Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Deep neural networks with weighted spikes

Jaehyun Kim[a], Heesu Kim[a], Subin Huh[a], Jinho Lee[b], Kiyoung Choi[a,*]

[a] *Department of Electrical and Computer Engineering, Seoul National University, Seoul, Republic of Korea*
[b] *IBM Research, Burnet Rd, Austin, TX 11501 USA*

ARTICLE INFO

Article history:
Received 7 November 2017
Revised 5 April 2018
Accepted 25 May 2018
Available online 31 May 2018

Communicated by Dr. Deng Cheng

Keywords:
Spiking neural network
Weighted spike
Supervised learning

ABSTRACT

Spiking neural networks are being regarded as one of the promising alternative techniques to overcome the high energy costs of artificial neural networks. It is supported by many researches showing that a deep convolutional neural network can be converted into a spiking neural network with near zero accuracy loss. However, the advantage on energy consumption of spiking neural networks comes at a cost of long classification latency due to the use of Poisson-distributed spike trains (rate coding), especially in deep networks. In this paper, we propose to use weighted spikes, which can greatly reduce the latency by assigning a different weight to a spike depending on which time phase it belongs. Experimental results on MNIST, SVHN, CIFAR-10, and CIFAR-100 show that the proposed spiking neural networks with weighted spikes achieve significant reduction in classification latency and number of spikes, which leads to faster and more energy-efficient spiking neural networks than the conventional spiking neural networks with rate coding. We also show that one of the state-of-the-art networks the deep residual network can be converted into spiking neural network without accuracy loss.

1. Introduction

Nowadays deep neural networks (*DNNs*) are continuously expanding their influences on application areas such as image classification [1], speech recognition [2], natural language processing [3] and many others. However, its heavy computational load and high energy consumption still block the broader use of DNNs in practical applications that require large-scale data processing in real-time [4]. As an alternative, spiking neural networks (*SNNs*) have been studied by many researchers for the purpose of building neuromorphic hardware [5–9] with low energy consumption. A spiking neural network consists of spiking neurons, each of which fires an output spike only when its membrane potential is charged above a certain threshold [10]. The generated spike is propagated into the neurons in the next layer and increases/decreases their membrane potentials. In this manner, the communication between neurons is performed by spikes. In SNNs, the processing of each input spike in a neuron accompanies only a single simple addition operation onto the membrane potential, while conventional artificial neural networks (*ANNs*) require multi-bit input signals and a multiplication operation in addition to accumulation, which consumes much larger energy.

There are various approaches to train a spiking neural network. Among those, one popular way is to train an ANN with the same topology and then convert the synaptic weights to those of the SNN. The resulting SNN achieves high classification accuracy comparable to the ANN even for a deep topology. Most of the ANN-to-SNN conversion approaches use Poisson-distributed rate coding [11], where spike firing rate or the number of spikes generated within a certain time interval approximates the signal intensity. Rate coding inevitably requires a long time to represent high precision information, which means it has a low information capacity. Because of this, the classification latency increases much longer if the depth of SNN increases since spikes can be propagated into next neurons only after the membrane potentials of the current neurons are charged over the given threshold. Moreover, the information represented by the spike firing rate becomes more error-prone in deeper layers of the network [12], and thus larger number of spikes is required to reduce the approximation errors in deep neural networks. Since the number of additions is proportional to the number of spikes arriving at neurons in SNNs, larger number of spikes incurs more dynamic energy consumption and significantly diminishes the merit of low energy consumption in deep SNNs.

In this paper, we propose a deep spiking neural network with weighted spikes (*SNN-WS*)[1] to perform the image classification

* Corresponding author.
E-mail address: kchoi@snu.ac.kr (K. Choi).

[1] Note that the weight in this context does not mean synaptic weight.

https://doi.org/10.1016/j.neucom.2018.05.087

with shorter classification latency and less number of spikes compared to the SNN with conventional Poisson-distributed rate coding (*SNN-RC*). It assumes that a neuron can fire at most one spike within a time step. It assigns different weights to spikes depending on their phases (relative position of their time steps within a period) in order to transfer more information to the deeper layers in a short time. This scheme is partly inspired by the phase coding [13,14] in neural coding field of neuroscience. Exploiting the characteristics of the weighted spikes, we also propose an early decision algorithm to take the trade-off between classification accuracy and energy. We validate the proposed idea through experiments performed on various datasets such as MNIST, SVHN [15], CIFAR-10, and CIFAR-100 [16] with various network topologies.

The contributions of this paper are enumerated below.

- Proposing a novel spiking neural network model with weighted spikes inspired by phase coding, which has shorter classification latency and lower energy consumption than conventional SNN-RC.
- Providing a mathematical formulation showing that SNN-WS approximates the ReLU activations of ANN.
- Devising an update skipping scheme that reduces the effect of noise spikes and increases the convergence speed.
- Combining an early decision scheme for further energy saving by allowing a small accuracy loss.
- Extensive experimental results on MNIST, SVHN, CIFAR-10, and CIFAR-100 datasets.

