Memory-Efficient Reversible Spiking Neural Networks

Hong Zhang[1], **Yu Zhang**[1,2*]

[1]State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou, China

[2]Key Laboratory of Collaborative Sensing and Autonomous Unmanned Systems of Zhejiang Province, Hangzhou, China

{hongzhang99, zhangyu80}@zju.edu.cn

Abstract

Spiking neural networks (SNNs) are potential competitors to artificial neural networks (ANNs) due to their high energy-efficiency on neuromorphic hardware. However, SNNs are unfolded over simulation time steps during the training process. Thus, SNNs require much more memory than ANNs, which impedes the training of deeper SNN models. In this paper, we propose the reversible spiking neural network to reduce the memory cost of intermediate activations and membrane potentials during training. Firstly, we extend the reversible architecture along temporal dimension and propose the reversible spiking block, which can reconstruct the computational graph and recompute all intermediate variables in forward pass with a reverse process. On this basis, we adopt the state-of-the-art SNN models to the reversible variants, namely reversible spiking ResNet (RevSResNet) and reversible spiking transformer (RevSFormer). Through experiments on static and neuromorphic datasets, we demonstrate that the memory cost per image of our reversible SNNs does not increase with the network depth. On CIFAR10 and CIFAR100 datasets, our RevSResNet37 and RevSFormer-4-384 achieve comparable accuracies and consume $3.79\times$ and $3.00\times$ lower GPU memory per image than their counterparts with roughly identical model complexity and parameters. We believe that this work can unleash the memory constraints in SNN training and pave the way for training extremely large and deep SNNs.

Introduction

Spiking neural networks (SNNs), brain-inspired models based on binary spiking signals, are regarded as the third generation of neural networks (Maass 1997). Due to the sparsity and event-driven characteristics, SNNs can be deployed on neuromorphic hardware with low energy consumption. With the help of backpropagation through time framework (BPTT) and surrogate gradient, direct training SNNs are developing towards deeper and larger models. Advanced spiking architectures such as ResNet-like SNNs (Hu et al. 2021; Fang et al. 2021a; Zhang et al. 2023) and spiking vision transformers (Zhou et al. 2022, 2023) have been proposed in succession, indicating that SNNs are potential competitors to artificial neural networks (ANNs).

*Corresponding Author.

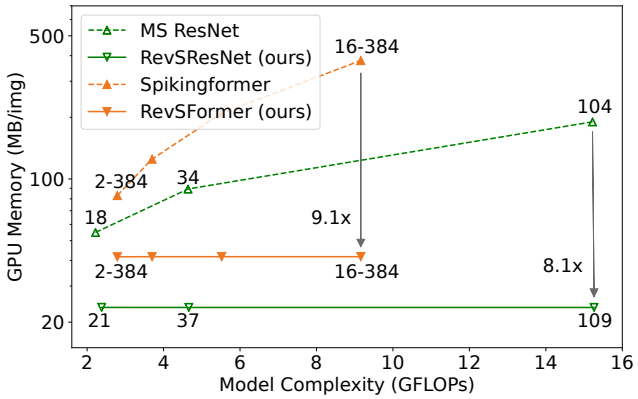

Figure 1: Reversible spiking neural networks are more memory-efficient. Our proposed RevSResNet and RevSFormer need less GPU memory than their non-reversible counterparts during training. Besides, the memory cost of reversible SNNs does not increase with the network depth.

Although the inference process of SNNs on neuromorphic chips is relatively mature (Davies et al. 2018; Roy, Jaiswal, and Panda 2019), these chips cannot support the training process. Therefore, SNNs are still trained on graphics processing units (GPUs). During the training process based on the BPTT framework, SNNs are unfolded over simulation time steps T. Thus, SNNs usually require higher computing resources and memory bandwidth compared to ANNs. The computational requirements can be compensated by some AI accelerators (Okuta et al. 2017) or spending more time in training. However, there is currently no solution to the memory constraints. Under such constraints, some SNNs are trained with a small batch size (Zhang, Fan, and Zhang 2023), indirectly affecting the final accuracy (Wu and Johnson 2021). Also, deeper SNNs are prevented from training.

The high memory consumption of SNNs comes from several aspects. On the one hand, like ANNs, the memory required by SNNs increases linearly with the depth of the network. The deeper the network, the more parameters and intermediate activations need storage. On the other hand, unlike ANNs, the memory cost of SNNs increases with simulation time step T. SNNs need to store T times more intermediate activations, and the membrane potentials of spiking

neurons need also to be stored for gradient computation. It is evident that a significant amount of memory consumption comes from storing intermediate activations and membrane potentials (Gomez et al. 2017). By reducing this part of the consumption, we can decouple the memory growth from the depth to a large extent.

In this work, we propose reversible spiking neural networks to reduce the memory cost of SNN training. The intention of reversibility is that each layer's input variables and membrane potentials can be re-computed by its output variables. Therefore, even if no intermediate variables are stored, we can quickly reconstruct them through such reversible transformation. In this work, we first extend the reversible architecture (Gomez et al. 2017) along the temporal dimension to adapt to the BPTT training framework. On this basis, we propose spiking reversible block, which is reversible along spatial dimension and consistent along temporal dimension. Then, we present the reversible spiking ResNet (RevSResNet) and reversible spiking transformer (RevSFormer), which are the reversible counterparts of MS ResNet (Hu et al. 2021) and Spikingformer (Zhou et al. 2022) (the latest ResNet-like and transformer-like SNNs). As is shown in Figure 1, our networks consume much less memory per image than their counterparts. We verify the effect of RevSResNet and RevSFormer on static datasets (CIFAR10 and CIFAR100 (Krizhevsky, Hinton et al. 2009)) and neuromorphic datasets (CIFAR10-DVS (Li et al. 2017) and DVS128 Gesture (Amir et al. 2017)). The experiments show that RevSResNet and RevSFormer have competitive performance to their non-reversible counterparts. At the same time, our reversible models significantly reduce memory cost during the training process, saving $3.79\times$ on the RevSResNet37 and $3.00\times$ on the RevSFormer-4-384 model.

