta, A., Ye, Y., Wang, R. Y., Liu, C., and Roy, K. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience*, 13, 2018.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Srinivasan, K. and Cowan, G. Subthreshold cmos implementation of the izhikevich neuron model. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1062–1066. IEEE, 2022.

Stanojevic, A., Woźniak, S., Bellec, G., Cherubini, G., Pantazi, A., and Gerstner, W. An exact mapping from relu networks to spiking neural networks. *Neural Networks*, 168:74–88, 2023.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Izacard, G., Joulin, A., Synnaeve, G., Verbeek, J., and J'egou, H. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:5314–5321, 2021.

Valtolina, S. and Hu, L. Charlie: A chatbot to improve the elderly quality of life and to make them more active to fight their sense of loneliness. In *CHItaly 2021: 14th Biannual Conference of the Italian SIGCHI Chapter*, pp. 1–5, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, B., Cao, J., Chen, J., Feng, S., and Wang, Y. A new ann-snn conversion method with high accuracy, low latency and good robustness. In *International Joint Conference on Artificial Intelligence*, 2023a.

Wang, Y., Zhang, M., Chen, Y., and Qu, H. Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion. In *International Joint Conference on Artificial Intelligence*, 2022.

Wang, Z., Jiang, R., Lian, S. J., Yan, R., and Tang, H. Adaptive smoothing gradient learning for spiking neural networks. In *International Conference on Machine Learning*, 2023b.

Ward, M. and Rhodes, O. Beyond lif neurons on neuromorphic hardware. *Frontiers in Neuroscience*, 16:881598, 2022.

Wightman, R. et al. Pytorch image models, 2019.

Wu, C.-J., Raghavendra, R., Gupta, U., Acun, B., Ardalani, N., Maeng, K., Chang, G., Aga, F., Huang, J., Bai, C., et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813, 2022.

Yang, J.-Q., Wang, R., Wang, Z.-P., Ma, Q.-Y., Mao, J.-Y., Ren, Y., Yang, X., Zhou, Y., and Han, S.-T. Leaky integrate-and-fire neurons based on perovskite memristor for spiking neural networks. *Nano Energy*, 74:104828, 2020.

Yao, M., Zhao, G., Zhang, H., Hu, Y., Deng, L., Tian, Y., Xu, B., and Li, G. Attention spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:9393–9410, 2022.

Yin, B., Corradi, F., and Bohté, S. M. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Machine Intelligence*, 3(10): 905–913, 2021.

Zenke, F. and Neftci, E. O. Brain-inspired learning on neuromorphic substrates. *Proceedings of the IEEE*, 109 (5):935–950, 2021.

Zhang, S.-Q. and Zhou, Z.-H. Theoretically provable spiking neural networks. *Advances in Neural Information Processing Systems*, 35:19345–19356, 2022.

Zhang, S.-Q., Zhang, Z.-Y., and Zhou, Z.-H. Bifurcation spiking neural network. *The Journal of Machine Learning Research*, 22(1):11459–11479, 2021.

Zhou, Z., Zhu, Y., He, C., Wang, Y., Shuicheng, Y., Tian, Y., and Yuan, L. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2022.

Zhu, R., Zhao, Q., and Eshraghian, J. K. Spikegpt: Generative pre-trained language model with spiking neural networks. *ArXiv*, abs/2302.13939, 2023.

Zhu, Y., Yu, Z., Fang, W., Xie, X., Huang, T., and Masquelier, T. Training spiking neural networks with event-driven backpropagation. In *Neural Information Processing Systems*, 2022.

Zuo, F., Panda, P., Kotiuga, M., Li, J., Kang, M., Mazzoli, C., Zhou, H., Barbour, A., Wilkins, S., Narayanan, B., et al. Habituation based synaptic plasticity and organismic learning in a quantum perovskite. *Nature communications*, 8(1):240, 2017.

A. Related Works: Review on SNN Training.

Training a SNN from scratch is an alternative of ANN-to-SNN conversion to build a high-performance SNN. Learning methodologies of SNN largely categorizes into two groups: (i) biologically plausible synaptic plasticity and (ii) backpropagation methods (Li et al., 2021a). Spike timing-dependent plasticity(STDP), a representative synaptic plasticity mechanism, updates a synaptic weight based on the spike timing difference of two connected neurons (Kheradpisheh et al., 2018). STDP falls behind backpropagation methods since it is a local unsupervised learning rule without globally guided error (Dong et al., 2022). Backpropagation methods for rate coding unfolds SNN backwards through entire time-steps (Fang et al., 2021; Zhu et al., 2022). To address the non-differentiablility of spiking mechanism, a common strategy is to use a surrogate gradient, which is a smooth relaxation of the spiking mechanism (Neftci et al., 2019). Still, they are computationally expensive and memory-intensive since they unfold a network backwards the entire time-steps (Li et al., 2021a). A line of works efficiently approximate the costly backpropagation process with mixed-mode differentiations (Zenke & Neftci, 2021) or eligibility traces (Bellec et al., 2020; Frenkel & Indiveri, 2022). Backpropagation methods for temporal coding computes backward only up to the last spike's timing, and hence it is event-driven and more resource-efficient (Zhu et al., 2022). Among these methods, surrogate gradient methods achieve state-of-the-art performance. Unfortunately, their performances still has a large margin with trained DNNs, (Fang et al., 2021; Zhou et al., 2022; Yao et al., 2022; Zhu et al., 2023), due to SNN-friendly architectural modification and gradient mismatch (Wang et al., 2023b).

B. Discussions & Future Works

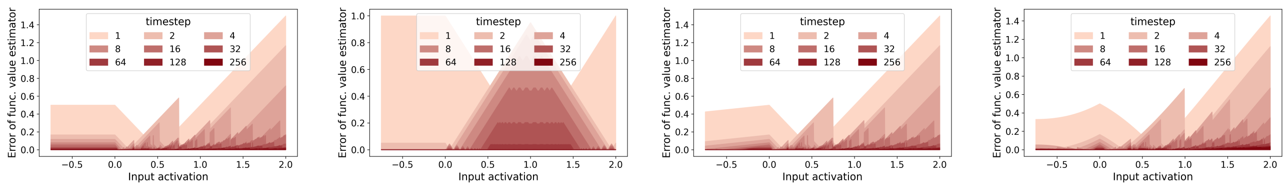
Biophysical Neurons. Our theoretical framework explains the behavior of integrate-and-fire models. Since these models are a simplified discretization of biological neurons, we may extend our framework to continuous and biophysically-detailed models which incorporate known physiology and synaptic anatomy of a neuron cell (Chizhov & Graham, 2021). For example, Hodgkin-huxley cell model (Hodgkin & Huxley, 1952) is an experimentally discovered mathematical model of how biological neurons initiate and propagate spikes. It is a continuous dynamical system of four ordinary differential equations over membrane potentials and conductances that cannot be solved analytically (Gerstner et al., 2014). This model is a basis of biologically plausible neuron models (Izhikevich, 2003). To explain biologically realistic models, it may be helpful to extrapolate our framework to continuous dynamics.

Hardware Implementability. Our signGD-based neuron may need more complex hardware to realize than integrate-and-fire models. Studies in efficient neuromorphic hardwares largely focus on LIF neuron (Yang et al., 2020; Davies et al., 2018; Akopyan et al., 2015). Computational neuroscientists invest on simulating more complex and biologically realistic models, e.g., SpiNNaker (Furber et al., 2014a; Mayr et al., 2019), and make them resource-efficient (Ward & Rhodes, 2022; Rhodes et al., 2020) or introduce new electrical elements for diversity (Guo et al., 2021a). Our neuronal dynamics is much more simple than biophysically detailed models (Hodgkin & Huxley, 1952), but slightly more complex than LIF neuron. For example, our neuron requires two or more internal variables, and GELU or layer norm approximation requires complex operators over internal variables, e.g., exponential or square function. Exponential operator is used in the computational neuroscience works to simulate exponential integrate-and-fire models (Fourcaud-Trocmé et al., 2003; Brette & Gerstner, 2005). Square function is used in the izhikevich model (Izhikevich, 2003). Thus, these operators are implemented in SpiNNaker (Furber et al., 2014b), and their efficient hardware implementations are also studied (Srinivasan & Cowan, 2022; Fang et al., 2022). Note that operator requirements are dependent to spike mechanism; For instance, ReLU or LeakyReLU approximation with our neuron does not need such operators.

Reset-to-Zero. We focus on the reset-by-subtraction mechanism and does not explain the reset-to-zero mechanism. Reset-by-subtraction keeps the leftover information after thresholding operation, hence accelerating the approximation speed (Rueckauer et al., 2016). In contrast, reset-to-zero discards the leftover information at reset (Liu et al., 2022a), causing an unaccountable error for ReLU approximation (Rueckauer et al., 2017). Two reset mechanisms are both valid discretizations of thresholding operation in continuous dynamics (See Appendix E.), since both converge taken the continuum limit of time. Thus, it would be intriguing to interpret the reset-to-zero mechanism in an optimization-theoretic perspective, but gradient-based methods may not suit to formulate its behavior.

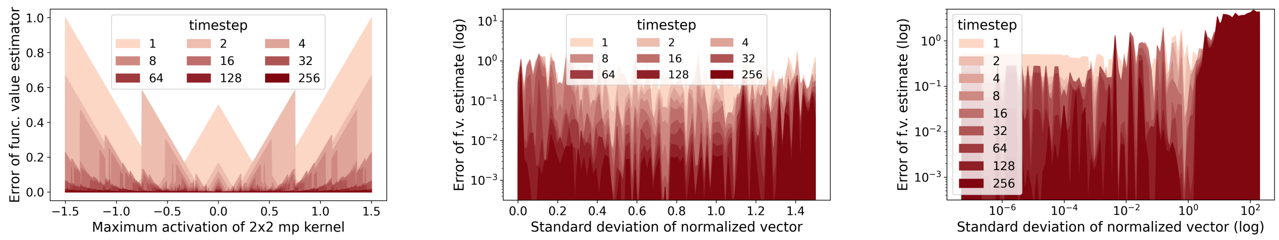
SNN Training. We plan to investigate how our framework extends to a SNN training. Specifically, we newly interpret backpropagation (Rumelhart et al., 1986) and forward-forward algorithms (Hinton, 2022) with our optimization-theoretic perspective of neuronal dynamics. Conversely, we may analyze synaptic plasticity-based SNN learning rules (Kheradpisheh et al., 2018) with our theory, and discover methods to advance their training performance by applying state-of-the-art techniques of ANN training. It would also be appealing to apply short-term plasticity, e.g., habituation (Glanzman, 2009;

Zuo et al., 2017), with our framework to facilitate few-shot adaptation of SNN.



(a) ReLU approximation. $\eta(t) = \frac{1}{t+1}$ (b) ReLU approximation. $\eta(t) = (0.95)^t$ (c) LeakyReLU (α=0.1) approx. $\eta(t) = \frac{1}{t+1}$ (d) GELU approximation. $\eta(t) = \frac{1}{t+1}$

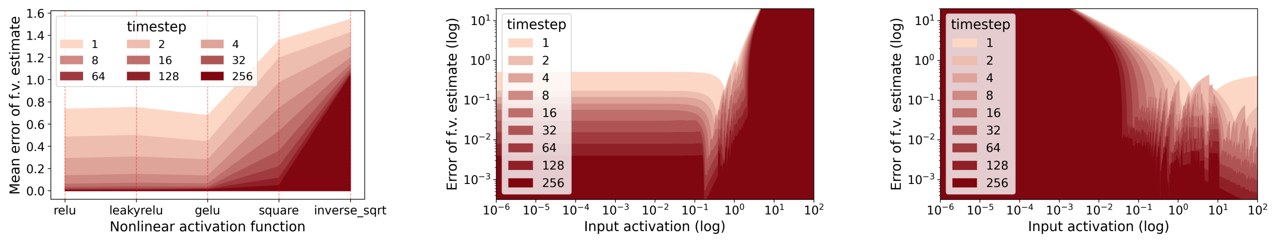
Figure 8. SignGD-based neurons approximating unary nonlinear functions, based on Corollary 5.4. We measure time-evolution of error between the spike neuron output and the nonlinear function output. Input value is deterministically spike-encoded with the Algorithm 2.



(a) Max Pooling approximation with Corollary 5.5. $\eta(t) = \frac{1}{t+1}$.

(b) LayerNorm approximation with Corollary 5.6, 5.7. $\eta(t) = \frac{1}{t+1}$

(c) LayerNorm approximation plot as in Fig. 9(b), with log-scaled X-axis $(10^{-7}, 200)$



(d) Time-evolution of approximation error for distinct nonlinearities, inverse schedule.

(e) Square approximation with Corollary 5.6. $\eta(t) = \frac{1}{t+1}$

(f) Inverse sqrt approximation with Corollary 5.7. $\eta(t) = \frac{1}{t+1}$

Figure 9. SignGD-based neurons approximating n-ary nonlinear operators. Input value is float-encoded with the Algorithm 1.

C. Empirical validation of multi-operand nonlinearity approximation with our signGD-based neuronal dynamics

We conduct toy experiments to validate the approximation capability of our signGD-based neuron on n-ary nonlinearities. Figure 9(a) is the time-evolution of error from signGD-based neurons on the max pooling operator. X axis is the maximum x_{max} of 2x2 patch. Rest of the patch elements x are sampled by $x = \min(x_{max}, \epsilon - 1)$ where $\epsilon \sim \mathcal{N}(0, 1)$. The figure shows that our neuron approximates the max pooling operation through time-steps. Figure 9(b) shows the time-evolution of error of our spike-based approximation of the layer normalization operator. For the experiment, we sample 10-dim vector from $\mathcal{N}(0, 1)$, normalize and scale it to make its standard deviation σ. We plot the error of the first vector element. The figure shows that the approximation is noticeably slow as $\sigma \to 0$ or σ gets larger. Figure 9(c) further illustrates this trend with logarithmic σ values. We analyze in-depth through Figure 9(d)- 9(f) and find that the slow convergence is because the step size is invariant with the input activation. The step size of signGD solely depends on the learning rate schedule. In Figure 9(e), precise square approximation for a small input value, e.g., 0.1, is incompatible with a square approximation of large value like 5, 10. For example, if we decrease the initial learning rate to swiftly approximate $0.01 = (0.1)^2$, then it needs more time-steps to approximate $25 = 5^2$. Also, Figure 9(f) shows that inverse sqrt approximation gets slower or even fails to converge as the denominator, the input value, tends to 0. It is difficult to design a learning rate schedule that

Algorithm 1 Float encoding for signed schedule coding (Definition 5.1)

```
1: Input: A real-valued input activation $x$, learning rate schedule $\eta : \mathbb{N} \to \mathbb{R}$, total time-steps $T$
2: Output: A float train, i.e., time-series of floats, S = $[s(1), s(2), \cdots, s(T)] \in \mathbb{R}^T$
3: Initialize $f \leftarrow 0$, $S \leftarrow [\,]$.
4: for $t = 1$ to $T$ do
5:     grad $\leftarrow f - x$.
6:     current $\leftarrow 0.5 \cdot (1 + \text{grad})$.
7:     $f \leftarrow f - \eta(t) \cdot \text{grad}$.
8:     $S$.append( current )
9: end for
```

Algorithm 2 Deterministic spike-based encoding for signed schedule coding (Definition 5.1)

```
1: Input: A real-valued input activation $x$, learning rate schedule $\eta : \mathbb{N} \to \mathbb{R}$, total time-steps $T$
2: Output: A spike train, i.e., time-series of binary spikes, S = $[s(1), s(2), \cdots, s(T)] \in \{0, 1\}^T$
3: Initialize $f \leftarrow 0$, $S \leftarrow [\,]$.
4: for $t = 1$ to $T$ do
5:     current $\leftarrow \mathbb{H}(f - x)$.
6:     grad $\leftarrow 2 \cdot \text{current} - 1$.
7:     $f \leftarrow f - \eta(t) \cdot \text{grad}$.
8:     $S$.append( current )
9: end for
```

mitigates these approximation issues. In Figure 9(d), we compare the time-evolution of mean approximation error over five unary nonlinearities, given a Gaussian-sampled input vector of length 1000. Errors of square and inverse sqrt functions converge slower than others, explaining why it is difficult to approximate the layer normalization. Faster approximation of layer normalization is crucial to accelerate the inference of SNN-converted MLP-Mixer, ConvNext, and Transformer, but we leave it as a future work.

