An Improved STBP for Training High-Accuracy and Low-Spike-Count Spiking Neural Networks

Pai-Yu Tan[1], Cheng-Wen Wu[1,2], and Juin-Ming Lu[1,3]
[1]*Department of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan*
[2]*Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan*
[3]*Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan*

***Abstract*—Spiking Neural Networks (SNNs) that facilitate energy-efficient neuromorphic hardware are getting increasing attention. Directly training SNN with backpropagation has already shown competitive accuracy compared with Deep Neural Networks. Besides the accuracy, the number of spikes per inference has a direct impact on the processing time and energy once employed in the neuromorphic processors. However, previous direct-training algorithms do not put great emphasis on this metric. Therefore, this paper proposes four enhancing schemes for the existing direct-training algorithm, Spatio-Temporal Back-Propagation (STBP), to improve not only the accuracy but also the spike count per inference. We first modify the reset mechanism of the spiking neuron model to address the information loss issue, which enables the firing threshold to be a trainable variable. Then we propose two novel output spike decoding schemes to effectively utilize the spatio-temporal information. Finally, we reformulate the derivative approximation of the non-differentiable firing function to simplify the computation of STBP without accuracy loss. In this way, we can achieve higher accuracy and lower spike count per inference on image classification tasks. Moreover, the enhanced STBP is feasible for the on-line learning hardware implementation in the future.**

***Index Terms*—artificial intelligence (AI), backpropagation, direct-training algorithm, image classification, machine learning, neuromorphic computing, spiking neural network (SNN).**

I. Introduction

Spiking neural networks (SNNs) are the third-generation artificial neural network models that try to mimic the actual mechanisms of biological neurons. This model aims to bridge the gap between neuroscience and machine learning. SNNs use spatio-temporal dynamics to mimic neural behaviors and binary spike signals to communicate between neurons. This event-driven property enables specialized neuromorphic hardware, such as IBM TrueNorth [1], Intel Loihi [2], and Tianjic [3], to efficiently process the machine learning applications. When SNNs are processed in the neuromorphic hardware, a spike is seen as an event triggering a series of computations. As a result, the numbers of spikes are proportional to the processing time and energy on the hardware [4]. To enable the energy-efficient processing of the neuromorphic hardware, not only the accuracy but also the number of spikes should be evaluated when developing the training algorithms.

In recent years, there has been extensive research on the training algorithms for SNNs. Approaches in this field roughly fall into three categories. The first category develops the bio-inspired learning algorithms that imitate the learning process of the biological brains [5]–[9]. These methods are still in the early stage, targeting simple tasks and shallow networks. The second category proposes methods to convert pre-trained deep neural networks (DNNs) to the SNNs [10]–[16], which usually use rate coding to approximate the firing rate of SNNs to the activation of DNNs. However, the rate coding scheme needs large time steps for approximation, thus reducing the power efficiency once the networks are implemented on the hardware. Moreover, the accuracy of the converted SNN is limited to that of the original DNN. As a result, the third category aims to directly train the SNNs by backpropagation [17]–[26].

In [23], a spatio-temporal backpropagation (STBP) algorithm is proposed to train SNNs, which achieves competitive accuracy compared to DNNs. This algorithm is compatible with the existing deep learning frameworks, such as TensorFlow and PyTorch, so that the training process can be accelerated by GPUs. Therefore, STBP is a promising way towards large-scale networks and real-world applications. Note that in [23], the objective is mainly to maximize the accuracy, i.e., the accuracy is the only metric for evaluation on all benchmarks. However, as previously mentioned, the number of spikes per inference is also critical to the actual efficiency when processed on the neuromorphic processors.