In summary, our contributions are three-fold.

- We analyze the reversibility of SNNs in the spatial and temporal dimensions and propose spiking reversible block for the BPTT framework. On this basis, each block's input and intermediate variables can be calculated by its outputs.
- We propose the reversible spiking ResNet (RevSResNet) and reversible spiking transformer (RevSFormer). We redesign a series of structures (such as downsample layers, reversible spiking residual block, and reversible spiking transformer block) to match the performance of the non-reversible state-of-the-art spiking counterparts.
- The experiments show that RevSResNet and RevSFormer have competitive performance to their non-reversible counterparts. At the same time, our reversible models significantly reduce memory cost during the training process.

Related Works

Spiking Neural Networks

SNNs utilize binary spikes to transmit and compute information, while the spiking neurons (Gerstner and Kistler 2002; Yao et al. 2022) play a crucial role in converting analog membrane potentials into binary spikes. There are two methods to obtain deep SNNs: ANN-to-SNN conversion and direct training. The ANN-to-SNN conversion methods (Diehl et al. 2015; Bu et al. 2022; Deng and Gu 2021; Wang et al. 2022) convert the same structured ANNs into SNNs, which usually achieves high accuracy. However, this method is limited because the obtained SNN requires a large time step and is unable to handle neuromorphic data. The direct training method utilizes error backpropagation to train SNNs directly, where the BPTT framework (Shrestha and Orchard 2018) and surrogate gradient (Neftci, Mostafa, and Zenke 2019) techniques play a vital role. In recent years, direct training spiking structures have been proposed successively, including ResNet-like models (Lee et al. 2020; Fang et al. 2021a; Hu et al. 2021; Zhang et al. 2023), Spiking transformers (Zhou et al. 2022, 2023), NAS SNNs (Na et al. 2022; Kim et al. 2022), etc. These networks have lower latency, but the training process requires more computing resources and memory costs than ANNs. Among them, high memory cost limits the depth and time steps of the network. Thus, this article aims to reduce the memory cost of the SNN training based on reversible architectures.

Reversible Architectures

Reversible architectures are neural networks based on NICE reversible transformation (Dinh, Krueger, and Bengio 2014). Reversible ResNet (Gomez et al. 2017) is the first work that utilizes it for CNN-based image classification tasks. They employ reversible blocks to complete memory-efficient network training. The core of its memory saving is that the intermediate activation can be reconstructed through the reverse process. After that, other works (Hascoet et al. 2019; Sander et al. 2021; Li and Gao 2021) have further iterated on the CNN-based reversible architectures. Recently, (Mangalam et al. 2022) applied the reversible transformation to vision transformers and proposed Rev-ViT and Rev-MViT, two memory-efficient transformer structures. They found that reversible architectures have stronger inherent regularization than their non-reversible counterparts. In addition, reversible transformation has also been adopted in other networks, such as UNet (Brügger, Baumgartner, and Konukoglu 2019), masked convolutional networks (Song, Meng, and Ermon 2019), and graph neural networks (Li et al. 2021a).

It is worth noting that the above reversible architectures are reversible in the spatial dimension, in which the forward process propagates from shallow to deep layers, and the reverse process propagates from deep to shallow layers. Unlike them, reversible RNN (MacKay et al. 2018) is reversible in the temporal dimension. It calculates hidden states in the past by reversing them from the future. SNN is a network with both spatial and temporal dimensions, while our spiking reversible block is reversible along the spatial dimension and consistent along the temporal dimension.

Approach

In this section, we first explain the spiking neuron model, which is the preliminary of SNNs. Then, we present our proposed spiking reversible block. Furthermore, we apply

