A Noise-Based Novel Strategy for Faster SNN Training

Chunming Jiang
cji39@uclive.ac.nz
Yilei Zhang*
yilei.zhang@canterbury.ac.nz
Department of Mechanical Engineering, University of Canterbury, Canterbury CT2 7NX, New Zealand

Spiking neural networks (SNNs) are receiving increasing attention due to their low power consumption and strong bioplausibility. Optimization of SNNs is a challenging task. Two main methods, artificial neural network (ANN)-to-SNN conversion and spike-based backpropagation (BP), both have advantages and limitations. ANN-to-SNN conversion requires a long inference time to approximate the accuracy of ANN, thus diminishing the benefits of SNN. With spike-based BP, training high-precision SNNs typically consumes dozens of times more computational resources and time than their ANN counterparts. In this letter, we propose a novel SNN training approach that combines the benefits of the two methods. We first train a single-step SNN ($T = 1$) by approximating the neural potential distribution with random noise, then convert the single-step SNN ($T = 1$) to a multistep SNN ($T = N$) losslessly. The introduction of gaussian distributed noise leads to a significant gain in accuracy after conversion. The results show that our method considerably reduces the training and inference times of SNNs while maintaining their high accuracy. Compared to the previous two methods, ours can reduce training time by 65% to 75% and achieves more than 100 times faster inference speed. We also argue that the neuron model augmented with noise makes it more bioplausible.

1 Introduction

Spiking neural networks (SNNs) recently attracted increasing attention due to their biological plausibility. The SNN incorporates the concept of time into the model, and neurons in the SNN receive input spike trains that either increase or decrease their membrane potential. Through temporal accumulation, membrane potential may reach a specific firing threshold and neurons transmit information by firing discrete spike trains to neurons in other neurons.

*Yilei Zhang is the corresponding author.
the next layer. These characteristics emulate the information transmission and processing in the brain. It is therefore regarded as the next-generation neural network (Tavanaei et al., 2019).

Since SNNs use nondifferentiable spikes as information-carrying agents, gradient-based backpropagation (BP) that uses gradients to optimize synaptic connections and neuron parameters in ANNs is not directly applicable in SNNs. Thus, one of the main challenges is to train and optimize the network parameters in SNNs. At present, the available methods for training SNNs can be divided into three categories: (1) unsupervised learning, (2) indirect supervised learning, and (3) direct supervised learning.

In the first approach, weights are modulated to mimic synaptic interactions of biological neurons. A classic example is the spike-time-dependent plasticity (STDP; Diehl & Cook, 2015; Kheradpisheh et al., 2016; Querlioz et al., 2013). However, due to the reliance on local neuronal activity rather than global supervision, STDP-based unsupervised algorithms have been limited to training shallow SNNs and can produce only low accuracy on complex data sets (Han et al., 2020; Masquelier & Thorpe, 2007; Srinivasan et al., 2018).

In the second approach, an ANN model is first trained and then converted to an SNN with the same network structure, where the firing rate of the SNN neuron is approximated as the analog output of the ANN neuron. The ANN-to-SNN conversion has produced state-of-the-art (SOTA) performance in image recognition tasks (Roy et al., 2019).

The last approach is direct supervised learning, which uses a similar gradient descent technique used in ANNs to train SNNs directly. SpikeProp (Bohte et al., 2002) was the first BP-based supervised learning method for SNNs that uses a linear approximation to overcome the SNNs’ nondifferentiable, threshold-triggered firing mechanism. Further works include Tempotron (Gutig & Sompolinsky, 2006), ReSuMe (Ponulak & Kasinski, 2010), and SPAN (Mohemmed et al., 2012). However, they could be used only for training single-layer SNNs. The SuperSpike (Zenke & Ganguli, 2018) algorithm is designed to improve the accuracy of SNNs by optimizing the timing of the spikes. The algorithm uses a variant of backpropagation to adjust the weights of the network based on the timing of the spikes. A method proposed by Huh and Sejnowski (2018) also optimizes the spiking network dynamics for general supervised tasks on the timescale of individual spikes as well as the behavioral timescales. However, these methods still have a large performance gap with ANN when optimizing deep SNNs. A surrogate gradient algorithm proposed by Wu et al. (2018) introduces a differentiable surrogate function to approximate the derivative of spiking activity. It executes spatiotemporal BP in the training phase and is widely applied to train deep SNNs.

