Skip-Connected Self-Recurrent Spiking Neural Networks with Joint Intrinsic Parameter and Synaptic Weight Training

Wenrui Zhang

wenruizhang@ucsb.edu

Department of Electrical and Computer Engineering, University of California, Santa Barbara, Santa Barbara, CA, 93106, U.S.A.

Peng Li

lip@ucsb.edu

Department of Electrical and Computer Engineering, University of California, Santa Barbara, Santa Barbara, CA, 93106, U.S.A.

Keywords: Deep Recurrent Spiking Neural Networks, Spiking Neural Networks Architecture, Backpropagation, Intrinsic Plasticity.

Abstract

As an important class of spiking neural networks (SNNs), recurrent spiking neural networks (RSNNs) possess great computational power and have been widely used for processing sequential data like audio and text. However, most RSNNs suffer from two problems. 1. Due to the lack of architectural guidance, random recurrent connectivity is often adopted, which does not guarantee good performance. 2. Training of RSNNs is in general challenging, bottlenecking achievable model accuracy. To address these problems, we propose a new type of RSNNs called Skip-Connected Self-Recurrent SNNs (ScSr-SNNs). Recurrence in ScSr-SNNs is introduced in a stereotyped manner by adding self-recurrent connections to spiking neurons. The SNNs with self-recurrent connections can realize recurrent behaviors similar to those of more complex RSNNs while the error gradients can be more straightforwardly calculated due to the mostly feedforward nature of the network. The network dynamics is enriched by skip connections between nonadjacent layers. Moreover, we propose a new backpropagation (BP) method called backpropagated intrinsic plasticity (BIP) to further boost the performance of ScSr-SNNs by training intrinsic model parameters. Unlike standard intrinsic plasticity rules that adjust the neuron's intrinsic parameters according to neuronal activity, the proposed BIP method optimizes intrinsic parameters based on the backpropagated error gradient of a well-defined global loss function in addition to synaptic weight training. Based on challenging speech, neuromorphic speech, and neuromorphic image datasets, the proposed ScSr-SNNs can boost performance by up to 2.85% compared with other types of RSNNs trained by state-of-the-art BP methods.

1 Introduction

Recurrent neural networks (RNNs) are one of the most popular types of networks in artificial neural networks (ANNs). They are designed to better handle sequential information such as audio or text. RNNs make use of internal states to store past information, which is combined with the current input to determine the current network output. During the past few decades, RNNs have been widely studied with various structures such as Gated Recurrent Units (GRU) (Cho et al., 2014), Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), Legendre Memory Units (LMU) (Voelker, Kajić, and Eliasmith, 2019), Echo state networks (ESN) (Jaeger, 2001), and Deep RNNs (Graves, Mohamed, and Hinton, 2013).

Similarly, recurrent spiking neural networks (RSNNs) are competent for processing temporal signals such as time series or speech data (Ghani et al., 2008). Moreover, RSNNs are advantageous from a biological plausibility point of view compared with their non-spiking counterparts and enjoy their intrinsic spatiotemporal computing power attributed to the rich network dynamics created by the recurrence of connectivity. While mature network architectures have been developed for RNNs, the complex dynamics of RSNNs are not well understood and training of RSNNs is in general challenging. These difficulties lead to the demonstration of simple RSNN architectures, severely limiting the practical application of RSNNs.

In the previous works, most RSNNs have randomly generated recurrent connections. Training these recurrent connections is challenging, bottlenecking achievable model accuracy. One of the well-known RSNN models is the liquid State Machine (LSM) (Maass, Natschläger, and Markram, 2002) which has a single randomly connected reservoir layer followed by a readout layer. The reservoir weights typically are either fixed or trained by unsupervised learning rules like spike-timing-dependent plasticity (STDP) (Morrison, Diesmann, and Gerstner, 2008) with only the readout layer trained by supervision (Ponulak and Kasiński, 2010; Zhang et al., 2015; Jin and Li, 2016). In recent years, the reservoir of LSM has been extended to different structures. In (Wang and Li, 2016) and (Srinivasan, Panda, and Roy, 2018), multiple reservoirs are applied to process different parts of the input signals. (Maes, Barahona, and Clopath, 2020) proposed a recurrent structure in which the recurrent layer is organized in C clusters of excitatory neurons and a central cluster of inhibitory neurons. In (Panda and Srinivasa, 2018), a modified D/A based reservoir construction approach is applied to build the reservoir. (Bellec et al., 2018) proposed an architecture called long short-term memory SNNs with one recurrent layer. Its recurrent layer has a regular spiking portion with both inhibitory and excitatory spiking neurons and an adaptive neural population. The recurrent connections are trained by the BPTT method. (Zhang and Li, 2019b) demonstrates training of deep RSNNs by a backpropagation method called ST-RSBP. However, the recurrent connections in all of these works are sparsely and randomly generated with certain probabilities.

On the other hand, (Shrestha et al., 2017) and (Lotfi Rezaabad and Vishwanath, 2020) propose methods to convert a pre-trained ANN-based LSTM to a spiking LSTM. However, they either directly substitute the original non-spiking activation function with a spiking activation function or approximate the non-spiking activation function heuristically. The original ANN LSTM model is re-trained using a standard BP method for ANNs and no additional training is applied to the converted spiking LSTM model in the spatiotemporal domain. Hence, these ANN-to-SNN conversion approaches are unable to explore the intrinsic spatiotemporal computing capability of typical spiking neurons such as ones modeled using the Leaky-Integrate-and-Fire (LIF) model or spike response model (SRM). In addition, the converted spiking LSTMs are trained with the BP method of ANNs without considering the spikes.

In most existing works of RSNNs, the recurrent connections are sparsely and randomly generated. Whether a neuron is connected to another neuron in the same layer is randomly determined with a probability. In this paper, we propose a new RSNN structure called Skip-Connected Self-Recurrent SNNs (ScSr-SNNs) to offer a simple and structured approach for designing high-performance RSNNs and mitigating the training challenges resulted from random recurrent connections as in the prior works. The main contributions of this work are:

- We proposed the self-recurrent architecture for SNNs. The recurrence is only introduced by self-recurrent connections of individual spiking neurons, i.e., there exist no lateral connections between different neurons within a layer. We demonstrate that, with self-recurrent connections, SNNs are able to realize recurrent behaviors similar to those of more complex RSNNs while the error gradients can be more straightforwardly calculated due to the mostly feedforward nature of the network;

- We show that the skip connections can help the formation of recurrent structures, introduce more tunability, and thus improve performance.

- We rigorously derive the backpropagation algorithm that can handle self-recurrent connections and skip connections;

- We propose a new backpropagation (BP) method called backpropagated intrinsic plasticity (BIP) to further boost the performance of ScSr-SNNs by training intrinsic model parameters. Unlike standard intrinsic plasticity rules that adjust the neuron's intrinsic parameters according to neuronal activity, the proposed BIP method optimizes intrinsic parameters based on the backpropagated error gradient of a well-defined global loss function in addition to synaptic weight training.
- We implement the self-recurrent connections and BIP method not only to fully connected SNNs but also to spiking CNNs. Our experiments on the neuromorphic image dataset show that the proposed methods also benefit the image learning tasks on Spiking CNNs.

Based on challenging speech datasets TI46 (Liberman et al., 1991), neuromorphic speech dataset N-TIDIGITS (Anumula et al., 2018), neuromorphic image dataset Dvs-Gesture (Amir et al., 2017) and N-MNIST (Orchard et al., 2015), the proposed ScSr-SNNs can boost performance by up to 2.85% compared with other types of RSNNs trained by state-of-the-art BP methods.

2 Background

2.1 Spiking Neuron Model

SNNs employ more biologically plausible spiking neuron models than ANNs. In this work, the leaky integrate-and-fire (LIF) neuron model (Gerstner and Kistler, 2002) are adopted.

First, we notate the input spike train from pre-synaptic neuron j to post-synaptic neuron i as

$$s_j(t) = \sum_{t_j^{(f)}} \delta(t - t_j^{(f)}), \tag{1}$$

where δ is the Dirac delta function, $t_j^{(f)}$ the firing time of presynaptic neuron j.

Then, the incoming spikes are converted into an unweighted postsynaptic current (PSC) $a_j(t)$ through a synaptic model. The first order synaptic model (Gerstner and Kistler, 2002) is adopted and defined as

$$\tau_s \frac{a_j(t)}{dt} = -a_j(t) + s_j(t), \tag{2}$$

where τ_s is the synaptic time constant.

The neuronal membrane voltage $u_i(t)$ of post-synaptic neuron i at time t is given by

$$\tau_m \frac{du_i(t)}{dt} = -u_i(t) + R \sum_j w_{ij} a_j(t), \tag{3}$$

where R and τ_m are the effective leaky resistance and time constant of the membrane, w_{ij} the synaptic weight from neuron j to neuron i, and $a_j(t)$ the PSC induced by the spikes from neuron j.