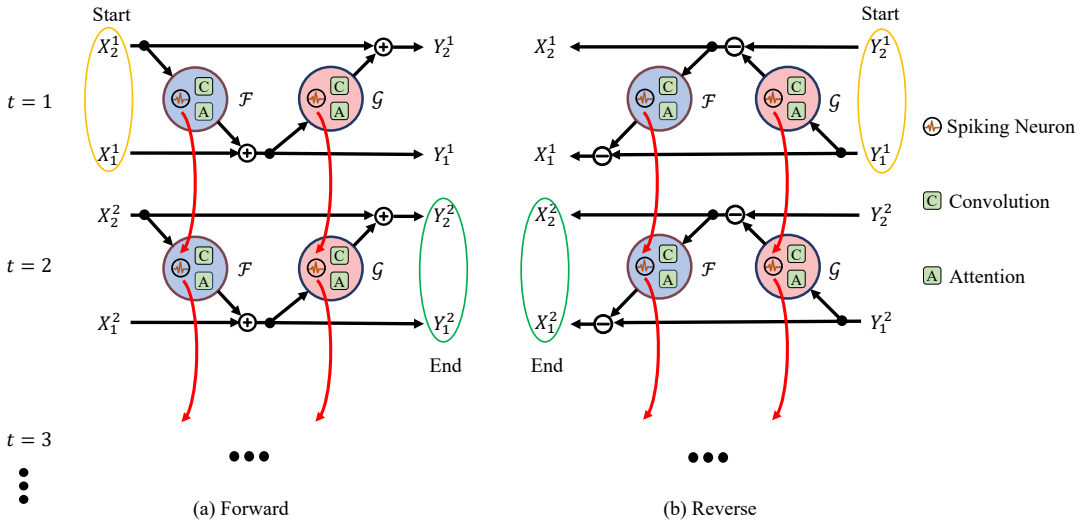

Figure 2: Illustration of the forward (a) and reverse (b) process of our spiking reversible block. We can recompute all intermediate variables in forward pass with the reverse process. Note that there is a reset process between them.

it to spiking ResNet-like and transformer-like structures and propose the reversible spiking ResNet and reversible spiking transformer. They both support memory-efficient end-to-end training.

Spiking Neuron Model

The spiking neuron, which plays the role of activation function, is the fundamental unit used in SNNs. It converts analog membrane potentials to binary spiking signals. The leaky-integrate-and-fire (LIF) neuron is a widely used spiking neuron whose discrete-time dynamics can be formulated as follows:

$$H[t] = V[t-1] + \frac{1}{\tau_m}\left(I[t] - (V[t-1] - V_{reset})\right) \quad (1)$$

$$S[t] = \Theta\left(H[t] - V_{th}\right) \quad (2)$$

$$V[t] = H[t](1 - S[t]) + V_{reset}S[t] \quad (3)$$

where $V[t]$ represents the membrane potential at time t, and $H[t]$ is the hidden membrane potential before trigger time t. $I[t]$ is the synaptic current, which is the input from other neurons. Once $H[t]$ exceeds the firing threshold V_{th}, the neuron will file a spike expressed by $S[t]$. Then, the membrane potential $V[t]$ will be reset to reset potential V_{rest}.

In addition to LIF, we also use (integrate-and-fire) IF neuron in this work, which is a simplified version of LIF. Its integrate dynamics (Eq.4) differs from LIF, while the fire and reset processes remain unchanged.

$$H[t] = V[t-1] + I[t] \quad (4)$$

Spiking Reversible Block

Computation Graph of Spiking Reversible Block During standard backpropagation training, a single-batch is computed with a forward-backward process. In contrast, for a reversible block, this computation turns to a forward-reverse-backward process. The added reverse process utilizes the output of the block to compute the input in reverse. Then we can delete all inputs and intermediate variables after the forward process and save only the output. RevNet (Gomez et al. 2017) and RevRNN (MacKay et al. 2018) implement the reversible blocks in the spatial and temporal dimensions, respectively.

For SNNs, as long as the network is designed in a two-residual-stream manner in (Gomez et al. 2017), we can establish the reverse process in the spatial dimension. However, in the temporal dimension, the reverse means that the input potential of all neurons must be calculated through their output spikes, which is theoretically impossible for spiking neurons in Eq. 1. Therefore, spiking reversible block should be reversible along the spatial dimension and consistent along the temporal dimension. We extend the single-batch computation process to forward-reset-reverse-backward. The computation graphs for forward and reverse processes are shown in Figure 2, where $\mathcal{F}$ and $\mathcal{G}$ can be set as arbitrary spiking modules composed of spiking neurons, convolutional layers, attention mechanisms, etc. Since spiking neurons have different membrane potentials at different time steps, $\mathcal{F}$ and $\mathcal{G}$ vary with time. We use $\mathcal{F}^t$ and $\mathcal{G}^t$ to represent these two modules at the time step t.

In the forward process, the starting node of the graph lies in the input node at time step 1, and the end node is the output at time T, where T is the total time steps of the SNN. At each time step t, output Y^t is calculated using formula 5, as the horizontal arrows in Figure 2a. From time step t to $t+1$, the edges of the computation graph are established through the inherited membrane potential of all spiking neurons in $\mathcal{F}$ and $\mathcal{G}$, as the red arrows illustrate in Figure 2a.

$$\begin{aligned} Y_1^t &= X_1^t + \mathcal{F}^t\left(X_2^t\right) \\ Y_2^t &= X_2^t + \mathcal{G}^t\left(Y_1^t\right) \end{aligned} \quad (5)$$

Before the reverse process, all spiking neurons are reset by resetting membrane potential to the initial state, which is named the reset process.

In the reverse process, the starting node of the graph lies in the output node at time step 1, and the end node is the input at time step T. For each time step t, input X^t is calculated using formula 6, as the reversed horizontal arrows in Figure 2b. From time step t to $t+1$, same as forward process, the edges of the computation graph are established through the inherited membrane potential of all spiking neurons in $\mathcal{F}$ and $\mathcal{G}$, as the red arrows show in Figure 2b.

$$\begin{aligned} X_2^t &= Y_2^t - \mathcal{G}^t\left(Y_1^t\right) \\ X_1^t &= Y_1^t - \mathcal{F}^t\left(X_2^t\right) \end{aligned} \tag{6}$$