D. Neuronal codec for signGD-based neuronal dynamics

Neurons in SNN communicate with spike train. spike trains share the same neural coding scheme different from the float(or quantized integer)-based activations used in DNN. We can thus think of a *neuronal codec* which encodes a real number into a spike train, and also decodes a spike train into a a real number. For example, rate coding has multiple encoding algorithms, e.g., float encoding (Li et al., 2021a) or poisson encoding (Sengupta et al., 2018). A pair of encoding and decoding scheme consists a neuronal codec.

To leverage signGD-based neurons for practical SNN inference, we develop neuronal codecs for the signed schedule coding (Definition 5.1). We develop three spike encoding algorithms for signed schedule coding: float-based (Algorithm 1), deterministic spike-based (Algorithm 2), and stochastic spike-based encoding (Algorithm 3). In practice, We can eliminate total number of time-steps T from the input arguments of algorithms; The encoding process can be lazy-evaluated to

Algorithm 3 Stochastic spike-based sigmoidal encoding for signed schedule coding (Definition 5.1)

```
1: Input: A real-valued input activation $x$, learning rate schedule $\eta : \mathbb{N} \to \mathbb{R}$, total time-steps $T$, stochasticity $c \in [0, 1]$
2: Output: A spike train, i.e., time-series of binary spikes, S = $[s(1), s(2), \cdots, s(T)] \in \{0, 1\}^T$
3: Initialize $f \leftarrow 0$, $S \leftarrow [\,]$.
4: for $t = 1$ to $T$ do
5:     $p \leftarrow \text{Sigmoid}(c \cdot (f - x))$.
6:     current $\sim \text{Bernoulli}(p)$.
7:     grad $\leftarrow 2 \cdot \text{current} - 1$.
8:     $f \leftarrow f - \eta(t) \cdot \text{grad}$.
9:     $S$.append( current )
10: end for
```

generate an infinite-length spike (or float) train.

E. Continuous dynamics of one-dimensional integrate-and-fire models

The continuous neuronal dynamics of general one-dimensional integrate-and-fire neuron is a linear differential equation with thresholding criterion (Gerstner et al., 2014).

$$\tau_m \frac{du}{dt} = -f(u(t)) + RI(t) \qquad \text{(Integration)}$$

$$\lim_{\delta \to 0; \delta > 0} u(t+\delta) = u_{rest} \qquad \text{if } u(t) \geq \theta_{th} \qquad \text{(Thresholding)}$$

with time $t \in \mathbb{R}^+$, dynamics function $f(u) : \mathbb{R} \to \mathbb{R}$, input current $I(t)$ at time t, membrane potential u, resting potential u_{rest}, threshold θ_{th}, membrane resistance R, membrane capacitance C and membrane constant $\tau_m = RC$.

F. Experimental setup & Implementation details

We use Top-1 classification accuracy as a measure of SNN inference performance. Following a standard normalization practice in conversion literatures (Diehl et al., 2015; Rueckauer et al., 2017; Wang et al., 2022), we scale inputs and outputs of each ReLU layer with layer-wise maximum activation value over random 100 batches before the conversion. Algorithm 4 in the appendix describes the end-to-end ANN-to-SNN conversion with our neuron. To train DNN models for CIFAR datasets, we use SGD with learning rate 0.1, momentum 0.9, weight decay 5×10^{-4}, and cosine annealing schedule (Loshchilov & Hutter, 2016) of $T_{max} = 300$. For ImageNet classification models, we use pretrained DNN weights of VGG, ResNet, ConvNext, and RegNetX from torchvision (maintainers & contributors, 2016),VGG and ResNet without max pooling from (Li et al., 2021a), and MLP-Mixer and ResMLP from timm (Wightman et al., 2019). We run our experiments on a machine with AMD EPYC 7313 CPU, 512GB RAM, and NVIDIA RTX A6000.

G. Theoretical energy consumption analysis

To analyze the energy consumption of our proposed neuron model, we use a theoretical energy cost estimation framework that is widely recognized in prior research (Zhou et al., 2022; Kundu et al., 2021; Yin et al., 2021; Yao et al., 2022). We choose this approach because a direct hardware implementation and empirical evaluation of our neuron are beyond the current scope. However, we plan to explore this direction in future work. We compare the energy costs of LIF/IF neurons and our neuron as follows.

G.1. Single synaptic operation

We first compare the energy consumption of single spike-based accumulation (AC) operation (synaptic operation, SOP) on 45nm CMOS processors (Horowitz, 2014). SOP energy consumption of our neuron varies with the selection of dynamics coefficients. If we choose $\eta(t) = \eta(0)\gamma^t$ following the evaluation setup of Table 1, 4 and 5, the coefficients satisfying Theorem 5.3 and Corollary 5.4 can be chosen as $\alpha_1(t) = 1/\gamma$, $\beta_1(t) = \gamma$, $\alpha_2(t) = \beta_2(t) = 1$. The difference of our neuron with LIF neuron is a new internal variable $u(t)$ and a weight term W at the dynamics of $v(t)$. Since the W term addition is absorbed into an existing operation, This results in an energy overhead 0.9pJ added to the 0.9pJ consumption of LIF neuron (Zhou et al., 2022) to become 1.8 pJ, which is twice over the LIF neuron. Table 2 compares the per-operation energy consumption, which we denote as E_{SOP}.

	IF / LIF neuron (AC)	signGD-based neuron (ours)	FLOPs (MAC)
E_{SOP}	0.9pJ	1.8pJ	4.6pJ

Table 2. Theoretically derived energy consumption of a single operation on 45nm CMOS processors (Horowitz, 2014).

G.2. End-to-End inference of converted DNN models

We now compare the energy consumption of converted DNN models. In terms of the end-to-end SNN model, our neuron differs with prior spiking neurons only in the computation of neuronal dynamics, and everything else remains the same.

CIFAR10, Timesteps T = 64	Neurons	Firing Rates(%)	N_{SOP} (M)	Energy (uJ)
ResNet18 (He et al., 2015)	IF Neuron	20.97	22.431	20.188
	LIF Neuron	21.17	22.651	20.386
	signGD-based Neuron	16.78	17.943	32.298 (1.59×)
VGG16 (Simonyan & Zisserman, 2014)	IF Neuron	20.30	10.778	9.700
	LIF Neuron	20.71	10.993	9.893
	signGD-based Neuron	16.75	8.892	16.006 (1.65×)

Table 3. Inference energy consumption of ReLU Networks converted with different spiking neuron models.

Table 3 lists firing rates $fr = \frac{\text{Number of spikes}}{\text{Timesteps * Neurons}}$, number of SOPs $N_{\text{SOP}} =$ (Number of spikes), and energy consumption estimated by $E = N_{\text{SOP}} \cdot E_{SOP}$.

Our theoretical analysis outlines that more complex internal dynamics of the signGD-based neuron lead to higher energy consumption in same time-steps compared to IF and LIF neurons. The relative increase in energy usage is manageable with an approximate factor of $1.6\times$. This increase is offset by advantages our approach provides, including broader support for nonlinear operator approximations and the reduced time steps in SNN inference. For example, in Table 4, ResNet18 converted with subgradient-based neurons achieve 94% accuracy in 64 steps, while the model converted with our neuron model achieves the same accuracy in only 22 steps.

Besides, the energy efficiency of SNN comes not only from computation but also from memory storage. For example, while IF neurons theoretically offer a $5.11\times$ energy efficiency in SOP compared to traditional FLOPS, the VGG model implemented on a neuromorphic SNN chip achieves even higher efficiency, ranging from $35\times$ to $560\times$ (Bu et al., 2022), thanks to in-memory computing. To accurately assess this, a real hardware implementation is crucial, and we plan to explore this in our future work.

H. Mathematical Proofs.

Theorem H.1. *Dynamical system of IF neuron (Eq. 2-4) with rate-coded input $\tilde{x}(t) = \frac{1}{t}\sum_{i=1}^{t} I(i)$ and output $y(t) = \frac{1}{t}\sum_{i=1}^{t} s(i)$ is equivalent to a subgradient method over an optimization problem $\min_{y\in\mathbb{R}} \mathcal{L}(y;x)$, approximated with $x \leftarrow \tilde{x}(t+1)$ as,*

$$\tilde{f}(t) = \tilde{f}(t-1) - \frac{1}{t+1} \cdot \tilde{g}\big(\tilde{f}(t-1); \tilde{x}(t)\big) \tag{17}$$

$$\mathcal{L}(y;x) = h\big(\frac{R}{\theta_{th}}x - y\big) + \frac{1}{2}y^2 \quad \textit{(objective fn.)} \tag{18}$$

$$\tilde{f}(t) = \frac{t}{t+1}y(t) - \frac{u(0)-\theta_{th}}{\theta_{th}(t+1)} \ \textit{(t-th approx.)} \tag{19}$$

where $\tilde{g}(y;x)$ is a subgradient of $\mathcal{L}(y;x)$, $h(x) =$ ReLU(x) and $u(0)$ an initial membrane potential. Solution of the problem is ReLU1$(\frac{R}{\theta_{th}}x)$.

Proof. Applying equation 3 to equation 1, we rearrange the membrane equation for u_{pre} as

$$u_{pre}(t+1) = (u_{pre}(t) - \theta_{th}s(t)) + RI(t+1) \tag{20}$$

Spike $s(t)$ links the recurrence relation (20) of the membrane potential u_{pre} and coding scheme (6) of output activation y.

$$s(t) = \frac{-1}{\theta_{th}}(u_{pre}(t+1) - u_{pre}(t) - RI(t+1)) = ty(t) - (t-1)y(t-1) \tag{21}$$

Solving the recurrence relation (21) through time t yields a linear relation of membrane potential u_{pre}, input $\tilde{x}$ and output y.

Note that $u_{pre}(1) = u(0) + RI(1)$.

$$\begin{aligned}
ty(t) - (t-1)y(t-1) &= \frac{-1}{\theta_{th}}\big(u_{pre}(t+1) - u_{pre}(t) - RI(t+1)\big) \\
(t-1)y(t-1) - (t-2)y(t-2) &= \frac{-1}{\theta_{th}}\big(u_{pre}(t) - u_{pre}(t-1) - RI(t)\big) \\
&\vdots \\
1y(1) - 0y(0) &= \frac{-1}{\theta_{th}}\big(u_{pre}(2) - u_{pre}(1) - RI(2)\big) \\
+ \overline{\qquad\qquad\qquad\qquad\qquad}&\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad} \\
ty(t) &= \frac{-1}{\theta_{th}}\big(u_{pre}(t+1) - u_{pre}(1) - R\sum_{i=2}^{t+1} I(i)\big) \\
&= \frac{-1}{\theta_{th}}\big(u_{pre}(t+1) - u(0) - RI(1) - R\sum_{i=2}^{t+1} I(i)\big) \\
&= \frac{-1}{\theta_{th}}\big(u_{pre}(t+1) - u(0) - R\sum_{i=1}^{t+1} I(i)\big) \\
&= \frac{-1}{\theta_{th}}\big(u_{pre}(t+1) - u(0) - R\cdot(t+1)\cdot\tilde{x}(t+1)\big)
\end{aligned}$$

Rearranging the above equation for u_{pre},

$$\therefore u_{pre}(t+1) = -\theta_{th}\cdot t\cdot y(t) + u(0) + R\cdot(t+1)\cdot\tilde{x}(t+1) \tag{22}$$

With equation (22), we remove the u_{pre} term from rate coding equation (6) to obtain a discrete dynamical system of the input $\tilde{x}$ and output y.

$$y(t) = \frac{t-1}{t}y(t-1) + \frac{1}{t}\mathbb{H}(-\theta_{th}\cdot(t-1)\cdot y(t-1) + u(0) + R\cdot t\cdot\tilde{x}(t) - \theta_{th}) \tag{23}$$

By definition of $\tilde{f}(t)$ in (19), we substitute y with $\tilde{f}$ from the dynamical system (23).

$$\begin{aligned}
\theta_{th}\cdot(t+1)\cdot\tilde{f}(t) &= \theta_{th}\cdot t\cdot y(t-1) - (u(0) - \theta_{th}) \\
\theta_{th}\cdot t\cdot\tilde{f}(t-1) &= \theta_{th}\cdot(t-1)\cdot y(t-1) - (u(0) - \theta_{th}) \\
-\theta_{th}\cdot t\cdot\tilde{f}(t-1) &= -\theta_{th}\cdot(t-1)\cdot y(t-1) + u(0) - \theta_{th} \\
\frac{t-1}{t}y(t-1) &= \tilde{f}(t-1) + \frac{u(0)-\theta_{th}}{\theta_{th}\cdot t} \\
y(t) &= \frac{t+1}{t}\tilde{f}(t) + \frac{u(0)-\theta_{th}}{\theta_{th}\cdot t} \\
\therefore \frac{t+1}{t}\tilde{f}(t) + \frac{u(0)-\theta_{th}}{\theta_{th}\cdot t} &= \tilde{f}(t-1) + \frac{u(0)-\theta_{th}}{\theta_{th}\cdot t} + \frac{1}{t}\mathbb{H}\big(-\theta_{th}\cdot t\cdot\tilde{f}(t-1) + R\cdot t\cdot\tilde{x}(t)\big)
\end{aligned}$$

By the scale-invariance of step function $\mathbb{H}$, we obtain

$$\begin{aligned}
\tilde{f}(t) &= \frac{t}{t+1}\tilde{f}(t-1) + \frac{1}{t+1}\mathbb{H}(R\cdot t\cdot\tilde{x}(t) - \theta_{th}\cdot t\cdot\tilde{f}(t-1)) \\
&= \frac{t}{t+1}\tilde{f}(t-1) + \frac{1}{t+1}\mathbb{H}(\frac{R}{\theta_{th}}\tilde{x}(t) - \tilde{f}(t-1)) \\
&= \tilde{f}(t-1) - \frac{1}{t+1}(\tilde{f}(t-1) - \mathbb{H}(\frac{R}{\theta_{th}}\tilde{x}(t) - \tilde{f}(t-1))) \\
&= \tilde{f}(t-1) - \frac{1}{t+1}\cdot\tilde{g}(\tilde{f}(t-1);\tilde{x}(t))
\end{aligned}$$

where $\mathcal{L}(y;x) = f\big(\frac{R}{\theta_{th}}x - y\big) + \frac{1}{2}y^2$. It corresponds to the subgradient method over an optimization problem $\min_{y\in\mathbb{R}} \mathcal{L}(y;x)$ with a diminishing step size $\frac{1}{t+1}$, approximated with $x \leftarrow \tilde{x}(t+1)$. The objective function $\mathcal{L}(y;x)$ is a convex l^2-regularized negative ReLU function. We now show that the $\arg\min_y \mathcal{L}(y;x) = \text{ReLU1}(\frac{R}{\theta_{th}}x)$ through 3 different cases. Note that the sub-gradient of $\mathcal{L}(y;x)$ is

$$\tilde{g}(y;x) = y - \mathbb{H}(\frac{R}{\theta_{th}}x - y)$$

Case 1, $x < 0$. If $y \le \frac{R}{\theta_{th}}x < 0$, then $\tilde{g}(y;x) = y - 1 < 0$. If $\frac{R}{\theta_{th}}x < y < 0$, then $\tilde{g}(y;x) = y - 0 < 0$. Finally, $y \ge 0$ then $\frac{R}{\theta_{th}}x - y < 0$ so $\tilde{g}(y;x) = y > 0$. Therefore, $\arg\min_y \mathcal{L}(y;x) = 0$

Case 2, $x \ge \frac{\theta_{th}}{R}$. If $y \le 1$ then $\tilde{g}(y;x)(y;x) = y - 1 \le 0$. If $y \ge 1$ then $\tilde{g}(y;x)L(y;x) \ge y - 1 \ge 0$. Therefore, $\arg\min_y \mathcal{L}(y;x) = 1$.