The rest of the paper is organized as follows. Section 2 enumerates the previous researches related to this work. Section 3 explains the basics of conventional spiking neural network with rate coding. Section 4 describes the details of the proposed weighted spike scheme, and Section 5 introduces further optimization techniques. Section 6 and 7 shows the experimental conditions and results, and Section 8 gives concluding remarks.

2. Related work

2.1. Training SNNs

Many approaches have been proposed for more efficient and accurate training of SNNs. However, it is not trivial and often suffers from a significantly low accuracy. One approach is an unsupervised learning by implementing spike timing-dependent plasticity (*STDP*) [17,18], which is inspired by the learning behavior of biological neural networks in brains. However, the existing methods based on the STDP learning suffer from low classification accuracy and training difficulty on deep networks [19–23].

Another approach is a supervised learning by exploiting backpropagation algorithm [24], which is a very popular method used to train conventional ANNs. Many previous researches have applied the backpropagation algorithm in order to train SNNs, and most of them have achieved significantly better classification accuracy and training stability even in deep networks. The SNN researches using the backpropagation algorithm can be categorized into two groups.

One is to train synaptic weights of SNNs directly by using the spikes which are propagated in the networks. *SpikeProp* [25] is one of the first supervised learning algorithm for SNNs. It extends the traditional backpropagation algorithm to transfer the information in the timing of spikes. It shows that the learning algorithm can be used to solve real-world classification problems, and its learning efficiency has been improved by following researches [26–29]. However, they can control the timing of single-spike output only and limit the information capacity they can handle. To overcome the problem, a multi-spike learning method called remote supervised learning method (*ReSuMe*) is proposed [30]. It trains SNNs to produce desired output spike train by exploiting STDP and anti-STDP processes based on the Widrow-Hoff rule [31,32]. *DL-ReSuMe* [33], which combines delay shifting on the ReSuMe, is proposed to enhance the learning performance further. The work in [34,35] exploits perceptron neurons to improve training speed significantly, and the recent work in [36] proposes relative ordering learning (*ROL*) to remove the strict timing constraint of the previous work for more robust learning. These learning methods of direct spike handling are shown to be successful for SNNs. However, only easy problems are solved by the methods and the networks used in the experiments are small and shallow compared to the networks used in conventional ANNs. There has been no attempt yet to show the feasibility of those methods for problems more complex than MNIST [37].

The other is to convert the synaptic weights of an ANN with the same topology to those of the SNN. The ANN-to-SNN conversion achieves accuracy nearly the same as that of the ANN even for deep networks and more complex datasets, because it uses synaptic weights well optimized by the backpropagation algorithm for ANNs. Early researches on the ANN-to-SNN conversion [38,39] use more biologically realistic neurons with leakage and refractory periods, but suffer from significant accuracy loss. The work in [40] exploits ReLU activation function [41] to achieve more accurate results. The work in [42] achieves nearly lossless accuracy compared to ANNs on MNIST dataset. They propose an ANN-to-SNN conversion method by fixing the threshold of neurons to 1.0 and normalizing synaptic weights such that the maximum possible positive input to a neuron can only generate one spike to avoid the approximation error caused by activation values larger than 1.0. There is a similar weight normalization technique [12] tested on more complex CIFAR-10 [16] dataset. It analyzes the distribution of ReLU activation values and selects the 99.9th percentile value instead of the maximum activation as a normalization factor, which leads to faster classification.

2.2. Spike coding schemes

Rate coding [11,43] is the most frequently used coding technique for spikes in SNNs. Thus we use it as the control group in our experiments. In rate coding, the number of spikes occurred within a period of time is counted, and the spike firing rate is used as the signal intensity. As a result, rate coding has much redundancy and also tolerance to a certain amount of noise. For this reason, along with the fact that it fits well with the nature of SNNs, it is often used in many of SNN researches [12,38,39,42]. However, it often suffers from a long classification latency.

While the rate coding drops out most of the timing information of the spikes, various approaches of temporal coding have been proposed to use timing information to mitigate the slowdown. For instance, time-to-first-spike [44] is a well-known temporal coding. In this technique, only the first spike within a certain time range has a meaning, and the arrival time of the first spike is used as the information. Time-to-first-spike coding is often used in many approaches [25,45], because it is simple to understand, yet it can save useful information that is lost in rate coding. There are many other coding schemes, such as rank-order coding [46] and resonant burst model [47]. Among them, it would be worth mentioning phase coding [13,14]. In phase coding, a background periodic oscillation function is defined as a reference. The value of a spike is determined by the phase of the reference function at the arrival time of the spike. Our work is conceptually close to phase coding because the spike weights are determined by the current phase.