Learning without Caching Intermediate Variables During network training, the backward process is essential for updating the network weights. Consider the presynaptic weight W_l of a spiking neuron in the l_{th} layer. Its gradient is calculated as follows:

$$\frac{\partial L}{\partial W_l} = \sum_t \left(\frac{\partial L}{\partial S_l^t}\frac{\partial S_l^t}{\partial U_l^t} + \frac{\partial L}{\partial U_l^{t+1}}\frac{\partial U_l^{t+1}}{\partial U_l^t} \right) \frac{\partial U_l^t}{\partial W_l} \tag{7}$$

where S_l^t and U_l^t are the output spike (activation) and membrane potential at time step t, which are calculated using the spiking neuron dynamics. It can be found that the gradient calculation requires all output spikes and membrane potentials at all time steps. In fact, almost all intermediate variables in the forward process are needed in the backward process. In standard training, these variables are cached in GPU memory after the forward process. Because of the sequential nature of the network, all intermediate variables for all layers at all time steps should be stored. Thus, peak memory usage becomes linearly dependent on the network depth D and time steps T. Its spatial complexity is $O(D \cdot T)$.

For the training of the spiking reversible block, we propose Theorem 1, which means all intermediate variables in the forward process can be recomputed from output in the reverse process. Then, only output Y needs caching in the forward process. Furthermore, if spiking reversible blocks are sequentially placed, we only need to store the output of the last block. Before the backward process of any block, we can recompute all intermediate variables with the output. In this process, the peak memory usage is the memory required for a single block whose spatial complexity is $O(T)$. Since direct training SNNs often have relatively small T (such as 4), the peak memory usage during training is much smaller.

Theorem 1 *Consider a spiking reversible block with T time steps, if the forward and reverse functions are formulated as Eq. 5 and Eq. 6, and outputs of forward process are fed into the reverse process, then X^t, Y^t and all intermediate variables (including the intermediate activations and membrane potentials) in $\mathcal{F}^t$ and $\mathcal{G}^t$ in the forward process are the identical to those in the reverse process.*

Proof. The proof of Theorem 1 is presented in the Appendix (Zhang and Zhang 2023).

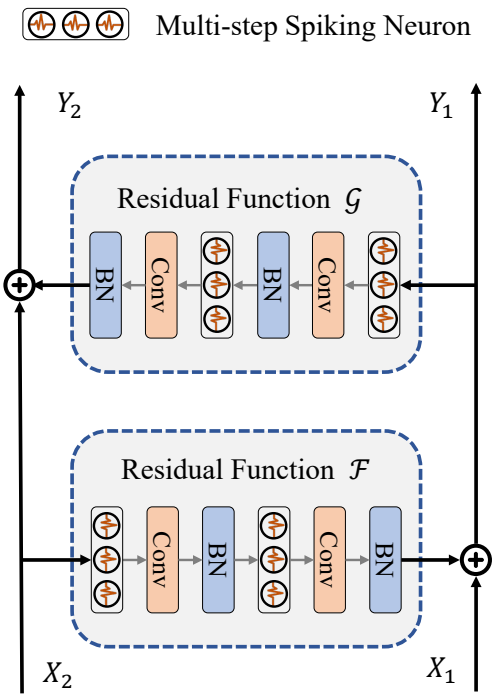

Figure 3: Basic block of RevSResNet. We utilize two residual functions with the same structure as $\mathcal{F}$ and $\mathcal{G}$.

Reversible Spiking Residual Neural Network

ResNet (He et al. 2016) is one of the most popular deep convolutional neural networks (CNNs), and residual learning is also the best solution for CNN-based SNNs to tackle the gradient degradation problem (Fang et al. 2021a). With the help of our spiking reversible block, we propose the reversible spiking residual neural network, which completes the training of deep SNNs with much less memory usage.

Basic Block In ANN ResNet, the parameterized residual function is wrapped around a single residual stream in each block. We adopt it to the spiking reversible block and propose the two-residual-stream architecture in Figure 3. The input X is partitioned into tensors X_1 and X_2 in halves along the channel dimension. The forward process follows transformation in Eq. 5 to ensure reversibility. We utilize two residual functions with the same structure as $\mathcal{F}$ and $\mathcal{G}$. To ensure that all operations are spike computations, we adopt the Activation-Conv-BatchNorm paradigm (Hu et al. 2021). Each residual function consists of two sequentially connected multi-step spiking neurons, convolutional layers, and batch normalization.

Downsample Block Due to the reversibility of the basic block, the feature dimensions of X and Y are identical. Therefore, residual functions $\mathcal{F}$ and $\mathcal{G}$ must be equidimensional in input and output spaces, which means that downsample layers (such as maxpooling or convolution with a stride of 2) cannot appear in spiking reversible blocks. To replace the downsampling basic blocks in ResNet, we set up

Total layers	$N = 5 + 4 * \sum n_i$
conv1	3×3, 128
reversible sequence 1	$\begin{pmatrix} 3\times3, 64 \\ 3\times3, 64 \end{pmatrix} \times 2 \times n_1$
reversible sequence 2	$\begin{pmatrix} 3\times3, 128 \\ 3\times3, 128 \end{pmatrix}^* \times 2 \times n_2$
reversible sequence 3	$\begin{pmatrix} 3\times3, 256 \\ 3\times3, 256 \end{pmatrix}^* \times 2 \times n_3$
reversible sequence 4	$\begin{pmatrix} 3\times3, 448 \\ 3\times3, 448 \end{pmatrix}^* \times 2 \times n_4$
	average pool, fc, softmax

Table 1: Architectures of RevSResNet. The stride of conv1 are set to 2 for downsampling. $*$ means that a downsample block is set at the beginning of the reversible sequence. N represents the total number of layers.

a downsample block at the start of the stages where downsampling is required. We first use a 3×3 average pooling with a stride of 2 to downsample the image scale and then increase the feature channels using a 1×1 convolutional layer with a stride of 1.