Case 3, $0 < x < \frac{\theta_{th}}{R}$. If $0 < \frac{R}{\theta_{th}}x \le y$ then $\tilde{g}(y;x)(y;x) = y - 0 \ge 0$. If $y \le \frac{R}{\theta_{th}}x \le 1$ then $\tilde{g}(y;x)(y;x) \ge y - 1 \le 0$. Therefore, $\arg\min_y \mathcal{L}(y;x) = \frac{R}{\theta_{th}}x$. □

Theorem H.2. *Dynamical system of LIF neuron (Eq. 2,3, 5) with EMA-coded input $\tilde{x}(t)$ and output $y(t)$ (Eq. 7), if $\tau_m = \tau$, is equivalent to a subgradient method over an optimization problem $\min_{y\in\mathbb{R}} \mathcal{L}(y;x)$, approximated with $x \leftarrow \tilde{x}(t+1)$ as,*

$$\begin{aligned}
\tilde{f}(t+1) &= \tilde{f}(t) - \frac{1}{\tau}\cdot \tilde{g}\big(\tilde{f}(t);\tilde{x}(t+1)\big) \\
\mathcal{L}(y;x) &= \frac{1}{2}y^2 + \frac{u_{rest}}{\theta_{th}(\tau-1)}y + h(\frac{R-\theta_{th}}{\theta_{th}(\tau-1)}x - y) \\
\tilde{f}(t) &= y(t) - \frac{1}{\tau\theta_{th}}(\frac{\tau-1}{\tau})^t u(0) - \frac{u_{rest}\sum_{i=0}^{t}(\frac{\tau-1}{\tau})^{t-i}}{\theta_{th}\tau(\tau-1)}
\end{aligned}$$

where $\tilde{g}(y;x)$ is a subgradient of $\mathcal{L}(y;x)$, $h(x) =$ ReLU(x) and $u(0)$ an initial membrane potential. If $u_{rest} = 0$, the solution to the problem is ReLU1$(\frac{R}{\theta_{th}(\tau-1)}x - \frac{1}{\tau-1})$.

Proof. $u_{pre}(t) = u(t-1) - \frac{u(t-1)-u_{rest}}{\tau_m} + \frac{R}{\tau_m}I(t)$ $s(t) = \mathbb{H}(u_{pre}(t) - \theta_{th})u(t) = u_{pre}(t) - \theta_{th}s(t)$ We apply equation (3) on equation (5) to obtain the membrane equation of u_{pre}.

$$\begin{aligned}
u_{pre}(t) &= (u_{pre}(t-1) - \theta_{th}s(t-1)) - \frac{(u_{pre}(t-1) - \theta_{th}s(t-1)) - u_{rest}}{\tau_m} + \frac{R}{\tau_m}I(t) \\
&= \frac{\tau_m - 1}{\tau_m}u_{pre}(t-1) - \theta_{th}\frac{\tau_m-1}{\tau_m}s(t-1) + \frac{1}{\tau_m}u_{rest} + \frac{R}{\tau_m}I(t)
\end{aligned} \tag{24}$$

Rearranging the equation (24) for the spike $s(t)$,

$$\begin{aligned}
-\frac{\theta_{th}}{\tau_m}s(t-1) &= \frac{1}{\tau_m - 1}u_{pre}(t) - \frac{1}{\tau_m}u_{pre}(t-1) - \frac{R}{\tau_m(\tau_m-1)}I(t) - \frac{1}{\tau_m(\tau_m-1)}u_{rest} \\
-\frac{\theta_{th}}{\tau_m}s(t) &= \frac{1}{\tau_m - 1}u_{pre}(t+1) - \frac{1}{\tau_m}u_{pre}(t) - \frac{R}{\tau_m(\tau_m-1)}I(t+1) - \frac{1}{\tau_m(\tau_m-1)}u_{rest}
\end{aligned}$$

With the definition of $y(t)$ and $\tau = \tau_m$, we yield the recurrence relation of y and u_{pre}.

$$\begin{aligned}
\frac{-\theta_{th}}{\tau}s(t) &= -\theta_{th}\cdot y(t) + \theta_{th}\cdot\frac{\tau-1}{\tau}y(t-1) \\
-\theta_{th}\cdot y(t) + \theta_{th}\cdot\frac{\tau-1}{\tau}y(t-1) &= \frac{1}{\tau-1}u_{pre}(t+1) - \frac{1}{\tau}u_{pre}(t) - \frac{R}{\tau(\tau-1)}I(t+1) - \frac{1}{\tau(\tau-1)}u_{rest}
\end{aligned}$$

We solve the recurrence relation through time, deriving a linear relation of u_{pre}, I and y.

$$
\begin{aligned}
-\theta_{th}\cdot y(t)+\theta_{th}\cdot\frac{\tau-1}{\tau}y(t-1) &= \frac{1}{\tau-1}u_{pre}(t+1)-\frac{1}{\tau}u_{pre}(t)-\frac{R}{\tau(\tau-1)}I(t+1)-\frac{1}{\tau(\tau-1)}u_{rest}\\
\frac{\tau-1}{\tau}\Big(-\theta_{th}\cdot y(t-1)+\theta_{th}\cdot\frac{\tau-1}{\tau}y(t-2)\Big) &\\
&= \frac{\tau-1}{\tau}\Big(\frac{1}{\tau-1}u_{pre}(t)-\frac{1}{\tau}u_{pre}(t-1)-\frac{R}{\tau(\tau-1)}I(t)-\frac{1}{\tau(\tau-1)}u_{rest}\Big)\\
&\vdots\\
(\frac{\tau-1}{\tau})^{t-1}\Big(-\theta_{th}\cdot y(1)+\theta_{th}\cdot\frac{\tau-1}{\tau}y(0)\Big) &\\
&= (\frac{\tau-1}{\tau})^{t-1}\Big(\frac{1}{\tau-1}u_{pre}(2)-\frac{1}{\tau}u_{pre}(1)-\frac{R}{\tau(\tau-1)}I(2)-\frac{1}{\tau(\tau-1)}u_{rest}\Big)\\
+\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}&\\
-\theta_{th}\cdot y(t)+\theta_{th}(\frac{\tau-1}{\tau})^{t}y(0) &= \frac{1}{\tau-1}u_{pre}(t+1)-\frac{1}{\tau}(\frac{\tau-1}{\tau})^{t-1}u_{pre}(1)\\
&\quad -\frac{R}{\tau(\tau-1)}\sum_{i=2}^{t+1}(\frac{\tau-1}{\tau})^{t+1-i}I(i)-\frac{u_{rest}}{\tau(\tau-1)}\sum_{i=1}^{t}(\frac{\tau-1}{\tau})^{t-i}
\end{aligned}
$$

Since $u_{pre}(1)=\frac{\tau-1}{\tau}u(0)+\frac{1}{\tau}u_{rest}+\frac{R}{\tau(\tau-1)}I(1)$,

$$
\begin{aligned}
-\frac{1}{\tau}(\frac{\tau-1}{\tau})^{t-1}u_{pre}(1) &= -\frac{1}{\tau}(\frac{\tau-1}{\tau})^{t-1}\Big(\frac{\tau-1}{\tau}u(0)+\frac{1}{\tau}u_{rest}+\frac{R}{\tau}I(1)\Big)\\
&= -\frac{1}{\tau}(\frac{\tau-1}{\tau})^{t}u(0)-\frac{u_{rest}}{\tau(\tau-1)}(\frac{\tau-1}{\tau})^{t}-\frac{1}{\tau}(\frac{\tau-1}{\tau})^{t-1}\cdot\frac{R}{\tau}I(1)\\
&= -\frac{1}{\tau}(\frac{\tau-1}{\tau})^{t}u(0)-\frac{u_{rest}}{\tau(\tau-1)}(\frac{\tau-1}{\tau})^{t}-\frac{R}{\tau(\tau-1)}(\frac{\tau-1}{\tau})^{t}I(1)
\end{aligned}
$$

Removing the term $y(0)=0$ and $u_{pre}(1)$ from the above linear relation results in the linear relation of u_{pre}, y and $\tilde{x}$.

$$
\begin{aligned}
-\theta_{th}\cdot y(t) &= \frac{1}{\tau-1}u_{pre}(t+1)-\frac{R}{\tau(\tau-1)}\sum_{i=2}^{t+1}(\frac{\tau-1}{\tau})^{t+1-i}I(i)-\frac{u_{rest}}{\tau(\tau-1)}\sum_{i=1}^{t}(\frac{\tau-1}{\tau})^{t-i}\\
&\quad -\frac{1}{\tau}(\frac{\tau-1}{\tau})^{t-1}u_{pre}(1)\\
&= \frac{1}{\tau-1}u_{pre}(t+1)-\frac{R}{\tau(\tau-1)}\sum_{i=2}^{t+1}(\frac{\tau-1}{\tau})^{t+1-i}I(i)-\frac{u_{rest}}{\tau(\tau-1)}\sum_{i=1}^{t}(\frac{\tau-1}{\tau})^{t-i}\\
&\quad -\frac{1}{\tau}(\frac{\tau-1}{\tau})^{t}u(0)-\frac{u_{rest}}{\tau(\tau-1)}(\frac{\tau-1}{\tau})^{t}-\frac{R}{\tau(\tau-1)}(\frac{\tau-1}{\tau})^{t}I(1)\\
&= \frac{1}{\tau-1}u_{pre}(t+1)-\frac{1}{\tau}(\frac{\tau-1}{\tau})^{t}u(0)-\frac{R}{\tau(\tau-1)}\sum_{i=1}^{t+1}(\frac{\tau-1}{\tau})^{t+1-i}I(i)-\frac{u_{rest}}{\tau(\tau-1)}\sum_{i=0}^{t}(\frac{\tau-1}{\tau})^{t-i}\\
&= \frac{1}{\tau-1}u_{pre}(t+1)-\frac{1}{\tau}(\frac{\tau-1}{\tau})^{t}u(0)-\frac{R}{\tau-1}\tilde{x}(t+1)-\frac{u_{rest}}{\tau(\tau-1)}\sum_{i=0}^{t}(\frac{\tau-1}{\tau})^{t-i}
\end{aligned}
$$

By definition $\tilde{f}(t)=y(t)-\frac{1}{\tau\theta_{th}}(\frac{\tau-1}{\tau})^{t}u(0)-\frac{u_{rest}}{\theta_{th}\tau(\tau-1)}\sum_{i=0}^{t}(\frac{\tau-1}{\tau})^{t-i}$, thus

$$
\begin{aligned}
-\theta_{th}\tilde{f}(t) &= \frac{1}{\tau-1}u_{pre}(t+1)-\frac{R}{\tau-1}\tilde{x}(t+1)\\
\therefore u_{pre}(t+1) &= R\cdot\tilde{x}(t+1)-\theta_{th}\cdot(\tau-1)\cdot\tilde{f}(t)
\end{aligned}
$$

We now reformulate the EMA coding equation (7) with $\tilde{f}$ and $\tilde{x}$.

$$\begin{aligned}
y(t) - \frac{\tau-1}{\tau}y(t-1) &= \tilde{f}(t) + \frac{1}{\tau\theta_{th}}(\frac{\tau-1}{\tau})^t u(0) + \frac{u_{rest}}{\theta_{th}\cdot\tau(\tau-1)}\sum_{i=0}^{t}(\frac{\tau-1}{\tau})^{t-i} \\
&- \frac{\tau-1}{\tau}\Big(\tilde{f}(t-1) + \frac{1}{\tau\theta_{th}}(\frac{\tau-1}{\tau})^{t-1}u(0) + \frac{u_{rest}}{\theta_{th}\cdot\tau(\tau-1)}\sum_{i=0}^{t-1}(\frac{\tau-1}{\tau})^{t-1-i}\Big) \\
&= \tilde{f}(t) - \frac{\tau-1}{\tau}\tilde{f}(t-1) + \frac{u_{rest}}{\theta_{th}\cdot\tau(\tau-1)} = \frac{1}{\tau}s(t) \\
\frac{1}{\tau}s(t) =& \tilde{f}(t) - \frac{\tau-1}{\tau}\tilde{f}(t-1) + \frac{u_{rest}}{\theta_{th}\cdot\tau(\tau-1)} \\
=& \frac{1}{\tau}\mathbb{H}(u_{pre}(t)-\theta_{th}) \\
=& \frac{1}{\tau}\mathbb{H}(R\cdot\tilde{x}(t) - \theta_{th}\cdot(\tau-1)\cdot\tilde{f}(t-1) - \theta_{th}) \\
\tilde{f}(t) =& \frac{\tau-1}{\tau}\tilde{f}(t-1) - \frac{u_{rest}}{\theta_{th}\cdot\tau(\tau-1)} + \frac{1}{\tau}\mathbb{H}(R\cdot\tilde{x}(t) - \theta_{th}\cdot(\tau-1)\cdot\tilde{f}(t-1) - \theta_{th}) \\
\therefore\ \tilde{f}(t) =& \tilde{f}(t-1) - \frac{1}{\tau}\Big(\tilde{f}(t-1) + \frac{u_{rest}}{\theta_{th}(\tau-1)} - \mathbb{H}(R\cdot\tilde{x}(t) - \theta_{th}\cdot(\tau-1)\cdot\tilde{f}(t-1) - \theta_{th})\Big) \\
=& \tilde{f}(t-1) - \frac{1}{\tau}\Big(\tilde{f}(t-1) + \frac{u_{rest}}{\theta_{th}(\tau-1)} - \mathbb{H}(\frac{R}{\theta_{th}(\tau-1)}\tilde{x}(t) - \tilde{f}(t-1) - \frac{1}{\tau-1})\Big)
\end{aligned}$$

The subgradient $\tilde{g}(y;x)$ of the objective function $\mathcal{L}(y;x) = \frac{1}{2}y^2 + \frac{u_{rest}}{\theta_{th}(\tau-1)}y + \text{ReLU}(\frac{R}{\theta_{th}(\tau-1)}x - y - \frac{1}{\tau-1})$ is

$$\tilde{g}(y;x) = y + \frac{u_{rest}}{\theta_{th}(\tau-1)} - \text{ReLU}(\frac{R}{\theta_{th}(\tau-1)}x - y - \frac{1}{\tau-1}) \tag{25}$$