Although the ANN-to-SNN conversion and surrogate gradient-based algorithm can train deep SNNs, there are some limitations. For the ANN-to-SNN conversion, training an ANN model is fast. Nevertheless, the
approach requires considerable inference time (from hundreds to thousands of time steps) to approximate the analog outputs (Cao et al., 2015; Hunsberger & Eliasmith, 2015; Rueckauer et al., 2017; Sengupta et al., 2019; Stöckl & Maass, 2021), which leads to high memory consumption, larger latency, and decreased energy efficiency, diminishing the benefits of SNNs (Deng & Gu, 2021; Roy et al., 2019; Severa et al., 2019). For the surrogate gradient-based direct training method, although it is possible to train SNNs with arbitrary time steps, the fewer the time steps, the lower the accuracy of the trained model would be. Training high-accuracy SNNs with this approach often requires many times more training time and computational resources than training ANNs.

In this letter, we propose a novel SNN training method that combines the conversion concept and direct training using the surrogate gradient. The method consists of two phrases: single-step SNN($T = 1$) direct training and conversion to multistep SNN($T = N$). Specifically, during the training phase, SNN ($T = 1$) augmented with gaussian noise is trained by surrogate gradient-based BP and then converted losslessly to a multistep SNN model to promote its generalization capability. Our training technique not only greatly reduces training and inference time but also achieves high accuracy, which significantly improves the operating efficiency of SNN.

The primary contributions of this letter follow:

1. We propose a novel noise-based SNN training algorithm, which speeds up the training and inference time of SNN.
2. We compare our method’s training and inference time with those of current methods. The experiments demonstrate that our method is three to five times faster for training than the surrogate-gradient-based method and more than 100 times faster than the ANN-to-SNN conversion for inference.
3. We argue that introducing noise in SNN has biological plausibility.

2 Methods

2.1 Leaky Integrate-and-Fire Model. The leaky integrate-and-fire (LIF) model is a fundamental unit in SNNs. It is a simplified representation of biological neurons that describes the nonlinear relationship between input and output. The LIF neuron receives spikes over a specific period and integrates them into its membrane potential, whose dynamics are governed by

$$ H(t) = \lambda \cdot V(t - 1) + \sum_{i} w_i \cdot S_i(t) $$

(2.1)

$$ S(t) = \begin{cases} 1, & H(t) > V_{th} \\ 0, & H(t) \leq V_{th} \end{cases} $$

(2.2)
\[ V(t) = H(t)(1 - S(t)) + V_{\text{reset}} \cdot S(t), \tag{2.3} \]

where \( H(t) \) and \( V(t) \) represent the membrane potentials before and after triggering a spike at time \( t \), respectively. \( \lambda \) represents the decay factor with a value of 0.5. \( S(t) \) denotes the output of a neuron at time \( t \), which equals 1 if there is a spike and 0 otherwise. \( w_i \cdot S_i(t) \) is the weighted input of \( i \)th neuron in the last layer at time step \( t \). When the membrane potential of the LIF neuron reaches the firing threshold \( V_{th} (= 1) \), the neuron fires one spike and the membrane potential is reset to the resting potential \( V_{\text{reset}} \) (here is 0).

2.2 Surrogate Gradient Method. The method of surrogate gradient defines the derivative of the threshold-triggered firing mechanism using a surrogate derivative, which was proposed and developed to overcome the nondifferential property of spiking neurons (Neftci et al., 2019; Wu et al., 2018). It approximates the gradient of the spike function, which allows for the computation of gradients and the application of traditional optimization algorithm stochastic gradient descent (SGD). After the training process is completed, the final weights obtained from the surrogate gradient method can be used to produce spikes in the actual SNN, allowing it to function as intended. The surrogate gradient method has been shown to be effective in training SNNs on a variety of tasks and has been widely used in the development of SNN-based systems.