Considering the discrete time steps simulation, we use the fixed-step first-order Euler method to discretize Eq.(3) to

$$u_i[t] = \theta_m u_i[t-1] + \sum_j w_{ij} a_j[t], \tag{4}$$

where we name $\theta_m = 1 - \frac{1}{\tau_m}$ the time constant factor. The ratio of R and τ_m is absorbed into the synaptic weights. Moreover, the firing output of the neuron i is expressed as

$$s_i[t] = H\left(u_i[t] - V_{th}\right), \tag{5}$$

where V_{th} is the firing threshold and $H(\cdot)$ the Heaviside step function.

2.2 SNNs Forward Pass

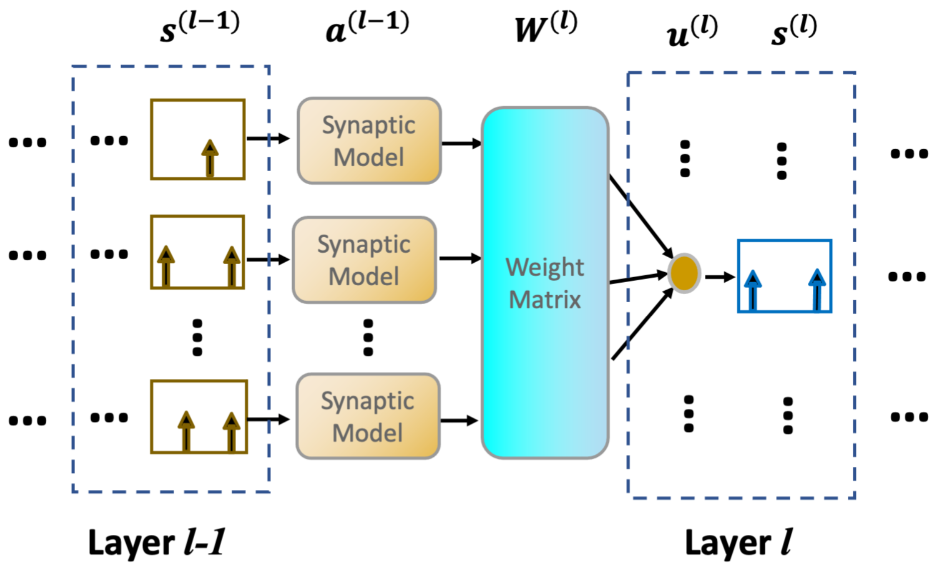

Figure 1: Forward evaluation pass of SNNs.

Figure 1 describes a feedforward SNN with the aforementioned LIF model and synaptic model. Consider a layer l with N_l neurons, we denote the presynaptic weights by $\boldsymbol{W}^{(l)} = \left[\boldsymbol{w}_1^{(l)}, \cdots, \boldsymbol{w}_{N_l}^{(l)}\right]^T$, where $\boldsymbol{w}_i^{(l)}$ is a column vector of weights from all the neurons in layer $l-1$ to the neuron i of layer l. In addition, we also denote PSCs from neurons in layer $l-1$ by $\boldsymbol{a}^{(l-1)}[t] = \left[a_1^{(l-1)}[t], \cdots, a_{N_{l-1}}^{(l-1)}[t]\right]^T$, output spike trains of the

$l-1$ layer by $\boldsymbol{s}^{(l-1)}[t] = \left[s_1^{(l-1)}[t], \cdots, s_{N_{l-1}}^{(l-1)}[t]\right]^T$, membrane potentials of the l layer neurons by $\boldsymbol{u}^{(l)}[t] = \left[u_1^{(l)}[t], \cdots, u_{N_l}^{(l)}[t]\right]^T$, where variables associated with the layer l have l as the superscript. Therefore, the network forward pass can be described as

$$\begin{aligned}
&\boldsymbol{a}^{(l-1)}[t] = (1 - \frac{1}{\tau_s})\boldsymbol{a}^{(l-1)}[t-1] + \boldsymbol{s}^{(l-1)}[t], \\
&\boldsymbol{u}^{(l)}[t] = \theta_m \boldsymbol{u}^{(l)}[t-1] + \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)}[t], \\
&\boldsymbol{s}^{(l)}[t] = H\left(\boldsymbol{u}^{(l)}[t] - V_{th}\right),
\end{aligned} \tag{6}$$

In the forward pass, the spike trains $\boldsymbol{s}^{(l-1)}[t]$ of the $l-1$ layer generate the (unweighted) PSCs $\boldsymbol{a}^{(l-1)}[t]$ according to the synaptic model. Then, $\boldsymbol{a}^{(l-1)}[t]$ are multiplied by the synaptic weights and passed onto the neurons of layer l. The integrated PSCs alter the membrane potentials and trigger the output spikes of the layer l neurons when the membrane potentials exceed the threshold.

3 ScSr-SNN Architecture

In this section, we first introduce the two components in ScSr-SNNs, the self-recurrent connections, and the skip connections. We illustrate that self-recurrent layers can realize recurrent behaviors similar to one fully connected recurrent layer while implementing extra local memory. Then, we introduce the skip-connected structure which can benefit neuronal dynamics and correlations.

3.1 Self-recurrent Architecture

The idea of connecting neurons back to themself in ANNs was introduced in (Mikolov et al., 2014). It presented that by introducing constraints on the recurrent weight matrix, RNNs can learn longer-term patterns with standard stochastic gradient descent. More specifically, it claimed that a kind of longer-term memory can be formed by making part of the recurrent weight matrix close to the identity matrix. It is the same as adding self-recurrent connections to part of hidden neurons. After that, (Li et al., 2018a) proposed an independently recurrent neural network (IndRNN) with self-recurrent connections where neurons in one layer are independent of each other. It is shown that multiple IndRNNs can be stacked to construct a deep network especially combined with residual connections over layers, and the deep ANNs can be trained robustly. In this work, we apply a similar idea to SNNs. We show that applying self-recurrence to SNNs can realize recurrent behaviors similar to those of more complex RSNNs. In addition, the self-recurrent connection can also improve local memory.

For simplicity, we adopt the method in (Huh and Sejnowski, 2018) to replace the threshold and synaptic model with a gate function $g()$. Thus, the PSC is defined as

$$\boldsymbol{a}^{(l)}[t] = g\left(\boldsymbol{u}^{(l)}[t]\right), \tag{7}$$

$g()$ reveals the relation between the membrane potential and the postsynaptic PSC. It is introduced here to simplify the analysis. In the experiments, we still use (6) to simulate the network activities.

In a single layer, the computation of each neuron is independent while neurons are correlated through multiple layers. Neurons in the same self-recurrent layer only recurrently connected to themselves and can be described as

$$\begin{aligned} \boldsymbol{u}^{(l)}[t] &= \theta_m \boldsymbol{u}^{(l)}[t-1] \\ &\quad + \boldsymbol{W}^{(l)} \boldsymbol{a}^{(l-1)}[t] + \boldsymbol{W}_s^{(l)} \circ g\left(\boldsymbol{u}^{(l)}[t-1]\right), \end{aligned} \tag{8}$$

where $\boldsymbol{W}_s^{(l)} = \left[w_{s,1}^{(l)}, \cdots, w_{s,N_l}^{(l)}\right]$ is a vector weight matrix of self-recurrent connections in layer l , $w_{s,i}^{(l)}$ the weight of neuron i's self-recurrent connection in layer l, and $\circ$ the Hadamard product.

Two self-recurrent layers can work similarly to the fully connected recurrent layer. To illustrate this, we approximately derive the behavior of two self-recurrent layers and compare it with a fully connected recurrent layer.

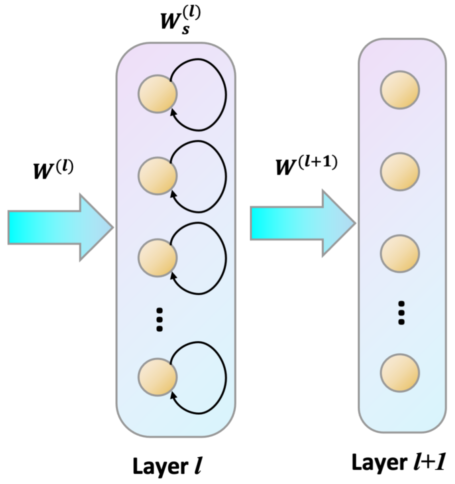

Figure 2: Self-recurrent pass of SNNs.