Therefore, this paper aims to reduce the spike count per inference without reducing the accuracy. We propose four enhancing schemes for the STBP algorithm to make each spike more informative. We first modify the integrate-and-fire neuron model to prevent information loss due to the reset mechanism, which allows the firing threshold to be a trainable variable, so that the SNN can adjust the spiking rate to the most efficient value by itself. Then, we propose two novel output spike decoding schemes to replace the original rate decoding scheme. Each output spike is prioritized and contributes differently to the loss function for effective back-propagation. Finally, we modify the derivative approximation of the non-differentiable firing function to simplify the computation of STBP. Experimental results show that the simplified computation can further improve both the accuracy and the spike count. As a result, by adopting the proposed enhancing schemes, the SNNs achieve higher accuracy and lower spike count per inference. High-performance SNN models thus can be efficiently processed on the event-driven neuromorphic processors to facilitate on-line training.

II. Spatio-Temporal Backpropagation for SNNs

In this section, we briefly introduce the spatio-temporal backpropagation (STBP) for training SNNs proposed in [22],

978-3-9819263-5-4/DATE21/©2021 EDAA

where they use two techniques for STBP to take full advantage of the spatio-temporal dynamics. First, they introduce an iterative leaky integrate-and-fire (LIF) model that is suitable for the error backpropagation algorithm. Second, they propose the derivative approximation method for the non-differentiable neuron firing function.

The integrate-and-fire (IF) model is the most commonly used neuron model for SNNs, as shown in Fig. 1a. The IF model is simply the LIF model without the leakage term. The mathematical model of the IF model can be written as:

$$V^{t,l} = V^{t-1,l}(1 - S^{t-1,l}) + \sum_{i=1}^{M^{l-1}} S_i^{t,l-1} W_i^l, \tag{1}$$

$$S^{t,l} = \begin{cases} 1, & \text{if } V^{t,l} \geq V_{th}^l \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The reset-and-integration function and the firing function are given in (1) and (2), respectively. Both equations represent the function at time step t in layer l. In (1), the *membrane potential* ($V^{t,l}$) is first reset to 0 if the neuron has fired a *spike* at the previous time step, i.e., $S^{t-1,l} = 1$, as shown in Fig. 1b. It then accumulates (integrates) the synapse weights (W^l) triggered by input spikes from the previous layer ($S^{t,l-1}$), where there are M^{l-1} inputs. After the reset-and-integration mechanism, the output spikes of layer-l neurons at the current time step ($S^{t,l}$) is set to 1 if the updated membrane potential ($V^{t,l}$) exceeds the *firing threshold* (V_{th}^l), as shown in (2). With (1) and (2), the entire network can be constructed.

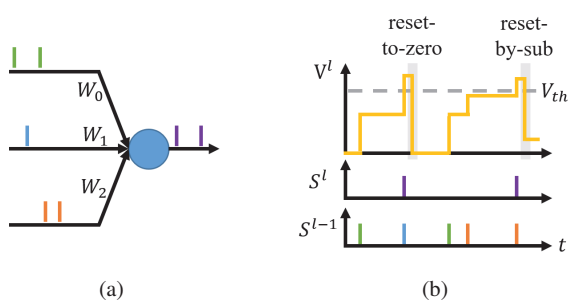

Fig. 1: Illustrations of (a) the IF model communicating through spikes, and (b) Eqs. (1), (2), and (8).

Note that backpropagation normally deals with differentiable functions only, when their gradients can be properly calculated. However, a problem in training SNNs with backpropagation is the non-differentiable firing function, i.e., Eq. (2). To obtain its gradient, STBP [23] approximates the derivative of the firing function by a simple rectangle function as shown in (3).

$$\frac{dS^{t,l}}{dV^{t,l}} = h(V^{t,l}) = \begin{cases} 1, & \text{if } |V^{t,l} - V_{th}^l| < 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Therefore, we can compute the gradients of $V^{t,l}$, $S^{t,l}$, and W_t^l by (4), (5), and (6), respectively. Based on these equations, all the gradients in SNNs can be computed accordingly through backpropagation.