The form of surrogate gradient function varies. In this letter, we use equation 2.4 \((a = 3)\) to approximate the gradient of spike function, which is plotted in Figure 1:

\[ y(x) = \frac{a}{2 \left(1 + (\frac{a}{2} \cdot (x - V_{th}))^2 \right)} \tag{2.4} \]

2.3 Training Single-Step SNN\((T = 1)\) and Converting to Multistep SNN\((T = N)\). We first train the single-step SNN \((T = 1)\) and then convert it to a multistep SNN \((T = N)\). When the total simulation step \( T = 1 \), the time dimension disappears and the network propagates forward only once. Consequently, the single-step SNN is actually an ANN with a Heaviside step function used as the activation function. Equation 2.1 is formulated as

\[ H = \sum_i w_i \cdot S_i \tag{2.5} \]

Comparing equations 2.1 and 2.2, the output of the multistep LIF neuron depends on both the input and the accumulated potential, while the output of a single-step LIF neuron depends only on the input.

Due to the absence of the potential accumulation term in equation 2.5 compared with equation 2.1, we introduce a noise distribution \( N_{\text{noise}} \).
representing the missing accumulated membrane potential during training the single-step SNN in order to perform the conversion into a multi-step SNN later, as equation 2.6 shows. In particular, we assume that $N_{\text{noise}}$ is a gaussian distribution and distributes in each layer of the network independently. Thus, the dynamic of the neuron in a single-step SNN could be described by

$$H = N_{\text{noise}} + \sum_i w_i \cdot S_i. \quad (2.6)$$

Our method consists of the three following stages:

Stage 1: Train the single-step SNN ($T = 1$) with gaussian noise (see stage 1 in Figure 2) by the surrogate gradient. Notably, the added noise $N_{\text{noise}}$ varies randomly for each training iteration. It is not the same frozen noise over iterations. After training is completed, all parameters except $N_{\text{noise}}$ are frozen, and there is no gradient calculation or training in stages 2 and 3.

Stage 2: Change the value of simulation time from $T = 1$ to $T = N$ by simply modifying the value of $T$. When $T = N$, the input is fed $N$ times in the time dimension into the SNN model, and the noise at each time step $t$ is randomly varying. We average the outputs of $N$ times as the final output of the SNN model (see stage 2 in Figure 2).

Stage 3: Add the potential accumulation term using equation 2.7 to establish the temporal correlation among different time steps (see stage 3.

Figure 1: Surrogate gradients for the Heaviside function used in our experiments. The orange line represents the step function responsible for spike generation, and the blue line approximates its derivative (surrogate gradient).
in Figure 2). The dynamic of SNN model after step 2 is formally different from the real SNN’s dynamic because it lacks the process of potential accumulation and the temporal correlation among different time steps. Consequently, we must add the item $\lambda \cdot V(t - 1)$ to keep the formal consistency with equation 2.1. We decompose $N_{\text{noise}}$ into

$$N_{\text{noise}} = N(\lambda \cdot V(t - 1)) + N'_{\text{noise}}.$$  \hspace{1cm} (2.7)

$N_{\text{noise}}$ could be represented as the addition of two gaussian distribution items: the accumulated membrane potential distribution and the new noise distribution.

For the first item, we normalize $\lambda \cdot V(t - 1)$ according to equations 2.8 to 2.10 to approximate a gaussian distribution $N(\lambda \cdot V(t - 1))$. The reason is that we have added the gaussian noise distribution $N_{\text{noise}}$ in stage 1 and assume that it has an approximate distribution with the potential accumulation term. But the actual potential accumulation term $\lambda \cdot V(t - 1)$ does not necessarily obey a gaussian distribution, so we constrain the potential accumulation term in stage 3 to an approximate gaussian distribution by the item $N(\lambda \cdot V(t - 1))$ in equation 2.7. In this way, we can introduce the noise term $N_{\text{noise}}$ for training with $T = 1$ in stage 1 and replace it with the potential accumulation term after training is completed.