Network Architecture The high-level structure of RevSResNet is the same as its non-reversible counterpart MS ResNet (Hu et al. 2021). The first convolution is regarded as the encoding layer which performs the initial downsampling. Then the spiking features propagate through the four stages with basic blocks. We set up a downsample block at the start of the second to fourth stages. The network ends with an average pooling and fully connected layer.

When spiking reversible blocks are sequentially connected (we call it reversible sequence), we only need to store the output of the last block to complete the training. Leave out the downsample block, all stages in RevSResNet are reversible sequences. No matter how the number of blocks in a reversible sequence grows, the memory usage required by intermediate variables does not increase. The detailed architectures of RevSResNet are summarized in Table 1. RevSResNet-N means the network with N layers.

Reversible Spiking Transformer

Vision transformer has taken the accuracy of computer vision tasks to a new level. Combining our spiking reversible block with the spiking transformer (Zhou et al. 2023), we propose RevSFormer and prove the feasibility of reversible structures in transformer-like SNNs.

Basic Block Unlike ResNet, a spiking transformer block has two relatively independent residual functions: spiking self-attention (SSA) and spiking MLP block (MLP). They are wrapped around their residual connection, respectively. Under this condition, we respectively consider SSA and MLP as $\mathcal{F}$ and $\mathcal{G}$, and propose the basic block in RevSFormer, as is shown in Figure 4. We adopt the same SSA and MLP structure as Spikingformer (Zhou et al. 2023), so our basic block's computational complexity and parameter

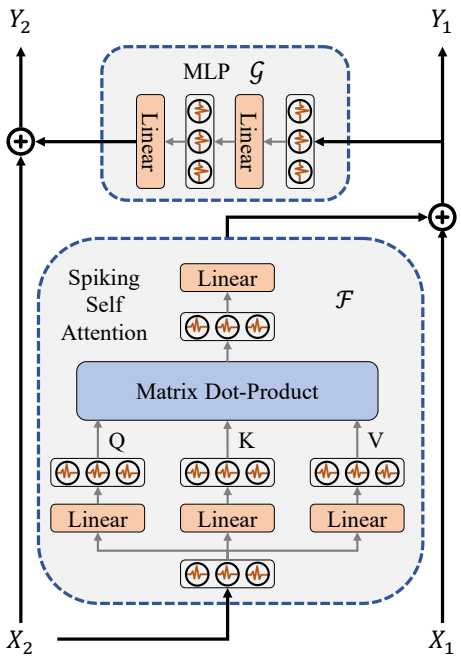

Figure 4: Basic block of RevSFormer. We consider spiking self-attention and MLP block as $\mathcal{F}$ and $\mathcal{G}$, respectively.

numbers are consistent with the original spiking transformer block.

Network Structure The high-level structure of RevSFormer is the same as its non-reversible counterpart Spikingformer. The network includes a spiking tokenizer, L basic blocks, and a classification head. The spiking tokenizer computes the patch embedding of the image and projects the embedding into a fixed size with several convolutional and maxpooling layers. The classification head is composed of a spiking neuron and a fully connected layer. It is worth mentioning that all downsampling operations of RevSFormer are placed in the spiking tokenizer. Since there are no other downsampling or irreversible operations between all basic blocks, RevSFormer has only one reversible sequence composed of L basic blocks. As L grows, the memory required to store intermediate variables is expected to stay the same. The detailed configurations of RevSFormer are the same as Spikingfomer. And RevSFormer-L-D means the network has L blocks and the embedding dimension is D.

Experiments

We evaluate the performance of our reversible structures on static datasets (CIFAR10 and CIFAR100) and neuromorphic datasets (CIFAR10-DVS and DVS128 Gesture). The metrics include parameters, time steps, FLOPS, memory per image, and the top-1 accuracy. The memory per image is measured as the peak GPU memory each image occupies during training. To ensure direct comparability with non-reversible counterparts, we match the model complex-

Methods	Architecture	Param (M)	Time Step	FLOPS (G)	Memory (MB/img)	CIFAR10 Top-1 Acc	CIFAR100 Top-1 Acc
Hybrid training (Rathi et al. 2020)	VGG-11	9.27	125	-	-	92.22	67.87
Diet-SNN (Rathi and Roy 2020)	ResNet-20	0.27	10	-	-	92.54	64.07
STBP (Wu et al. 2018)	CIFARNet	17.54	12	-	-	89.83	-
STBP NeuNorm (Wu et al. 2019)	CIFARNet	17.54	12	-	-	90.53	-
TSSL-BP (Zhang and Li 2020)	CIFARNet	17.54	5	-	-	91.41	-
STBP-tdBN (Zheng et al. 2021)	ResNet-19	12.63	4	-	-	92.92	70.86
TET (Deng et al. 2022)	ResNet-19	12.63	4	-	-	94.44	74.47
DS-ResNet (Feng et al. 2022)	ResNet20	4.32	4	-	-	94.25	-
Spikformer (Zhou et al. 2022)	Spikformer-4-384	9.32	4	-	-	95.19	77.86
MS ResNet (Hu et al. 2021)	MS ResNet18	11.22	4	2.22	54.83	94.40	75.06
RevSResNet (ours)	RevSResNet21	11.05	4	2.38	**23.59** $\downarrow 2.32\times$	94.53	75.46
MS ResNet (Hu et al. 2021)	MS ResNet34	21.33	4	4.64	89.33	94.69	75.34
RevSResNet (ours)	RevSResNet37	23.59	4	4.66	**23.58** $\downarrow 3.79\times$	94.77	76.34
Spikingformer (Zhou et al. 2023)	Spikingformer-2-384	5.76	4	2.79	83.05	95.12	77.96
RevSFormer (ours)	RevSFormer-2-384	5.76	4	2.79	**41.68** $\downarrow 1.99\times$	95.29	78.04
Spikingformer (Zhou et al. 2023)	Spikingformer-4-384	9.32	4	3.70	125.06	95.35	79.02
RevSFormer (ours)	RevSFormer-4-384	9.32	4	3.70	**41.74** $\downarrow 3.00\times$	95.34	79.04

