Training Feedback Spiking Neural Networks by Implicit Differentiation on the Equilibrium State

Anonymous Author(s)
Affiliation
Address
email

Abstract

Spiking neural networks (SNNs) are brain-inspired models that enable energy-efficient implementation on neuromorphic hardware. However, the supervised training of SNNs remains a hard problem due to the discontinuity of the spiking neuron model. Most existing methods imitate the backpropagation framework and feedforward architectures for artificial neural networks, and use surrogate derivatives or compute gradients with respect to the spiking time to deal with the problem. These approaches either accumulate approximation errors or only propagate information limitedly through existing spikes, and usually require information propagation along time steps with large memory costs and biological implausibility. In this work, we consider feedback spiking neural networks, which are more brain-like, and propose a novel training method that does not rely on the exact reverse of the forward computation. First, we show that the average firing rates of SNNs with feedback connections would gradually evolve to an equilibrium state along time, which follows a fixed-point equation. Then by viewing the forward computation of feedback SNNs as a black-box solver for this equation, and leveraging the implicit differentiation on the equation, we can compute the gradient for parameters without considering the exact forward procedure. In this way, the forward and backward procedures are decoupled and therefore the problem of non-differentiable spiking functions is avoided. We also briefly discuss the biological plausibility of implicit differentiation, which only requires computing another equilibrium. Extensive experiments on MNIST, Fashion-MNIST, N-MNIST, CIFAR-10, and CIFAR-100 demonstrate the superior performance of our method for feedback models with fewer neurons and parameters in a small number of time steps.

1 Introduction

Spiking neural networks (SNNs) have gained increasing attention recently due to their inherent energy-efficient computation [21, 36, 39, 31, 8]. Inspired by the neurons in the human brain, biologically plausible SNNs transmit spikes between neurons, enabling event-based computation which can be carried out on neuromorphic chips with less energy consumption [1, 7, 28, 31]. Meanwhile, SNNs are computationally more powerful than artificial neural networks (ANNs) theoretically and are therefore regarded as the third generation of neural network models [23].

Despite the advantages, directly supervised training of SNNs remains a hard problem, which hampers the real applications of SNNs compared with popular ANNs. The main obstacle lies in the complex spiking neuron model. While backpropagation [33] works well for ANNs, it suffers from the discontinuity of spike generation which is non-differentiable in SNN training. Most recent successful SNN training methods still imitate the backpropagation through time (BPTT) [37] framework by error propagation through the computational graph unfolded along time steps, and they deal with the spiking

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

function by applying surrogate derivatives to approximate the gradients [39, 5, 14, 36, 40, 25, 16, 45], or by computing the gradients with respect to the spiking time only on the spiking neurons [6, 44, 16]. However, these methods either accumulate approximation error along time steps, or suffer from the "dead neuron" problem [36], i.e. learning would not occur when no neuron spikes. At the same time, BPTT requires memorizing intermediate variables at all time steps and backpropagating along them, which is memory-costing and biologically implausible. So it is necessary to consider if there could be training methods other than backpropagation along computational graphs that fit SNNs better.

On the other hand, most recent SNN models simply imitate the architectures of popular feedforward ANNs [39, 36, 40, 44, 45], which ignores the ubiquitous feedback connections in the human brain and significantly losses the variety of neuron connections. Feedback (recurrent) circuits are critical to human's vision system for object recognition [15]. Meanwhile, [18] demonstrates that shallow ANNs with recurrence achieve higher functional fidelity of human brains and similarly high performance on large-scale vision recognition tasks, compared with deep ANNs. So incorporating feedback computation enables neural networks to be shallower and more efficient, as well as more brain-like. As for SNNs, feedback connections were popular in the early models like Liquid State Machine [24], which leverages a recurrent reservoir layer with weights fixed or trained by unsupervised methods. And compared with the uneconomical cost for ANNs to incorporate feedback connections by unfolding along time, SNNs naturally compute with multiple time steps, which inherently supports feedback computation. Most recent SNN models follow the feedforward architecture because they were once lacking effective supervised learning methods and thus they borrow everything from successful ANNs. We focus on another direction, i.e. feedback SNN, which is a natural choice for visual tasks as well.

In this work, we consider the training of feedback spiking neural networks (FSNN), and propose a novel method based on the Implicit Differentiation on the Equilibrium state (IDE). Inspired by recent advances in implicit models [3, 4], which treat weight-tied ANNs as solving a fixed-point equilibrium equation and propose alternative implicit models defined by the equation, we derive that when the average inputs converge to an equilibrium, the average firing rates of FSNNs would gradually evolve to an equilibrium state along time, which follows a fixed-point equation as well. Then we view the forward computation of FSNN as a black-box solver for the fixed-point equation, and borrow the idea of implicit differentiation from implicit models [3, 4] to calculate the gradients, which only relies on the equation rather than the exact forward procedure. In this way, gradient calculation is agnostic to the spiking function in SNN, thus avoiding the common difficulties in SNN training. While implicit differentiation may seem too abstract to be computed in the brain, we briefly discuss the biological plausibility and show that it only requires computing another equilibrium along the inverse connections of neurons. Besides, we incorporate the multi-layer structure into the feedback model for better representation ability. Our contributions include:

1. We are the first to theoretically derive the equilibrium states with a fixed-point equation for the average firing rates of FSNNs with the (leaky) integrate and fire model under both continuous and discrete views. According to this, the forward computation of FSNNs can be interpreted as solving a fixed-point equation.
2. We propose a novel training method for FSNNs based on the implicit differentiation on the equilibrium state, which is decoupled from the forward computational graph and avoids SNN training problems, e.g. non-differentiability and large memory costs. We also discuss the biological plausibility and demonstrate the connection to the Hebbian learning rule.
3. We conduct extensive experiments on MNIST, Fashion-MNIST, N-MNIST, CIFAR-10, and CIFAR-100, which demonstrate the superior results of our methods with fewer neurons and parameters in a small number of time steps for both static images and neuromorphic inputs. Especially, our directly trained model can outperform the state-of-the-art SNN performance on the complex CIFAR-100 dataset with only 30 time steps.