$$\Delta V^{t,l} = \Delta S^{t,l} h(V^{t,l}) + \Delta V^{t+1,l}(1 - S^{t,l}) \tag{4}$$

$$\Delta S^{t,l} = -\Delta V^{t+1,l} V^{t,l} + \sum_{i=1}^{M^{l+1}} \Delta V_i^{t,l+1} W_i^{l+1} \tag{5}$$

$$\Delta W_i^l = \sum_{t=1}^{T} \Delta V^{t,l} S_i^{t,l-1} \tag{6}$$

Because the STBP algorithm can be programmed by the existing deep learning frameworks, the gradient computation is automatically calculated by these frameworks. From (3), (4), and (5), we obtain:

$$\Delta V^{t,l} = \begin{cases} \Delta V^{t+1,l}(1 - S^{t,l} - V^{t,l}) + \sum_{i=1}^{M^{l+1}} \Delta V_i^{t,l+1} W_i^{l+1} & \text{, if } |V^{t,l} - V_{th}^l| < 0.5 \\ \Delta V^{t+1,l}(1 - S^{t,l}) & \text{, otherwise.} \end{cases} \tag{7}$$

Although STBP is a powerful algorithm to train SNN, (7) shows that its computation complexity is quite high. For online learning on future SNN hardware, we choose to simplify this equation. In what follows, we focus on energy-efficient training algorithms with reasonable hardware cost.

III. Proposed Enhancing Schemes for STBP

Although the SNN trained with STBP has shown good accuracy on many benchmarks, its performance on neuromorphic processors, such as Intel Loihi [2] and IBM TrueNorth [1], has not been evaluated. For these event-driven neuromorphic processors, the number of spikes and the length of time window are proportional to their processing time and energy. Therefore, to enable energy-efficient processing on the neuromorphic hardware, not only the accuracy but also the number of spikes should be considered.

In this section, we first introduce the modified reset mechanism for the IF neuron model and then derive the corresponding formulas for computing the gradients. Based on the modified IF neuron model, we then illustrate how to make the firing threshold a trainable variable. Following that, we propose a modified derivative for the firing function to reduce the computation complexity of the gradients for the STBP algorithm. Finally, we propose two novel output spike decoding schemes.

A. Reset by Subtraction

The original reset mechanism of the IF neuron model directly resets the membrane potential to zero, as shown in (1). However, this *reset-to-zero* mechanism over-adjusts the membrane potential in most cases–when the "fire" condition is met, the membrane potential is higher than the firing threshold with a positive slack. The membrane potential stores the information from the input spikes, so the reset-to-zero operation may result in information loss. To solve this issue, we propose to replace reset-to-zero with *reset-by-subtraction* for the IF neuron model, as shown in the second reset in Fig. 1b. This is inspired by [14] that proposes to reset the membrane potential by subtracting the firing threshold. The purpose is to improve the DNN-to-SNN conversion efficiency by reducing the approximation error between the SNN firing rate and DNN activation. Although reset-by-subtraction has

been used in DNN-to-SNN conversion, it has not been applied to the direct-training method before.

With *reset-by-subtraction*, (1) is re-formulated as:

$$V^{t,l} = V^{t-1,l} - S^{t-1,l}V_{th}^{l} + \sum_{i=1}^{M^{l-1}} S_i^{t,l-1}W_i^{l} \tag{8}$$

Therefore, the gradients of $V^{t,l}$ and $S^{t,l}$ can be re-formulated as (9) and (10), respectively.

$$\Delta V^{t,l} = \Delta S^{t,l}h(V^{t,l}) + \Delta V^{t+1,l} \tag{9}$$

$$\Delta S^{t,l} = -\Delta V^{t+1,l}V_{th}^{l} + \sum_{i=1}^{M^{l+1}} \Delta V_i^{t,l+1}W_i^{l+1} \tag{10}$$

B. Trainable Firing Threshold

In the original STBP algorithm, the firing threshold V_{th}^{l} is a non-trainable and fixed parameter, since its gradient is hard to obtain. Yet, a proper V_{th}^{l} is crucial in stabilizing the firing activities of the whole network, i.e., we would like to ensure a fast response of the pre-synaptic stimulus, but avoid too many spikes that reduce the neuronal selectivity. As it is very difficult to set a perfect firing threshold by humans, we decide to make the firing threshold trainable, i.e., adjustable through the training process.