Therefore, we finally derive the approximation $\tilde{x}(t) \to x$ of the subgradient method of constant step-size $\frac{1}{\tau}$ over the optimization problem $\min_{y\in\mathbb{R}}\mathcal{L}(y;x)$. If $u_{rest}=0$, the solution to the problem is $\text{ReLU1}(\frac{R}{\theta_{th}(\tau-1)}x - \frac{1}{\tau-1})$.

$$\tilde{f}(t) = \tilde{f}(t-1) - \frac{1}{\tau}\cdot\tilde{g}\big(\tilde{f}(t-1);\tilde{x}(t)\big) \tag{26}$$

□

Theorem H.3. *Let $f^*(x) = \arg\min_{y\in\mathbb{R}}\mathcal{L}(y;x)$ be the minimizer of $\mathcal{L}(y;x) = ReLU(x-y) + \frac{1}{2}y^2$ over a true input $x\in\mathbb{R}$, and its t-th approximation $\tilde{f}(t)$ be defined as equation 8 and rate-coded input $\tilde{x}(t) = \frac{1}{t}\sum_{i=1}^{t}I(i)$. Denote $h(i) = \big(1-\sqrt{\frac{i-1}{i+1}}\big)$. If $\|\tilde{f}(t)\| < M \in \mathbb{R}^+$, then the approximation error $\|\tilde{f}(t) - f^*(x)\|$ is upper bounded as*

$$\begin{aligned}
\|\tilde{f}(t) - f^*(x)\|^2 \le& \frac{\|\tilde{f}(0) - f^*(x)\|^2}{t+1} \\
&+ \underbrace{\frac{M+1}{t+1}\sum_{i=1}^{t}h(i)}_{\textit{Nondifferentiability Error}} + \underbrace{\frac{4}{t+1}\sum_{i=1}^{t}\min(\|x-\tilde{x}(i)\|,1)}_{\textit{Input Error}}
\end{aligned} \tag{27}$$

Proof. For simplicity, we denote true objective function $\mathcal{L}(y) = \mathcal{L}(y;x) = \text{ReLU}(x-y) + \frac{1}{2}y^2$ and the estimated objective function $\mathbf{L}(y) = \mathcal{L}(y;\tilde{x}(t+1)) = f(\tilde{x}(t+1)-y) + \frac{1}{2}y^2$. Respectively, we denote their minimizers $f^* = f^*(x) = \arg\min_{y\in\mathbb{R}}\mathcal{L}(y)$ and $\mathbf{f}^* = \arg\min_{y\in\mathbb{R}}\mathbf{L}(y)$. We also use the fact that $f^* = \text{ReLU1}(x)$, $\mathbf{f}^* = \text{ReLU1}(\tilde{x}(t+1))$.

From equation 17 , $\tilde{f}(t) = \tilde{f}(t-1) - \frac{1}{t+1}\nabla_y\mathbf{L}(\tilde{f}(t-1))$. Hence

$$\|\tilde{f}(t+1) - f^*\|^2 = \|\tilde{f}(t) - \frac{1}{t+2}\nabla_y\mathbf{L}(\tilde{f}(t)) - f^*\|^2 \tag{28}$$

$$= \|\tilde{f}(t) - f^*\|^2 - \frac{2}{t+2}\nabla_y\mathbf{L}(\tilde{f}(t))\big(\tilde{f}(t) - f^*\big) + \frac{1}{(t+2)^2}\|\nabla_y\mathbf{L}(\tilde{f}(t))\|^2 \tag{29}$$

We first use the fact that $\mathbf{L}(y)$ is 1-strongly convex over y, since $\mathbf{L}(y) - \frac{1}{2}y^2$ is convex. First-order convexity over $\mathbf{L}$ show

$$\mathbf{L}(f^*) \geq \mathbf{L}(\tilde{f}(t)) - \nabla_y \mathbf{L}(\tilde{f}(t))(\tilde{f}(t) - f^*) + \frac{1}{2}\|\tilde{f}(t) - f^*\|^2 \tag{30}$$

The inequality 30 applies to the right hand-side of the equation 29 as follows.

$$\begin{aligned}&\frac{1}{(t+2)^2}\|\nabla_y \mathbf{L}(\tilde{f}(t))\|^2 - \frac{2}{t+2}(\mathbf{L}(\tilde{f}(t)) - \mathbf{L}(f^*))\\ &\geq \frac{1}{(t+2)^2}\|\nabla_y \mathbf{L}(\tilde{f}(t))\|^2 - \frac{2}{t+2}\nabla_y \mathbf{L}(\tilde{f}(t))(\tilde{f}(t) - f^*) + \frac{1}{t+2}\|\tilde{f}(t) - f^*\|^2\end{aligned} \tag{31}$$

Also, $\mathbf{L}(y)$ is piece-wise smooth since it is a maximum of two smooth functions. Define each piece function as $\mathbf{L}_i(y)$ for $i = 1, 2$. In other words,

$$\mathbf{L}(y) = \max(\frac{1}{2}y^2 - y + \tilde{x}(t+1), \frac{1}{2}y^2) = \max(\mathbf{L}_1(y), \mathbf{L}_2(y)) \tag{32}$$

The difference of two piece functions is a line.

$$\|\mathbf{L}_1(y) - \mathbf{L}_2(y)\| = \|y - \tilde{x}(t+1)\| \tag{33}$$

$$\therefore \mathbf{L}_i(y) \geq \mathbf{L}(y) - \|y - \tilde{x}(t+1)\| \tag{34}$$

Suppose $\mathbf{L}(y) = \mathbf{L}_i(y)$ at a point $y \in \mathbb{R}$. For any $\delta \in \mathbb{R}$,

$$\begin{aligned}\mathbf{L}_i(y+\delta) &\geq \mathbf{L}(y+\delta) - \|y+\delta - \tilde{x}(t+1)\| && (\because \text{Inequality } (34)) && (35)\\ &\geq \mathbf{L}(y+\delta) - (\|y - \tilde{x}(t+1)\| + \|\delta\|) && (\because \text{Triangle Inequality of Norm}) && (36)\\ &\geq \mathbf{L}(y+\delta) - \|\delta\| && (\because \text{Positiveness of Norm}) && (37)\end{aligned}$$

Note that the error term $\|y + \delta - \tilde{x}(t+1)\|$ appears only when $\tilde{x}(t+1) \in [y, y+\delta]$, i.e., when the segment $[y, y+\delta]$ crosses the singularity $\tilde{x}(t+1)$. The error maximizes only if $y = \tilde{x}(t+1)$.
The smoothness constant is 2 for both $\mathbf{L}_1$ and $\mathbf{L}_2$, since

$$\begin{aligned}\|\nabla_y \mathbf{L}_1(x_1) - \nabla_y \mathbf{L}_1(x_2)\| &= \|2x_1 - 1 - 2x_2 + 1\| = 2\|x_1 - x_2\|\\ \|\nabla_y \mathbf{L}_2(x_1) - \nabla_y \mathbf{L}_2(x_2)\| &= \|2x_1 - 2x_2\| = 2\|x_1 - x_2\|\end{aligned}$$

We now use the smoothness of $\mathbf{L}_i$ and the inequality 34, 37 to show the main result. Without loss of generality, let $\mathbf{L}(\tilde{f}(t)) = \mathbf{L}_i(\tilde{f}(t))$. For any $\alpha \in \mathbb{R}$, by the property of 2-smooth function,

$$\begin{aligned}\mathbf{L}_i(\tilde{f}(t) - \alpha\nabla_y \mathbf{L}_i(\tilde{f}(t))) &\leq \mathbf{L}_i(\tilde{f}(t)) + \nabla_y \mathbf{L}_i(\tilde{f}(t)))(-\alpha\nabla_y \mathbf{L}_i(\tilde{f}(t)))) + \frac{2}{2}\alpha^2\|\nabla_y \mathbf{L}_i(\tilde{f}(t)))\|^2 && (\because \text{2-smooth})\\ &= \mathbf{L}_i(\tilde{f}(t)) + (\alpha^2 - \alpha)\|\nabla_y \mathbf{L}_i(\tilde{f}(t)))\|^2\\ \mathbf{L}_i(\tilde{f}(t) - \alpha\nabla_y \mathbf{L}_i(\tilde{f}(t))) &\geq \mathbf{L}(\tilde{f}(t) - \alpha\nabla_y \mathbf{L}_i(\tilde{f}(t))) - \|\tilde{f}(t) - \alpha\nabla_y \mathbf{L}_i(\tilde{f}(t)) - \tilde{x}(t+1)\| && (\because (34))\\ &\geq \mathbf{L}(\tilde{f}(t) - \alpha\nabla_y \mathbf{L}_i(\tilde{f}(t))) - \alpha\|\nabla_y \mathbf{L}_i(\tilde{f}(t))\| && (\because (37))\end{aligned}$$

$$\therefore \mathbf{L}(\tilde{f}(t) - \alpha\nabla_y \mathbf{L}_i(\tilde{f}(t))) \leq \mathbf{L}(\tilde{f}(t)) + (\alpha^2 - \alpha)\|\nabla_y \mathbf{L}_i(\tilde{f}(t)))\|^2 + \alpha\|\nabla_y \mathbf{L}_i(\tilde{f}(t))\| \tag{38}$$

Since $f^* = \mathrm{ReLU1}(x)$, $\mathbf{f}^* = \mathrm{ReLU1}(\tilde{x}(t+1))$ and $x^2 - y^2 \geq -\|x+y\|\|x-y\|$,

$$\begin{aligned}
\mathbf{L}(\tilde{f}(t) - \alpha\nabla_y \mathbf{L}_i(\tilde{f}(t))) &\geq \mathbf{L}(\mathbf{f}^*) \qquad (\because \mathbf{f}^* = \operatorname*{arg\,min}_{y\in\mathbb{R}} \mathbf{L}(y)) \qquad (39)\\
&= \mathbf{L}(f^*) + \big(\mathbf{L}(\mathbf{f}^*) - \mathbf{L}(f^*)\big)\\
&= \mathbf{L}(f^*) + \left(\mathrm{ReLU}(\tilde{x}(t+1) - \mathbf{f}^*) + \frac{1}{2}(\mathbf{f}^*)^2 - \mathrm{ReLU}(\tilde{x}(t+1) - f^*) - \frac{1}{2}(f^*)^2\right)\\
&= \mathbf{L}(f^*) + \left(\frac{1}{2}(\mathbf{f}^*)^2 - \frac{1}{2}(f^*)^2 + \mathrm{ReLU}(\tilde{x}(t+1) - \mathbf{f}^*) - \mathrm{ReLU}(\tilde{x}(t+1) - f^*)\right)\\
&\geq \mathbf{L}(f^*) - \frac{1}{2}\|\mathbf{f}^* + f^*\|\|\mathbf{f}^* - f^*\| - \|\mathbf{f}^* - f^*\| \qquad (\because \text{Definition of ReLU})\\
&= \mathbf{L}(f^*) - \frac{1}{2}\|\mathrm{ReLU1}(\tilde{x}(t+1)) + \mathrm{ReLU1}(x) + 2\|\|\mathrm{ReLU1}(\tilde{x}(t+1)) - \mathrm{ReLU1}(x)\|\\
&\geq \mathbf{L}(f^*) - \frac{1}{2}\cdot 4 \cdot \|\mathrm{ReLU1}(\tilde{x}(t+1)) - \mathrm{ReLU1}(x)\|\\
&\geq \mathbf{L}(f^*) - 2\|\min(x - \tilde{x}(t+1), 1)\| \qquad (40)
\end{aligned}$$

Merging two inequalities (38) and (40), we yield the following inequality.

$$\begin{aligned}
\mathbf{L}(\tilde{f}(t)) + (\alpha^2 - \alpha)\|\nabla_y \mathbf{L}_i(\tilde{f}(t)))\|^2 + \|\nabla_y \mathbf{L}_i(\tilde{f}(t))\| &\geq \mathbf{L}(f^*) - 2\|\min(x - \tilde{x}(t+1), 1)\|\\
2\|\min(x - \tilde{x}(t+1), 1)\| + \alpha\|\nabla_y \mathbf{L}_i(\tilde{f}(t))\| &\geq (\alpha - \alpha^2)\|\nabla_y \mathbf{L}_i(\tilde{f}(t)))\|^2 - (\mathbf{L}(\tilde{f}(t)) - \mathbf{L}(f^*))\\
\frac{4}{t+2}\|\min(x - \tilde{x}(t+1), 1)\| + \frac{2\alpha}{t+2}\|\nabla_y \mathbf{L}_i(\tilde{f}(t))\| &\geq \frac{2(\alpha - \alpha^2)}{t+2}\|\nabla_y \mathbf{L}_i(\tilde{f}(t)))\|^2 - \frac{2}{t+2}(\mathbf{L}(\tilde{f}(t)) - \mathbf{L}(f^*))
\end{aligned}$$

We let $\alpha = \frac{1}{2} - \frac{1}{2}\sqrt{\frac{t}{t+2}}$, since $\alpha = \frac{1}{2} \pm \frac{1}{2}\sqrt{1 - \frac{2}{t+2}} = \frac{1}{2} \pm \frac{1}{2}\sqrt{\frac{t}{t+2}}$ satisfies $2(\alpha - \alpha^2) = \frac{1}{t+2}$.

$$\begin{aligned}
&\frac{4}{t+2}\|\min(x - \tilde{x}(t+1), 1)\| + \frac{1}{t+2}\left(1 - \sqrt{\frac{t}{t+2}}\right)\|\nabla_y \mathbf{L}_i(\tilde{f}(t))\|\\
&\geq \frac{1}{(t+2)^2}\|\nabla_y \mathbf{L}_i(\tilde{f}(t)))\|^2 - \frac{2}{t+2}(\mathbf{L}(\tilde{f}(t)) - \mathbf{L}(f^*))\\
&\geq \frac{1}{(t+2)^2}\|\nabla_y \mathbf{L}(\tilde{f}(t))\|^2 - \frac{2}{t+2}\nabla_y \mathbf{L}(\tilde{f}(t))(\tilde{f}(t) - f^*) + \frac{1}{t+2}\|\tilde{f}(t) - f^*\|^2 \qquad (\because \text{Inequality (31)})\\
&= \|\tilde{f}(t+1) - f^*\|^2 - \frac{t+1}{t+2}\|\tilde{f}(t) - f^*\|^2 \qquad (\because \text{Equation (29)})\\
&\therefore \|\tilde{f}(t+1) - f^*\|^2 \leq \frac{t+1}{t+2}\|\tilde{f}(t) - f^*\|^2 + \frac{4}{t+2}\|\min(\tilde{x}(t+1) - x, 1)\| + \frac{1}{t+2}\left(1 - \sqrt{\frac{t}{t+2}}\right)\|\nabla_y \mathbf{L}(\tilde{f}(t))\|
\end{aligned}$$

We now analyze the sub-gradient magnitude term $\|\nabla_y \mathbf{L}(\tilde{f}(t))\|$. By definition of $\mathbf{L}$ and triangle inequality,

$$\|\nabla_y \mathbf{L}(\tilde{f}(t))\| = \|\tilde{f}(t) - \mathbb{H}(\tilde{x}(t+1) - \tilde{f}(t))\| \leq \|\tilde{f}(t)\| + 1$$

We assumed that $\|\tilde{f}(t)\| \leq M \in \mathbb{R}^+$ for any $t \in \mathbb{N}$. Thus,

$$\|\tilde{f}(t+1) - f^*\|^2 \leq \frac{t+1}{t+2}\|\tilde{f}(t) - f^*\|^2 + \frac{4}{t+2}\|\tilde{x}(t+1) - x\| + \frac{1}{t+2}\left(1 - \sqrt{\frac{t}{t+2}}\right)(M+1) \qquad (41)$$

Solving the recurrence relation (41) through time t leads to the upper bound of the approximation error.

$$\|\tilde{f}(t+1) - f^*\|^2 \leq \frac{1}{t+2}\|\tilde{f}(0) - f^*\|^2 + \frac{4}{t+2}\sum_{i=1}^{t+1}\|\min(x - \tilde{x}(i), 1)\| + \frac{M+1}{t+2}\sum_{i=0}^{t}\left(1 - \sqrt{\frac{i}{i+2}}\right) \qquad (42)$$

Recovering the notations $f^* \to f^*(x)$ and $t+1 \to t$, the above inequality becomes the desired inequality.

$$\|\tilde{f}(t) - f^*(x)\|^2 \leq \frac{1}{t+1}\|\tilde{f}(0) - f^*(x)\|^2 + \frac{4}{t+1}\sum_{i=1}^{t}\|\min(x - \tilde{x}(i), 1)\| + \frac{M+1}{t+1}\sum_{i=1}^{t}\left(1 - \sqrt{\frac{i-1}{i+1}}\right) \qquad (43)$$

□

Lemma H.4. *The harmonic number* $H_n = \sum_{i=1}^{n} \frac{1}{i}$ *satisfies* $\lim_{n\to\infty} \frac{1}{n} H_n = 0$.