2 Related Works

2.1 Training Methods for Spiking Neural Networks

Early works apply biologically inspired method, spike-time dependent plasticity (STDP) [9], to formulate a bottom-up unsupervised learning rule, or choose reward-modulated STDP [22] with limited supervision. Since the rise of successful ANNs, error backpropagation and gradient descent have inspired many methods. One direction is to convert a trained ANN to SNN [13, 32, 35, 30, 8, 42]. However, they suffer from conversion errors and extremely large simulation time steps. The other

methods are to directly calculate the gradient and train SNNs. These methods either compute the gradient with respect to spiking times [6, 44], or leverage a surrogate derivative for discontinuous spiking functions [21, 39, 5, 14, 36, 40, 25, 45], or combine them [16]. However, they suffer from the "dead neuron" problem [36] or accumulated approximation error, and typically require backpropagation along the computational graph to be unfolded by many time steps, which is memory-consuming and biologically implausible. As for SNN with feedback connection, [43] proposes the ST-RSBP method, which backpropagates errors at the spike-train level. They simply approximate the activation function of a neuron as a linear operation, and require long time steps for satisfactory results. In this work, we propose a new training method that does not rely on the exact reverse of the forward computation, which avoids problems of non-differentiability and large memory costs, and only requires short time steps for superior performance. There are also works trying methods other than BP along computational graphs to train SNNs, such as equilibrium propagation [27]. However, [27] defines a complex computation form rather than common SNN models, and can hardly achieve satisfactory results on the simple MNIST dataset. Instead, our work is based on SNN models applicable on neuromorphic hardware and demonstrates promising results on more complex datasets.

2.2 Equilibrium of Neural Networks

The study on the equilibrium of neural networks originates from energy-based models, e.g. Hopfield Network [11, 12]. They view the dynamics or iterative procedures of feedback (recurrent) neural networks as minimizing an energy function, which will converge to a minimum of the energy. Based on the energy, several training methods are proposed, including recurrent backpropagation [2, 29] and more recent equilibrium propagation (EP) [34]. They heavily rely on the energy function and can hardly achieve competitive results as deep neural networks do. Deep equilibrium models [3, 4], on the other hand, are recently proposed models which treat weight-tied deep ANNs as solving a fixed-point equilibrium point, and propose alternative implicit models defined by the fixed-point equations rather than energy functions. They express the entire deep network as an equilibrium computation and solve and train it by root-finding methods and implicit differentiation, respectively, which achieves superior results. Most of these works are based on ANNs, except that [27] generalizes the EP method to neurons with binary communications. They define a complex neuron computation form and follow the methodology of energy-based EP method to approximate the gradients. Differently, we are the first to derive the equilibrium state with a fixed-point equation for average firing rates of FSNNs with common SNN models, and propose to train SNNs by exact gradients through implicit differentiation.

3 Preliminaries

3.1 Spiking Neural Network Models

Spiking neurons, different from real-valued ANNs, communicate with each other by binary spike trains along time. Each neuron maintains a membrane potential, which integrates input spike trains, and the neuron would generate an output spike once the membrane potential exceeds a threshold. The commonly used integrate and fire (IF) model and leaky integrate and fire (LIF) model describe the dynamics of the membrane potential as:

$$\begin{aligned} \textbf{IF:} \quad & \frac{du}{dt} = R \cdot I(t), && u < V_{th}, \\ \textbf{LIF:} \quad & \tau_m \frac{du}{dt} = -(u - u_{rest}) + R \cdot I(t), && u < V_{th}, \end{aligned} \tag{1}$$

where u is the membrane potential, I is the input current, V_{th} is the spiking threshold, and R and τ_m are resistance and time constant, respectively. Once u reaches V_{th} at time t^f, a spike is generated and u is reset to the resting potential $u = u_{rest}$, which is usually taken as 0. The spike train is expressed by the Dirac delta function: $s(t) = \sum_{t^f} \delta(t - t^f)$. We consider the simple current model $I_i(t) = \sum_j w_{ij} s_j(t) + b$, where w_{ij} is the weight from neuron j to neuron i, which is the same as ANN. After discretization, the general computation form for the IF and LIF model is described as:

$$\begin{cases} u_i[t+1] = \lambda u_i[t] + \sum_j w_{ij} s_j[t] + b, \\ s_i[t+1] = H(u_i[t+1] - V_{th}), \\ u_i[t+1] = u_i[t+1] - V_{th} s_i[t+1], \end{cases} \tag{2}$$

where $H(x)$ is the Heaviside step function, i.e. the non-differentiable spiking function, $s_i[t]$ is the binary spike train of neuron i, and λ is 1 for the IF model while $\lambda < 1$ is a leaky term related to the

constant τ_m and discretization time interval for the LIF model. The constant R, τ_m, and time step size are absorbed into the weights w_{ij} and bias b. We use subtraction as the reset operation.

3.2 Implicit Differentiation on the Fixed-Point Equation

We consider a fixed-point equation $a = f_\theta(a)$ parameterized by θ. Let $\mathcal{L}(a^*)$ denote the objective function with respect to the equilibrium state a^*, and let $g_\theta(a) = f_\theta(a) - a$. The implicit differentiation on the equation satisfies $\left(I - \frac{\partial f_\theta(a^*)}{\partial a^*}\right)\frac{\mathrm{d}a^*}{\mathrm{d}\theta} = \frac{\partial f_\theta(a^*)}{\partial \theta}$ [3]. Therefore, the differentiation of $\mathcal{L}(a^*)$ with respect to parameters can be calculated based on implicit differentiation:

$$\frac{\partial \mathcal{L}(a^*)}{\partial \theta} = -\frac{\partial \mathcal{L}(a^*)}{\partial a^*}\left(J_{g_\theta}^{-1}|_{a^*}\right)\frac{\partial f_\theta(a^*)}{\partial \theta}, \tag{3}$$