Figure 2 presents the layer l and layer $l+1$ of a network while layer l has self-recurrent connections. In this case, the behavior can be expressed as

$$
\begin{aligned}
\boldsymbol{u}^{(l)}[t] &= \theta_m \boldsymbol{u}^{(l)}[t-1] \\
&\quad + \boldsymbol{W}^{(l)} \boldsymbol{a}^{(l-1)}[t] + \boldsymbol{W}_s^{(l)} \circ g\left(\boldsymbol{u}^{(l)}[t-1]\right), \\
\boldsymbol{u}^{(l+1)}[t] &= \theta_m \boldsymbol{u}^{(l+1)}[t-1] + \boldsymbol{W}^{(l+1)} g\left(\boldsymbol{u}^{(l)}[t]\right).
\end{aligned} \tag{9}
$$

More aggressively, we approximate the $g(v)$ as a linear function $g(v) = hv$. h is a constant value to represent the proportion from membrane potential to the postsynaptic PSC. Then, the Eq.(9) can be combined as

$$
\begin{aligned}
\boldsymbol{u}^{(l+1)}[t] &= \theta_m \boldsymbol{u}^{(l+1)}[t-1] + \boldsymbol{W}^{(l)} \boldsymbol{a}^{(l-1)}[t] \\
&\quad + \boldsymbol{W}_1^{(l)} \boldsymbol{a}^{(l+1)}[t-1] + \boldsymbol{W}_2^{(l)} \boldsymbol{a}^{(l+1)}[t-2], \\
\boldsymbol{W}_1^{(l)} &= \boldsymbol{W}^{(l+1)} \left(\frac{\theta_m}{h} + \boldsymbol{W}_s^{(l)}\right) (\boldsymbol{W}^{(l+1)})^{-1} \\
\boldsymbol{W}_2^{(l)} &= -\theta_m \boldsymbol{W}_1^{(l)}.
\end{aligned} \tag{10}
$$

The behavior of two layers network illustrated in Eq.(10) is similar to the single fully connected recurrent layer which is described as

$$
\boldsymbol{u}^{(l)}[t] = \theta_m \boldsymbol{u}^{(l)}[t-1] + \boldsymbol{W}^{(l)} \boldsymbol{a}^{(l-1)}[t] + \boldsymbol{W}_r^{(l)} \boldsymbol{a}^{(l)}[t-1], \tag{11}
$$

where $\boldsymbol{W}_r^{(l)}$ is recurrent connection weights matrix.

Compared to the Eq.(11), two layers self-recurrent structure can be viewed as a single recurrent layer with two groups of recurrent connections which have the delay of 1 and 2 time steps. Eq.(10) also indicates the constraints of two layers self-recurrent structure that: 1. the recurrent weights $\boldsymbol{W}_1^{(l)}$ and $\boldsymbol{W}_2^{(l)}$ should be diagonalizable; 2.

$\boldsymbol{W}_2^{(l)}$ is the negative of $\boldsymbol{W}_1^{(l)}$ with the scalar θ_m.

Moreover, by including the reset mechanism and self-recurrent connections, Eq.(8) can be written as

$$\boldsymbol{u}^{(l)}[t] = \theta_m \boldsymbol{u}^{(l)}[t-1](1-\boldsymbol{s}^{(l)}[t-1]) + \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)}[t] + \boldsymbol{W}_s^{(l)} \circ \boldsymbol{a}^{(l)}[t-1]. \quad (12)$$

As shown in the first right term, a neuron will lose previous information after it fires and resets the membrane potential. With the self-recurrent connections, the output signals are passed back to the original neurons. From the neuronal perspective, positive feedback can be viewed as compensation for the information loss caused by the firing-and-resetting mechanism. On the other hand, negative feedback can control the neuron's firing activity and regulate the network dynamics.

To sum it up, a two layers self-recurrent structure behaves similarly to a fully connected recurrent layer. However, compared to the existing random connected RSNNs, ScSr-SNNs brings several benefits from the self-recurrent connections:

- ScSr-SNNs have a simpler but more structured architecture than the randomly generated RSNNs.
- ScSr-SNNs simplify the forward and backward computation in the recurrent structure. The error gradients can be calculated straightforwardly in ScSr-SNNs due to the mostly feedforward nature of the network. Thus, the straightforward calculation also cost less computational resources.
- Since the self connections are local within the layer, they can be not only applied to fully connected SNNs but also applicable to other structures like spiking CNNs.

3.2 Skip Connections

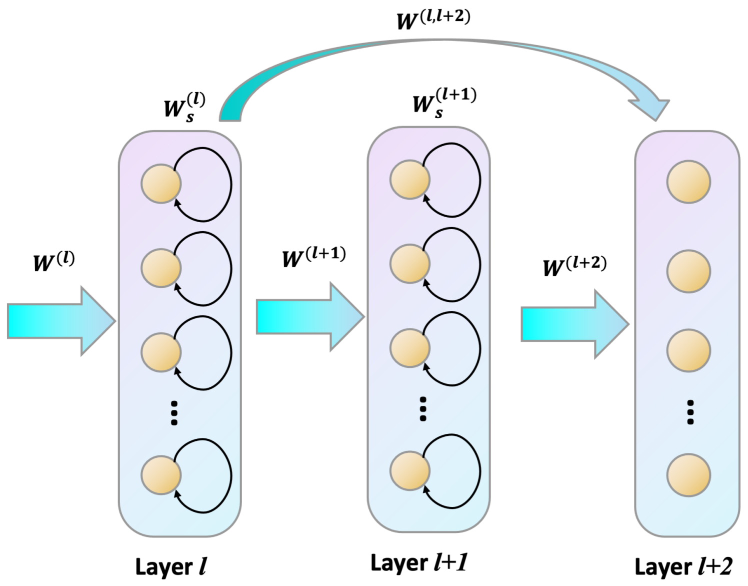

Figure 3: Skip-connected pass of SNNs.

As shown in Figure 3, we adopt the skip connections which directly connect two non-adjacent layers to enrich the network dynamics. The skip connections benefit the network for three reasons. First, two layers with self-recurrent connections can work similarly to a fully connected recurrent layer. Therefore, in Figure 3, we can view layer l and $l + 1$ as a recurrent layer while the $l + 1$ and $l + 2$ layers also form an equivalent recurrent layer. With the additional skip connections from layer l to $l + 2$, these two layers can also be treated as an approximate recurrent layer. The skip connections bring more possibilities and dynamics to form different structures. Second, the skip connections pass high-layer information to a certain layer and introduce more features. Third, during training, the skip connections form an alternative path for the calculation of backpropagated error gradient and provide additional tunability for the network. Owing to these three features, in Section 6.2, we demonstrate that the performance improvement and faster convergence speed are achieved by applying the skip connections.

4 Backpropagation for ScSr-SNNs

In this paper, we adopt the TSSL-BP method proposed in (Zhang and Li, 2020) to train the ScSr-SNNs. TSSL-BP is a BPTT based backpropagation method for SNNs. It captures the error backpropagation across two types of inter-neuron and intra-neuron dependencies and leads to state-of-the-art performance with extremely low latency. In the original paper, the TSSL-BP is proposed only for feedforward SNNs. In this paper, we extend its applicability to the self-recurrent connections and skip connections.

4.1 Loss Fuction

During the training, we denote the desired and the actual spike trains in the output layer by $\boldsymbol{d} = [\boldsymbol{d}[t_0], \cdots, \boldsymbol{d}[t_{N_t}]]$ and $\boldsymbol{s} = [\boldsymbol{s}[t_0], \cdots, \boldsymbol{s}[t_{N_t}]]$ where N_t is the number of the considered time steps, $\boldsymbol{d}[t]$ and $\boldsymbol{s}[t]$ the desired and actual firing events for all output neurons at time t, respectively. Note that, in this work, the output layer has neither self-recurrent connections nor skip connections.

The loss function L is defined by the square error for each output neuron at each time step:

$$L = \sum_{k=0}^{N_t} E[t_k] = \sum_{k=0}^{N_t} \frac{1}{2}((\epsilon * \boldsymbol{d})[t_k] - (\epsilon * \boldsymbol{s})[t_k])^2, \tag{13}$$

where $\epsilon(\cdot)$ is a kernel function which measures the so-called Van Rossum distance between the actual spike train and desired spike train.

We consider the backpropagation in three cases: 1. feedforward layer, 2. self-recurrent layer, 3. self-recurrent layer with skip-connections to a post layer.

4.2 Case 1: Feedforward Layer

As shown in Figure 1, when the layer l is feedforward layer, it could be the output layer or a hidden layer.