$$A = \lambda \cdot V(t - 1),$$ \hspace{1cm} (2.8)

$$\hat{A} = \frac{A - \mu}{\sigma},$$ \hspace{1cm} (2.9)

$$N(\lambda \cdot V(t - 1)) = \frac{\hat{A}}{\alpha \cdot \max(\text{abs}(\hat{A}))} + \beta.$$ \hspace{1cm} (2.10)
Table 1: Network Structures and Training Epoch for Different Data Sets.

| Data Set     | Epoch | Network Structure          |
|--------------|-------|----------------------------|
| MNIST        | 100   | 64C3-AP2-128C3-AP2-128C3-AP2-512FC-10FC |
| Fashion-MNIST| 200   |                             |
| CIFAR-10     | 1000  | VGG-16                     |
|              | 800   | ResNet-18                  |

Note: nC3: Convolutional layer with $n$ output channels, kernel size = 3 and stride = 1. AP2-2D: Average-pooling layer with kernel size = 2 and stride = 2. FC: Fully connected layer.

Equation 2.9 converts the membrane potential distribution to a standard normal distribution approximately. $\mu$ and $\sigma$ are the mean and standard deviation of $A$, respectively. Equation 2.1 guarantees that the potential distribution in each layer has the mean $\beta$ and distributes between the interval $(-1/\alpha, 1/\alpha)$. By changing the values of $\alpha$ and $\beta$, we are able to change the mean and range of the distribution. (max() and abs() represent taking the maximum value and absolute value, respectively.)

For the second item, $N'_{\text{noise}}$, we also keep it as a random gaussian distribution. The mean and range of $N'_{\text{noise}}$ also depend on $\alpha$ and $\beta$, because we need to guarantee that the two items added in equation 2.7 almost have the same mean and range as $N_{\text{noise}}$ to avoid conversion loss. The reason we keep the random noise distribution $N'_{\text{noise}}$ is that introducing the noise plays the important role in promoting accuracy, which we discuss in detail in section 3.

It is well known that a SNN model with more simulation steps $T$ can increase performance. However, training an SNN with large $T$ directly would increase not only the training and inference time but also the memory by $T$ folds, so it is not very practical. In our proposed method, the whole training process is completed in stage 1. In stages 2 and 3, we need to change the value of simulation time from $T = 1$ to $T = N$ and replace $N_{\text{noise}}$ using equation 2.7. The approach allows training SNNs using only one time step, which make it possible to train models with much less memory cost.

3 Results and Discussion

We conduct our experiments as Table 1 shows. SNNs are trained with MSE loss and an Adam (Kingma & Ba, 2014) optimizer. The initial learning rate is set to $1e-4$. The cosine annealing warm restart (Loshchilov & Hutter, 2016) learning rate scheduler with $T_{\text{max}} = 100$ adjusts the learning rate over training. Unless specified, all results are generated by default for $\alpha = 4$, $\beta = 0.5$, and $N_{\text{noise}}$ in the range $[0, 1]$. 

3.1 Inference Accuracy. In Figure 3, we plot the inference accuracy of single-step SNNs ($T = 1$) on different data sets and the inference accuracy of multistep SNNs with total simulation steps $T = 5$ and $T = 10$, respectively. It can be shown that as $T$ increases, the accuracy of the SNN improves dramatically. The simulation step $T$ could be directly changed to any values in real time. The fluctuation of curves in the figure is because the scheduler with warm restart resets the learning rate every 100 epochs.

3.2 Comparison of Training and Inference Time with Related Work. In Figure 4, we compare the training speed of the direct training method to train SNN ($T = 10$) models with our method of training. The $x$-axis represents the training time, and the $y$-axis represents the inference accuracy. The direct training method directly trains SNN models with $T = 10$ by the surrogate gradient, while our method executes stages 1 to 3 described in section 2.3. Clearly our method is substantially faster than the direct training method with the same training time.
To intuitively assess the difference in training time, we selected a benchmark inference accuracy for each model and halted training when benchmark inference accuracy was achieved. The benchmark inference accuracy of the three models is 92%, 88%, and 90%, respectively. As shown in Figure 5, on FashionMNIST, our method saves about 75% of the time: it takes only 135 s to reach the benchmark accuracy when direct training takes 528 s. In CIFAR10/VGG-16, our method requires 3132 s, whereas the other requires 9900 s, a time savings of about 70%. In CIFAR10/ResNet-18, our model requires 7290 s while direct training requires 19,998 s, a time savings of approximately 65%.