Proof. Let $1 \le k \le n$. For every $i \le k$, $\frac{1}{i} \le 1$. For every $k < i \le n$, $\frac{1}{i} \le \frac{1}{k}$. Thus

$$H_n = \sum_{i=1}^{n} \frac{1}{i} \le 1 \cdot k + \frac{1}{k} \cdot (n-k) = k + \frac{1}{k} - 1$$

If we choose $\sqrt{n} \le k \le \sqrt{n} + 1$, then $k - 1 \le \sqrt{n}$ and $\frac{1}{k} \le \sqrt{n}$. Thus, $H_n \le 2\sqrt{n}$, and

$$\frac{1}{n} H_n \le \frac{2}{\sqrt{n}}$$

Therefore, $\lim_{n\to\infty} \frac{1}{n} H_n = 0$. □

Lemma H.5. *Let* $t \in \mathbb{N}$. $\lim_{t\to\infty} \frac{1}{t+1} \sum_{i=1}^{t} \left(1 - \sqrt{\frac{i-1}{i+1}}\right) = 0$

Proof.

$$0 \le 1 - \sqrt{\frac{i-1}{i+1}} = 1 - \frac{\sqrt{(i-1)(i+1)}}{i+1} = \frac{i+1-\sqrt{(i-1)(i+1)}}{i+1} \tag{44}$$

$$\le \frac{i+1-\sqrt{(i-1)(i-1)}}{i+1} = \frac{i+1-(i-1)}{i+1} = \frac{2}{i+1} \tag{45}$$

$$\therefore 0 \le \frac{1}{t+1} \sum_{i=1}^{t} (1 - \sqrt{\frac{i-1}{i+1}}) \le \frac{1}{t+1} \sum_{i=1}^{t} \frac{2}{i+1} \tag{46}$$

By Lemma H.4,

$$\lim_{t\to\infty} \frac{1}{t+1} \sum_{i=1}^{t} \frac{2}{i+1} = \lim_{t\to\infty} \frac{2}{t+1}(H_t - 1) = 0 - 0 = 0$$

Therefore, $0 \le \lim_{t\to\infty} \frac{1}{t+1} \sum_{i=1}^{t} \left(1 - \sqrt{\frac{i-1}{i+1}}\right) \le 0$, and thus $\lim_{t\to\infty} \frac{1}{t+1} \sum_{i=1}^{t} \left(1 - \sqrt{\frac{i-1}{i+1}}\right) = 0$. □

Corollary H.6. *Let* $\|\tilde{f}(t)\| < M \in \mathbb{R}^+$*, then*

- *(Exact Input) If* $\tilde{x}(t) = x$*, then* $\|\tilde{f}(t) - f^*\| \to 0$ *as* $t \to \infty$.
- *(Deterministic Input) If* $\|\tilde{x}(t) - x\| = \mathcal{O}(\frac{1}{t})$*, then* $\|\tilde{f}(t) - f^*\| \to 0$ *as* $t \to \infty$.
- *(Stochastic Input) If* $\mathbb{E}[\|\tilde{x}(t) - x\|] = \mathcal{O}(\frac{1}{t})$*, then* $\mathbb{E}[\|\tilde{f}(t) - f^*\|] \to 0$ *as* $t \to \infty$.

Proof. **(Exact Input)** We first prove it for the case of exact input, i.e., if $\tilde{x}(t) = x$, then $\|\tilde{f}(t) - f^*\| \to 0$ as $t \to \infty$.

$$\|\tilde{f}(t) - f^*(x)\|^2 \le \frac{\|\tilde{f}(0) - f^*(x)\|^2}{t+1} + \frac{M+1}{t+1} \sum_{i=1}^{t} \left((1 - \sqrt{\frac{i-1}{i+1}} \right) + \frac{4}{t+1} \sum_{i=1}^{t} \min(\|x - \tilde{x}(i)\|, 1) \quad \text{(Theorem 4.3)}$$

$$= \frac{\|\tilde{f}(0) - f^*(x)\|^2}{t+1} + \frac{M+1}{t+1} \sum_{i=1}^{t} \left(1 - \sqrt{\frac{i-1}{i+1}} \right) \quad \text{(Assumption on } \tilde{x}(t))$$

By Lemma H.4 and H.5,

$$\lim_{t\to\infty} \|\tilde{f}(t) - f^*(x)\|^2 \le \lim_{t\to\infty} \frac{\|\tilde{f}(0) - f^*(x)\|^2}{t+1} + \lim_{t\to\infty} \frac{M+1}{t+1} \sum_{i=1}^{t} \left(1 - \sqrt{\frac{i-1}{i+1}} \right) = 0 + (M+1) \cdot 1 \cdot 0 = 0$$

(Deterministic Input) We show the deterministic input case that if $\|\tilde{x}(t)-x\| = \mathcal{O}(\frac{1}{t})$, then $\|\tilde{f}(t)-f^*\| \to 0$ as $t \to \infty$.

$$\begin{aligned}
\|\tilde{f}(t)-f^*(x)\|^2 &\leq \frac{\|\tilde{f}(0)-f^*(x)\|^2}{t+1} + \frac{M+1}{t+1}\sum_{i=1}^{t}\left(1-\sqrt{\frac{i-1}{i+1}}\right) + \frac{4}{t+1}\sum_{i=1}^{t}\min(\|x-\tilde{x}(i)\|,1) \quad \text{(Theorem 4.3)} \\
&\leq \frac{\|\tilde{f}(0)-f^*(x)\|^2}{t+1} + \frac{M+1}{t+1}\sum_{i=1}^{t}\left(1-\sqrt{\frac{i-1}{i+1}}\right) + \frac{4}{t+1}\sum_{i=1}^{t}\min\left(\frac{C}{t},1\right) \quad \text{(Assumption on } \tilde{x}(t)) \\
&\leq \frac{\|\tilde{f}(0)-f^*(x)\|^2}{t+1} + \frac{M+1}{t+1}\sum_{i=1}^{t}\left(1-\sqrt{\frac{i-1}{i+1}}\right) + \frac{4C}{t+1}\sum_{i=1}^{t}\frac{1}{t}
\end{aligned}$$

By Lemma H.4 and H.5,

$$\begin{aligned}
\lim_{t\to\infty}\|\tilde{f}(t)-f^*(x)\|^2 &\leq \lim_{t\to\infty}\frac{\|\tilde{f}(0)-f^*(x)\|^2}{t+1} + \lim_{t\to\infty}\frac{M+1}{t+1}\sum_{i=1}^{t}\left(1-\sqrt{\frac{i-1}{i+1}}\right) + \lim_{t\to\infty}\frac{4C}{t+1}\sum_{i=1}^{t}\frac{1}{t} \\
&= 0 + (M+1)\cdot 1\cdot 0 + 4C\cdot 1\cdot 0 = 0
\end{aligned}$$

(Stochastic Input) We now demonstrate the stochastic input case that if $\mathbb{E}[\|\tilde{x}(t)-x\|] = \mathcal{O}(\frac{1}{t})$, then $\mathbb{E}[\|\tilde{f}(t)-f^*\|] \to 0$ as $t \to \infty$.

$$\begin{aligned}
\mathbb{E}\left[\|\tilde{f}(t)-f^*(x)\|^2\right] &\leq \mathbb{E}\left[\frac{\|\tilde{f}(0)-f^*(x)\|^2}{t+1} + \frac{M+1}{t+1}\sum_{i=1}^{t}(1-\sqrt{\frac{i-1}{i+1}}) + \frac{4}{t+1}\sum_{i=1}^{t}\min(\|x-\tilde{x}(i)\|,1)\right] \\
&= \frac{\|\tilde{f}(0)-f^*(x)\|^2}{t+1} + \frac{M+1}{t+1}\sum_{i=1}^{t}(1-\sqrt{\frac{i-1}{i+1}}) + \mathbb{E}\left[\frac{4}{t+1}\sum_{i=1}^{t}\min(\|x-\tilde{x}(i)\|,1)\right] \\
&\leq \frac{\|\tilde{f}(0)-f^*(x)\|^2}{t+1} + \frac{M+1}{t+1}\sum_{i=1}^{t}(1-\sqrt{\frac{i-1}{i+1}}) + \frac{4}{t+1}\sum_{i=1}^{t}\min(\frac{C}{t},1) \quad \text{(Assumption on } \tilde{x}(t)) \\
&\leq \frac{\|\tilde{f}(0)-f^*(x)\|^2}{t+1} + \frac{M+1}{t+1}\sum_{i=1}^{t}(1-\sqrt{\frac{i-1}{i+1}}) + \frac{4C}{t+1}\sum_{i=1}^{t}\frac{1}{t}
\end{aligned}$$

By Lemma H.4 and H.5,

$$\begin{aligned}
\lim_{t\to\infty}\mathbb{E}\left[\|\tilde{f}(t)-f^*(x)\|^2\right] &\leq \lim_{t\to\infty}\frac{\|\tilde{f}(0)-f^*(x)\|^2}{t+1} + \lim_{t\to\infty}\frac{M+1}{t+1}\sum_{i=1}^{t}\left(1-\sqrt{\frac{i-1}{i+1}}\right) + \lim_{t\to\infty}\frac{4C}{t+1}\sum_{i=1}^{t}\frac{1}{t} \\
&= 0 + (M+1)\cdot 1\cdot 0 + 4C\cdot 1\cdot 0 = 0
\end{aligned}$$

□

Theorem H.7. *Let an input activation $\tilde{x}(t)$ be signed schedule coded over an N weighted input spike trains, i.e.,*

$$\tilde{x}^{(k)}(t) = \tilde{x}^{(k)}(t-1) - \sum_{i=1}^{N} W_i\left(\eta(t)(2I_i^{(k)}(t)-1)\right) \tag{47}$$

$\tilde{f}(t) = \tilde{f}(t-1) - \eta(t)\big(2\cdot s(t)-1\big)$ is signed schedule coded with the spike output $s(t)$ of signGD-based neuronal dynamics. If α_1, α_2 β_1, β_2 and η satisfies $\eta(1) = \alpha_2(1) = \beta_2(1)$ and

$$\frac{\eta(t)}{\eta(t-1)} = \beta_1(t)\frac{\beta_2(t)}{\beta_2(t-1)} = \frac{1}{\alpha_1(t-1)}\frac{\alpha_2(t)}{\alpha_2(t-1)} \tag{48}$$

Then the dynamical system of $\tilde{f}(t)$ is equivalent to a sign gradient descent method

$$\tilde{f}(t) = \tilde{f}(t-1) - \eta(t)sgn(\nabla_y\mathcal{L}(\tilde{f}(t-1);\tilde{x}(t))) \tag{49}$$

Proof. We start from the equation (49) to derive the signGD-based neuron in Definition 5.2. We define $u(t), v(t)$ to be

$$u(t) = \frac{\beta_2(t-1)}{\eta(t-1)}\tilde{f}(t-1) \qquad v(t) = \frac{\alpha_2(t)}{\eta(t)}\tilde{x}(t) \qquad s(t) = \mathbb{H}\big(\nabla_y \mathcal{L}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t); \frac{\eta(t)}{\alpha_2(t)}v(t))\big) \tag{50}$$

From equation 49, by substituting $\tilde{f}(t)$ with $u(t)$ and $\tilde{x}(t)$ with $v(t)$,

$$\begin{aligned}
\frac{\eta(t)}{\beta_2(t)}u(t+1) &= \frac{\eta(t-1)}{\beta_2(t-1)}u(t) - \eta(t)\text{sgn}(\nabla_y \mathcal{L}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t); \frac{\eta(t)}{\alpha_2(t)}v(t))) \\
u(t+1) &= \frac{\beta_2(t)}{\eta(t)}\frac{\eta(t-1)}{\beta_2(t-1)}u(t) - \frac{\beta_2(t)}{\eta(t)}\eta(t)\text{sgn}(\nabla_y \mathcal{L}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t); \frac{\eta(t)}{\alpha_2(t)}v(t))) \\
&= \frac{\beta_2(t)}{\eta(t)}\frac{\eta(t-1)}{\beta_2(t-1)}u(t) - \beta_2(t)\text{sgn}(\nabla_y \mathcal{L}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t); \frac{\eta(t)}{\alpha_2(t)}v(t)))
\end{aligned} \tag{51}$$

By equation (48), $\beta_1(t) = \frac{\eta(t)}{\beta_2(t)}\frac{\beta_2(t-1)}{\eta(t-1)}$. The equation (51) becomes identical to the equation (14) in the definition of signGD-based neuron.

$$u(t+1) = \beta_1(t)u(t) - \beta_2(t)(2\cdot s(t) - 1) \tag{52}$$

We now derive the membrane equation for $v^{(k)}(t)$.

$$\begin{aligned}
v^{(k)}(t+1) - \alpha_1(t)v^{(k)}(t) &= \frac{\alpha_2(t+1)}{\eta(t+1)}\tilde{x}^{(k)}(t+1) - \alpha_1(t)\frac{\alpha_2(t)}{\eta(t)}\tilde{x}^{(k)}(t) \\
&= \frac{\alpha_2(t+1)}{\eta(t+1)}\Big(\tilde{x}^{(k)}(t) - \sum_{i=1}^{N} W_i\big(\eta(t+1)(2I_i^{(k)}(t+1) - 1)\big)\Big) - \alpha_1(t)\frac{\alpha_2(t)}{\eta(t)}\tilde{x}^{(k)}(t)
\end{aligned} \tag{53}$$