Eq.(6) reveals that the membrane potentials $\boldsymbol{u}^{(l)}$ of the neurons in layer l at time t have contribution to all future fires and losses. Therefore, the error gradient with respect to the presynaptic weights matrix $\boldsymbol{W}^{(l)}$ can be defined as

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{W}^{(l)}} &= \sum_{k=0}^{N_t} \frac{\partial E[t_k]}{\partial \boldsymbol{W}^{(l)}} = \sum_{k=0}^{N_t} \sum_{m=0}^{k} \frac{\partial E[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} \frac{\partial \boldsymbol{u}^{(l)}[t_m]}{\partial \boldsymbol{W}^{(l)}} \\ &= \sum_{m=0}^{N_t} \boldsymbol{a}^{(l-1)}[t_m] \sum_{k=m}^{N_t} \frac{\partial E[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]}.\end{aligned} \tag{14}$$

We use $\delta[t_m]$ to denote the backpropagated error at time t_m:

$$\boldsymbol{\delta}^{(l)}[t_m] = \sum_{k=m}^{N_t} \frac{\partial E[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} = \sum_{k=m}^{N_t} \frac{\partial E[t_k]}{\partial \boldsymbol{a}^{(l)}[t_k]} \frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]}. \tag{15}$$

If l is the output layer, from Eq.(13), the first term of Eq.(15) is given by

$$\frac{\partial E[t_k]}{\partial \boldsymbol{a}^{(l)}[t_k]} = (\epsilon * \boldsymbol{d})[t_k] - (\epsilon * \boldsymbol{s})[t_k]. \tag{16}$$

If l is a hidden layer, $\boldsymbol{\delta}^{(l)}[t_m]$ can be derived from the error δ of the post layer:

$$\begin{aligned}
\boldsymbol{\delta}^{(l)}[t_m] &= \sum_{j=m}^{N_t}\sum_{k=m}^{j} \frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} \left(\frac{\partial \boldsymbol{u}^{(l+1)}[t_k]}{\partial \boldsymbol{a}^{(l)}[t_k]} \frac{\partial E[t_j]}{\partial \boldsymbol{u}^{(l+1)}[t_k]} \right) \\
&= \sum_{k=m}^{N_t} \frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} \sum_{j=k}^{N_t} \boldsymbol{W}^{(l+1)} \frac{\partial E[t_j]}{\partial \boldsymbol{u}^{(l+1)}[t_k]} \\
&= (\boldsymbol{W}^{(l+1)})^T \sum_{k=m}^{N_t} \frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} \boldsymbol{\delta}^{(l+1)}[t_k].
\end{aligned} \tag{17}$$

Eq.(17) demonstrates that membrane potentials $\boldsymbol{u}^{(l)}$ of the neurons in layer l influence their (unweighted) PSCs $\boldsymbol{a}^{(l)}$ through fired spikes, and $\boldsymbol{a}^{(l)}$ further affects the membrane potentials $\boldsymbol{u}^{(l+1)}$ in the next layer.

The calculation of the term $\frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]}$ is one of the key contributions of (Zhang and Li, 2020). We do not repeat the derivatives here but treat it as a known term in this paper.

4.3 Case 2: Self-recurrent Layer

The structure of the self-recurrent layer is shown in Figure 2. Similar to the feedforward case, the weights of the incoming synapses can be calculated according to Eq.(14). Based on Eq.(8), the error gradient with respect to the weights of self-recurrent layer can be expressed as:

$$\frac{\partial L}{\partial \boldsymbol{W}_s^{(l)}} = \sum_{m=1}^{N_t} \boldsymbol{a}^{(l)}[t_m - 1]\boldsymbol{\delta}^{(l)}[t_m]. \tag{18}$$

Compared to the feedforward case, the main difference of the self-recurrent case comes from the derivation of $\boldsymbol{\delta}^{(l)}[t_m]$. Except the error signals from the post layer, we also need to take the errors backpropagated from self-recurrent connections into consideration. Thus, the error $\boldsymbol{\delta}^{(l)}[t_m]$ is calculated by:

$$\begin{aligned}
\boldsymbol{\delta}^{(l)}[t_m] &= \sum_{j=m}^{N_t}\sum_{k=m}^{j} \frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} \left(\frac{\partial \boldsymbol{u}^{(l+1)}[t_k]}{\partial \boldsymbol{a}^{(l)}[t_k]} \frac{\partial E[t_j]}{\partial \boldsymbol{u}^{(l+1)}[t_k]} \right) \\
&+ \sum_{j=m+1}^{N_t}\sum_{k=m}^{j} \frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} \left(\frac{\partial \boldsymbol{u}^{(l)}[t_k+1]}{\partial \boldsymbol{a}^{(l)}[t_k]} \frac{\partial E[t_j]}{\partial \boldsymbol{u}^{(l)}[t_k+1]} \right) \\
&= (\boldsymbol{W}^{(l+1)})^T \sum_{k=m}^{N_t} \frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} \boldsymbol{\delta}^{(l+1)}[t_k] \\
&+ \boldsymbol{W}_{\boldsymbol{s}}^{(l)} \circ \sum_{k=m}^{N_t-1} \frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} \boldsymbol{\delta}^{(l)}[t_k+1].
\end{aligned} \tag{19}$$

In Eq.(19), the first term is the same as Eq.(17) which represent the error backpropagated from the post layer. The second term is caused by self-recurrent connections. It reveals that membrane potentials $\boldsymbol{u}^{(l)}$ of the neurons in layer l influence their (unweighted) PSCs $\boldsymbol{a}^{(l)}$ through fired spikes, and $\boldsymbol{a}^{(l)}$ further affect the membrane potentials of $\boldsymbol{u}^{(l)}$ at the next time step.

4.4 Case 3: Self-recurrent Layer with skip connections

As shown in Figure 3, we suppose layer l has self-recurrent connections. In addition, it also connects to layer $l+1$ the same as the feedforward layer and layer $l+2$ through the skip connections. The changes of BP method still come from the derivation of $\boldsymbol{\delta}^{(l)}[t_m]$. In this case, the output (unweighted) PSCs $\boldsymbol{a}^{(l)}$ of layer l further directly affects the membrane potentials of $\boldsymbol{u}^{(l+2)}$ at layer $l+2$.

Similar to the derivation of Eq.(19), an additional term should be added when the skip connections are taken into account. Therefore, the $\boldsymbol{\delta}^{(l)}[t_m]$ can be derived as

$$\begin{aligned}
\boldsymbol{\delta}^{(l)}[t_m] &= (\boldsymbol{W}^{(l+1)})^T \sum_{k=m}^{N_t} \frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} \boldsymbol{\delta}^{(l+1)}[t_k] \\
&+ \boldsymbol{W}_{\boldsymbol{s}}^{(l)} \circ \sum_{k=m}^{N_t-1} \frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} \boldsymbol{\delta}^{(l)}[t_k+1] \\
&+ (\boldsymbol{W}^{(l,l+2)})^T \sum_{k=m}^{N_t} \frac{\partial \boldsymbol{a}^{(l)}[t_k]}{\partial \boldsymbol{u}^{(l)}[t_m]} \boldsymbol{\delta}^{(l+2)}[t_k],
\end{aligned} \tag{20}$$

where $\boldsymbol{W}^{(l,l+2)}$ represents weights of skip connections from layer l to layer $l+2$.

4.5 Backpropagation Based Intrinsic Plasticity

Apart from the ScSr-SNNs, we also propose a new backpropagation (BP) method called backpropagated intrinsic plasticity (BIP) to further boost the performance of ScSr-SNNs by training intrinsic model parameters.

Intrinsic plasticity (IP) is a widely used self-adaptive mechanism that maintains homeostasis and shapes the dynamics of neural circuits. In short, IP is an inner neuronal mechanism that adjusts the neuron's activity. IP has been observed in various biological neurons (Marder et al., 1996; Baddeley et al., 1997; Desai, Rutherford, and Turrigiano, 1999). However, there's still no conclusion about how the IP exactly works.

The most straight forward idea of IP is boosting the neuron's activity if the neuron rarely fires and depressing the neuron if it fires too frequently. In SNNs, there are mainly two ways to apply the IP method. First, the most famous way is called the dynamic threshold. It heuristically increases the neuronal firing threshold when the neuron spikes and exponentially decays the firing threshold during the rest of the time (Lazar, Pipa, and Triesch, 2007; Bellec et al., 2018; Li et al., 2018b). The mechanism of the dynamic threshold method is the same as the adaptive LIF (ALIF) neuron model in (Bellec et al., 2018). The ALIF neuron adapts the firing rate by adjusting the threshold. On the other hand, (Triesch, 2005) first proposed a mathematical IP rule on ANNs to adjust the neuron so that its Kullback–Leibler (KL) divergence from an exponential distribution to the actual output firing rate distribution is minimized. After that several works tried to apply this idea to SNNs by adjusting the neuronal parameters to optimize the neuron's output firing rate distribution (Schrauwen et al., 2008; Li and Li, 2013; Zhang and Li, 2019a).

In (Zhang and Li, 2019a), the IP method trains the time constant to minimize the loss which is defined by the KL divergence between the desired and actual firing rate distributions. In this work, we also train the time constant of the neuron. However, the typical IP method targets to adjust network firing rates. It may contradict the objective of the BP rule that tries to minimize the output loss and hinder the training. Therefore, we propose a backpropagated intrinsic plasticity (BIP) method to joint both the IP and BP methods for the same goal, the output loss. More specifically, unlike the typical IP rule which is unsupervised, we apply the IP rule together with the BP rule. During backpropagation, the error of each neuron is either directly from the loss function (the output layer) or backpropagated from the post layer. Once the error is obtained, the neuron's time constant can be updated according to the error by the BIP method while the corresponding weights are updated by the BP method.

According to Eq.(6), the neuronal parameters θ_m of each neuron are trained with the backpropagated errors. We rewrite the forward pass with self-recurrent connections as

$$\boldsymbol{u}^{(l)}[t] = \boldsymbol{\theta}^{(l)} \circ \boldsymbol{u}^{(l)}[t-1] + \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)}[t] + \boldsymbol{W}_s^{(l)} \circ \boldsymbol{a}^{(l)}[t-1], \tag{21}$$

where $\boldsymbol{\theta}^{(l)} = \left[\theta_1^{(l)}, \cdots, \theta_{N_l}^{(l)}\right]^T$ is the vector of the time constant factor defined in (4).