We consider that the addition of noise during training makes the LIF neurons sensitive to the initial potential values inside the neurons at each time $t$, so that when we change the total simulation time $T = 1$ to $T = N$ in stage 2, the SNN model can be seen as the parallelization of $N$ different models in the time dimension to get accuracy improvement, which is similar to the concept of ensemble learning except that the ensemble learning
Figure 5: Comparison of training time for training SNN ($T = 10$) by the direct training method versus our method. The benchmark inference accuracy of the three models is 92%, 88%, and 90%, respectively.

is the combination of several different models in space, while our method uses the ensemble of the same SNN models with different initial potential states in different time steps.

Therefore, our method needs to train only an SNN ($T = 1$) model, which can be quickly converted into a model with SNN ($T = 10$). It is obviously faster than directly training a SNN ($T = 10$) model by the surrogate gradient. Also, since only a single-step SNN ($T = 1$) needs to be trained in our method, we can choose a larger batch size and thus achieve faster parallel training. It is more convenient and feasible for groups that lack sufficient computational resources.

For inference time, we compare current advanced methods listed in Table 2 with our method. As demonstrated, the accuracy of extending to multistep SNN (no more than 10 time steps) is able to attain an approximate accuracy of ANN-to-SNN conversion methods. In contrast, ANN-to-SNN conversion requires hundreds to thousands of time steps, which is hundreds of times slower than our method. Compared with spike-based BP methods, our method also requires fewer time steps to reach close accuracy.

3.3 The Impact of Noise. With equations 2.7 and 2.10, we know that the parameter $\alpha$ controls the range of noise fluctuation $N'_{\text{noise}}$. Here, we attempt to alter the value of $\alpha$ to observe how the model’s performance varies.
Figure 6: The impact of $\alpha$ on CIFAR10/VGG-16 and CIFAR10/ResNet-18. The black dashed line represents the accuracy of trained single-step SNN ($T = 1$).

We plot the accuracy in stage 3 for different values of $\alpha$ in Figure 6. When $\alpha$ is equal to 2, the potential range is $[0, 1]$, so the noise term $N_{\text{noise}}'$ in equation 2.7 no longer exists. $N_{\text{noise}} = N(\lambda \cdot V(t - 1))$ and the SNN does not include noise in stage 3. It can be seen that there is only a slight improvement in accuracy with increasing time steps. In contrary, when $\alpha > 2$, $N_{\text{noise}}'$ is present; the accuracy improvement is obvious, and different $\alpha$ values make models converge to close accuracy. These results indicate that noise plays the vital role in enhancing the accuracy of conversion.
When $\alpha$ is positive infinity, membrane potential item disappears so $N_{\text{noise}}' = N_{\text{noise}}$; the curve represents the accuracy of stage 2. Comparing this curve with others, we can see that the conversion from stage 2 to stage 3 has minor accuracy gap according to Figure 6.

3.4 The Impact of Noise Type. In the previous sections, all our experiments were performed by training the SNN model with gaussian noise. In order to investigate whether it is only the uniform noise that brings the improvement of model accuracy, we replace the gaussian noise with uniform noise during training to observe the effect of the noise type on models. As shown in Figure 7, we trained CIFAR-10 with uniform noise on both VGG-16 and ResNet-18 models. Models trained by uniform noise behave the same as with uniform noise: the accuracy is significantly improved when models are extended to multistep SNNs. For single-step SNNs, models with gaussian noise reach higher accuracy than those with uniform noise. But after conversion, the gap is no longer obvious.
Figure 8: (a) Neural potential dynamic in the absence of input. (b) Neural potential dynamic when receiving input. (c) Subthreshold membrane potential oscillation. Source: Figure (c) is cited from (Boehmer et al., 2000).