where $J_{g_\theta}^{-1}|_{a^*}$ is the inverse Jacobian of g_θ evaluated at a^*. For the proof please refer to [3]. To solve the inverse Jacobian, we follow [3, 4] and solve an alternative linear system $\left(J_{g_\theta}^T|_{a^*}\right)x^T + \left(\frac{\partial \mathcal{L}(a^*)}{\partial a^*}\right)^T = 0$. We can leverage Broyden's method [3, 4], which is a second-order quasi-Newton approach; or we can alternatively use a fixed-point update scheme $x^T = (J_{f_\theta}^T|_{a^*})x^T + \left(\frac{\partial \mathcal{L}(a^*)}{\partial a^*}\right)^T$ since $J_{g_\theta}^T|_{a^*} = J_{f_\theta}^T|_{a^*} - I$, and it converges with linear convergence rate as long as $\|J_{f_\theta}^T|_{a^*}\| \leq 1$. In this way, gradients for the parameters can be calculated only with the equilibrium state and equation.

4 Proposed IDE Method

In this section, we first derive the equilibrium state of FSNNs under both continuous and discrete views, and demonstrate that FSNNs can be treated as solving a fixed-point equation. Then we introduce how to train the network by the proposed IDE method based on the equation and briefly discuss the biological plausibility. Finally, we incorporate the multi-layer structure into the model for more non-linearity and stronger representation ability.

4.1 Derivation of Equilibrium States for Feedback Spiking Neural Networks

4.1.1 Continuous View

We first consider a group of spiking neurons with feedback connections. Let $\mathbf{u}(t)$ and $\mathbf{s}(t)$ denote the membrane potentials and spikes of these neurons at time t respectively, $\mathbf{x}(t)$ denote the inputs, W denote the feedback weight matrix, F denote the weight matrix from inputs to these neurons, and $\mathbf{b}$ denote a constant bias. Under the IF model, the dynamics of membrane potentials are expressed as:

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = W\mathbf{s}(t - \Delta t_d) + F\mathbf{x}(t) + \mathbf{b} - V_{th}\mathbf{s}(t), \tag{4}$$

where Δt_d is a time delay of feedback connections, and V_{th} is the threshold. Note that W and F represent linear operations including both fully-connected and convolutional layers. Define the average firing rates as $\mathbf{a}(t) = \frac{1}{t}\int_0^t \mathbf{s}(\tau)\mathrm{d}\tau$, and the average inputs as $\overline{\mathbf{x}}(t) = \frac{1}{t}\int_0^t \mathbf{x}(\tau)\mathrm{d}\tau$. Then through integration, we have:

$$\mathbf{a}(t) = \frac{1}{V_{th}}\left(\frac{t - \Delta t_d}{t}W\mathbf{a}(t - \Delta t_d) + F\overline{\mathbf{x}}(t) + \mathbf{b} - \frac{\mathbf{u}(t)}{t}\right). \tag{5}$$

Eq.(5) roughly follows a fixed-point update scheme except the existence of $\mathbf{u}(t)$. Now we dive into $\mathbf{u}(t)$. Since neurons will not spike when the accumulated $\mathbf{u}(t)$ is negative, if $v_i(t) = \left(\frac{t-\Delta t_d}{t}W\mathbf{a}(t - \Delta t_d) + F\overline{\mathbf{x}}(t) + \mathbf{b}\right)_i < 0$, ideally neuron i generates no spike and its accumulated negative term remains in $\mathbf{u}_i(t)$. So $\mathbf{u}_i(t)$ can be divided as $\mathbf{u}_i(t) = \mathbf{u}_i^-(t) + \mathbf{u}_i^+(t)$, where $\frac{1}{t}\mathbf{u}_i^-(t) = \min(v_i(t), 0)$ is the remaining negative term, and $\mathbf{u}_i^+(t)$ is the positive one typically bounded in the range between 0 and V_{th}. There could be random error in $\mathbf{u}_i^+(t)$ in the context of random arrival of spikes rather than the average condition (e.g. the average is 0, but a large positive input followed by a negative one will generate an unexpected spike). Despite this, we can still assume $\mathbf{u}_i^+(t)$ to be bounded by a constant when average inputs converge. By this decomposition, we have the equation with the element-wise ReLU function ($\mathrm{ReLU}(x) = \max(x, 0)$) and bounded $\mathbf{u}^+(t)$:

$$\mathbf{a}(t) = \mathrm{ReLU}\left(\frac{1}{V_{th}}\left(\frac{t - \Delta t_d}{t}W\mathbf{a}(t - \Delta t_d) + F\overline{\mathbf{x}}(t) + \mathbf{b} - \frac{\mathbf{u}^+(t)}{t}\right)\right). \tag{6}$$

With Eq.(6), we can derive that the average firing rate will gradually converge to an equilibrium state.

Theorem 1. *If the average inputs converge to an equilibrium point $\overline{\mathbf{x}}(t) \to \mathbf{x}^*$, and there exists constant c and $\gamma < 1$ such that $|\mathbf{u}_i^+(t)| \leq c, \forall i, t$ and $\|W\|_2 \leq \gamma V_{th}$, then the average firing rates of FSNN with continuous IF model in Eq.(6) will converge to an equilibrium point $\mathbf{a}(t) \to \mathbf{a}^*$, which satisfies the fixed-point equation $\mathbf{a}^* = ReLU\left(\frac{1}{V_{th}}\left(W\mathbf{a}^* + F\mathbf{x}^* + \mathbf{b}\right)\right)$.*

The proof can be found in Appendix A. Theorem 1 rigorously shows the equilibrium state under the IF model, and we can view the forward computation of FSNN as solving this fixed-point equation.