By equation (48), $\alpha_1(t)\frac{\alpha_2(t)}{\eta(t)} = \frac{\alpha_2(t+1)}{\eta(t+1)}$. Since $I^{(k)}(t+1) = \sum_{i=1}^N W_i(I_i^{(k)})(t+1)$ and $W = \sum_{i=1}^N W_i$,

$$\begin{aligned}
v^{(k)}(t+1) - \alpha_1(t)v^{(k)}(t) &= \frac{\alpha_2(t+1)}{\eta(t+1)}\Big(\tilde{x}^{(k)}(t) - \sum_{i=1}^{N} W_i\big(\eta(t+1)(2I_i^{(k)}(t+1) - 1)\big)\Big) - \alpha_1(t)\frac{\alpha_2(t)}{\eta(t)}\tilde{x}^{(k)}(t) \\
&= -\frac{\alpha_2(t+1)}{\eta(t+1)}\sum_{i=1}^{N} W_i\big(\eta(t+1)(2I_i^{(k)}(t+1) - 1)\big) \\
&= -\alpha_2(t+1)\sum_{i=1}^{N} W_i(2I_i^{(k)}(t+1) - 1) = -\alpha_2(t+1)\big(2(\sum_{i=1}^{N} W_i I_i^{(k)}(t+1)) - W\big) \\
&= -\alpha_2(t+1)(2\cdot I^{(k)} - W) \\
\therefore v^{(k)}(t+1) &= \alpha_1(t)v^{(k)}(t) - \alpha_2(t+1)(2\cdot I^{(k)} - W)
\end{aligned} \tag{54}$$

We thus derive the equation for $v^{(k)}(t)$. The proof direction from signGD-based neuronal dynamics to optimization algorithm can be similarly derived by defining $\tilde{f}(t)$ and $\tilde{x}(t)$ as (50). □

Corollary H.8. *SignGD-based neuronal dynamics (Def. 5.2) satisfying Eq. 15 and $\eta(t) = \alpha_2(t) = \beta_2(t)$ is equivalent to signGD (Eq. 16) if for $s(t)$ (Eq. 13),*

- ***(ReLU)** $\mathcal{L}(y;x) = \frac{1}{2}\|y - \text{ReLU(x)}\|^2$ and $s(t) = \mathbb{H}\big(v(t)\big)\mathbb{H}\big(u(t) - \beta_1(t)v(t)\big) + \mathbb{H}\big(-v(t)\big)\mathbb{H}\big(u(t)\big)$*
- ***(Sigmoid approximation of GELU)** (Hendrycks & Gimpel, 2016)) $\mathcal{L}(y;x) = \frac{1}{2}\|y - \frac{x}{1+e^{-1.702x}}\|^2$ and $s(t) = \mathbb{H}\big((1 + (e^{-1.702})^{v(t)})u(t) - \beta_1(t)v(t)\big)$*
- ***(LeakyReLU)** $\mathcal{L}(y;x) = \frac{1}{2}\|y - \text{LeakyReLU(x, }\delta)\|^2$, where δ is the negative slope, and $s(t) = \mathbb{H}(v(t))\mathbb{H}\big(u(t) - \beta_1(t)v(t)\big) + \mathbb{H}(-v(t))\mathbb{H}\big(u(t) - \delta\beta_1(t)v(t)\big)$*

Proof. **(ReLU)** We first show that signGD-based neuronal dynamics with a spiking mechanism $s(t) = \mathbb{H}\big(v(t)\big)\mathbb{H}\big(u(t) - \beta_1(t)v(t)\big) + \mathbb{H}\big(-v(t)\big)\mathbb{H}\big(u(t)\big)$ is equivalent to the signGD algorithm (Eq. 16) with the objective function $\mathcal{L}(y;x) = \frac{1}{2}\|y - \text{ReLU}(x)\|^2$, which has ReLU$(x)$ as its minimizer.

Since $\mathcal{L}(y;x)=\frac{1}{2}\|y-\mathrm{ReLU}(x)\|^2$, $\nabla_y\mathcal{L}(y;x)=(y-\mathrm{ReLU}(x))=\mathbb{H}(x)(y-x)+\mathbb{H}(-x)y$

$$\begin{aligned}
s(t)&=\mathbb{H}\Big(\nabla_y\mathcal{L}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t);\frac{\eta(t)}{\alpha_2(t)}v(t))\Big)\\
&=\mathbb{H}\Big(\mathbb{H}(\frac{\eta(t)}{\alpha_2(t)}v(t))\big(\frac{\eta(t-1)}{\beta_2(t-1)}u(t)-\frac{\eta(t)}{\alpha_2(t)}v(t)\big)+\mathbb{H}(-\frac{\eta(t)}{\alpha_2(t)}v(t))\frac{\eta(t-1)}{\beta_2(t-1)}u(t)\Big)\\
&=\mathbb{H}\Big(\frac{\eta(t)}{\alpha_2(t)}v(t)\Big)\mathbb{H}\Big(\frac{\eta(t-1)}{\beta_2(t-1)}u(t)-\frac{\eta(t)}{\alpha_2(t)}v(t)\Big)+\mathbb{H}\Big(-\frac{\eta(t)}{\alpha_2(t)}v(t)\Big)\mathbb{H}\Big(\frac{\eta(t-1)}{\beta_2(t-1)}u(t)\Big)\\
&=\mathbb{H}(v(t))\mathbb{H}\Big(u(t)-\frac{\eta(t)\beta_2(t-1)}{\eta(t-1)\alpha_2(t)}v(t)\Big)+\mathbb{H}(-v(t))\mathbb{H}(u(t))\\
&=\mathbb{H}(v(t))\mathbb{H}\Big(u(t)-\beta_1(t)\frac{\beta_2(t)}{\alpha_2(t)}v(t)\Big)+\mathbb{H}(-v(t))\mathbb{H}(u(t))\\
&=\mathbb{H}(v(t))\mathbb{H}\Big(u(t)-\beta_1(t)v(t)\Big)+\mathbb{H}(-v(t))\mathbb{H}(u(t))
\end{aligned}$$

(GELU) We now show that signGD-based neuronal dynamics with a spiking mechanism $s(t)=\mathbb{H}\big((1+(e^{-1.702})^{v(t)})u(t)-\beta_1(t)v(t)\big)$ is equivalent to the signGD algorithm (Eq. 16) with the objective function $\mathcal{L}(y;x)=\frac{1}{2}\|y-\frac{x}{1+e^{-1.702x}}\|^2$, which has the sigmoid approximation of GELU function value $\frac{x}{1+e^{-1.702x}}$ as its minimizer.

Since $\mathcal{L}(y;x)=\frac{1}{2}\|y-g(x)\|^2$, $\nabla_y\mathcal{L}(y;x)=(y-g(x))=y-\frac{x}{1+e^{-1.702x}}$

$$\begin{aligned}
s(t)&=\mathbb{H}\Big(\nabla_y\mathcal{L}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t);\frac{\eta(t)}{\alpha_2(t)}v(t))\Big)\\
&=\mathbb{H}\Big(\frac{\eta(t-1)}{\beta_2(t-1)}u(t)-\frac{\frac{\eta(t)}{\alpha_2(t)}v(t)}{1+e^{-1.702\frac{\eta(t)}{\alpha_2(t)}v(t)}}\Big)\\
&=\mathbb{H}\Big((1+e^{-1.702\frac{\eta(t)}{\alpha_2(t)}v(t)})\frac{\eta(t-1)}{\beta_2(t-1)}u(t)-\frac{\eta(t)}{\alpha_2(t)}v(t)\Big)\\
&=\mathbb{H}\Big((1+e^{-1.702\frac{\eta(t)}{\alpha_2(t)}v(t)})u(t)-\frac{\eta(t)\beta_2(t-1)}{\eta(t-1)\alpha_2(t)}v(t)\Big)\\
&=\mathbb{H}\Big((1+e^{-1.702\frac{\eta(t)}{\alpha_2(t)}v(t)})u(t)-\beta_1(t)\frac{\beta_2(t)}{\alpha_2(t)}v(t)\Big)\\
&=\mathbb{H}\Big((1+e^{-1.702v(t)})u(t)-\beta_1(t)v(t)\Big)
\end{aligned}$$

(LeakyReLU) Finally, we show that signGD-based neuronal dynamics with a spiking mechanism $s(t)=\mathbb{H}(v(t))\mathbb{H}\big(u(t)-\beta_1(t)v(t)\big)+\mathbb{H}(-v(t))\mathbb{H}\big(u(t)-\delta\beta_1(t)v(t)\big)$ is equivalent to the signGD algorithm (Eq. 16) with the objective function $\mathcal{L}(y;x)=\frac{1}{2}\|y-\text{LeakyReLU(x, }\delta)\|^2$, which has LeakyReLU(x, δ) as its minimizer.

Since $\mathcal{L}(y;x)=\frac{1}{2}\|y-\mathrm{LeakyReLU}(x,\delta)\|^2$, $\nabla_y\mathcal{L}(y;x)=y-\mathrm{LeakyReLU}(x,\delta)=\mathbb{H}(x)(y-x)+\mathbb{H}(-x)(y-\delta x)$

$$\begin{aligned}
s(t)&=\mathbb{H}\Big(\nabla_y\mathcal{L}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t);\frac{\eta(t)}{\alpha_2(t)}v(t))\Big)\\
&=\mathbb{H}(\frac{\eta(t)}{\alpha_2(t)}v(t))\mathbb{H}\big(\frac{\eta(t-1)}{\beta_2(t-1)}u(t)-\frac{\eta(t)}{\alpha_2(t)}v(t)\big)+\mathbb{H}(-\frac{\eta(t)}{\alpha_2(t)}v(t))\mathbb{H}\big(\frac{\eta(t-1)}{\beta_2(t-1)}u(t)-\delta\frac{\eta(t)}{\alpha_2(t)}v(t)\big)\\
&=\mathbb{H}(v(t))\mathbb{H}\big(u(t)-\frac{\eta(t)\beta_2(t-1)}{\eta(t-1)\alpha_2(t)}v(t)\big)+\mathbb{H}(-v(t))\mathbb{H}\big(u(t)-\delta\frac{\eta(t)\beta_2(t-1)}{\eta(t-1)\alpha_2(t)}v(t)\big)\\
&=\mathbb{H}(v(t))\mathbb{H}\big(u(t)-\beta_1(t)\frac{\beta_2(t)}{\alpha_2(t)}v(t)\big)+\mathbb{H}(-v(t))\mathbb{H}\big(u(t)-\delta\beta_1(t)\frac{\beta_2(t)}{\alpha_2(t)}v(t)\big)\\
&=\mathbb{H}(v(t))\mathbb{H}\big(u(t)-\beta_1(t)v(t)\big)+\mathbb{H}(-v(t))\mathbb{H}\big(u(t)-\delta\beta_1(t)v(t)\big)
\end{aligned}$$

□

Corollary H.9. *SignGD (Eq. 16) with $\mathcal{L}\big(y;(x_1,x_2)\big)=\frac{1}{2}\|y-\max(x_1,x_2)\|^2$ is equivalent to the signGD-based neuronal dynamics (Def. 5.2) satisfying equation 15, $\alpha_2(t)=\beta_2(t)$ and*

$$s(t)=\mathbb{H}(v^{(1)}(t)-v^{(2)}(t))(u(t)-\beta_1(t)v^{(1)}(t))+\mathbb{H}(v^{(2)}(t)-v^{(1)}(t))(u(t)-\beta_1(t)v^{(2)}(t))$$

Proof. Since $\mathcal{L}\big(y;(x_1,x_2)\big)=\frac{1}{2}\|y-\max(x_1,x_2)\|^2$, $\nabla_y\mathcal{L}(y;x_1,x_2)=y-\max(x_1,x_2)=\mathbb{H}(x_1-x_2)(y-x_1)+\mathbb{H}(x_2-x_1)(y-x_2)$.

$$\begin{aligned}
s(t)&=\mathbb{H}\Big(\nabla_y\mathcal{L}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t);\frac{\eta(t)}{\alpha_2(t)}v(t))\Big)\\
&=\mathbb{H}(\frac{\eta(t)}{\alpha_2(t)}v^{(1)}(t)-\frac{\eta(t)}{\alpha_2(t)}v^{(2)}(t))\mathbb{H}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t)-\frac{\eta(t)}{\alpha_2(t)}v^{(1)}(t))\\
&+\mathbb{H}(\frac{\eta(t)}{\alpha_2(t)}v^{(2)}(t)-\frac{\eta(t)}{\alpha_2(t)}v^{(1)}(t))\mathbb{H}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t)-\frac{\eta(t)}{\alpha_2(t)}v^{(2)}(t))\\
&=\mathbb{H}(v^{(1)}(t)-v^{(2)}(t))\mathbb{H}(u(t)-\frac{\eta(t)\beta_2(t-1)}{\eta(t-1)\alpha_2(t)}v^{(1)}(t))+\mathbb{H}(v^{(2)}(t)-v^{(1)}(t))\mathbb{H}(u(t)-\frac{\eta(t)\beta_2(t-1)}{\eta(t-1)\alpha_2(t)}v^{(2)}(t))\\
&=\mathbb{H}(v^{(1)}(t)-v^{(2)}(t))\mathbb{H}(u(t)-\frac{\eta(t)\beta_2(t-1)}{\eta(t-1)\alpha_2(t)}v^{(1)}(t))+\mathbb{H}(v^{(2)}(t)-v^{(1)}(t))\mathbb{H}(u(t)-\frac{\eta(t)\beta_2(t-1)}{\eta(t-1)\alpha_2(t)}v^{(2)}(t))\\
&=\mathbb{H}(v^{(1)}(t)-v^{(2)}(t))\mathbb{H}(u(t)-\beta_1(t)\frac{\beta_2(t)}{\alpha_2(t)}v^{(1)}(t))+\mathbb{H}(v^{(2)}(t)-v^{(1)}(t))\mathbb{H}(u(t)-\beta_1(t)\frac{\beta_2(t)}{\alpha_2(t)}v^{(2)}(t))\\
&=\mathbb{H}(v^{(1)}(t)-v^{(2)}(t))\mathbb{H}(u(t)-\beta_1(t)v^{(1)}(t))+\mathbb{H}(v^{(2)}(t)-v^{(1)}(t))\mathbb{H}(u(t)-\beta_1(t)v^{(2)}(t))
\end{aligned}$$

□

Corollary H.10. *SignGD (Eq. 16) with $\mathcal{L}(y;x)=\frac{1}{2}\|y-x^2\|^2$ is equivalent to the signGD-based neuronal dynamics (Def. 5.2) satisfying equation 15, $\eta(t)=\beta_2(t)=\alpha_2(t)$ and*

$$s(t)=\mathbb{H}\Big(u(t)-\beta_1(t)v(t)^2\Big)$$

Proof. Since $\mathcal{L}(y;x)=\frac{1}{2}\|y-x^2\|^2$, $\nabla_y\mathcal{L}(y;x)=y-x^2$.

$$\begin{aligned}
s(t)&=\mathbb{H}\Big(\nabla_y\mathcal{L}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t);\frac{\eta(t)}{\alpha_2(t)}v(t))\Big)\\
&=\mathbb{H}\Big(\frac{\eta(t-1)}{\beta_2(t-1)}u(t)-(\frac{\eta(t)}{\alpha_2(t)}v(t))^2\Big)\\
&=\mathbb{H}\Big(u(t)-\frac{\eta(t)^2\beta_2(t-1)}{\eta(t-1)\alpha_2(t)^2}v(t)^2\Big)\\
&=\mathbb{H}\Big(u(t)-\beta_1(t)\frac{\eta(t)\beta_2(t)}{\alpha_2(t)^2}v(t)^2\Big)\\
&=\mathbb{H}\Big(u(t)-\beta_1(t)v(t)^2\Big)
\end{aligned}$$