Similar to Eq.(14), the $\boldsymbol{\theta}^{(l)}$ is updated based on the backpropagated error $\boldsymbol{\delta}^{(l)}$. The error gradient can be described as

$$\frac{\partial L}{\partial \boldsymbol{\theta}^{(l)}} = \sum_{m=1}^{N_t} \boldsymbol{u}^{(l)}[t_m - 1]\boldsymbol{\delta}^{(l)}[t_m]. \tag{22}$$

Figure 4 summarizes the forward pass and backward pass of the proposed ScSr-SNNs with BIP method at the single-neuron level. In the figure, neurons in layer l have skip connections to neurons in layer $l+q$. Neurons in layer $l-p$ and $l+1$ are also connected through skip connections. As shown, at the spatial level, the neuron communicates with neurons of different layers via regular feedforward connections and skip connections. However, within the same layer, the neuron only depends on itself at the temporal level via intrinsic parameter and self-recurrent connection. This simple structure results in efficient learning with rich neural dynamics.

In addition, at the spatial level, the self-recurrent connections and intrinsic parameters are independent without connecting to other neurons. Therefore, both self-recurrent connections and BIP can be easily applied to different SNNs like Spiking CNNs. In Section 5.5, we demonstrate that the proposed methods can also improve the performance of Spiking CNNs on the neuromorphic image dataset.

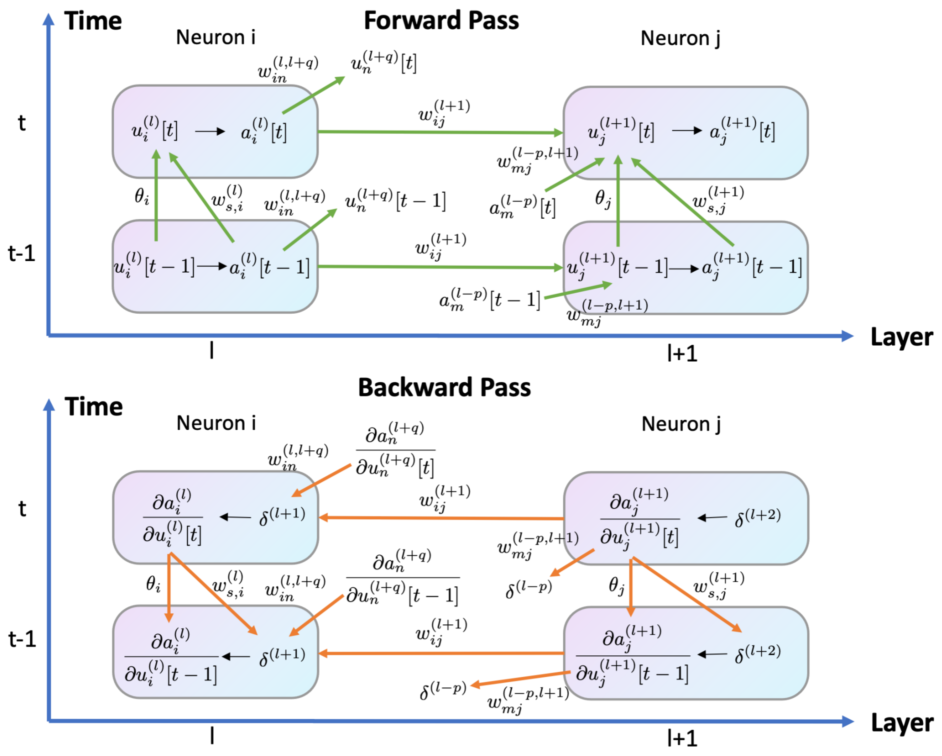

Figure 4: Forward and backward pass of ScSr-SNNs with BIP method.

5 Experiments and Results

5.1 Experimental Settings

In this section, we test the proposed ScSr-SNNs with BIP method on five datasets, the speech dataset TI46-Alpha (Liberman et al., 1991), TI46-Digits (Liberman et al., 1991), neuromorphic speech dataset N-TIDIGITS (Anumula et al., 2018), neuromorphic image dataset DvsGesture (Amir et al., 2017), and N-MNIST (Orchard et al., 2015). We compare ScSr-SNNs with several state-of-the-art results of different structures including feedforward SNNs, RSNNs, Liquid State Machine(LSM), spiking CNNs, and ANNs.

All reported experiments are conducted on an NVIDIA Titan XP GPU. The implementation of ScSr-SNNs is on the Pytorch framework (Paszke et al., 2019). The simulation step size is set to 1 ms. The parameters like threshold and learning rate are empirically tuned. Table 1 lists the typical values adopted for each dataset.

Parameter	**TI46-Alpha**	**TI46-Digit**	**N-TIDIGITS**	**DvsGesture**	**N-MNIST**
τ_m	16 ms	16 ms	64 ms	64 ms	16 ms
τ_s	8ms	8 ms	8 ms	8 ms	4 ms
learning rate	0.0005	0.0005	0.0002	0.0001	0.0005
Threshold V_{th}	1 mV	1 mV	1 mV	1 mV	1 mV
Batch Size	50	50	50	5	50
Time steps	100	100	300	300	100
Epochs	400	400	400	100	100

Table 1: Parameters settings.

The weight matrices of self-recurrent connections and skip connections are initialized following the normal distribution with a mean of 0 and std of 1. For the BIP method, the time constant of membrane potential in Table 1 is applied as the initial value. No axon and synaptic delay or refractory period are adopted in the feedforward pass whereas the self-recurrent connections have 1 ms delay. No dropout or normalization is applied. Adam (Kingma and Ba, 2014) is adopted as the optimizer.

For practical issues such as desired output selection, warm-up mechanism, the boundary of derivatives, we follow the solutions in (Zhang and Li, 2020).

In the results, the mean and standard deviation (std) reported are obtained by repeating the experiments five times. Moreover, we also separately apply the proposed self-recurrent structure, skip-connected structure, and BIP method to the networks to reveal their individual effects of boosting performance. For all the results reported in this paper, we follow Table 2 to name the structures and methods.

Name	Structures
Sr-SNNs	SNNs with self-recurrent connections in each hidden layer
Sr-SNNs-BIP	Sr-SNNs with BIP in each hidden layer
ScSr-SNNs	Sr-SNNs with skip connections between two layers
ScSr-SNNs-BIP	ScSr-SNNs with BIP in each hidden layer

Table 2: Structures.

5.2 TI46-Alpha

TI46-Alpha is the full alphabets subset of the TI46 Speech corpus (Liberman et al., 1991) with spoken English alphabets audios from 16 speakers. There are 4142 and 6628 spoken English examples in 26 classes for training and testing, respectively. The continuous temporal speech waveforms are preprocessed by Lyon's ear model (Lyon, 1982). Each speech is encoded into 78 channels with real-valued intensity. In this case, we directly apply the preprocessed real-valued results to the input layer. The sample rate of this dataset is 12.5 kHz. The decimation factor of Lyon's ear model is 125. Each preprocessed sample is simulated for 100 time steps.

In Table 3, we compare the proposed structures with several existing results. In (Wijesinghe et al., 2019), the state vector of the reservoir is used to train the single readout layer using BP. Its result shows that only training the single readout layer of a recurrent LSM is inadequate for this challenging task, demonstrating the necessity of a more structured and deep SNN. (Zhang and Li, 2019b) demonstrates the best-reported performance of TI46-Alpha. It has one recurrent layer and two feedforward layers trained with the ST-RSBP method. However, both recurrent networks in (Wijesinghe et al., 2019) and (Zhang and Li, 2019b) are randomly generated. We show that, with the proposed ScSr-SNNs-BIP method, we can achieve a performance of 96.20% with a mean of 95.76% and a standard deviation of 0.29% which is 2.85% better than the best-reported result.

In the table, we also demonstrate that the three proposed methods, the self-recurrent connections, skip connections, and BIP method, can all improve performance when they are applied separately. In addition, with the same training rule TSSL-BP, our proposed methods can boost 3.06% performance on the same network size compared to (Zhang and Li, 2020).

Method	Network Size	Mean	Std	Best
HM2BP (Jin, Zhang, and Li, 2018)	400-400	89.83%	0.71%	90.60%
Liquid Ensembles (Wijesinghe et al., 2019)	LSM: R2000[a]	-	-	85.5%
ST-RSBP (Zhang and Li, 2019b)	400-R400-400	93.06%	0.21%	93.35%
TSSL-BP (Zhang and Li, 2020)	400-400-400	93.01%	0.13%	93.14%
Sr-SNNs	400-400-400	94.17%	0.31%	94.62%
Sr-SNNs-BIP	400-400-400	94.90%	0.16%	95.04%
ScSr-SNNs	400-400-400[b]	94.98%	0.17%	95.13%
ScSr-SNNs-BIP	400-400-400[b]	95.76%	0.29%	**96.20%**

[a] R represent recurrent layer.
[b] Skip connections from first layer to third layer.