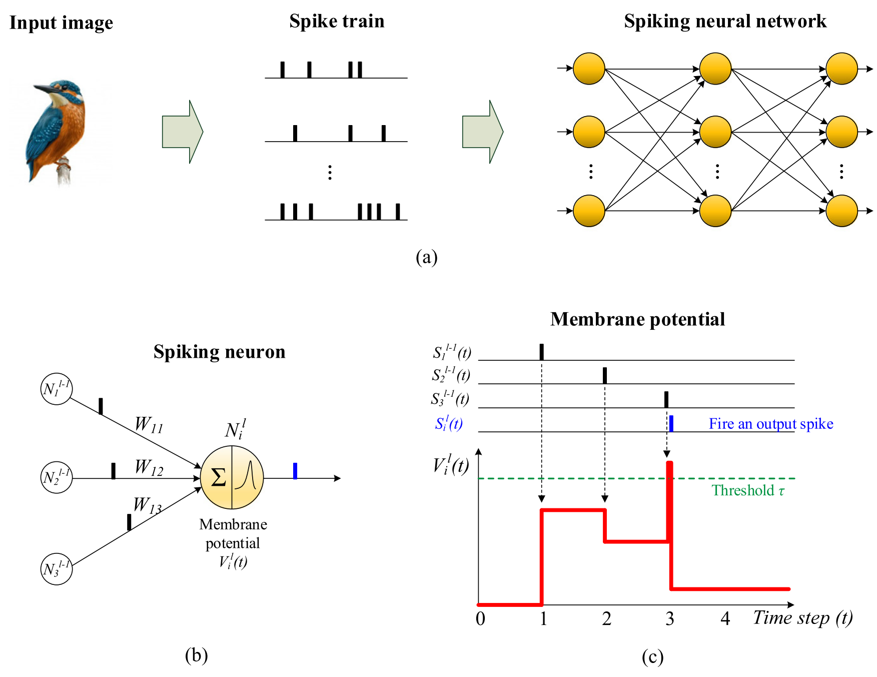


Fig. 1. Example of a spiking neural network. Each pixel of an image is encoded into a spike train, and the resulting spike trains are fed into the spiking neural network. The network consists of spiking neurons, each of which integrates postsynaptic potentials and fires a new spike if its membrane potential exceeds a certain threshold.

3. Background

3.1. Spiking neural network

Spiking neural network is a class of artificial neural networks that emulates the behavior of a brain with spiking neurons. In a brain, neurons communicate with spike signals. An incoming spike is mediated by a synapse, and the synapse produces a postsynaptic potential (*PSP*) which is affected by the weight of the synapse. The post-synaptic potential then changes the membrane potential, the internal state of a neuron. The membrane potential can be represented as the sum of the PSPs generated from all synapses that receive input spikes. If the membrane potential goes above a certain threshold, the neuron fires a new spike and it is propagated into other neurons.

Fig. 1 demonstrates a process of image classification by a spiking neural network. First, spike trains are generated according to an input image. Each pixel in an image is encoded as a spike train based on its intensity, and the resulting spike trains are applied as an input of the SNN. When spikes are propagated through synapses, PSPs are induced with different intensity based on synaptic weights. A spiking neuron integrates all incoming PSPs into its membrane potential, and fires a new spike if the membrane potential exceeds a certain threshold. With this mechanism, input spike trains lead to new spike trains at intermediate layers of the SNN, and a classification decision can be made by comparing the firing rate or timing of the output spikes.

The main advantage of SNNs compared to conventional ANNs is high energy efficiency. While a neuron of an ANN requires high precision multiplications of input values and synaptic weights, that of an SNN requires only additions or integrations of synaptic weights into its membrane potential only at the presence of spikes. Due to the operational simplicity, it can be implemented with low energy consumption. The energy consumption required to handle a spike is just a few pJ in current SNN implementations [5–9], and TrueNorth chip [7], a popular SNN implementation, consumes only 72mW for 1 million neurons. Moreover, the in-situ placement of synapses near neurons in SNNs avoids huge energy consumption required for external memory access in ANN implementations.

3.2. Spiking neuron model

To model the behavior of a spiking neuron, we use a simple *integrate-and-fire model* that is similar to the previous works [12,42]. In the model, we define An occurrence of an output spike by the i-th neuron in layer l at time t as

$$s_i^l(t) = \begin{cases} 1, & \text{if there is a spike at time } t \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

An input current, the sum of postsynaptic potentials, which flows into the ith neuron in layer l is represented as

$$z_i^l(t) = \sum_{j=1}^{M^{l-1}} W_{ij}^l s_j^{l-1}(t) + b_i^l, \quad (2)$$

where $s_j^{l-1}(t)$ is the output spike of jth neuron in layer $l-1$ and b_i^l is the bias for the ith neuron in layer l. Although it is not clear that there exists a bias in a biological neuron, the bias term is added to accurately convert a trained ANN to an SNN.