### 3.5 Biological Plausibility of Uniform Noise Distribution in Neurons.

It is believed to be more biologically plausible when we keep some part of noise during conversion: biological neural systems have lots of kinds of noise. We plot the dynamic of the spiking neuron in Figures 8a and 8b. Figure 8a depicts the neural potential dynamic in the absence of input. The potential oscillates between the reset potential and the threshold, but no spikes fire. In Figure 8b, when a neuron receives inputs, it begins to accumulate potential and fires spikes. Such behavior is thought to be similar to the form of the subthreshold neural oscillation mechanism in biological neurons. Neural oscillations are rhythmic patterns of activity generated by the neurological system (Başar, 2013). Many cognitive activities, including information transfer, perception, and memory, are believed to be related to neural oscillations. These oscillations are mostly caused by the interaction of individual neurons. Neural oscillation can emerge as oscillating membrane potentials or rhythmic action potentials in individual neurons. Subthreshold membrane potential oscillations are membrane oscillations that are below the firing threshold and hence cannot directly initiate action potentials. However, they can aid in sensory signal processing. As a result of subthreshold membrane potential oscillations, sensory systems, particularly for vision and smell, evolve. Subthreshold membrane potential oscillation (see Figure 8c) in the visual system helps process visual input and adjust to sensory input (Purves et al., 2008). Additionally, oscillatory
activity influences excitatory postsynaptic potentials, refining postneural activities (Desmaisons et al., 1999).

4 Discussion and Conclusion

We have used the proposed method to handle static vision problems. For dynamic vision problems, such as video data, which include time series information, we should feed all the information into the network at once and use a 3D convolutional network rather than a 2D convolution network to deal with the input.

In this letter, we propose a novel way of training SNNs that achieves accuracy improvement in multistep SNNs by fitting the neural network to noise, which greatly spares the training and inference time of SNNs and allows fast training of SNNs with arbitrary simulation time compared to previous methods. Our approach combines the advantages of both direct training of SNN and ANN-to-SNN conversion. With a good balance of accuracy and training time and a great saving of computational resources, this method can be used to train large SNNs quickly or pretrain SNNs. The inclusion of noise is also proved to be more consistent with the dynamic mechanism of biological neurons. These points make our method promising for training deep SNN. In addition, we focus on computer vision classification tasks and convolutional neural networks in this letter. In subsequent work, our algorithms could be further validated in more tasks, such as natural language processing, and by combining SNNs with more structures, like Transformer.

Acknowledgments

We gratefully acknowledge the financial support provided by the University of Canterbury, New Zealand.

References

Ba¸sar, E. (2013). Brain oscillations in neuropsychiatric disease. Dialogues in Clinical Neuroscience, 15(3), 291.
Boehmer, G., Greffrath, W., Martin, E., & Hermann, S. (2000). Subthreshold oscillation of the membrane potential in magnocellular neurones of the rat supraoptic nucleus. Journal of Physiology, 526(pt. 1), 115. 10.1111/j.1469-7793.2000.t01-1-00115.x
Bohte, S. M., Kok, J. N., & La Poutre, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing, 48(1–4), 17–37. 10.1016/S0925-2312(01)00658-0
Cao, Y., Chen, Y., & Khosla, D. (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. International Journal of Computer Vision, 113(1), 54–66. 10.1007/s11263-014-0788-3
Cheng, X., Hao, Y., Xu, J., & Xu, B. (2020). LISNN: Improving spiking neural networks with lateral interactions for robust object recognition. In Proceedings of the International Joint Conference on Artificial Intelligence.

Deng, S., & Gu, S. (2021). Optimal conversion of conventional artificial neural networks to spiking neural networks. arXiv:2103.00476.

Desmaisons, D., Vincent, J.-D., & Lledo, P.-M. (1999). Control of action potential timing by intrinsic subthreshold oscillations in olfactory bulb output neurons. Journal of Neuroscience, 19(24), 10727–10737. 10.1523/JNEUROSCI.19-24-10727.1999

Diehl, P. U., & Cook, M. (2015). Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In Proceedings of the International Joint Conference on Neural Networks.