As for the LIF model, we can similarly define the weighted average firing rate $\hat{\mathbf{a}}(t) = \frac{\int_0^t \kappa(t-\tau)\mathbf{s}(\tau)\mathrm{d}\tau}{\int_0^t \kappa(t-\tau)\mathrm{d}\tau}$ and the weighted average inputs $\hat{\mathbf{x}}(t) = \frac{\int_0^t \kappa(t-\tau)\mathbf{x}(\tau)\mathrm{d}\tau}{\int_0^t \kappa(t-\tau)\mathrm{d}\tau}$, where $\kappa(\tau) = \exp(-\frac{\tau}{\tau_m})$ is the response kernel of the LIF model. In this setting, however, there could be random errors caused by $\mathbf{u}^+(t)$ as its denominator $\int_0^t \kappa(t-\tau)\mathrm{d}\tau$ does not go to infinity. We consider it as an approximate solver for the equilibrium with random errors, as shown in Proposition 1. Please refer to Appendix C for details.

Proposition 1. *If the weighted average inputs converge to an equilibrium point $\hat{\mathbf{x}}(t) \to \mathbf{x}^*$, and there exists constant c and $\gamma < 1$ such that $|\mathbf{u}_i^+(t)| \leq c, \forall i, t$ and $\|W\|_2 \leq \gamma V_{th}$, then the weighted average firing rates $\hat{\mathbf{a}}(t)$ of FSNN with continuous LIF model gradually approximate an equilibrium point $\mathbf{a}^*$ with bounded random errors, which satisfies $\mathbf{a}^* = ReLU\left(\frac{1}{V_{th}}\left(W\mathbf{a}^* + F\mathbf{x}^* + \mathbf{b}\right)\right)$.*

4.1.2 Discrete View

In practice, we will simulate SNNs with discretization. Now consider the computation in Eq.(2). With feedback connections, the update equation of membrane potentials under the IF model is:

$$\mathbf{u}[t+1] = \mathbf{u}[t] + W\mathbf{s}[t] + F\mathbf{x}[t] + \mathbf{b} - V_{th}\mathbf{s}[t+1], \tag{7}$$

where we treat the feedback delay in one time step for simplicity. Define the average firing rates as $\mathbf{a}[t] = \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{s}[\tau]$, the average inputs as $\overline{\mathbf{x}}(t) = \frac{1}{t+1}\sum_{\tau=0}^{t}\mathbf{x}[\tau]$, and $\mathbf{u}[0] = \mathbf{0}, \mathbf{s}[0] = \mathbf{0}$. By summation, we have:

$$\mathbf{a}[t+1] = \frac{1}{V_{th}}\left(\frac{t}{t+1}W\mathbf{a}[t] + F\overline{\mathbf{x}}[t] + \mathbf{b} - \frac{\mathbf{u}[t+1]}{t+1}\right). \tag{8}$$

Different from the continuous view, $\mathbf{a}[t]$ is bounded in the range of $[0, 1]$, since there could be at most t spikes during t time steps. Therefore, $\mathbf{u}_i[t]$ will maintain both the negative terms and the exceeded positive ones. Similarly, $\mathbf{u}_i[t] = \mathbf{u}_i^-[t] + \mathbf{u}_i^+[t]$, where $\frac{1}{t}\mathbf{u}_i^-[t] = \min\left(\max\left(v_i[t] - V_{th}, 0\right), v_i[t]\right)$ is the exceeded term, and $\mathbf{u}_i^+[t]$ is assumed to be bounded by a constant as previously indicated. Then:

$$\mathbf{a}[t+1] = \sigma\left(\frac{1}{V_{th}}\left(\frac{t}{t+1}W\mathbf{a}[t] + F\overline{\mathbf{x}}[t] + \mathbf{b} - \frac{\mathbf{u}^+[t+1]}{t+1}\right)\right), \text{where}\, \sigma(x) = \begin{cases} 1, & x > 1 \\ x, & 0 \leq x \leq 1 \\ 0, & x < 0 \end{cases}. \tag{9}$$

With Eq.(9), we can derive the equilibrium state under discrete view.

Theorem 2. *If the average inputs converge to an equilibrium point $\overline{\mathbf{x}}[t] \to \mathbf{x}^*$, and there exists constant c and $\gamma < 1$ such that $|\mathbf{u}_i^+(t)| \leq c, \forall i, t$ and $\|W\|_2 \leq \gamma V_{th}$, then the average firing rates of FSNN with discrete IF model in Eq.(9) will converge to an equilibrium point $\mathbf{a}[t] \to \mathbf{a}^*$, which satisfies the fixed-point equation $\mathbf{a}^* = \sigma\left(\frac{1}{V_{th}}\left(W\mathbf{a}^* + F\mathbf{x}^* + \mathbf{b}\right)\right)$.*

The proof can be found in Appendix A. And for the LIF model, define the weighted average firing rate $\hat{\mathbf{a}}[t] = \frac{\sum_{\tau=1}^{t}\lambda^{t-\tau}\mathbf{s}[\tau]}{\sum_{\tau=1}^{t}\lambda^{t-\tau}}$ and the weighted average inputs $\hat{\mathbf{x}}[t] = \frac{\sum_{\tau=0}^{t}\lambda^{t-\tau}\mathbf{x}[\tau]}{\sum_{\tau=0}^{t}\lambda^{t-\tau}}$, then we can similarly consider it as an approximation solver for the equilibrium state with random errors, as shown in Proposition 2. Please refer to Appendix C for details.

Proposition 2. *If the weighted average inputs converge to an equilibrium point $\hat{\mathbf{x}}[t] \to \mathbf{x}^*$, and there exists constant c and $\gamma < 1$ such that $|\mathbf{u}_i^+(t)| \leq c, \forall i, t$ and $\|W\|_2 \leq \gamma V_{th}$, then the weighted average firing rates $\hat{\mathbf{a}}[t]$ of FSNN with discrete LIF model gradually approximate an equilibrium point a^* with bounded random errors, which satisfies $\mathbf{a}^* = \sigma\left(\frac{1}{V_{th}}\left(W\mathbf{a}^* + F\mathbf{x}^* + \mathbf{b}\right)\right)$.*

4.2 Training of Feedback Spiking Neural Networks

Based on the derivation in Section 4.1, we can view the forward computation of FSNNs as a black-box solver for the fixed-point equilibrium equations, with some errors caused by finite time steps or the LIF model. Then we assume the (weighted) average firing rates $\mathbf{a}[T]$ after T time steps approximately follow the equations. We will demonstrate how to train FSNNs and its biological plausibility.