The membrane potential of a neuron, which gradually increases as it takes input currents induced by PSPs and decreases after generating outgoing spikes when it goes over a certain threshold τ, can be represented as

$$V_i^l(t) = V_i^l(t-1) + z_i^l(t) - \tau s_i^l(t). \quad (3)$$

This is called a *reset by subtraction model* in which a membrane potential decreases by the amount of the threshold level τ when it fires an output spike. We do not model membrane potential leakage that makes the membrane potential decrease over time, and the refractory period that makes the membrane potential settle into a base level after firing a spike. It is because an adoption of these factors could rather deteriorate classification accuracy and incur computational overhead.

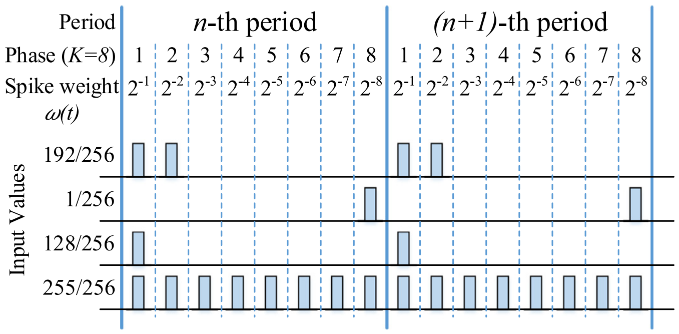


Fig. 2. Example of input spike trains when the number of phases in a period (K) is 8. A rectangle indicates a spike, and each spike has the spike weight $\omega(t)$ depending on the phase of global reference clock. The same input spike trains are repeated until the end of classification.

3.3. Rate coding

A rate coding is one of the most popular encoding schemes used in conventional SNNs to represent a signal with a spike train. In the rate coding scheme, a spike firing frequency of a neuron represents a signal intensity. The spike firing rate r_i of ith neuron can be formulated as

$$r_i = \frac{N_s}{T} = \frac{\sum_{t=1}^{T} s_i(t)}{T}, \tag{4}$$

when N_s spikes are fired during T time steps. There can be a spike or no spike in each time step. The firing rate is 1.0 if a neuron fires every time step, and the firing rate is 0.0 if it fires nothing for all time steps. In an image classification task, for example, if a pixel intensity of an image is in the range of [0.0, 1.0], it can be represented by a spike train with the firing rate of [0.0, 1.0] as an input to the SNN.

The Poisson-distributed spike train [11] transmitted to ith input neuron which corresponds to a pixel intensity P can be generated by comparing a random number and P independently for each time step as

$$s_i^0(t) = \begin{cases} 1, & \text{if Random(0.0, 1.0)} < P \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The rate coding suffers from low information capacity. It requires many time steps to recognize a signal intensity correctly since the number of fired spikes has to be counted over the whole period. To recognize a signal of K-bit precision, it requires at least 2^K time steps. This intrinsic property of rate coding increases a classification latency and the number of spikes generated in a network. The increased number of spikes incurs higher dynamic energy consumption due to more frequent changes of membrane potentials, and the increased running time increases leakage energy consumption, which offsets the energy efficient nature of SNN.

4. SNN with weighted spikes

4.1. Weighted spikes

The main idea of the weighted spikes is assigning different weights to different phases (or to spikes in those phases) in order to pack more information into the spikes. This is the major difference from a conventional rate coding scheme that assigns the same weight to every spike. Fig. 2 shows an example of weighted spike trains corresponding to the input values. A spike train consists of a sequence of periods, each of which has K phases of different weights. In our approach, we assign weights of $2^{-1}, 2^{-2}, \ldots, 2^{-K}$ respectively to the 1st, 2nd, ..., Kth phases, so that a period consists of phases divided by time, each of which corresponds to a different spike weight. Thus, spike weight $\omega(t)$ at current time $t \in \mathbb{N}^+$ (t is the total number of time steps elapsed from the 1st phase of the 1st period) is given by

$$\omega(t) = 2^{-(1+mod(t-1,K))}. \tag{6}$$

We normalize the values of all input signals and activations (ReLU outputs) to fit into the range $[0, 1-2^{-K}]$ by using the *data-based normalization* technique introduced in [42]. Then the weighted spike train within a period corresponds to a binary representation of the value with K fractional bits. We assume the use of a conventional frame-based sensor which produces multi-bit outputs. The multi-bit outputs of the sensor are fed to our SNN in a bit-by-bit manner from MSB to LSB. For example, consider an SNN for image classification task, where an input image sensor produces values corresponding to the intensities of 10 x 10 pixels. The most significant bits of all the 100 pixel outputs are fed to the SNN in time step 1 and then the second most significant bits are fed to the SNN in time step 2. This is continued to the end of the period at time step K. Then the same sequence is repeated for the following periods until the image is recognized. In the same way, a set of K-bit data is transferred between layers. This leads to significantly faster communication compared to SNN-RC since SNN-WS needs only K-slots to represent a K-bit data whereas SNN-RC would require 2^K-slots to represent the same data.

Due to the weighted spikes, SNN-WS can transmit more information with spikes than SNN-RC for the same time duration. It results in shortened classification latency and reduced number of spikes generated, which leads to higher throughput and energy-efficiency.