Table 3: Performances on TI46-Alpha

5.3 TI46-Digit

TI46-Digits is the full digits subset of the TI46 Speech corpus (Liberman et al., 1991). It contains 1594 training examples and 2542 testing examples of 10 utterances for each of digits ”0” to ”9” spoken by 16 different speakers. The same preprocessing steps of TI46-Alpha is adopted. In the Table 4, SpiLinC proposed in (Srinivasan, Panda, and Roy, 2018) is an LSM with multiple reservoirs in parallel. Weights between inputs and reservoirs are trained using STDP. The excitatory neurons in the reservoir are tagged with the classes for which they spiked at the highest rate during training. The neurons with the same tag are grouped accordingly for inference. As shown in table, with the proposed methods, We can achieve 99.69% with a mean of 99.52% and a standard deviation of 0.13%. The performance is 0.3% better than the best-reported results in (Zhang and Li, 2019b) with only half of the network size. Moreover, our proposed method also achieves 6 times latency reduction. It proves that the proposed methods can improve the network dynamic when processing sequential signals and thus boost the performance.

Method	Network Size	Mean	Std	Best
SpiLinC (Srinivasan, Panda, and Roy, 2018)	LSM: R3200	-	-	86.66%
Liquid Ensembles (Wijesinghe et al., 2019)	LSM: R5000	-	-	97.25%
ST-RSBP (Zhang and Li, 2019b)	200-R200-200	99.25%	0.13%	99.39%
TSSL-BP (Zhang and Li, 2020)	100-100-100	98.49%	0.18%	98.66%
Sr-SNNs	100-100-100	99.01%	0.20%	99.17%
Sr-SNNs-BIP	100-100-100	99.08%	0.09%	99.31%
ScSr-SNNs	100-100-100[a]	99.16%	0.08%	99.35%
ScSr-SNNs-BIP	100-100-100[a]	99.52%	0.13%	**99.69%**

[a] Skip connections from first layer to third layer.

Table 4: Performances on TI46-Digits

5.4 N-TIDIGITS

The N-Tidigits (Anumula et al., 2018) is the neuromorphic version of the well-known speech dataset Tidigits (Leonard and Doddington, 1993). It consists of recorded spike responses from a 64-channel CochleaAMS1b sensor in response to the original audios. The dataset includes both single digits and connected digit sequences with a vocabulary consisting of 11 digits including “oh,” “zero” and the digits “1” to “9”. In this experiment, There are 55 male and 56 female speakers with 2,475 single digit examples for training and the same number of examples for testing. In the original dataset, each sample lasts about $0.9s$. We reduce the time resolution to speed up the simulation. Thus, the preprocessed samples only have about 300 time steps. We determine that a channel has a spike at a certain time step in the preprocessed sample if there's at least one spike among the corresponding several time steps of the original sample.

For N-TIDIGITS dataset, we not only compare the performance with previous work on RSNNs but also with the well-known RNNs in non-spiking networks such as Gated Recurrent Unit (GRU) and Long-Short Term Memory (LSTM). The GRU and LSTM networks are trained by the non-spiking BP method. As shown in Table 5, the proposed methods achieve 95.07% with a mean of 94.79% and a standard deviation of 0.22% which is the best results compared to state-of-the-art performances. Note that although the number of neurons in the GRU and LSTM networks are less than our proposed methods. The number of tunable parameters is nearly the same.

Structures	Network Size	Mean	Std	Best
GRU (Anumula et al., 2018)	G200-G200-100[a]	-	-	89.69%
LSTM (Anumula et al., 2018)	L250-L250[b]	-	-	90.90%
ST-RSBP (Zhang and Li, 2019b)	400-R400-400	93.63%	0.27%	93.90%
TSSL-BP (Zhang and Li, 2020)	400-400-400	89.55%	0.28%	89.85%
Sr-SNNs	400-400-400	93.77%	0.25%	94.02%
Sr-SNNs-BIP	400-400-400	94.19%	0.14%	94.35%
ScSr-SNNs	400-400-400[c]	94.25%	0.18%	94.46%
ScSr-SNNs-BIP	400-400-400[c]	94.79%	0.22%	**95.07%**

[a] G represents Gated Recurrent Unit (GRU) layer.
[b] L represents long-short term memory (LSTM) layer.
[c] Skip connections from first layer to third layer.

Table 5: Performances on N-TIDIGITS

5.5 DvsGesture

The DvsGesture dataset (Amir et al., 2017) have recordings of 29 different individuals (subjects) performing hand and arm gestures. 11 hand and arm gestures are recorded using a DVS camera under three different lighting conditions for one subject in each trail. There are 122 trails in total. Samples from the first 23 subjects are used for training and the last 6 subjects are used for testing. The problem is to classify the action sequence video into an action label. Each action (sample) is recorded for about 6 s. Similar to the preprocessing steps in (Shrestha et al., 2017), only the first 1.5 second of action video for each class are used to classify the actions. The temporal resolution of simulation is 5 ms which means it takes 300 time steps for each sample.

Since a neuron with self-recurrent connection and the BIP method only need to communicate itself within the layer, these two proposed methods are also applicable to spiking CNNs. As shown in Table 6, our proposed method on spiking CNNs can achieve the performance about 2% better than the best results of existing works with the same or similar network sizes. Note that the self-recurrent connections and BIP only increase 2 additional tunable parameters for each neuron. Therefore, the total number of tunable parameters keeps at the same level.

Method	Network	Accuracy
TrueNorth (Amir et al., 2017)	CNN-based 16 layers	91.77%
SLAYER (Shrestha et al., 2017)	CNN-based 8 layers	93.64%
RNN (He et al., 2020)	CNN-based	92.01%
LSTM (He et al., 2020)	CNN-based	93.75%
SNN (He et al., 2020)	CNN-based 8 layers	93.40%
Sr-SNNs-BIP	CNN-based 8 layers	**95.49%**

Table 6: Performances on DvsGesture

5.6 N-MNIST

The N-MNIST (Orchard et al., 2015) dataset is a neuromorphic version of the MNIST dataset generated by tilting a Dynamic Version Sensor (DVS) in front of static digit images on a computer monitor. The movement induced pixel intensity changes at each location is encoded as spike trains. Each sample contains two kinds of ON and OFF events spike events which represent the increment and decrement of the intensity. The recorded sample of the N-MNIST is a spatio-temporal pattern with $34 \times 34 \times 2$ spike sequences lasting for 300ms with a resolution of 1us. In the experiments, we follow the preprocessing steps in (Zhang and Li, 2020) to reduce the time resolution of the N-MNIST samples by 3000 times to speed up the simulation. Therefore, the preprocessed samples only have about 100 time steps. Whether a channel has a spike at a certain time step of the preprocessed sample is determined by if there's at least one spike among the corresponding 3000 time steps of the original sample. In addition, in the experiment, only the first 30 time steps of each sample are used for training and inference.

Same as the experiment on DvsGesture, a spiking CNN network is adopted for the simulation on N-MNIST. The self-recurrent connections and the BIP method are applied to the two convolution layers. As shown in Table 7, the proposed method achieves 99.32% accuracy which outperform the state-of-the-art performance in (Zhang and Li, 2020). According to the experimental results on DvsGesture and N-MNIST, it implies that the proposed method can also benefit the learning of Spiking CNNs.

Method	Network	Accuracy	Time Steps
HM2BP (Jin, Zhang, and Li, 2018)	400-400	98.88%	600
SLAYER (Shrestha et al., 2017)	12C5-P2-64C5-P2	99.22%	300
TSSL-BP (Zhang and Li, 2020)	12C5-P2-64C5-P2	92.28%	30
Sr-SNNs-BIP	12C5-P2-64C5-P2	**99.32%**	30

Table 7: Performances on N-MNIST

6 Analysis

6.1 Effects of Self-recurrent Connections

In the proposed method, the self-recurrent connections are constructed so that the network can realize recurrent behaviors similar to those of more complex RSNNs while the error gradients can be calculated more straightforwardly. The weights of self-recurrent connections are randomly initialized following the normal distribution with the mean of 0 and std of 1. By the BP method, the weights of the connections can be trained to minimize the loss function.

We take the well-trained ScSr-SNN on TI46-Alpha as an example. The network contains 3 layers with 400 neurons in each layer. We record the weights of self-recurrent connections after training. Figure 5 shows the distribution of the 1200 self-recurrent weights. As shown, there're about 60% positive self-recurrent connections. From the network perspective, the well-trained self-recurrent weights guarantee the complex dynamics of the RSNNs and minimize the output loss. From the single neuron level, on one hand, the positive self-recurrent connections refresh the information of the neuron and thus maintain the single-neuron memory. On the other hand, the negative self-recurrent connection depresses the neuron and can be considered as a regulation for the neuron's activity.