4.2.1 Loss and Gradient Computation

Suppose that we simulate the SNN by T time steps. Let $\mathbf{a}[T]$ denote the final (weighted) average firing rates. We configure a readout layer after these spiking neurons, which performs as a fully-connected classification layer, with the number of outputs as class numbers. We assume that these neurons will not spike or reset, and do classification based on the accumulated membrane potential. Then the outputs are equivalent as a linear transformation on $\mathbf{a}[T]$, i.e. $\mathbf{o} = W^o\mathbf{a}[T]$. The loss L is defined on $\mathbf{o}$ and labels $\mathbf{y}$ by commonly used loss functions $\mathcal{L}(\mathbf{o}, \mathbf{y})$, and we leverage the cross-entropy loss.

Let f_θ denote the function in the fixed-point equilibrium equation, e.g. $f_\theta(\mathbf{a}^*, \mathbf{x}^*) = \sigma\left(\frac{1}{V_{th}}\left(W\mathbf{a}^* + F\mathbf{x}^* + \mathbf{b}\right)\right)$. The gradient for parameters can be calculated based on the implicit differentiation as described in Section 3.2 by substituting $\mathbf{a}^*$ with $\mathbf{a}[T]$. Then parameters can be optimized based on common gradient descent methods, e.g. SGD [33] and its variants.

4.2.2 Biological Plausibility of Implicit Differentiation

While implicit differentiation may seem too abstract for information propagation compared with back-propagation, we will briefly discuss the biological possibility for this calculation and its connection to the Hebbian learning rule [10]. Consider the equilibrium state $\mathbf{a}^*$ following $\mathbf{a}^* = f_\theta(\mathbf{a}^*, \mathbf{x}^*) = \text{ReLU}\left(\frac{1}{V_{th}}\left(W\mathbf{a}^* + F\mathbf{x}^* + \mathbf{b}\right)\right)$. Let $\mathbf{m} = f'_\theta(\mathbf{a}^*, \mathbf{x}^*) = H\left(\frac{1}{V_{th}}\left(W\mathbf{a}^* + F\mathbf{x}^* + \mathbf{b}\right)\right) = H(\mathbf{a}^*), M = \text{Diag}(\mathbf{m}), \widetilde{W} = MW$, where $H(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$. As indicated in Section 3.2, the gradient can be calculated as $\nabla_\theta \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial \theta}\right)^T = \left(\frac{\partial f_\theta(\mathbf{a}^*, \mathbf{x}^*)}{\partial \theta}\right)^T \left(I - \frac{1}{V_{th}}\widetilde{W}^T\right)^{-1} \left(\frac{\partial \mathcal{L}}{\partial \mathbf{a}^*}\right)^T$, and we can leverage a fixed-point update scheme to solve for $\boldsymbol{\beta}^* = \left(I - \frac{1}{V_{th}}\widetilde{W}^T\right)^{-1} \left(\frac{\partial \mathcal{L}}{\partial \mathbf{a}^*}\right)^T$ by iterating $\boldsymbol{\beta}^{k+1} = \frac{1}{V_{th}}\widetilde{W}^T\boldsymbol{\beta}^k + \frac{\partial \mathcal{L}}{\partial \mathbf{a}^*}$, $\boldsymbol{\beta}^k \to \boldsymbol{\beta}^*$. This can be viewed as computing another equilibrium for these neurons: in this stage, neurons receive the inputs $\frac{\partial \mathcal{L}}{\partial \mathbf{a}^*}$ and they use the inverse directions of connections with a mask (M can be viewed as a mask matrix based on the firing condition in the first stage, which may be realized by some inhibition mechanisms) to compute for the equilibrium $\boldsymbol{\beta}^*$. If W is symmetric and $V_{th} = 1$, it is similar to the energy-based method equilibrium propagation [34] to use the same weight as the forward computation for a second equilibrium computation. Plugging $\boldsymbol{\beta}^*$ into the gradient and calculating $\left(\frac{\partial f_\theta(\mathbf{a}^*, \mathbf{x}^*)}{\partial \theta}\right)^T$ explicitly, we have: $\nabla_W \mathcal{L} = \frac{1}{V_{th}} M\boldsymbol{\beta}^*\mathbf{a}^{*T}, \nabla_F \mathcal{L} = \frac{1}{V_{th}} M\boldsymbol{\beta}^*\mathbf{x}^{*T}$. It is interesting to find that the change of weight from neuron j to neuron i is proportional to the the equilibrium state of neuron j in the first stage and that of neuron i in the second stage, and is related to whether neuron i fires in the first stage, because $\nabla_{W_{i,j}} \mathcal{L} = \frac{1}{V_{th}} m_i \beta_i^* a_j^*, \nabla_{F_{i,j}} \mathcal{L} = \frac{1}{V_{th}} m_i \beta_i^* x_j^*$. It is to some extent similar to the locally updated Hebbian learning rule $\Delta w_{i,j} \propto x_i x_j$ meaning that neurons wire together if they fire together [10], except that we take average firing rates and some temporal information (two stages) into account. Therefore, implicit differentiation calculation is only related to two equilibrium states by the neurons with a mask possibly realized by inhibition mechanisms, and may correspond to modified locally updated rules, which is more biologically plausible than BPTT with surrogate derivatives for SNNs. Please note that 'biological plausibility' here is a brief discussion in the context of the above properties, while there may be other aspects of biological implausibility as well.