4.2. Spiking neuron model for weighted spikes

The adoption of weighted spikes requires modifications of equations for the conventional spiking neuron model described in Section 3. The modified spiking neuron model for weighted spikes is described in this section.

In ANN, the ReLU activation of the ith neuron in layer l is given by

$$a_i^l = max\left(0, \sum_{j=1}^{M^{l-1}} W_{ij}^l a_j^{l-1} + b_i^l\right), \tag{7}$$

where M^{l-1} is the number of neurons in layer $l-1$, W_{ij}^l is the synaptic weight between layers $l-1$ and l, and b_i^l is the bias for the ith neuron in layer l.

Similarly, in SNN-WS, the input current $z_i^l(t)$, which will be integrated into the membrane potential of the ith neuron in layer l at time t—$z_i^l(t)$ corresponds to a bit of a_i^l in ANN—is represented as

$$z_i^l(t) = \omega(t)\left(\sum_{j=1}^{M^{l-1}} W_{ij}^l \tau s_j^{l-1}(t) + \beta_i^l\right), \tag{8}$$

where τ is the threshold of spiking neurons, $s_j^{l-1}(t)$ is the output spike of jth neuron in layer $l-1$, and β_i^l is the normalized value of bias b_i^l in ANN and given by $\beta_i^l = b_i^l/\left(1-2^{-K}\right)$, which leads to $\sum_{t=1+(n-1)K}^{nK} \omega(t)\beta_i^l = b_i^l$ for $n \in \mathbb{N}^+$.

Eq. (1) representing an occurrence of a spike still holds for SNN-WS. It can also be calculated by using the relation between an output spike and a membrane potential as

$$s_i^l(t) = U\left(V_i^l(t-1) + z_i^l(t) - \omega(t)\tau\right), \tag{9}$$

where $U(x)$ is a unit step function. $V_i^l(t)$ is the membrane potential of the ith neuron in layer l, given by

$$V_i^l(t) = V_i^l(t-1) + z_i^l(t) - \omega(t)\tau s_i^l(t). \tag{10}$$

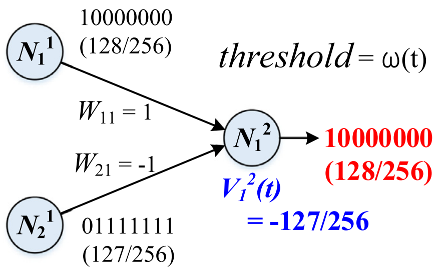


Fig. 3. Example of a noise spike output. The neuron N_1^2 generates the noise spike train (10000000) instead of the correct one (00000001). The negatively charged membrane potential $V_1^2(t)$ prevents further generations of noise spike trains in next periods.

In summary, as shown in (8), $z_i^l(t)$ is obtained by first taking the sum of input spikes multiplied by the corresponding synaptic weights, adding the adjusted bias, and then scaling by $\omega(t)$. As shown in (9) and (10), the membrane potential is updated by integrating $z_i^l(t)$ (adding spikes to the membrane potential every time step). As shown in (9), the neuron fires an output spike with spike weight $\omega(t)$ if the updated membrane potential is greater than or equal to the threshold scaled by $\omega(t)$. Finally, in (10), the membrane potential is subtracted by the threshold scaled by $\omega(t)$ if an output spike is generated (i.e., $s_i^l = 1$). In this work, we fix τ to 1 for all neurons and thus it is omitted in the rest of this paper for simplicity.

4.3. Noise spike

The bit-by-bit computation of weighted spikes in a spiking neuron can cause noise spikes[2] since a spiking neuron does not consider *a priori* information of the future input bits. Fig. 3 shows a simple case of generating a noise spike when K is 8. In the figure, neuron N_1^1 generates a spike train 10000000 (128/256) and N_2^1 generates 01111111 (127/256) in the same period of 8 time steps. In the first time step, neuron N_1^2 receives a spike from N_1^1 and increases its membrane potential by 128/256 because the spike weight of the first phase is 2^{-1} and the synaptic weight is 1. Then, N_1^2 fires an output spike because its membrane potential 128/256 is sufficient to fire a 2^{-1} weighted spike. In the remaining time steps in the period, however, only N_2^1 outputs spikes, charging the membrane potential of N_1^2 in the negative direction and no longer producing any output spike. As a result, N_1^2 produces a spike train 10000000 and have membrane potential of -127/256, which is far from the correct output spike train 00000001 and the membrane potential 1/256.