So far the gradient of V_{th}^{l} is not available, because V_{th}^{l} is not involved in the forward computation. However, reset-by-subtraction gets V_{th}^{l} involved in the computation (see (8)). The gradient of V_{th}^{l} thus can be expressed as (11), and thus V_{th}^{l} becomes a trainable variable. This is the other benefit of reset-by-subtraction in our direct-training method.

$$\Delta V_{th}^{l} = \sum_{t=2}^{T} -\Delta V^{t,l}S^{t-1,l} \tag{11}$$

C. Modified Derivative Approximation of the Firing Function

After reset-by-subtraction, (7) can be re-formulated as follows.

$$\begin{aligned}\Delta V^{t,l} = \Delta V^{t+1,l}(1-V_{th}^{l}h(V^{t,l}))+ \\ h(V^{t,l})\sum_{i=1}^{M^{l+1}} \Delta V_i^{t,l+1}W_i^{l+1}\end{aligned} \tag{12}$$

Then, from (3), we have

$$\Delta V^{t,l} = \begin{cases}\Delta V^{t+1,l}(1-V_{th}^{l}) + \sum_{i=1}^{M^{l+1}} \Delta V_i^{t,l+1}W_i^{l+1} \\ \qquad\qquad\qquad\qquad ,\text{if } |V^{t,l}-V_{th}^{l}|< 0.5 \\ \Delta V^{t+1,l} \qquad\qquad\qquad ,\text{otherwise.}\end{cases} \tag{13}$$

This equation is simpler than (7), but is still too complicated for hardware implementation. To further simply the equation, we modify the derivative approximation of the firing function, i.e., (3):

$$\frac{dS^{t,l}}{dV^{t,l}} = h(V^{t,l}) = \begin{cases}\frac{1}{V_{th}^{l}}, & \text{if } |V^{t,l}-V_{th}^{l}|< \frac{V_{th}^{l}}{2} \\ 0, & \text{otherwise.}\end{cases} \tag{14}$$

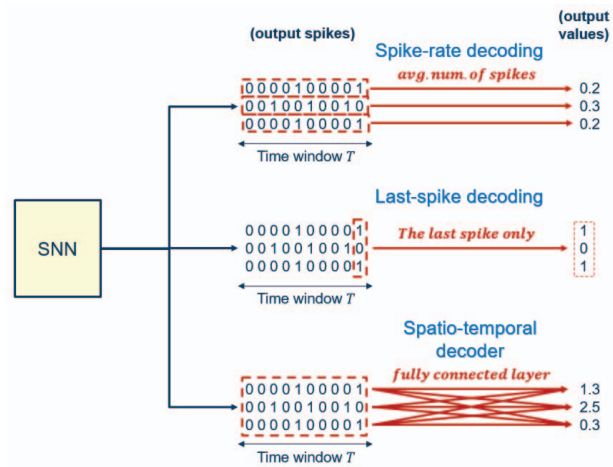

Fig. 2: Output spike decoding schemes.

Then we recalculate (13) with this modified derivative, and get the following equation:

$$\Delta V^{t,l} = \begin{cases}\sum_{i=1}^{M^{l+1}}(\Delta V_i^{t,l+1}W_i^{l+1})/V_{th}^{l} & ,\text{if } |V^{t,l}-V_{th}^{l}|< \frac{V_{th}^{l}}{2} \\ \Delta V^{t+1,l} & ,\text{otherwise.}\end{cases} \tag{15}$$

Compared to (7), (15) is simpler, and it consists of fewer operations. $\Delta V^{t,l}$ is computed with the gradients from either the spatial domain ($\Delta V^{t,l+1}$) or the temporal domain ($\Delta V^{t+1,l}$). This approach is hardware-efficient, getting us one step closer to developing on-line learning hardware. Note that this modification does not affect the accuracy of training SNNs, which will be discussed in Sec. IV.