Gutig, R., & Sompolinsky, H. (2006). The tempotron: A neuron that learns spike timing–based decisions. Nature Neuroscience, 9(3), 420–428. 10.1038/nn1643

Han, B., Srinivasan, G., & Roy, K. (2020). RMP-SNN: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Huh, D., & Sejnowski, T. J. (2018). Gradient descent for spiking neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), Advances in neural information processing systems, 31. Curran.

Hunsberger, E., & Eliasmith, C. (2015). Spiking deep networks with LIF neurons. arXiv:1510.08829.

Kheradpisheh, S. R., Ganjtabesh, M., & Masquelier, T. (2016). Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition. Neurocomputing, 205, 382–392. 10.1016/j.neucom.2016.04.029

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv:1412.6980.

Lee, C., Sarwar, S. S., Panda, P., Srinivasan, G., & Roy, K. (2020). Enabling spike-based backpropagation for training deep neural network architectures. Frontiers in Neuroscience, 14.

Loshchilov, I., & Hutter, F. (2016). SGDR: Stochastic gradient descent with warm restarts. arXiv:1608.03983.

Masquelier, T., & Thorpe, S. J. (2007). Unsupervised learning of visual features through spike timing dependent plasticity. PLOS Computational Biology, 3(2), e31. 10.1371/journal.pcbi.0030031

Mohemmed, A., Schlibs, S., Matsuda, S., & Kasabov, N. (2012). SPAN: Spike pattern association neuron for learning spatio-temporal spike patterns. International Journal of Neural Systems, 22(4), 1250012. 10.1142/S0129065712500128

Neftci, E. O., Mostafa, H., & Zenke, F. (2019). Surrogate gradient learning in spiking neural networks. IEEE Signal Processing Magazine, 36, 61–63. 10.1109/MSP.2019.2931595

Ponulak, F., & Kasinski, A. (2010). Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting. Neural Computation, 22(2), 467–510. 10.1162/neco.2009.11-08-901
Purves, D., Cabeza, R., Huettel, S. A., LaBar, K. S., Platt, M. L., Woldorff, M. G., & Brannon, E. M. (2008). *Cognitive neuroscience*. Sinauer.

Querlioz, D., Bichler, O., Dollfus, P., & Gamrat, C. (2013). Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Transactions on Nanotechnology, 12*(3), 288–295. 10.1109/TNANO.2013.2250995

Rathi, N., & Roy, K. (2021). DIET-SNN: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems, PP*(99), 1–9.

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

Roy, K., Jaiswal, A., & Panda, P. (2019). Towards spike-based machine intelligence with neuromorphic computing. *Nature, 575*(7784), 607–617. 10.1038/s41586-019-1677-2

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., & Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience, 11*, 682. 10.3389/fnins.2017.00682

Sengupta, A., Ye, Y., Wang, R., Liu, C., & Roy, K. (2019). Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience, 13*, 95. 10.3389/fnins.2019.00095

Severa, W., Vineyard, C. M., Dellana, R., Verzi, S. J., & Aimone, J. B. (2019). Training deep neural networks for binary communication with the whetstone method. *Nature Machine Intelligence, 1*(2), 86–94. 10.1038/s42256-018-0015-y

Srinivasan, G., Panda, P., & Roy, K. (2018). STDP-based unsupervised feature learning using convolution-over-time in spiking neural networks for energy-efficient neuromorphic computing. *ACM Journal on Emerging Technologies in Computing Systems, 14*(4), 1–12. 10.1145/3266229

Stöckl, C., & Maass, W. (2021). Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nature Machine Intelligence, 3*(3), 230–238. 10.1038/s42256-021-00311-4

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., & Maida, A. (2019). Deep learning in spiking neural networks. *Neural Networks, 111*, 47–63. 10.1016/j.neunet.2018.12.002

Wu, Y., Deng, L., Li, G., Zhu, J., & Shi, L. (2018). Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience, 12*, 331. 10.3389/fnins.2018.00331

Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., & Shi, L. (2019). Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zenke, F., & Ganguli, S. (2018). Superspike: Supervised learning in multilayer spiking neural networks. *Neural Computation, 30*(6), 1514–1541. 10.1162/neco_a_01086

Received November 10, 2022; accepted May 7, 2023.