Table 2: Comparison to prior works on static datasets, CIFAR100 and CIFAR10. Note that results of MS ResNet and Spikingformer are based on our implementation for a fair comparison. Bold values denotes the memory usage of our reversible SNNs.

ity (FLOPS in metric) and number of parameters as closely as possible. The dataset introduction, detailed network configuration, and other experimental settings are presented in the Appendix (Zhang and Zhang 2023).

Experiment on Static Datasets

CIFAR10 and CIFAR100 each provides 50000 train and 10000 test images. On these datasets, we establish two comparisons (MS ResNet18 vs. RevSResNet21, MS ResNet34 vs. RevSResNet37) for ResNet-like structures. For transformer-like structures, the network configuration and model complexity of RevSFormer are identical to Spikingformer. Results are shown in Table 2.

From an accuracy perspective, we find that the performance of RevSResNet and RevSFormer is comparable to their counterparts with similar complexity. RevSResNet37 achieves 94.77% and 76.34% accuracy on CIFAR10 and CIFAR100 datasets, respectively, while RevSFormer-4-384 achieves 95.34% and 79.04% accuracy with a time step of 4. The performance of RevSResNet and RevSFormer is even slightly better than MS ResNet and SpikingFormer, which may be due to stronger inherent regularization of reversible architectures than vanilla networks (Mangalam et al. 2022).

From the memory perspective, our reversible SNNs are much more memory-efficient than vanilla SNNs. On one hand, RevSResNet37 and RevSFormer-4-384 consume 23.58 and 41.74 MB GPU memory per image, which is $3.79\times$ and $3.00\times$ lower than their counterparts. On the other hand, the memory usage does not increase with depth in our networks, which will be further discussed later.

Experiment on Neuromorphic Datasets

On the neuromorphic datasets, we conduct experiments with two different time steps, 10 and 16. And we establish one network comparison (MS ResNet20 vs. RevSResNet24) for ResNet-like structures. For transformer-like structures, the network configuration are identical between reversible and non-reversible structures.

Results are shown in Table 3. The relative changes in accuracy and memory are similar to those on static datasets. Our RevSResNet and RevSFormer achieve a memory usage reduction of $2.01\times$ and $1.30\times$, respectively. And the magnitude of the reduction stays consistent across different time steps. In terms of performance, RevSResNet24 and RevSFormer-2-256 achieve 76.4% and 82.2% accuracy on CIFAR10-DVS dataset with a time step of 16.

Ablation Studies

Memory Usage vs. Depth Theoretically, for a reversible sequence, the memory usage required by intermediate variables does not increase with the number of reversible blocks because we only need to save the output of the whole sequence. Thus, for RevSResNet with 4 reversible sequences and RevSFormer with 1 sequence, the memory usage per image should not increase with depth. Figure 1 plots the memory usage for our reversible SNNs and their counterparts. For ResNet-like structures, the relative memory saving magnitude increases up to $8.1\times$ as the model goes deeper. For transformer networks, our RevSFormer-16-384 saves $9.1\times$ GPU memory per image. It is expected that this memory saving magnitude will increase further with increasing depth.

Memory Usage vs. Time Step The memory required by an SNN is T times larger than an ANN. Thus, the GPU memory required per image grows linearly with the total time steps T. Figure 5 shows the relationship between memory usage and time steps. As is seen, for each model, the memory usage increases with a certain slope m. In our reversible SNNs, intermediate variables in the non-reversible parts (e.g., the downsample layers and the spiking tokenizer) and the output of each reversible sequence still need caching. Thus, memory usage is not decoupled from time steps T.

Methods	FLOPS (G)	Memory (MB/img)	CIFAR10-DVS		DVS128 Gesture	
			Time Step	Top-1 Acc	Time Step	Top-1 Acc
LIAF-Net (Wu et al. 2021)	-	-	10	70.40	60	97.56
TA-SNN (Yao et al. 2021)	-	-	10	72.00	60	98.61
Rollout (Kugele et al. 2020)	-	-	48	66.75	240	97.16
tdBN (Zheng et al. 2021)	-	-	10	67.80	40	96.87
PLIF (Fang et al. 2021b)	-	-	20	74.80	20	97.57
SEW ResNet (Fang et al. 2021a)	-	-	16	74.40	16	97.92
Dspike (Li et al. 2021b)	-	-	10	75.40	-	-
DSR (Meng et al. 2022)	-	-	10	77.27	-	-
DS-ResNet (Feng et al. 2022)	-	-	10	70.36	40	97.29
Spikformer (Zhou et al. 2022)	-	-	16	80.60	16	97.90
MS ResNet20 (Hu et al. 2021)	0.42	50.72	10	76.00	10	94.79
RevSResNet24 (ours)	0.43	**24.97** $\downarrow 2.03\times$	10	75.50	10	94.44
MS ResNet20 (Hu et al. 2021)	0.67	79.38	16	75.80	16	97.57
RevSResNet24 (ours)	0.69	**39.52** $\downarrow 2.01\times$	16	76.40	16	96.53
Spikingformer-2-256 (Zhou et al. 2023)	3.78	295.73	10	78.50	10	96.88
RevSFormer-2-256 (ours)	3.78	**227.50** $\downarrow 1.30\times$	10	81.40	10	97.22
Spikingformer-2-256 (Zhou et al. 2023)	6.05	466.08	16	80.30	16	98.26
RevSFormer-2-256 (ours)	6.05	**359.58** $\downarrow 1.30\times$	16	82.20	16	97.57