D. Output Spike Decoding Schemes

The SNNs generate spatio-temporal spike sequences. We need to decode these binary sequences to the meaningful values, which are then computed within the loss function for backpropagation. A common decoding scheme is to average the number of spikes in a given time window. This scheme is commonly used in DNN-to-SNN conversion, and is also used in the STBP work [22]. However, we find that this decoding scheme is not suitable for STBP. Backpropagation aims to minimize the error between the output values and the labels, so that the output spikes tend to be all 1 or 0 through the training process. Based on our experience, SNNs need a period of time to warm up, so the output spikes in the first few time steps are insignificant. Trying to force the first few spikes to 1 or 0 may make the SNNs hard to converge. In other words, for STBP, we should not take the first few spikes into the training process. Based on this observation, we propose two novel output spike decoding schemes for STBP as shown in Fig. 2.

The first decoding scheme we propose is the *Last-Spike Decoding*. In this scheme, only the output spikes at the last time step are taken. This scheme is an extreme case for the previously mentioned concept, which ignores all spikes but the last ones. We find that this decoding scheme is simple and effective for STBP.

TABLE I: Network Structures for CIFAR10 Evaluation

	Network Structures
AlexNet	96C3-256C3-AP2-384C3-AP2-384C3-256C3-1024FC-1020FC-Decoding
CIFARNet	128C3-256C3-AP2-512C3-AP2-1024C3-512C3-1024FC-510FC-Decoding

The second proposed decoding scheme is the *Spatio-Temporal Decoding*. In this scheme, we consider all spikes, but each spike has a different weight on the output values. Note that in the original Average-Spike Decoding scheme, all spikes have the same impact on the output values. In the Last-Spike Decoding scheme, only the last spikes have an impact on the output values, while other spikes do not. The Spatio-Temporal Decoder takes another strategy, which aims to prioritize the importance of every spike. We implement the Spatio-Temporal Decoder by a trainable fully-connected layer, which takes all spikes as the inputs, and generates the decoded values. The weights of the spikes are adjusted through the training process.

IV. Experimental Results

A. Evaluation on Accuracy and Spike Count

We evaluate the proposed methods on the CIFAR10 classification dataset [27], which consists of 50,000 training images and 10,000 testing images in 10 classes. We use the network structures as listed in Tab. I for our evaluation, which are the same network structures used in the original STBP work [23]. In the table, nC3 represents a convolutional layer with n 3×3 kernels. AP2 means an average-pooling layer with a 2×2 pooling window and a stride of 2. nFC means a fully-connected layer with n neurons. In training and inference, we use rate coding to encode the input images to the input spikes by the IF neurons. We feed each pixel value to an IF neuron in every time step, then the IF neuron generates a spike when its membrane potential exceeds the firing threshold. As a result, the higher the pixel value is, the more spikes the IF neuron generates. Note that the firing thresholds of these input IF neurons can be trainable variables when using the proposed method mentioned in Sec. III-B.

In our experiments, Tensorflow is taken as the training framework for all neural networks. All networks are trained using the gradient descent optimizer, with a Nesterov momentum of 0.9 and a batch size of 50. The loss function is defined as the mean square error between the decoded values and the labels. For the pre-processing, we apply the usual data augmentation techniques, including horizontal flipping, pixel shifting, and rotation. The initial neuron firing threshold is set to 0.5. When the trainable firing thresholds are applied, multiple neurons in the same channel share the same firing threshold, instead of using separate firing thresholds. For example, for a layer with $h \times w \times c$ neurons, there will be $h \times w$ neurons using the same firing threshold, resulting in c firing thresholds in this layer.

Table II shows the impact of the proposed methods on accuracy and spike count per inference for AlexNet-like SNN on CIFAR10 ($T = 8$). In the "decode" columns, "last" and "FC" represent the *Last-Spike Decoding* and the *Spatio-Temporal Decoder*, respectively. As the table shows, when we apply more methods, the networks can achieve higher accuracy and lower spike count per inference, as compared with the baseline model. In other words, each spike contains more information from training with the proposed methods. In addition, the Last-Spike Decoding has almost the same effect as the Spatio-Temporal Decoder. This indicates that the output spikes at the last time step are actually the most important. Although the Spatio-Temporal Decoder achieves a little better results than the Last-Spike Decoding, it involves much more computations. This experiment also shows that the proposed derivative approximation of the firing function, as shown in (14), does not have a negative impact on the accuracy. Therefore, the computation of ΔV can be reduced without a negative effect. Finally, we also find that none of the proposed methods can be removed without reducing the improvement.