4.3 Incorporating Multi-layer Structure into The Feedback Model

The multi-layer structure is commonly adopted in ANNs due to its stronger non-linearity and representation ability. To enhance the non-linearity of the fixed-point equilibrium equation, we propose to incorporate multi-layer structure into FSNNs as well. We configure N subgroups of neurons as different layers, where the inputs have connections to the first layer, the $(l-1)$-th layer has connections to the l-th layer, and the last layer has feedback connections to the first layer. Let

$\mathbf{u}^l(t)$ and $\mathbf{s}^l(t)$ denote the l-th layer, $\mathbf{x}(t)$ denote the inputs, W^1 denote the feedback connection from the last layer to the first layer, and F^l denote the weight from the $(l-1)$-th layer (or input) to the l-th layer. The discrete update equations of membrane potentials are expressed as:

$$\begin{cases} \mathbf{u}^1[t+1] = \lambda \mathbf{u}^1[t] + W^1 \mathbf{s}^N[t] + F^1 \mathbf{x}[t] + \mathbf{b}^1 - V_{th}\mathbf{s}^1[t+1], \\ \mathbf{u}^{l+1}[t+1] = \lambda \mathbf{u}^{l+1}[t] + F^{l+1}\mathbf{s}^l[t+1] + \mathbf{b}^{l+1} - V_{th}\mathbf{s}^{l+1}[t+1], \quad l = 1, 2, \cdots, N-1. \end{cases} \tag{10}$$

With similar definitions of average firing rates $\mathbf{a}^l[t]$ for different layers, and the similar separation $\mathbf{u}_i^l[t] = \mathbf{u}_i^{l^-}[t] + \mathbf{u}_i^{l^+}[t]$, we can have the equilibrium state as the following.

Theorem 3. *If the average inputs converge to an equilibrium point $\overline{\mathbf{x}}[t] \to \mathbf{x}^*$, and there exists constant c and $\gamma < 1$ such that $|\mathbf{u}_i^{l^+}[t]| \leq c, \forall i, l, t$ and $\|W^1\|_2 \|F^N\|_2 \cdots \|F^2\|_2 \leq \gamma V_{th}^N$, then the average firing rates of multi-layer FSNN with discrete IF model will converge to equilibrium points $\mathbf{a}^l[t] \to \mathbf{a}^{l^*}$, which satisfy the fixed-point equations $\mathbf{a}^{1^*} = f_1\left(f_N \circ \cdots \circ f_2(\mathbf{a}^{1^*}), \mathbf{x}^*\right)$ and $\mathbf{a}^{l+1^*} = f_{l+1}(\mathbf{a}^{l^*})$, where $f_1(\mathbf{a}, \mathbf{x}) = \sigma\left(\frac{1}{V_{th}}(W^1\mathbf{a} + F^1\mathbf{x} + \mathbf{b}^1)\right)$ and $f_l(\mathbf{a}) = \sigma\left(\frac{1}{V_{th}}(F^l\mathbf{a} + \mathbf{b}^l)\right)$.*

The proof can be found in Appendix B. There is also a similar proposition for the LIF model, please refer to Appendix C for details. We will do classification based on the (weighted) average firing rate of the last layer and calculate the implicit differentiation for the equation on $\mathbf{a}^N[T]$. The loss, solution for implicit differentiation, and optimization methods are the same as those in Section 4.2.1.

5 Experiments

In this section, we conduct extensive experiments to demonstrate the superior performance of our proposed method. Please refer to Appendix D for implementation details, including restrictions on the spectral norm and batch normalization, as well as training parameters. Since few previous methods leverage feedback architectures, we compare the results of our IDE method for FSNNs with most feedforward SNNs and report network structures[1] as well as the number of neurons and parameters during computation (calculated according to the papers or released codes) for comparison.

5.1 MNIST and Fashion-MNIST

We first evaluate our method on simple static image datasets including MNIST [19] and Fashion-MNIST [41], and compare the results with other directly trained SNNs [21, 39, 36, 14, 43, 44] or similar ANNs. Inputs are the same images with binary or real values at all time steps, which can be regarded as input currents [44]. We leverage single-layer FSNNs, and adopt convolutional layers for MNIST while using fully-connected layers for Fashion-MNIST following [43]. As shown in Table 1, our models achieve comparable or better results with fewer neurons and parameters in a relatively small number of time steps, compared with other direct SNN training methods on feedforward or feedback architectures. Especially, our model achieves superior results on Fashion-MNIST with the similar structure in only 5 time steps. The LIF model performs slightly better than the IF model, probably because it leverages temporal information by encoding weighted average firing rates.

5.2 N-MNIST

We also evaluate our method on the neuromorphic dataset N-MNIST [26], whose inputs are spikes collected by dynamic vision sensors. We follow the same data pre-possessing as [44] and take 30 time steps, and we can view the (weighted) average inputs gradually converge to that at the last time step. Table 2 demonstrates the comparison results of our models and other directly trained models [14, 36, 44, 40]. It shows that our method can achieve satisfactory performance on neuromorphic data as well. Especially, only 30 time steps are required by our method for satisfactory performance.

5.3 CIFAR-10 and CIFAR-100

Then we evaluate our method on more complex CIFAR-10 and CIFAR-100 datasets [17]. We leverage multi-layer FSNNs with structures modified from AlexNet and CIFARNet proposed in [40], as indicated in the footnote of Table 3. We compare our model with SNNs converted from

[1] The notations are: '64C5' means a convolution with 64 output channels and kernel size 5, 's' after '64C5' means convolution with stride 2 while 'u' after that means a transposed convolution to upscale $2\times$, 'P2' means average pooling with size 2, '400' means fully-connected to 400 neurons, and 'F' means feedback layers.

Table 1: Performance on MNIST and Fashion-MNIST. Results are based on 5 runs of experiments.