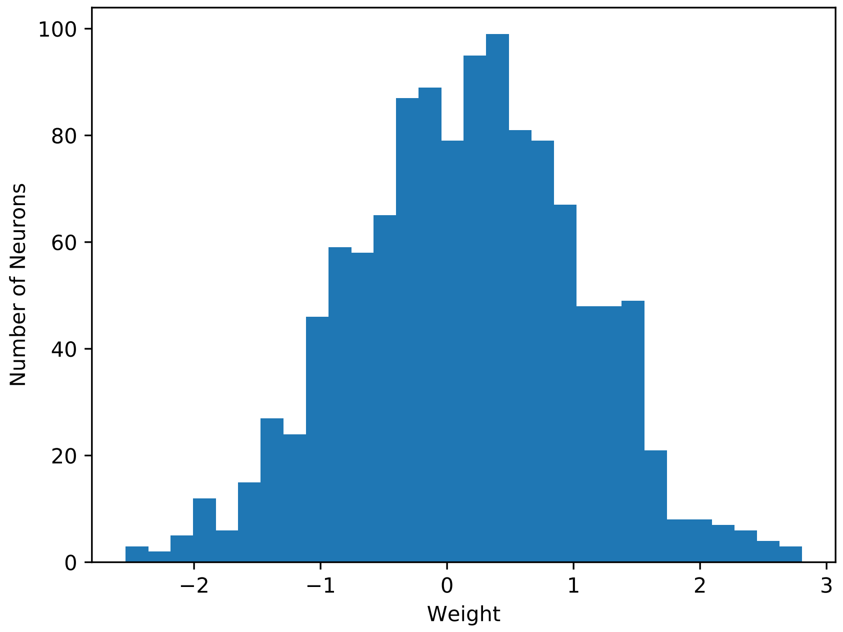

Figure 5: Weights distribution of well-trained self-recurrent connections

6.2 Effects of Skip Connections

The skip connections mainly play three roles. First, the skip connections combined with self-recurrent connections introduce additional recurrent structures and thus further enhance neural dynamics. Second, the skip connections pass high-level information to a certain layer and introduce more features. Finally, the skip connections provide an alternative path for the gradient.

When the network goes deeper, the effects of skip connections become even stronger. We compare two structures, ScSr-SNNs-BIP and Sr-SNNs-BIP, on the TI-Alpha dataset. The networks have six hidden layers with 100 neurons in each layer. The only difference between these two networks is that ScSr-SNNs-BIP has skip connections from the first layer to the fifth layer. As shown in Figure 6, with the skip connections, the performance can be improved for more than 1.5%. Apart from the performance improvement, the ScSr-SNNs-BIP network can also achieve the same loss with up to 50 fewer epochs than Sr-SNNs-BIP. The network dynamics and more features introduced by skip connections lead to the performance improvement. In the meanwhile, the faster convergence is benefited from the additional path to pass the backpropagated errors.

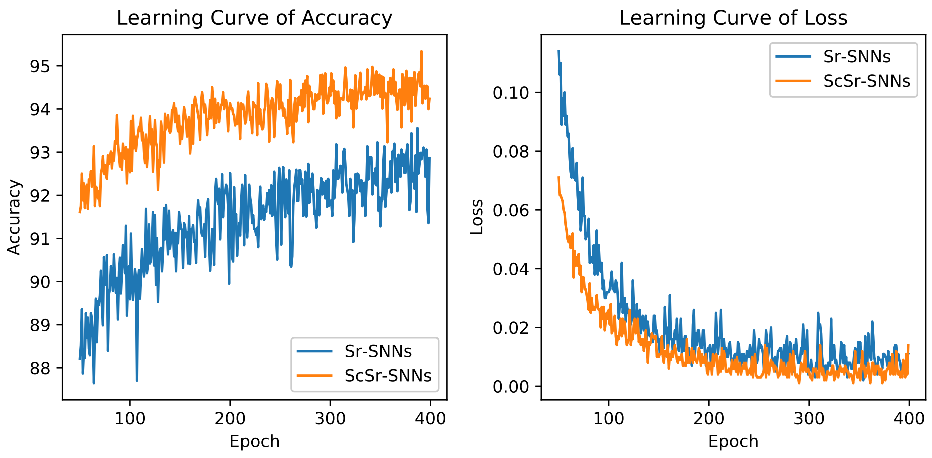

Figure 6: Effects of skip connections

When the network goes deeper, there are many possible ways to connect non-adjacent layers with skip connections. It's necessary to know if more skip connections mean better performance. We conduct a few experiments on the TI46-Alpha dataset. The network has 4 hidden layers with 200 neurons in each layer. As shown in Table 8, we compare two kinds of connections. The one skip connections network only has skip connections from the first hidden layer to the third hidden layer. The two skip connections means the first hidden layer to the third hidden layer and the second hidden layer to the fourth hidden layer are both connected. The results show that the proposed methods with only skip connections from the first layer to the third layer achieve the best result no matter whether the BIP method is applied. This may be because one skip connections can enrich the network dynamics while multiple skip connections lead to the instability of the network and also cause complexity for the proposed BP method to well train the network.

Structures	Best
Feedforward SNNs	92.08%
ScSr-SNNs: one skip connections	94.27%
ScSr-SNNs: two skip connections	93.56%
ScSr-SNNs-BIP: one skip connections	**95.04%**
ScSr-SNNs-BIP: two skip connections	94.76%

Table 8: Performances on TI46-Alpha with more layers

6.3 Computational Efficiency

Owing to the simple structure of self-recurrent connections, the proposed structure has lower computational complexity compared to fully connected RSNNs. For a layer with n neurons, the proposed self-recurrent connections and BIP method only introduce $2n$ more tunable parameters. However, for a fully connected recurrent layer, it has n^2 parameters in the recurrent weight matrix. During simulation, the time cost mainly comes from the error backpropagated through recurrent connections, because the gradient must be calculated time step by time step. Thus, the proposed structure can be more efficient than fully connected RSNNs.

More specifically, we use the networks in the experiments of TI46-Alpha as an example. Each network has 3 hidden layers with 400 neurons in each layer. In addition, the inputs have 78 channels and the output layer has 26 neurons. Therefore, the number of tunable parameters of the feedforward network is $361,600$. For the proposed method, Sr-SNNs have $362,800$ parameters and Sr-SNNs-BIP have $364,000$ parameters. Thus, by comparing the number of parameters, we can conclude that there's almost no computational overhead by applying self-recurrent connections and the BIP method. After implementing the skip connections, ScSr-SNNs have $522,800$ parameters and ScSr-SNNs-BIP have $524,000$ parameters. The largely increased number of parameters may result in additional computational cost. Thus, a trade-off between the cost and performance should be taken into consideration when applying the skip-connections.

Moreover, compared to existing methods, the proposed method can train networks over a short temporal window of a few time steps (low latency) which leads to more efficient training. For TI-Alpha, we use 100 time steps to simulate each sample while (Wijesinghe et al., 2019) requires 600 time steps and (Zhang and Li, 2019b) need 700 time steps.

6.4 Firing Activity

When considering the implementation of the proposed structure on neuromorphic hardware, low power consumption becomes an important constraint. In most cases, the power consumption is highly related to the firing rate of the whole network. A good network should not only demonstrate decent performance but also keeps relatively low firing activities.

To demonstrate the firing sparsity of the proposed networks, we select the well-trained ScSr-SNN on TI46-Alpha. We randomly apply one sample to the well-trained network from the test set. The firing rate of each hidden neuron is recorded. As shown in Figure 7, about 55% neurons are silent while 26% neurons have firing rates less than

10%. In addition, only less than 3% neurons have firing rates that are higher than 40%. Therefore, from the firing activity point, the proposed method also demonstrates decent computational efficiency.

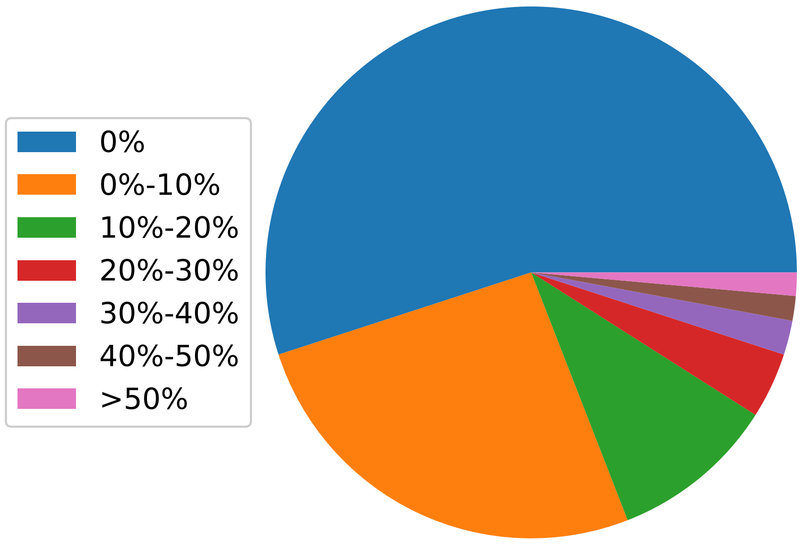

Figure 7: Firing acitivity of well-trained ScSr-SNNs-BIP

7 Conclusion

In this work, we propose the novel Skip-Connected Self-Recurrent SNNs (ScSr-SNNs) with backpropagated intrinsic plasticity (BIP). The benefits of each proposed method can be summarized as below:

- Self-recurrent Connections: (1) With the self-recurrent connections, the network can realize recurrent behaviors similar to those of more complex RSNNs. (2) A neuron loses all its information after generating a spike and resetting membrane potential to 0. With the positive self-recurrent connections, such information will be obtained by the neuron again. This process can maintain the single-neuron memory. (3) The structure is easy to interpret due to the independence of neurons in each layer. (4) It simplifies the forward and backward computation in the recurrent structure.