TABLE II: Impact on Accuracy and Spike Count

Reset-by-sub	Trainable V_{th}	Decode last	Decode FC	Modified dfire	Accuracy (%)	Spike count
✓					81.93	471,361
		✓			82.45	253,313
			✓		82.71	219,600
✓	✓				83.97	427,610
✓		✓			85.24	305,482
✓			✓		83.06	235,903
✓	✓	✓			88.14	215,787
✓	✓		✓		88.57	184,134
✓	✓	✓		✓	88.74	204,007
✓	✓		✓	✓	88.63	210,541
baseline (reset-to-zero, fixed V_{th}, rate decoding)					82.49	256,316

TABLE III: Comparison by Time Steps

Method	Time steps	Accuracy	Spike count
Baseline	4	81.86%	131,172
	8	82.49%	256,316
	16	82.46%	624,901
Ours (last)	4	84.75%	86,096
	8	88.74%	204,007
	16	88.65%	510,508
Ours (FC)	4	85.58%	72,593
	8	88.63%	210,541
	16	89.57%	412,163

Table III shows the training results with different numbers of time steps. From the table, the SNNs trained with the proposed enhanced STBP have higher accuracy and lower spike count than the baseline method, i.e., the spikes become more informative and contribute to higher accuracy by the proposed methods. Besides, the Spatio-Temporal Decoder benefits from the increase of time steps, since it takes all time steps into consideration. The table also shows that there is a trade-off between the SNN's accuracy and its time steps/spikes.

To better understand the importance of the output spikes at each time step, we plot the histograms and the output-spike-decoding weights of the Spatio-Temporal Decoder in Fig. 3, where Figs. 3a and 3b are for training with the total time steps of 8, while Figs. 3c and 3d are for training with the total time steps of 16. Note that the absolute weight values

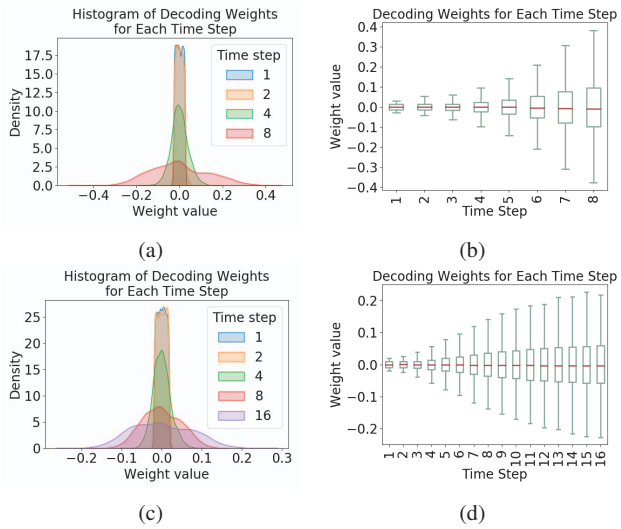

Fig. 3: Histograms and the output-spike-decoding weights of the *Spatio-Temporal Decoder* for each time step: (a) and (b) are for T=8; and (c) and (d) are for T=16.

TABLE IV: Comparison with the DNN-to-SNN Method

Training method	Total time steps	Accuracy (%)	Spike count	Spike count per time step
Convert from pre-trained	8	12.77	220,314	27,539
	16	28.93	463,794	28,987
	32	78.84	931,499	29,109
	64	89.88	1,867,580	29,180
	128	91.78	3,741,644	29,231
Ours (last)	8	88.74	204,007	25,500
Ours (FC)	8	88.63	210,541	26,317

are larger for the spikes in the later time steps, and smaller for the spikes in the earlier time steps. Therefore, the later spikes are more critical to the final predictions, which confirms the effectiveness of our novel Last-Spike Decoding scheme.