MNIST

Method	Network structure	Time steps	Mean±Std	Best	Neurons	Params
BP [21]	20C5-P2-50C5-P2-200	>200	/	99.31%	33K	518K
STBP [39]	15C5-P2-40C5-P2-300	30	/	99.42%	26K	607K
SLAYER [36]	12C5-P2-64C5-P2	300	99.36%±0.05%	99.41%	28K	51K
HM2BP [14]	15C5-P2-40C5-P2-300	400	99.42%±0.11%	99.49%	26K	607K
ST-RSBP [43]	15C5-P2-40C5-P2-300	400	**99.57%±0.04%**	**99.62%**	26K	607K
TSSL-BP [44]	15C5-P2-40C5-P2-300	5	99.50%±0.02%	99.53%	26K	607K
IDE-IF (ours)	64C5s (F64C5)	30	99.49%±0.04%	99.55%	13K	229K
IDE-LIF (ours)	64C5s (F64C5)	30	99.53%±0.04%	99.59%	13K	229K

Fashion-MNIST

Method	Network structure	Time steps	Mean±Std	Best	Neurons	Params
ANN [43]	512-512	/	/	89.01%	1.8K	670K
HM2BP [43]	400-400	400	/	88.99%	1.6K	478K
TSSL-BP [44]	400-400	5	89.75%±0.03%	89.80%	1.6K	478K
ST-RSBP [43]	400 (F400)	400	90.00%±0.14%	90.13%	1.2K	478K
IDE-IF (ours)	400 (F400)	**5**	**90.04%±0.09%**	**90.14%**	1.2K	478K
IDE-LIF (ours)	400 (F400)	**5**	**90.07%±0.10%**	**90.25%**	1.2K	478K

Table 2: Performance on N-MNIST. Results are based on 5 runs of experiments.

Method	Network structure	Time steps	Mean±Std	Best	Neurons	Params
HM2BP [14]	400-400	600	98.88%±0.02%	98.88%	3K	1.1M
SLAYER [36]	500-500	300	98.89%±0.06%	98.95%	3K	1.4M
SLAYER [36]	12C5-P2-64C5-P2	300	99.20%±0.02%	99.22%	40K	61K
TSSL-BP [44]	12C5-P2-64C5-P2	30	99.23%±0.05%	99.28%	40K	61K
STBP w/o NeuNorm [40]	CNN[1]	60	/	99.44%	414K	17.3M
IDE-IF (ours)	64C5s (F64C5)	30	99.30%±0.04%	99.35%	21K	291K
IDE-LIF (ours)	64C5s (F64C5)	**30**	**99.42%±0.04%**	**99.47%**	21K	291K

[1] 128C3-128C3-P2-128C3-256C3-P2-1024

ANNs [35, 8, 30, 42] and directly trained SNNs [40, 44, 20, 38]. For CIFAR-100, no result of directly trained SNN is reported, so we only compare with the converted ones. Table 3 demonstrates the superior results of our directly trained models with fewer neurons and parameters in a small number of time steps. Especially, our model can outperform the state-of-the-art SNN performance on CIFAR-100 with only 30 time steps, and achieves 1.59% accuracy improvement when 100 time steps are adopted. Please refer to Appendix E for more comparison results between IF and LIF models.

5.4 Convergence to the Equilibrium

To verify the convergence of FSNNs to equilibrium states, we plot the difference norm on the fixed-point equation at each time step, i.e. $\|f_\theta(\mathbf{a}[t]) - \mathbf{a}[t]\|$, where $f_\theta(x)$ is the fixed-point equilibrium equation and $\mathbf{a}[t]$ is the (weighted) average firing rate based on the first t time steps. Figure 1 demonstrates the convergence of different models. The convergence is almost the same among different samples. Since the numerical precision of average firing rates is only $\frac{1}{t}$, there would be a certain convergence error due to the finite time steps. And for the LIF model, there could be random errors compared with the IF model, as indicated in Section 4. The results conform to the theorems as the difference norm gradually decreases, i.e. firing rates converge to the equilibrium following the equation. It also indicates that exact precision is not necessary for satisfactory performance. Fully-connected layers with fewer neurons converge faster, so only a smaller number of time steps is needed, explaining why only 5 time steps are needed in the Fashion-MNIST experiment. For results on more datasets and different scales of time steps, please refer to Appendix E.

Table 3: Performance on CIFAR-10 and CIFAR-100. Results are based on 5 runs of experiments.

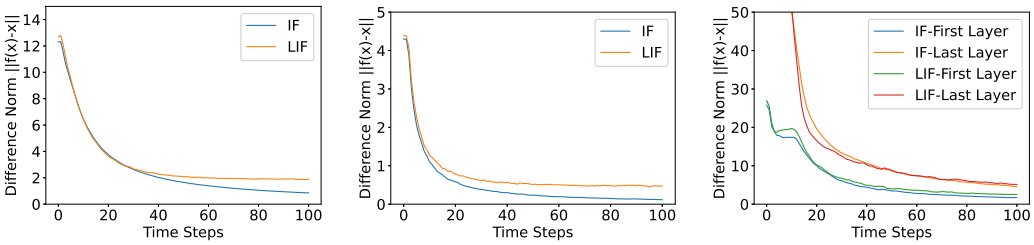
CIFAR-10

Method	Network structure	Time steps	Mean±Std	Best	Neurons	Params
ANN-SNN [8]	CIFARNet	400-600	/	90.61%	726K	45M
ANN-SNN [35]	VGG-16	2500	/	91.55%	311K	15M
ANN-SNN [8]	VGG-16	400-600	/	92.26%	318K	40M
Hybrid Training [30]	VGG-16	100	/	91.13%	318K	40M
STBP [40]	AlexNet	12	/	85.24%	595K	21M
TSSL-BP [44]	AlexNet	5	88.98%±0.27%	89.22%	595K	21M
STBP [40]	CIFARNet	12	/	90.53%	726K	45M
TSSL-BP [44]	CIFARNet	5	/	91.41%	726K	45M
Surrogate gradient [20]	VGG-9	100	/	90.45%	274K	5.9M
ASF-BP [38]	VGG-7	400	/	91.35%	>240K	>30M
IDE-LIF (ours)	AlexNet-F	30	91.74%±0.09%	91.92%	159K	3.7M
IDE-LIF (ours)	AlexNet-F	100	**92.03%±0.07%**	**92.15%**	**159K**	**3.7M**
IDE-LIF (ours)	CIFARNet-F	30	92.08%±0.14%	92.23%	232K	11.8M
IDE-LIF (ours)	CIFARNet-F	100	**92.52%±0.17%**	**92.82%**	**232K**	**11.8M**