Under this mechanism, unpredictable fluctuation of the output spike trains (noise spikes) generated during the first period makes it difficult to perform classification based on them. However, the negatively charged membrane potential $V_1^2(t)$ in the example above prevents additional noise spike generations in the following periods, which gradually compensates the error caused by the noise spike produced in the first time step. Therefore, the average of output spike trains over multiple periods can accurately approximate the desired value (we consider the ReLU activation of the corresponding neuron in ANN as the desired value). To compute the average of output spike trains, repetition of periods is required with the same input spikes until a certain level of classification accuracy can be reached. Nevertheless, the weighted spikes achieve the accuracy level much faster than the conventional rate coding since they deliver more information during the same time interval. Note that the weighted spike scheme requires only $O(logN)$ time steps to send one of N different values while the rate coding requires $O(N)$ time steps. This makes the SNN-WS achieve lossless accuracy with short classification latency and reduced number of spikes compared to the SNN-RC. In addition, we suggest to skip updating the output membrane potential for a few periods, so that the effect of noise spikes can be minimized and thus improve the classification accuracy and latency (see Section 5.1).

[2] The problem also exists in SNN-RC, but it's more significant in SNN-WS due to the higher information density.

4.4. Approximation of the ReLU activation

In SNN-RC, average firing rate of a neuron approximates ReLU activation of the corresponding neuron in ANN [12]. In SNN-WS, the value of an output spike train averaged over periods approximates the corresponding ReLU activation more precisely. In this section, we describe how the averaged output spike train in SNN-WS correctly approximates the corresponding ReLU activation.

The signal value from the ith input neuron (i.e., the ith sensor output value) for the nth period is given by

$$a_i^0 = \sum_{t=1+(n-1)K}^{nK} \omega(t) s_i^0(t), \tag{11}$$

for $n \in \mathbb{N}^+$. For the same input image, the signal value remains the same regardless of the value of n. In the ith neuron of the first hidden layer, the following equations for the sum of weighted output spikes accumulated over n periods with $V_i^1(0) = 0$ can be derived from (10) and (8):

$$\begin{aligned} \sum_{t=1}^{nK} \omega(t) s_i^1(t) &= \sum_{t=1}^{nK} z_i^1(t) - V_i^1(nK) \\ &= \sum_{t=1}^{nK} \omega(t) \left(\sum_{j=1}^{M^0} W_{ij}^1 s_j^0(t) + \beta_i^1 \right) - V_i^1(nK) \\ &= n \left(\sum_{j=1}^{M^0} W_{ij}^1 a_j^0 + b_i^1 \right) - V_i^1(nK) \\ &= n a_i^1 - V_i^1(nK), \end{aligned} \tag{12}$$

If $a_i^1 \le 0$, no output spike is generated, showing the behavior equivalent to ReLU. If $a_i^1 > 0$ and n is sufficiently large, $V_i^1(nK)$ becomes much smaller than na_i^1 (note that $V_i^1(nK) < \omega(t)\tau$) and can be ignored, which leads to

$$\frac{1}{n} \sum_{t=1}^{nK} \omega(t) s_i^1(t) \approx a_i^1. \tag{13}$$

In this manner, it can be generalized to other layers as follows:

$$\frac{1}{n} \sum_{t=1}^{nK} \omega(t) s_i^l(t) \approx a_i^l, \tag{14}$$

which means the average of the sum of weighted spikes over many periods correctly approximates the corresponding ReLU activation in ANN.

Therefore, the classification can be made by calculating

$$\arg\max_i \left(\sum_{t=1}^{nK} \omega(t) s_i^L(t) \right), \tag{15}$$

in the output layer L, which can be implemented with weighted spike counters at the end of the neurons in the output layer. However, instead of generating output spikes and counting them, we

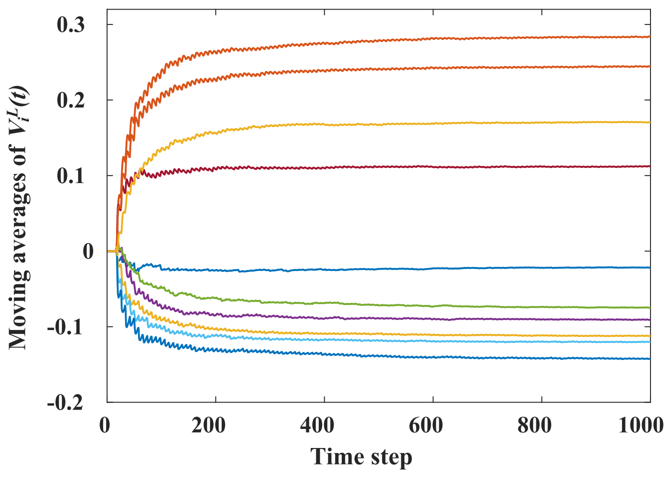


Fig. 4. Moving averages of the membrane potentials for 10 neurons in the output layer. It is calculated from a randomly selected CIFAR-10 test image.