For comparison, we also use the DNN-to-SNN conversion method proposed in [14] to get an AlexNet-like SNN. Table IV shows the results of accuracy and spike count per inference with different time steps. In the table, we can see that the DNN-converted SNN can achieve higher accuracy than the STBP-trained SNN when the time steps are long enough, e.g., higher than 64 in this case. However, the spike count of the DNN-converted SNN is much higher than that of the STBP-trained SNN at the same accuracy level. Taking both the accuracy and the number of spikes into account, the direct-training scheme is considered a better choice in most cases than the DNN-to-SNN conversion scheme.

B. Classification Performance

Table V shows the comparison with the original STBP method [23]. As the table shows, our enhanced methods achieve better accuracy when training the AlexNet-like SNN. While [23] applies NeuNorm to train their SNN, which introduces additional parameters and non-spiking operations to each layer, our result using the Last-Spike Decoding achieves better accuracy, without additional parameters and

TABLE V: Comparison with the Original STBP Method [23]

Structure	STBP method	Accuracy	Spike count
AlexNet	original w/ NeuNorm	85.24% [23]	-
	our enhanced (last, T=8)	88.74%	204,007
	our enhanced (FC, T=16)	89.57%	412,163
CIFARNet	original w/o NeuNorm	89.83% [23]	-
	original w/ NeuNorm	90.53% [23]	-
	our enhanced (last, T=8)	89.40%	203,220
	our enhanced (FC, T=16)	90.13%	341,590

operations. We can further improve the accuracy using the Spatio-Temporal Decoder, although additional parameters are required to decode the output spikes. We also find that larger networks do not tend to create more spikes, e.g., CIFARNet contains more synapses and neurons than AlexNet, but it achieves higher accuracy and lower spike count.

However, when training the CIFARNet structure, our results are close to the original method, which can probably be caused by choices of the implementation details. For instance, [23] applies an additional decay factor in the IF neuron model, while we disable it by setting it to 1 to simplify the computation. Also, [23] assigns the first layer as an encoding layer by using a non-spiking layer. Therefore, they can retain the precision of the input pixel values despite the small number of time steps. In comparison, our implementation simply uses rate coding to turn the input values into input spikes, because we want all layers to be spiking layers in the SNN. As a result, the precision of the input value is limited to the length of the time steps. Theoretically, at least 256 time steps are required to encode an 8-bit pixel value for rate coding, but we use at most 16 time steps, so our results can be limited to the resolution of the inputs.

Table VI shows the comparison with other direct-training methods using the same CIFARNet structure. To fairly compare with the original STBP [23], we use the same encoding scheme for training. In this case, our enhanced STBP achieves slightly better accuracy than the original STBP without NeuNorm. Although the original STBP with NeuNorm shows slightly better accuracy than our results, NeuNorm introduces additional parameters and non-spiking operations to each layer. As for the comparison with SpikeGrad [24], our results are very close. However, SpikeGrad uses ternary spikes, i.e., -1, 0, and 1, while we only use binary spikes, i.e., 0 and 1. Ternary spikes are thought to be more informative, but the extra negative spike can increase the total spike count and hardware overhead. We show the spike count per inference, which is an important metric to evaluate the efficiency of these SNNs employed in the neuromorphic processors. However, the previous works did not report this metric, so we can only compare the accuracy.

V. Conclusion

In this paper, we present four enhancing schemes for STBP to train high-accuracy and low-spike-count SNNs. We introduce reset-by-subtraction to STBP to solve the information loss issue and make the firing thresholds trainable. Besides, we propose two novel output spike decoding schemes to prioritize the spikes at each time step for effective training.