□

Corollary H.11. *SignGD (Eq. 16) with $\mathcal{L}(y;x_1,x_2)=\|y-\frac{x_1}{\sqrt{x_2}}\|^2$ is equivalent to the signGD-based neuronal dynamics (Def. 5.2) satisfying equation 15, and $\eta(t)=\alpha_2(t)=\beta_2(t)$ and*

$$\begin{aligned}
s(t)=&\,\mathbb{H}\big(u(t)\big)\mathbb{H}\big(v_1(t)\big)\mathbb{H}\big(v_2(t)u(t)^2-v_1(t)^2\big)\\
&+\mathbb{H}\big(-u(t)\big)\mathbb{H}\big(-v_1(t)\big)\mathbb{H}\big(v_1(t)^2-v_2(t)u(t)^2\big)\\
&+\mathbb{H}(u(t))\mathbb{H}(-v_1(t))
\end{aligned}$$

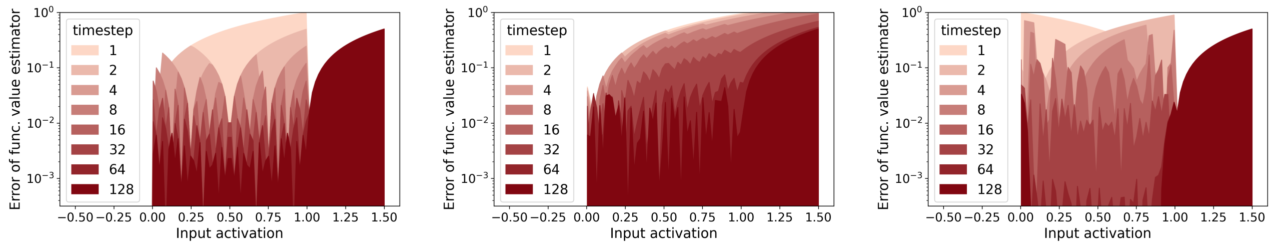
Proof. Since $\mathcal{L}(y;x_1,x_2) = \frac{1}{2}\|y - \frac{x_1}{\sqrt{x_2}}\|^2$, $\nabla_y \mathcal{L}(y;x) = y - \frac{x_1}{\sqrt{x_2}}$.

$$\mathbb{H}(y - \frac{x_1}{\sqrt{x_2}}) = \mathbb{H}(y)\mathbb{H}(x_1)\mathbb{H}(x_2y^2 - x_1^2) + \mathbb{H}(-y)\mathbb{H}(-x_1)\mathbb{H}(x_1^2 - x_2y^2) + \mathbb{H}(y)\mathbb{H}(-x_1)$$

Applying the equation to the spike mechanism $s(t)$,

$$\begin{aligned}
s(t) =&\mathbb{H}\Bigg(\nabla_y \mathcal{L}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t); \frac{\eta(t)}{\alpha_2(t)}v(t))\Bigg)\\
=&\mathbb{H}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t))\mathbb{H}(\frac{\eta(t)}{\alpha_2(t)}v^{(1)}(t))\mathbb{H}(\frac{\eta(t)}{\alpha_2(t)}v^{(2)}(t)(\frac{\eta(t-1)}{\beta_2(t-1)}u(t))^2 - (\frac{\eta(t)}{\alpha_2(t)}v^{(1)}(t))^2)\\
&+\mathbb{H}(-\frac{\eta(t-1)}{\beta_2(t-1)}u(t))\mathbb{H}(-\frac{\eta(t)}{\alpha_2(t)}v^{(1)}(t))\mathbb{H}((\frac{\eta(t)}{\alpha_2(t)}v^{(1)}(t))^2 - \frac{\eta(t)}{\alpha_2(t)}v^{(2)}(t)(\frac{\eta(t-1)}{\beta_2(t-1)}u(t))^2)\\
&+\mathbb{H}(\frac{\eta(t-1)}{\beta_2(t-1)}u(t))\mathbb{H}(-\frac{\eta(t)}{\alpha_2(t)}v^{(1)}(t))\\
=&\mathbb{H}(u(t))\mathbb{H}(v^{(1)}(t))\mathbb{H}(\frac{\alpha_2(t)}{\eta(t)}\frac{\eta(t-1)^2}{\beta_2(t-1)^2}v^{(2)}(t)u(t)^2 - v^{(1)}(t)^2)\\
&+\mathbb{H}(-u(t))\mathbb{H}(-v^{(1)}(t))\mathbb{H}(v^{(1)}(t)^2 - \frac{\alpha_2(t)}{\eta(t)}\frac{\eta(t-1)^2}{\beta_2(t-1)^2}v^{(2)}(t)u(t)^2)\\
&+\mathbb{H}(u(t))\mathbb{H}(-v^{(1)}(t))\\
=&\mathbb{H}(u(t))\mathbb{H}(v^{(1)}(t))\mathbb{H}(v^{(2)}(t)u(t)^2 - v^{(1)}(t)^2)\\
&+\mathbb{H}(-u(t))\mathbb{H}(-v^{(1)}(t))\mathbb{H}(v^{(1)}(t)^2 - v^{(2)}(t)u(t)^2)\\
&+\mathbb{H}(u(t))\mathbb{H}(-v^{(1)}(t))
\end{aligned}$$

□



(a) Inverse LR, $\eta(t) = \frac{1}{t+1} \approx$ IF + Rate of $\theta_{th} = R = 1$ (b) Constant LR, $\eta(t) = 0.05 \approx$ LIF + EMA of $\theta_{th} = 1, \tau = 20, u_{rest} = 0$ (c) Exp. LR, $\eta(t) = (0.9)^t$.

Figure 10. Toy experiment on subgradient-based neuronal dynamics with generalized learning rate schedule (Definition I.3). Time-evolution of error (Y-axis, log-scale) between ReLU function and spike-decoded neuron output, over a float input activation (X-axis)

I. Generalizing learning rate schedule over subgradient-based neuronal dynamics

To verify the effect of signGD dynamics on acceleration, we generalize the learning rate schedule of subgradient method-based neuronal dynamics, which underlies the simple integrate-and-fire models. In Figure 10, we also empirically verify that the subgradient-based neuronal dynamics are well-defined by approximating the clipped ReLU function with the subgradient-based neuron. We design neuronal dynamics with three different learning rate schedules: inverse (Figure 10(a)), exponential (Figure 10(c)), and constant (Figure 10(b)) schedule. We formulate them as follows.

Definition I.1. (Schedule coding) Let a spike train $s(t) \in \{0,1\}$, a step size schedule $\eta(t) \in \mathbb{R}^+$ for $t \in \mathbb{N}$ and $y(0) = 0$. Schedule coding with $\eta(t)$ decodes an activation $y(t)$ from $s(t)$ as

$$y(t) = (1 - \eta(t))y(t-1) + \eta(t)s(t) \tag{55}$$

Definition I.2. (Subgradient-based neuronal dynamics) Positive coefficients $\alpha_i(t) \in \mathbb{R}^+$, $\beta_i(t) \in \mathbb{R}^+$, and step size schedule $\eta(t) \in \mathbb{R}^+$ for $i = 1, 2$ and $t \in \mathbb{N}$. The subgradient-based neuronal dynamics over membrane potential $u(t) \in \mathbb{R}$ is

$$u_{pre}(t+1) = \alpha(t)u(t) + \gamma(t+1)I(t+1) \tag{56}$$

$$s(t+1) = \mathbb{H}\big(u_{pre}(t+1) - u_{pre}(0)\prod_{j=0}^{t}\alpha(j)\big) \tag{57}$$

$$u(t+1) = u_{pre}(t+1) - \beta(t+1)s(t+1) \tag{58}$$

Theorem I.3. *Let $\mathcal{L}(y;x) = ReLU\big(x-y\big) + \frac{1}{2}y^2$ and its sub-gradient $\tilde{g}(y;x)$ over y. If input $\tilde{x}(t) = (1-\eta(t))x(t-1) + \eta(t)I(t)$ and output activation $\tilde{f}(t) = (1-\eta(t))\tilde{f}(t-1) + \eta(t)s(t)$ are schedule coded with $\eta(t)$ and η, α, β, γ satisfies for $i, t \in \mathbb{N}$*

$$\frac{\beta(t)}{\eta(t)}(1-\eta(t)) = \frac{\beta(t-1)}{\eta(t-1)}\alpha(t-1) \tag{59}$$

$$\frac{\eta(i)}{\eta(t)}\prod_{j=i+1}^{t}(1-\eta(j)) = \frac{\gamma(i)}{\beta(t)}\prod_{j=i}^{t-1}\alpha(j) \tag{60}$$

Then the dynamical system of $\tilde{f}(t)$ and $\tilde{x}(t)$ is equivalent to a subgradient method

$$\tilde{f}(t) = \tilde{f}(t-1) - \eta(t)\cdot\tilde{g}\big(\tilde{f}(t-1);\tilde{x}(t)\big) \tag{61}$$

Proof. Subgradient $\tilde{g}(t) = \tilde{f}(t-1) - \mathbb{H}\big(\tilde{x}(t+1) - \tilde{f}(t)\big)$. We start from the neuronal dynamics (Definition I.2) to derive the subgradient method. By definition of $\tilde{f}(t)$,

$$s(t) = \frac{1}{\eta(t)}\tilde{f}(t) - \left(\frac{1}{\eta(t)} - 1\right)\tilde{f}(t-1)$$

The equation (58) applies to equation (56) to formulate the recurrence relation of $\tilde{f}(t)$ and $u(t)$.

$$u_{pre}(t+1) = \alpha(t)u_{pre}(t) - \alpha(t)\beta(t)s(t) + \gamma(t+1)I(t+1) \tag{62}$$

$$\frac{\alpha(t)\beta(t)}{\eta(t)}\left(\tilde{f}(t) - \big(1-\eta(t)\big)\tilde{f}(t-1)\right) = \alpha(t)u_{pre}(t) - u_{pre}(t+1) + \gamma(t+1)I(t+1) \tag{63}$$

By equation (60), $\frac{\alpha(t)\beta(t)}{\eta(t)}(1-\eta(t)) = \alpha(t)\frac{\alpha(t-1)\beta(t-1)}{\eta(t-1)}$. With $\tilde{f}(0) = 0$, solving the recurrence relation yields

$$\frac{\alpha(t)\beta(t)}{\eta(t)}\tilde{f}(t) = -u_{pre}(t+1) + u_{pre}(0)\prod_{j=0}^{t}\alpha(j) + \sum_{i=1}^{t+1}\gamma(i)I(i)\prod_{j=i}^{t}\alpha(j)$$

$$\tilde{f}(t) = -\frac{\eta(t)}{\alpha(t)\beta(t)}u_{pre}(t+1) + \frac{\eta(t)}{\alpha(t)\beta(t)}u_{pre}(0)\prod_{j=0}^{t}\alpha(j) + \frac{\eta(t)}{\alpha(t)\beta(t)}\sum_{i=1}^{t+1}\gamma(i)I(i)\prod_{j=i}^{t}\alpha(j)$$

By equation (59), $\eta(i)\prod_{j=i+1}^{t}(1-\eta(j)) = \frac{\eta(t)}{\alpha(t)\beta(t)}\gamma(i)\prod_{j=i}^{t}\alpha(j)$. Hence,

$$\begin{aligned}\tilde{f}(t) &= -\frac{\eta(t)}{\alpha(t)\beta(t)}u_{pre}(t+1) + \frac{\eta(t)}{\alpha(t)\beta(t)}u_{pre}(0)\prod_{j=0}^{t}\alpha(j) + \sum_{i=1}^{t+1}\eta(i)\prod_{j=i+1}^{t}(1-\eta(j))I(i) \\ &= -\frac{\eta(t)}{\alpha(t)\beta(t)}u_{pre}(t+1) + \frac{\eta(t)}{\alpha(t)\beta(t)}u_{pre}(0)\prod_{j=0}^{t}\alpha(j) + \tilde{x}(t+1)\end{aligned}$$

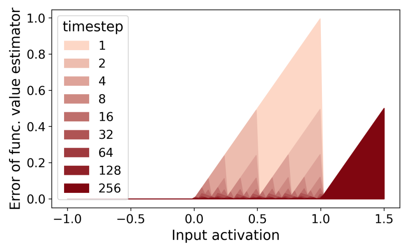
$$u_{pre}(t+1) - u_{pre}(0)\prod_{j=0}^{t}\alpha(j) = \frac{\alpha(t)\beta(t)}{\eta(t)}\big(\tilde{x}(t+1) - \tilde{f}(t)\big)$$

Substituting u_{pre} with $\tilde{f}$ and $\tilde{x}$ in the definition of s,

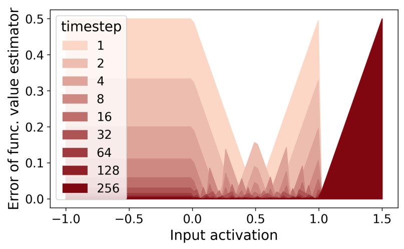
$$s(t+1) = \mathbb{H}\big(u_{pre}(t+1) - u_{pre}(0)\prod_{j=0}^{t}\alpha(j)\big) = \mathbb{H}\left(\frac{\alpha(t)\beta(t)}{\eta(t)}\big(\tilde{x}(t+1) - \tilde{f}(t)\big)\right) = \mathbb{H}\big(\tilde{x}(t+1) - \tilde{f}(t)\big)$$

$$\begin{aligned}\therefore \tilde{f}(t) &= (1-\eta(t))\tilde{f}(t-1) + \eta(t)\mathbb{H}\big(\tilde{x}(t) - \tilde{f}(t-1)\big)\\ &= \tilde{f}(t-1) - \eta(t)\cdot\tilde{g}(\tilde{f}(t-1);\tilde{x}(t))\end{aligned}$$

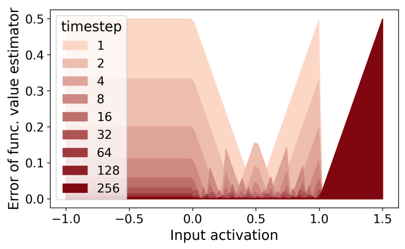
□



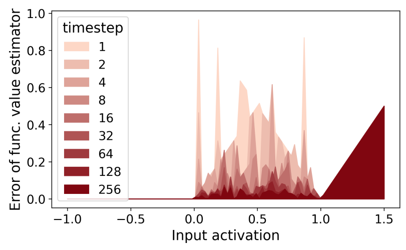
(a) IF neuron with $\theta_{th} = 1, R = 1$, float encoding.



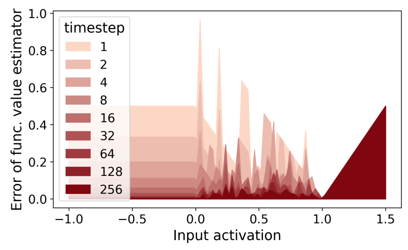
(b) Fig. 11(a) transformed to $\tilde{f}(t)$



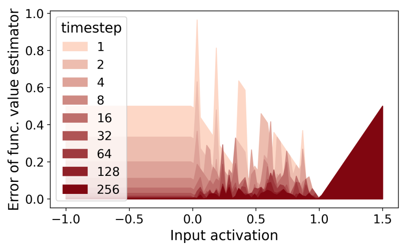
(c) Subgradient method on $\tilde{f}(t)$. Float encoding.