[1] AlexNet [40]: 96C3-256C3-P2-384C3-P2-384C3-256C3-1024-1024
[2] AlexNet-F: 96C3s-256C3-384C3s-384C3-256C3 (F96C3u)
[3] CIFARNet [40]: 128C3-256C3-P2-512C3-P2-1024C3-512C3-1024-512
[4] CIFARNet-F: 128C3s-256C3-512C3s-1024C3-512C3 (F128C3u)

CIFAR-100

Method	Network structure	Time steps	Mean±Std	Best	Neurons	Params
ANN [35]	VGG-16	/	/	71.22%	311K	15M
ANN-SNN [35]	VGG-16	2500	/	70.77%	311K	15M
ANN-SNN [8]	VGG-16	400-600	/	70.55%	318K	40M
ANN-SNN [42]	VGG-*	300	/	71.84%	540K	9.7M
IDE-IF (ours)	CIFARNet-F	**30**	**71.56%±0.31%**	**72.10%**	232K	14.8M
IDE-IF (ours)	AlexNet-F	**100**	**72.02%±0.16%**	**72.23%**	**159K**	**5.2M**
IDE-IF (ours)	CIFARNet-F	**100**	**73.07%±0.21%**	**73.43%**	232K	14.8M



(a) MNIST: 64C5s (F64C5) (b) Fashion-MNIST: 400 (F400) (c) CIFAR-10: CIFARNet-F

Figure 1: Convergence to the equilibrium of different models on a random sample in 100 time steps.

6 Conclusion

In this work, we propose a novel training method for feedback spiking neural networks based on implicit differentiation on the equilibrium state. We first derive the equilibrium states of (weighted) average firing rates for the IF and the LIF models of FSNNs under both continuous and discrete views. Then we propose to optimize parameters of FSNNs only based on the implicit differentiation on the underlying fixed-point equation. This enables the backward procedure to be decoupled from the forward computational graph and therefore avoids the common training problems for SNNs, such as non-differentiability and large memory costs. Meanwhile, we briefly discuss the biological plausibility for the calculation of implicit differentiation, which only requires computing another equilibrium and is related to the locally updated Hebbian learning rule. Extensive experiments demonstrate the superior results of our method and models with fewer neurons and parameters in a small number of time steps.

References

[1] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, 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.

[2] LB Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings, 1st First International Conference on Neural Networks*, volume 2, pages 609–618, 1987.

[3] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, pages 690–701, 2019.

[4] Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[5] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, pages 795–805, 2018.

[6] Sander M Bohte, Joost N Kok, and Han La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002.

[7] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[8] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations*, 2021.

[9] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9:99, 2015.

[10] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[11] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[12] John J Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81(10):3088–3092, 1984.

[13] Eric Hunsberger and Chris Eliasmith. Spiking deep networks with LIF neurons. *arXiv preprint arXiv:1510.08829*, 2015.

[14] Yingyezhe Jin, Wenrui Zhang, and Peng Li. Hybrid macro/micro level backpropagation for training deep spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 7005–7015, 2018.

[15] Kohitij Kar, Jonas Kubilius, Kailyn Schmidt, Elias B Issa, and James J DiCarlo. Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. *Nature Neuroscience*, 22(6):974–983, 2019.

[16] Jinseok Kim, Kyungsu Kim, and Jae-Joon Kim. Unifying activation-and timing-based learning rules for spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[18] Jonas Kubilius, Martin Schrimpf, Kohitij Kar, Rishi Rajalingham, Ha Hong, Najib Majaj, Elias Issa, Pouya Bashivan, Jonathan Prescott-Roy, Kailyn Schmidt, et al. Brain-like object recognition with high-performing shallow recurrent anns. *Advances in Neural Information Processing Systems*, 32:12805–12816, 2019.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 14, 2020.

[21] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10:508, 2016.

[22] Robert Legenstein, Dejan Pecevski, and Wolfgang Maass. A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Comput Biol*, 4(10):e1000180, 2008.

[23] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[24] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[25] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[26] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9:437, 2015.

[27] Peter O'Connor, Efstratios Gavves, and Max Welling. Training a spiking neural network with equilibrium propagation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1516–1523, 2019.

[28] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

[29] Fernando J Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59(19):2229, 1987.

[30] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In *International Conference on Learning Representations*, 2019.

[31] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[32] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11:682, 2017.

[33] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[34] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, 11:24, 2017.

[35] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience*, 13:95, 2019.

[36] Sumit Bam Shrestha and Garrick Orchard. Slayer: spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*, pages 1419–1428, 2018.

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

[38] Hao Wu, Yueyi Zhang, Wenming Weng, Yongting Zhang, Zhiwei Xiong, Zheng-Jun Zha, Xiaoyan Sun, and Feng Wu. Training spiking neural networks with accumulated spiking flow. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[39] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12:331, 2018.

[40] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1311–1318, 2019.

[41] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[42] Zhanglu Yan, Jun Zhou, and Weng-Fai Wong. Near lossless transfer learning for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[43] Wenrui Zhang and Peng Li. Spike-train level backpropagation for training deep recurrent spiking neural networks. In *Advances in Neural Information Processing Systems*, 2019.

[44] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[45] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Appendix F.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix F.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 4.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A, Appendix B, and Appendix C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Supplementary Materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix D.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.
   (b) Did you mention the license of the assets? [Yes] See Appendix D.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] We do not use new assets.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Appendix D.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Appendix D.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]