- Skip connections: (1) The skip connections combined with self-recurrent connections introduce additional recurrent structure as demonstrated in the paper. Thus, it further enhances network dynamics. (2) The skip connections pass high-layer information to a certain layer and introduce more features. (3) The skip connections provide an alternative path for the gradient.

- BIP: The BIP together with the BP method can improve the learning performance by concentrating on the output loss.

Moreover, the proposed methods can be easily applied to deep networks such as deep RSNNs and spiking CNNs.

On the speech datasets and neuromorphic datasets, we demonstrate that the proposed ScSr-SNNs with the BIP method significantly outperform the existing works including the state-of-the-art BP method in (Zhang and Li, 2019b). These great boosts also reveal the effectiveness of the proposed structures and methods.

This work has been prototyped based on the widely adopted Pytorch framework and will be made available to the public. We believe the proposed structures and BIP method will benefit the brain-inspired computing community from both a structural and algorithmic perspective.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Award Number DE-SC0021319, and the National Science Foundation (NSF) under Award Number 1948201.

Disclaimer

References

Amir, A.; Taba, B.; Berg, D.; Melano, T.; McKinstry, J.; Di Nolfo, C.; Nayak, T.; Andreopoulos, A.; Garreau, G.; Mendoza, M.; et al. 2017. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7243–7252.

Anumula, J.; Neil, D.; Delbruck, T.; and Liu, S.-C. 2018. Feature representations for neuromorphic audio spike streams. *Frontiers in neuroscience* 12: 23.

Baddeley, R.; Abbott, L. F.; Booth, M. C.; Sengpiel, F.; Freeman, T.; Wakeman, E. A.; and Rolls, E. T. 1997. Responses of neurons in primary and inferior temporal visual cortices to natural scenes. *Proceedings of the Royal Society of London B: Biological Sciences* 264(1389): 1775–1783.

Bellec, G.; Salaj, D.; Subramoney, A.; Legenstein, R.; and Maass, W. 2018. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, 787–797.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Desai, N. S.; Rutherford, L. C.; and Turrigiano, G. G. 1999. Plasticity in the intrinsic excitability of cortical pyramidal neurons. *Nature neuroscience* 2(6): 515.

Gerstner, W.; and Kistler, W. M. 2002. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

Ghani, A.; McGinnity, T. M.; Maguire, L. P.; and Harkin, J. 2008. Neuro-inspired speech recognition with recurrent spiking neurons. In *International Conference on Artificial Neural Networks*, 513–522. Springer.

Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, 6645–6649. IEEE.

He, W.; Wu, Y.; Deng, L.; Li, G.; Wang, H.; Tian, Y.; Ding, W.; Wang, W.; and Xie, Y. 2020. Comparing SNNs and RNNs on Neuromorphic Vision Datasets: Similarities and Differences. *arXiv preprint arXiv:2005.02183* .

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8): 1735–1780.

Huh, D.; and Sejnowski, T. J. 2018. Gradient descent for spiking neural networks. In *Advances in Neural Information Processing Systems*, 1433–1443.

Jaeger, H. 2001. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148(34): 13.

Jin, Y.; and Li, P. 2016. AP-STDP: A novel self-organizing mechanism for efficient reservoir computing. In *2016 International Joint Conference on Neural Networks (IJCNN)*, 1158–1165. IEEE.

Jin, Y.; Zhang, W.; and Li, P. 2018. Hybrid macro/micro level backpropagation for training deep spiking neural networks. In *Advances in Neural Information Processing Systems*, 7005–7015.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Lazar, A.; Pipa, G.; and Triesch, J. 2007. Fading memory and time series prediction in recurrent networks with different forms of plasticity. *Neural Networks* 20(3): 312–322.

Leonard, R. G.; and Doddington, G. 1993. Tidigits speech corpus. *Texas Instruments, Inc* .

Li, C.; and Li, Y. 2013. A spike-based model of neuronal intrinsic plasticity. *IEEE Transactions on Autonomous Mental Development* 5(1): 62–73.

Li, S.; Li, W.; Cook, C.; Zhu, C.; and Gao, Y. 2018a. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5457–5466.

Li, X.; Wang, W.; Xue, F.; and Song, Y. 2018b. Computational modeling of spiking neural network with learning rules from STDP and intrinsic plasticity. *Physica A: Statistical Mechanics and its Applications* 491: 716–728.

Liberman, M.; Amsler, R.; Church, K.; Fox, E.; Hafner, C.; Klavans, J.; Marcus, M.; Mercer, B.; Pedersen, J.; Roossin, P.; Walker, D.; Warwick, S.; and Zampolli, A. 1991. TI 46-Word LDC93S9.

Lotfi Rezaabad, A.; and Vishwanath, S. 2020. Long Short-Term Memory Spiking Networks and Their Applications. In *International Conference on Neuromorphic Systems 2020*, 1–9.

Lyon, R. 1982. A computational model of filtering, detection, and compression in the cochlea. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, volume 7, 1282–1285. IEEE.

Maass, W.; Natschläger, T.; and Markram, H. 2002. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation* 14(11): 2531–2560.

Maes, A.; Barahona, M.; and Clopath, C. 2020. Learning spatiotemporal signals using a recurrent spiking network that discretizes time. *PLoS computational biology* 16(1): e1007606.

Marder, E.; Abbott, L.; Turrigiano, G. G.; Liu, Z.; and Golowasch, J. 1996. Memory from the dynamics of intrinsic membrane currents. *Proceedings of the national academy of sciences* 93(24): 13481–13486.

Mikolov, T., Joulin, A., Chopra, S., Mathieu, M., and Ranzato, M. (2014). Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*.

Morrison, A.; Diesmann, M.; and Gerstner, W. 2008. Phenomenological models of synaptic plasticity based on spike timing. *Biological cybernetics* 98(6): 459–478.

Orchard, G., Jayawant, A., Cohen, G. K., and Thakor, N. 2015. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437.

Panda, P.; and Srinivasa, N. 2018. Learning to recognize actions from limited training examples using a recurrent spiking neural model. *Frontiers in neuroscience* 12: 126.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Ponulak, F.; and Kasiński, A. 2010. Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. *Neural computation* 22(2): 467–510.

Schrauwen, B.; Wardermann, M.; Verstraeten, D.; Steil, J. J.; and Stroobandt, D. 2008. Improving reservoirs using intrinsic plasticity. *Neurocomputing* 71(7-9): 1159–1171.

Shrestha, A.; Ahmed, K.; Wang, Y.; Widemann, D. P.; Moody, A. T.; Van Essen, B. C.; and Qiu, Q. 2017. A spike-based long short-term memory on a neurosynaptic processor. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 631–637. IEEE.

Srinivasan, G.; Panda, P.; and Roy, K. 2018. SpiLinC: Spiking Liquid-Ensemble Computing for Unsupervised Speech and Image Recognition. *Frontiers in neuroscience* 12.

Triesch, J. 2005. A gradient rule for the plasticity of a neuron's intrinsic excitability. In *International Conference on Artificial Neural Networks*, 65–70. Springer.

Voelker, A.; Kajić, I.; and Eliasmith, C. 2019. Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, 15570–15579.

Wang, Q.; and Li, P. 2016. D-lsm: Deep liquid state machine with unsupervised recurrent reservoir tuning. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2652–2657. IEEE.

Wijesinghe, P.; Srinivasan, G.; Panda, P.; and Roy, K. 2019. Analysis of Liquid Ensembles for Enhancing the Performance and Accuracy of Liquid State Machines. *Frontiers in Neuroscience* 13: 504.

Zhang, W.; and Li, P. 2019a. Information-theoretic intrinsic plasticity for online unsupervised learning in spiking neural networks. *Frontiers in neuroscience* 13: 31.

Zhang, W.; and Li, P. 2019b. Spike-Train Level Backpropagation for Training Deep Recurrent Spiking Neural Networks. In *Advances in Neural Information Processing Systems*, 7800–7811.

Zhang, W.; and Li, P. 2020. Temporal Spike Sequence Learning via Backpropagation for Deep Spiking Neural Networks. *arXiv preprint arXiv:2002.10085* .

Zhang, Y.; Li, P.; Jin, Y.; and Choe, Y. 2015. A digital liquid state machine with biologically inspired learning and its application to speech recognition. *IEEE transactions on neural networks and learning systems* 26(11): 2635–2649.