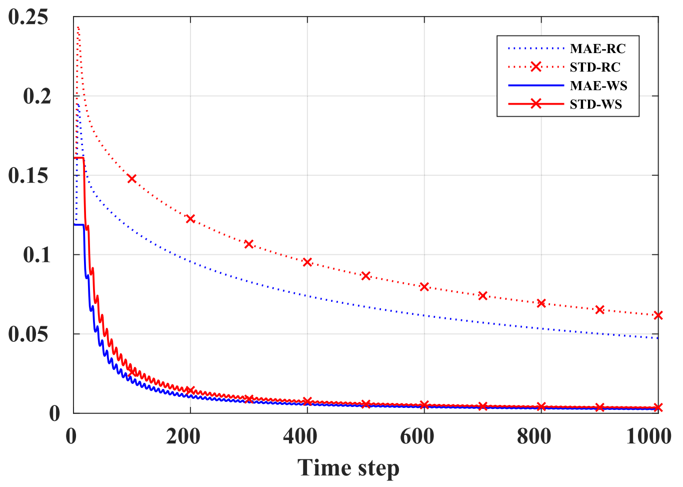


Fig. 5. Mean absolute error (MAE) and standard deviation (STD) of the approximation errors for all activations in a spiking neural network using rate coding (RC) and weighted spikes (WS). It is calculated from the test results with 10,000 CIFAR-10 test images.

found it better to keep accumulating the membrane potential of each output neuron as follows:

$$V_i^L(nK) = \sum_{t=1}^{nK} z_i^L(t) \approx na_i^L. \tag{16}$$

Then the classification is made by computing

$$\arg\max_i \left(V_i^L(nK)\right). \tag{17}$$

It reduces the classification latency by saving the time to charge the membrane potentials of the output neurons to generate spikes.

We experimented with a spiking neural network (Net4 in Table 1) on CIFAR-10 dataset to verify the approximation capability of SNN-WS empirically. Fig. 4 shows moving averages of the sum of weighted input spikes for 10 neurons in the output layer with a randomly selected CIFAR-10 test image. Each line converges to the value of its corresponding activation in ANN. Although the convergence requires around 200 time steps, the classification decision can be made much earlier (after 21 time steps) when the top membrane potential tends to maintain its top rank position. We also compared the approximation errors of the rate coding and the weighted spikes with Net4. The mean absolute errors (*MAE*) and the standard deviations (*STD*) of the approximation errors with SNN-RC and SNN-WS are depicted in Fig. 5. It clearly shows that the approximation errors diminish over time, and SNN-WS converges faster to more accurate values.

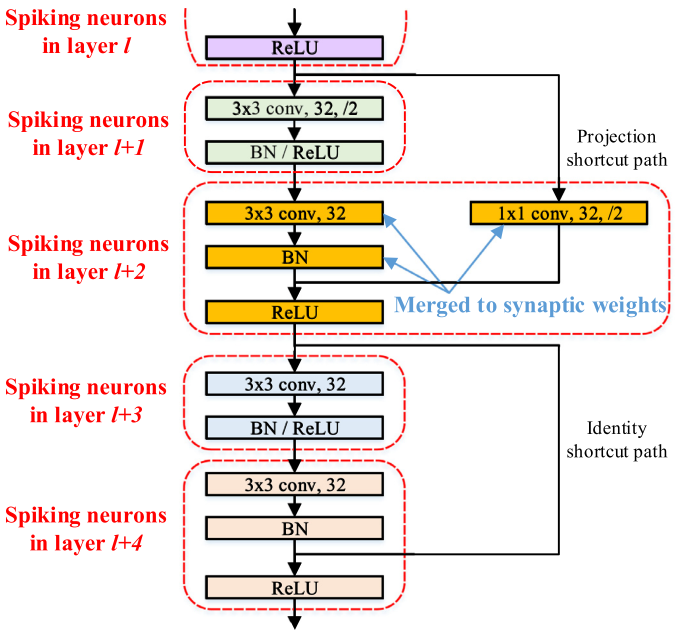

Spiking neurons in layer l
ReLU
Spiking neurons in layer l+1
3x3 conv, 32, /2
BN / ReLU
Projection shortcut path
Spiking neurons in layer l+2
3x3 conv, 32
1x1 conv, 32, /2
BN
ReLU
Merged to synaptic weights
Spiking neurons in layer l+3
3x3 conv, 32
BN / ReLU
Identity shortcut path
Spiking neurons in layer l+4
3x3 conv, 32
BN
ReLU

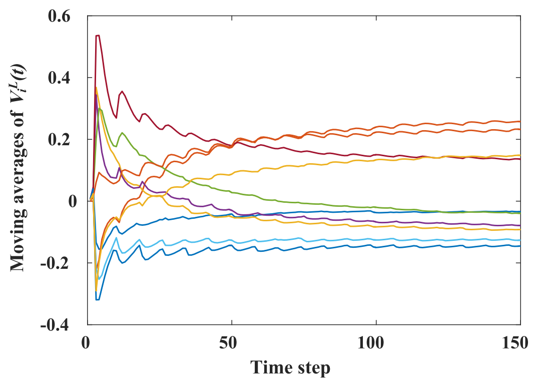
Fig. 6. ANN-to-SNN conversion for a deep residual network. The convolutional and batch normalization layers on the residual and shortcut paths are merged with the following ReLU layer, and they form a population of spiking neurons in a layer.