Table 3: Comparisons with prior works on neuromorphic datasets, CIFAR10-DVS and DVS128 Gesture. Note that results of MS ResNet and Spikingformer are based on our implementation for a fair comparison. Bold values denote the memory usage of our reversible SNNs.

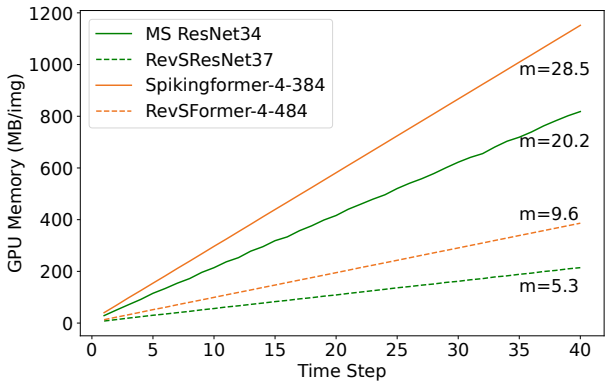

Figure 5: Relationship between memory and time step.

However, through reversible architecture, we have greatly reduced the slope of memory usage growth from 28.5 and 20.2 of non-reversible SNNs to 9.6 and 5.3 of our reversible networks.

Computational Overhead during Training In general, for a network with N operations, the forward and backward processes take N and $2N$ operations approximately (Gomez et al. 2017). Our spiking reversible block requires the extra reset and reverse processes. The reset process take negligible operations and the reverse process take N operations, same as forward. In summary, the reversible architectures need roughly 33% more computations than vanilla networks during training. Besides, reversible SNNs have larger maximum batch size, which may slightly influence the training speed and final performance (Wu and Johnson 2021).

The training time and maximum batch size of our reversible SNNs and their counterparts are shown in Table 4.

Architecture	Training time (seconds / epoch)	Maximum Batch size
MS ResNet34	98	239
RevSResNet37	**131** $\uparrow 1.33\times$	**644** $\uparrow 2.69\times$
Spikingformer-4-384	105	164
RevSFormer-4-384	**133** $\uparrow 1.27\times$	**286** $\uparrow 1.74\times$

Table 4: The training time and maximum batch size of our reversible structures and their non-reversible counterparts.

The values are measured on a single 24GB RTX3090 GPU under CIFAR10 dataset. Our RevSResNet37 takes $1.33\times$ more training time in practice. Besides, it achieves a $2.69\times$ increase in maximum batch size, and the increase magnitude will go larger on bigger models.

Conclusion

In this paper, we propose the reversible spiking neural network to reduce the memory cost of intermediate activations and membrane potentials during training of SNNs. We first extend the reversible architecture along temporal dimension and propose the reversible spiking block, which can reconstruct the computational graph of forward pass with a reverse process. On this basis, we present the RevSResNet and RevSFormer models, which are the reversible counterparts of the state-of-the-art SNNs. Through experiments on static and neuromorphic datasets, we demonstrate that the memory cost per image of our reversible SNNs does not increase with the network depth. In addition, RevSResNet and RevSFormer achieve comparative accuracies and consume much less GPU memory than their counterparts with roughly identical model complexity and parameters.

Acknowledgments

This work was supported by STI 2030-Major Projects 2021ZD0201403, in part by NSFC 62088101 Autonomous Intelligent Unmanned Systems.

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.

Brügger, R.; Baumgartner, C. F.; and Konukoglu, E. 2019. A partially reversible U-Net for memory-efficient volumetric image segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part III 22*, 429–437. Springer.

Bu, T.; Ding, J.; Yu, Z.; and Huang, T. 2022. Optimized Potential Initialization for Low-latency Spiking Neural Networks. *arXiv preprint arXiv:2202.01440*.

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

Deng, S.; and Gu, S. 2021. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*.

Deng, S.; Li, Y.; Zhang, S.; and Gu, S. 2022. Temporal efficient training of spiking neural network via gradient reweighting. *arXiv preprint arXiv:2202.11946*.

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

Dinh, L.; Krueger, D.; and Bengio, Y. 2014. Nice: Nonlinear independent components estimation. *arXiv preprint arXiv:1410.8516*.

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

Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; and Tian, Y. 2021b. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2661–2671.

Feng, L.; Liu, Q.; Tang, H.; Ma, D.; and Pan, G. 2022. Multi-level firing with spiking ds-resnet: Enabling better and deeper directly-trained spiking neural networks. *arXiv preprint arXiv:2210.06386*.