(d) IF neuron with $\theta_{th} = 1, R = 1$, poisson encoding.

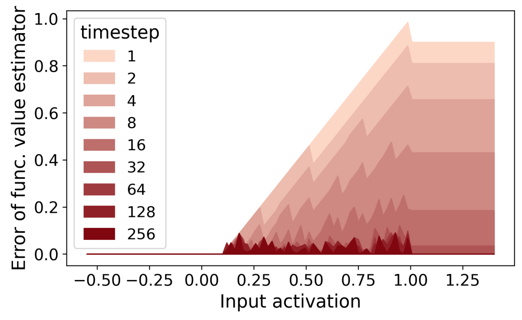


(e) Fig. 11(d) transformed to $\tilde{f}(t)$

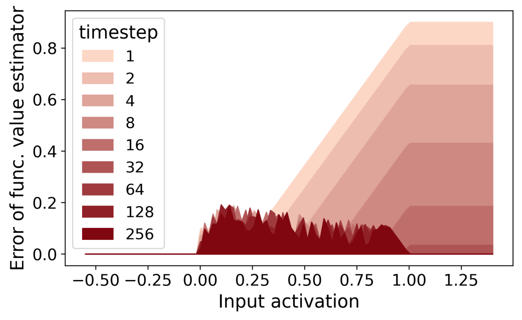


(f) Subgradient method on $\tilde{f}(t)$. Poisson encoding.

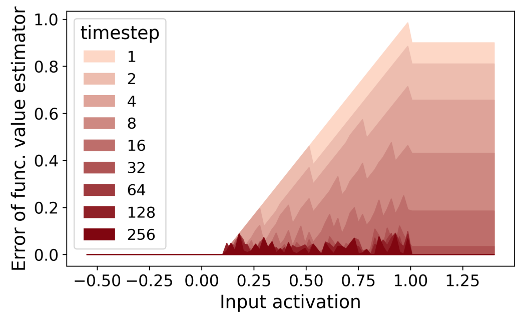
Figure 11. Equivalence of IF neuron (Eq. 2-4) + rate-coded input (Eq. 6) with the subgradient method defined as Theorem 4.1. We plot time-evolution of error between ReLU function and the rate-decoded output of IF neuron (Fig. 11(a), 11(d)) or the approximation of subgradient method (Fig. 11(c), 11(f)). Given input $x \in \mathbb{R}$, Float encoding is $I(t) = x\ \forall t$ (Li et al., 2021a), and Poisson encoding is a stochastic spike train representation of x that samples $I(t) \sim B(1, \text{ReLU1}(x)), \forall t$ (Sengupta et al., 2018)



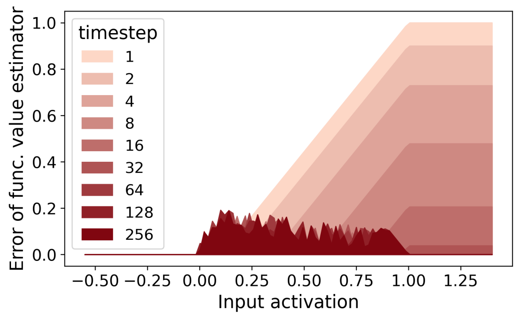
(a) LIF neuron, $\tau = 10, R = 1, \theta_{th} = 1$, Float encoding



(b) LIF neuron, $\tau = 10, R = 1, \theta_{th} = 1$, spike encoding.

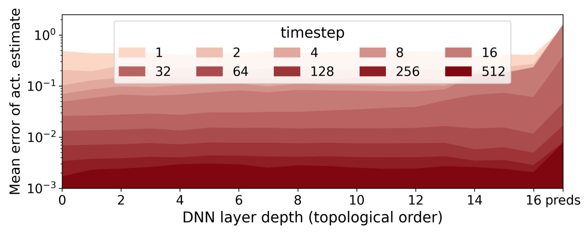


(c) Subgradient method, $\tau = 10, R = 1$, Float encoding.

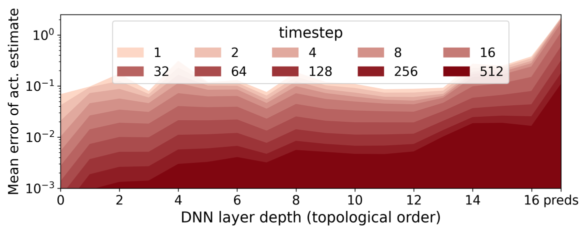


(d) Subgradient method $\tau = 10, R = 1$, spike encoding.

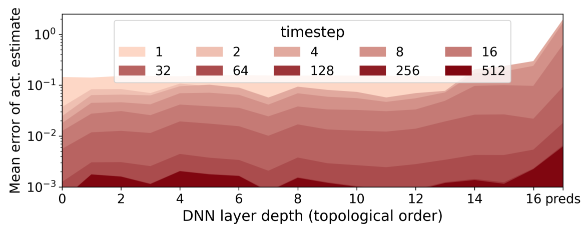
Figure 12. Equivalence of LIF neuron (Eq. 1-3) + EMA-coded input (Eq. 7) with the subgradient method defined as Theorem 4.2. We plot time-evolution of error between ReLU1$(\frac{Rx-\theta_{th}}{\tau-1})$ and the EMA-decoded output of neuron (Fig. 12(a), 12(b)) or the approximation of subgradient method (Fig. 12(c), 12(d)). Given input $x \in \mathbb{R}$, Float encoding is $I(t) = x \;\forall t$, and spike encoding is a deterministic spike train representation $I(t)$ of x defined as $I(t) = \mathbb{H}(x - \frac{\tau-1}{\tau}\tilde{x}(t-1)),\; \tilde{x}(t) = \frac{\tau-1}{\tau}\tilde{x}(t-1) + \frac{1}{\tau}I(t)$.



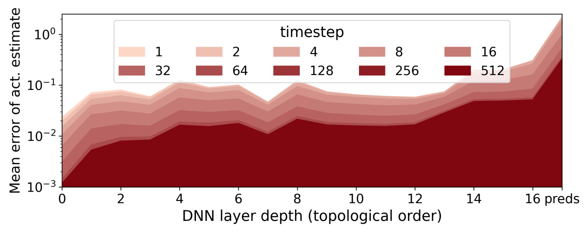
(a) SignGD-based neuron, $\eta(t) = \frac{1}{t+1}$



(b) Subgradient-based neuron, $\eta(t) = \frac{1}{t+1} \approx$ IF + Rate



(c) SignGD-based neuron, $\eta(t) = 0.15 \cdot (0.95)^t$.



(d) Subgradient-based neuron, $\eta(t) = 0.15 \cdot (0.95)^t$.

Figure 13. Layer-wise time-evolution of error between the true ANN activation and the spike-decoded SNN activations. We qualitatively compare two neuronal dynamics side-by-side with the same learning rate schedules; inverse schedule (Fig. 13(a), 13(b)) and exponential schedule (Fig. 13(c), 13(d)). Measured with ResNet-18 on a single instance of CIFAR-10 dataset.

Table 4. Comparing ANN-to-SNN conversion performance on CIFAR-10 (Krizhevsky et al., 2009) dataset. *No Spike-aware Activation Func* means ReLU functions in ANN architecture are not replaced with spike-aware functions before training, e.g., QCFS (Bu et al., 2022), SlipReLU (Jiang et al., 2023). For our signGD-based neuron, we used exponential schedule with initial LR 0.135 and decay factor 0.95. Results of RTS (Deng & Gu, 2021) are from (Li et al., 2021a).

Methods	No Spike-aware Activation Func.	ANN Acc.	Simulation time-steps T = 16	T = 32	T = 64	T = 128	T = 256
			ResNet-18 (He et al., 2015) CIFAR-10				
TSC (Han & Roy, 2020)	✔	91.47	-	-	69.38	88.57	90.10
RMP (Han et al., 2020)	✔	91.47	-	-	-	87.60	89.37
RTS (Deng & Gu, 2021)	✖	95.46	-	84.06	92.48	94.68	95.30
SNNC-AP (Li et al., 2021a)	✔	95.46	-	94.78	95.30	95.42	95.41
SNM (Wang et al., 2022)	✔	95.39	-	94.03	94.03	95.19	-
QCFS (Bu et al., 2022)	✖	96.04	95.92	96.08	96.06	-	-
SlipReLU (Jiang et al., 2023)	✖	96.15	**96.10**	96.12	96.22	-	-
SRP (Hao et al., 2023)	✖	95.64	95.55	95.55	95.58	-	-
Subgradient-based neuron (Thm. I.3)	✔	96.82	53.40	88.34	94.20	94.84	94.87
Ours (signGD-based neuron, Def. 5.2)	✔	96.82	80.74	**96.29**	**96.78**	**96.79**	**96.79**
			VGG-16 (Simonyan & Zisserman, 2014) CIFAR-10				
TSC (Han & Roy, 2020)	✔	93.63	-	-	92.79	93.27	93.45
RMP (Han et al., 2020)	✔	93.63	-	60.30	90.35	92.41	93.04
RTS (Deng & Gu, 2021)	✖	95.72	-	76.24	90.64	94.11	95.33
SNNC-AP (Li et al., 2021a)	✔	95.72	-	93.71	95.14	95.65	95.79
SNM (Wang et al., 2022)	✔	94.09	-	93.43	94.07	94.07	-
QCFS (Bu et al., 2022)	✖	95.52	95.40	**95.54**	95.55	-	-
SlipReLU (Jiang et al., 2023)	✖	95.60	95.20	95.66	95.65	-	-
SRP (Hao et al., 2023)	✖	95.52	**95.44**	95.42	95.40	-	-
Subgradient-based neuron (Thm. I.3)	✔	95.96	50.98	82.84	90.66	91.75	91.80
Ours (signGD-based neuron, Def. 5.2)	✔	95.96	81.06	95.53	**95.96**	**95.97**	**95.97**

Algorithm 4 ANN-to-SNN Conversion with signGD-based Neuron (Definition 5.2)

```
1: **Input:** Target ANN model $F$, learning rate schedule $\eta : \mathbb{N} \to \mathbb{R}$, key-value mapping $\mathcal{D}$ of $\{$Nonlinearity : signGD-based neuronal dynamics$\}$, number of training batches for normalization $N \in \mathbb{N}$
2: **Output:** Converted SNN model $S$
3: Extract the computational graph $G$ of tensor operators from ANN model $F$.
4: Decompose max pooling and layer normalization operators of $G$ to generate a new graph $G'$ (See Section 5.2.).
5: **if** ReLU operator $\in$ G **then**
6:     **for** any ReLU operator $f \in G'$ **do**
7:         Initialize the maximum ReLU output activation $M_f \leftarrow -\infty$.
8:     **end for**
9:     Register a callback for every ReLU operator $f$ to record its output activations $X_f$.
10:    **for** $t = 1$ **to** $N$ **do**
11:        Sample a batch $x$ from training dataset.
12:        Feed-forward $x$ through ANN model $F$ and record $X_f$.
13:        **for** any ReLU operator $f \in G'$ **do**
14:            $M_f \leftarrow \max(M_f, X_f)$
15:        **end for**
16:    **end for**
17:    **for** any ReLU operator $f \in G'$ **do**
18:        Replace the $f(\cdot)$ operator with a scaled ReLU operator $M_f \cdot f(\frac{\cdot}{M_f})$ to generate a new graph $G''$.
19:    **end for**
20: **else**
        $G'' \leftarrow G'$
21: **end if**
22: **for** nonlinearity $h \in$ domain of $\mathcal{D}$ **do**
23:     **for** any operator $f \in G''$ **do**
24:         **if** $f$ is $h$ **then**
25:             Initialize signGD-based neuron $n \leftarrow \mathcal{D}(f; \eta)$.
26:             Replace the operator $f$ of $G''$ with the neuron $n$ to generate a computational graph $G^S$ of SNN.
27:         **end if**
28:     **end for**
29: **end for**
30: Create a SNN model $S$ from the computational graph $G^S$.
31: Reset membrane potentials of SNN model $S$.
32: Register a callback for every neuron $h$ of SNN $S$ to record its influx current $I_h$.
33: Stimulate every neuron $h$ of SNN $S$ to emit a spike 1 for a single time-step.
34: Record the influx current $I_h^+ = I_h$ for every neuron $h$ of SNN $S$.
35: Depress every neuron $h$ of SNN $S$ to not emit a spike for a single time-step.
36: Record the idle current $I_h^- = I_h$ for every neuron $h$ of SNN $S$.
37: Store the sum of weights $W \leftarrow I_h^+ - I_h^-$ and bias $b \leftarrow I_h^-$ for every neuron.
```

Table 5. Comparing ANN-to-SNN conversion performance on CIFAR-100 (Krizhevsky et al., 2009) dataset. *No Spike-aware Activation Func* means ReLU functions in ANN architecture are not replaced with spike-aware functions before training, e.g., QCFS (Bu et al., 2022), SlipReLU (Jiang et al., 2023). For our signGD-based neuron, we used exponential schedule with initial LR 0.135 and decay factor 0.95. Results of RTS (Deng & Gu, 2021) are from (Li et al., 2021a).

Methods	No Spike-aware Activation Func.	ANN Acc.	Simulation time-steps T = 16	T = 32	T = 64	T = 128	T = 256
ResNet-20 (He et al., 2015) CIFAR-100							
TSC (Han & Roy, 2020)	✓	68.72	-	-	-	58.42	65.27
RMP (Han et al., 2020)	✓	68.72	-	27.64	46.91	57.69	64.06
RTS (Deng & Gu, 2021)	✗	77.16	-	51.27	70.12	75.81	77.22
SNNC-AP (Li et al., 2021a)	✓	77.16	-	76.32	77.29	77.73	77.63
SNM (Wang et al., 2022)	✓	78.26	-	74.48	77.59	77.97	-
QCFS (Bu et al., 2022)	✗	78.80	**79.48**	**79.62**	79.54	-	79.61
SlipReLU (Jiang et al., 2023)	✗	77.08	77.29	78.04	77.97	-	77.99
SRP (Hao et al., 2023)	✗	69.94	64.71	65.50	65.82	-	-
Subgradient-based neuron (Thm. I.3)	✓	81.19	22.39	57.79	71.22	73.08	73.14
Ours (signGD-based neuron, Def. 5.2)	✓	81.19	36.78	79.13	**81.10**	**81.22**	**81.23**
VGG-16 (Simonyan & Zisserman, 2014) CIFAR-100							
TSC (Han & Roy, 2020)	✓	71.22	-	-	-	69.86	70.65
RMP (Han et al., 2020)	✓	71.22	-	-	-	63.76	68.34
RTS (Deng & Gu, 2021)	✗	77.89	-	7.64	21.84	55.04	73.54
SNNC-AP (Li et al., 2021a)	✓	77.89	-	73.55	76.64	77.40	77.68
SNM (Wang et al., 2022)	✓	74.13	-	71.8	73.69	73.95	-
QCFS (Bu et al., 2022)	✗	76.28	76.24	**77.01**	77.10	-	77.08
SlipReLU (Jiang et al., 2023)	✗	70.03	69.35	70.65	71.23	-	-
SRP (Hao et al., 2023)	✗	76.28	**76.42**	76.45	76.37	-	-
Subgradient-based neuron (Thm. I.3)	✓	78.28	13.12	42.05	60.61	64.03	64.15
Ours (signGD-based neuron, Def. 5.2)	✓	78.28	39.42	76.33	**78.17**	**78.33**	**78.23**