TABLE VI: Comparison with Other Direct-Training Methods

Training method	Accuracy	Spike count
STBP w/o NeuNorm [23]	89.83%	-
STBP w/ NeuNorm [23]	90.53%	-
SpikeGrad [24]	89.99%	-
Ours (last, T=8)	89.40%	203,220
Ours (FC, T=16)	90.13%	341,590
Ours (last, T=8, same encoding as [23])	90.06%	324,525

Finally, we modify the derivative approximation of the firing function to simplify the computation of STBP for future online learning hardware implementation. Experimental results show that we can improve both the accuracy and the spike count per inference. Each spike contributes to more accuracy and becomes more informative by training with our enhanced STBP. We also show comparable accuracy with other direct-training methods. Moreover, processing energy and time are expected to be reduced when our low-spike-count SNNs are employed in the neuromorphic processors.

Acknowledgement

This work was supported in part by the Ministry of Science and Technology under Grant 109-2218-E-007-025, and the Industrial Technology Research Institute under Grant 109A-2069-EA.

References

[1] S. K. Essera, P. A. Merollaa, J. V. Arthura, A. S. Cassidya, R. Appuswamya, A. Andreopoulosa, D. J. Berga, J. L. McKinstrya, T. Melanoa, D. R. Barcha, *et al.*, "Convolutional networks for fast energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 41, pp. 11441–11446, 2016.

[2] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[3] L. Deng, G. Wang, G. Li, S. Li, L. Liang, M. Zhu, Y. Wu, Z. Yang, Z. Zou, J. Pei, *et al.*, "Tianjic: A unified and scalable chip bridging spike-based and continuous neural computation," *IEEE Journal of Solid-State Circuits*, 2020.

[4] P.-Y. Chuang, P.-Y. Tan, C.-W. Wu, and J.-M. Lu, "A 90nm 103.14 tops/w binary-weight spiking neural network cmos asic for real-time object classification," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020.

[5] S. R. Kheradpisheh, M. Ganjtabesh, and T. Masquelier, "Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition," *Neurocomputing*, vol. 205, pp. 382–392, 2016.

[6] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier, "Combining stdp and reward-modulated stdp in deep convolutional spiking neural networks for digit recognition," *arXiv preprint arXiv:1804.00227*, vol. 1, 2018.

[7] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh, "First-spike-based visual categorization using reward-modulated stdp," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 12, pp. 6178–6190, 2018.

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

[9] C. Lee, P. Panda, G. Srinivasan, and K. Roy, "Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning," *Frontiers in neuroscience*, vol. 12, p. 435, 2018.

[10] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015.

[11] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, ieee, 2015.

[12] E. Hunsberger and C. Eliasmith, "Spiking deep networks with lif neurons," *arXiv preprint arXiv:1510.08829*, 2015.

[13] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci, "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–8, IEEE, 2016.

[14] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, p. 682, 2017.

[15] A. Balaji, F. Corradi, A. Das, S. Pande, S. Schaafsma, and F. Catthoor, "Power-accuracy trade-offs for heartbeat classification on neural networks hardware," *Journal of Low Power Electronics*, vol. 14, no. 4, pp. 508–519, 2018.

[16] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, p. 95, 2019.

[17] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Advances in neural information processing systems*, pp. 1117–1125, 2015.

[18] D. Huh and T. J. Sejnowski, "Gradient descent for spiking neural networks," in *Advances in Neural Information Processing Systems*, pp. 1433–1443, 2018.

[19] Y. Jin, W. Zhang, and P. Li, "Hybrid macro/micro level backpropagation for training deep spiking neural networks," in *Advances in neural information processing systems*, pp. 7005–7015, 2018.

[20] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[21] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multilayer spiking neural networks," *Neural computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[22] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in neuroscience*, vol. 12, p. 331, 2018.

[23] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1311–1318, 2019.

[24] J. C. Thiele, O. Bichler, and A. Dupret, "Spikegrad: An ann-equivalent computation model for implementing backpropagation with spikes," *arXiv preprint arXiv:1906.00851*, 2019.

[25] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in Neuroscience*, vol. 14, 2020.

[26] S. R. Kheradpisheh and T. Masquelier, "S4nn: temporal backpropagation for spiking neural networks with one spike per neuron," *International Journal of Neural Systems*, vol. 30, no. 6, p. 2050027, 2020.

[27] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.