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

Gomez, A. N.; Ren, M.; Urtasun, R.; and Grosse, R. B. 2017. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30.

Hascoet, T.; Febvre, Q.; Zhuang, W.; Ariki, Y.; and Takiguchi, T. 2019. Layer-wise invertibility for extreme memory cost reduction of cnn training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 0–0.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hu, Y.; Deng, L.; Wu, Y.; Yao, M.; and Li, G. 2021. Advancing Spiking Neural Networks towards Deep Residual Learning. *arXiv preprint arXiv:2112.08954*.

Kim, Y.; Li, Y.; Park, H.; Venkatesha, Y.; and Panda, P. 2022. Neural architecture search for spiking neural networks. In *European Conference on Computer Vision*, 36–56. Springer.

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

Kugele, A.; Pfeil, T.; Pfeiffer, M.; and Chicca, E. 2020. Efficient processing of spatio-temporal data streams with spiking neural networks. *Frontiers in Neuroscience*, 14: 439.

Lee, C.; Sarwar, S. S.; Panda, P.; Srinivasan, G.; and Roy, K. 2020. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in neuroscience*, 119.

Li, D.; and Gao, S.-H. 2021. m-revnet: Deep reversible neural networks with momentum. *arXiv preprint arXiv:2108.05862*.

Li, G.; Müller, M.; Ghanem, B.; and Koltun, V. 2021a. Training graph neural networks with 1000 layers. In *International conference on machine learning*, 6437–6449. PMLR.

Li, H.; Liu, H.; Ji, X.; Li, G.; and Shi, L. 2017. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11: 309.

Li, Y.; Guo, Y.; Zhang, S.; Deng, S.; Hai, Y.; and Gu, S. 2021b. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34: 23426–23439.

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

MacKay, M.; Vicol, P.; Ba, J.; and Grosse, R. B. 2018. Reversible recurrent neural networks. *Advances in Neural Information Processing Systems*, 31.

Mangalam, K.; Fan, H.; Li, Y.; Wu, C.-Y.; Xiong, B.; Feichtenhofer, C.; and Malik, J. 2022. Reversible vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10830–10840.

Meng, Q.; Xiao, M.; Yan, S.; Wang, Y.; Lin, Z.; and Luo, Z.-Q. 2022. Training high-performance low-latency spiking neural networks by differentiation on spike representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12444–12453.

Na, B.; Mok, J.; Park, S.; Lee, D.; Choe, H.; and Yoon, S. 2022. Autosnn: Towards energy-efficient spiking neural networks. In *International Conference on Machine Learning*, 16253–16269. PMLR.

Neftci, E. O.; Mostafa, H.; and Zenke, F. 2019. 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.

Okuta, R.; Unno, Y.; Nishino, D.; Hido, S.; and Loomis, C. 2017. CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*.

Rathi, N.; and Roy, K. 2020. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*.

Rathi, N.; Srinivasan, G.; Panda, P.; and Roy, K. 2020. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv preprint arXiv:2005.01807*.

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

Sander, M. E.; Ablin, P.; Blondel, M.; and Peyré, G. 2021. Momentum residual neural networks. In *International Conference on Machine Learning*, 9276–9287. PMLR.

Shrestha, S. B.; and Orchard, G. 2018. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31.

Song, Y.; Meng, C.; and Ermon, S. 2019. Mintnet: Building invertible neural networks with masked convolutions. *Advances in Neural Information Processing Systems*, 32.

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

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; and Shi, L. 2018. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12: 331.

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; Xie, Y.; and Shi, L. 2019. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 1311–1318.

Wu, Y.; and Johnson, J. 2021. Rethinking" batch" in batchnorm. *arXiv preprint arXiv:2105.07576*.

Wu, Z.; Zhang, H.; Lin, Y.; Li, G.; Wang, M.; and Tang, Y. 2021. Liaf-net: Leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11): 6249–6262.

Yao, M.; Gao, H.; Zhao, G.; Wang, D.; Lin, Y.; Yang, Z.; and Li, G. 2021. Temporal-wise attention spiking neural networks for event streams classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10221–10230.

Yao, X.; Li, F.; Mo, Z.; and Cheng, J. 2022. Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. *Advances in Neural Information Processing Systems*, 35: 32160–32171.

Zhang, H.; Fan, X.; and Zhang, Y. 2023. Energy-Efficient Spiking Segmenter for Frame and Event-Based Images. *Biomimetics*, (4).

Zhang, H.; and Zhang, Y. 2023. Memory-Efficient Reversible Spiking Neural Networks. *arXiv preprint arXiv:2312.07922*.

Zhang, W.; and Li, P. 2020. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in Neural Information Processing Systems*, 33: 12022–12033.

Zhang, Y.; Zhang, H.; Li, Y.; He, B.; Fan, X.; and Wang, Y. 2023. Direct Training High-Performance Spiking Neural Networks for Object Recognition and Detection. *Frontiers in Neuroscience*, 17: 1229951.

Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; and Li, G. 2021. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11062–11070.

Zhou, C.; Yu, L.; Zhou, Z.; Zhang, H.; Ma, Z.; Zhou, H.; and Tian, Y. 2023. Spikingformer: Spike-driven Residual Learning for Transformer-based Spiking Neural Network. *arXiv preprint arXiv:2304.11954*.

Zhou, Z.; Zhu, Y.; He, C.; Wang, Y.; Yan, S.; Tian, Y.; and Yuan, L. 2022. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*.