4.5. ANN-to-SNN conversion

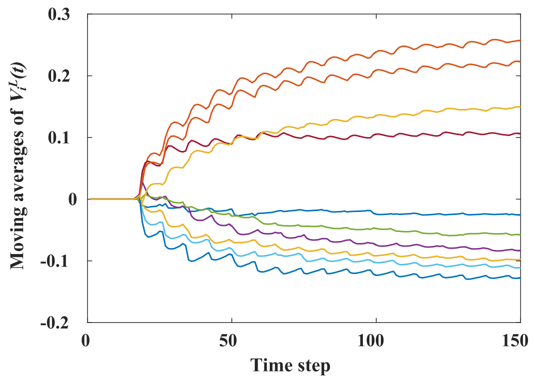
Since it is difficult to achieve high accuracy by training SNNs directly with spikes, it is a common practice to convert a trained ANN into an SNN. For the ANN-to-SNN conversion, we use the techniques introduced in [12] to reorganize biases and batch normalization layers [48]. To extend SNN-WS to convolutional neural networks (*CNNs*), we convert a max pooling layer of ANN by calculating the accumulated sum of weighted spikes for each neuron and then passing only the spike train that has the maximum accumulated sum. The softmax layer of ANN is not converted to a SNN since the softmax layer is dispensable in an inference stage. Instead, the membrane potentials of the neurons in the output layer are directly compared to make a classification decision.

The previous approach [12] uses analog inputs rather than spike inputs to achieve higher accuracy and lower classification latency. However, the use of analog input requires different kinds of neurons in the first hidden layer, each of which must perform a lot of precise analog multiplications of input signals and synaptic weights. Thus, we do not use analog inputs even though the use of analog input leads to further classification speed up.

Deep residual networks [49] show superior classification accuracy than plain networks for various image recognition tasks. If a residual network is converted to an SNN without accuracy loss, the resulting SNN will achieve higher classification accuracy than an SNN converted from a plain network. In a residual network, there is a join point that merges the residual path and the shortcut path. The merge is done by adding the convolution output from the residual path with the output from the shortcut path. The method to convert this structure into an SNN is depicted in Fig. 6. First the adders at the join point and its subsequent ReLU layer are replaced with new spiking neurons. Then the convolution weights of both residual and shortcut paths are moved to the synaptic weights of the new spiking neurons. The spiking neurons fed by the short-



(a) Do not skip the initial input currents in the output layer. The correct decision is made after 52 time steps.



(b) Skip the initial input currents in the output layer for 16 time steps. The correct decision is made after 21 time steps.

Fig. 7. Effect of skipping initial input currents in the output layer. Each line shows the moving average of membrane potential for a neuron in the output layer. It is calculated from a randomly selected CIFAR-10 test image. Note that Y-axis scales are different.

cut connection have twice as many synapses as the other spiking neurons.

5. Optimization techniques

5.1. Skipping initial input currents in the output layer

To reduce the adverse effect of noise spikes, we introduce an input current skipping technique. The current skipping technique ignores the noisy input currents to the neurons in the output layer for a few initial periods, thereby improving the classification latency by a great amount. As explained in Section 4.3, the highly-weighted bits of the input signals during these initial periods are propagated through the SNN without considering the following bits with lower weights and long term average. Moreover, during these periods, the membrane potentials of the neurons across the network are not initialized well. Thus, they may incur a lot of noise spikes and charge the membrane potentials of the spiking neurons in the output layer to an incorrect direction. Instead of accumulating the noisy input currents and then averaging out the effect in the following periods, we found it better to remove the effect just by ignoring the initial input currents and skipping the update of membrane potentials in the output neurons during the initial periods. This simple technique improves the classification latency significantly.

Fig. 7 shows the effect of initial period skipping in the output layer. The results are obtained from an inference experiment with Net4 in Table 1 on CIFAR-10 dataset. When the skipping method is not applied (Fig. 7a), the membrane potentials of the output neurons highly fluctuate due to the initial noise spikes and they rush into incorrect levels which are far from the correct converge points during the initial time steps. It takes long time steps to move the incorrectly initialized membrane potentials to the correct converge points. The classification decision is made by selecting the label of the output neuron which has the highest membrane potential. Thus, the correct decision can be made after 52 time steps. On the other hand, when the skipping method is applied (Fig. 7b), the membrane potentials of output neurons can avoid interferences caused by noise spikes during the initial 2 periods (16 time steps[3]), and move faster to the converge points. The correct decision can be made in only 5 time steps after the end of skipping the initial input currents, making the total delay to be 21.

The optimal number of initial periods to skip charging in the output layer depends on the depth of the network. In general, it is better to skip more periods when the network is deeper since it takes longer time for input spikes to traverse a deeper network. In terms of noise reduction, it is actually better to skip as many periods as possible. However, excessive skipping leads to a too long classification latency, which is also undesirable. In order to achieve a reasonable accuracy and latency, the length of the skipping periods should be tuned empirically for each network depending on its depth and topology.

5.2. The number of phases in a period

The number of phases in a period (K) is a tunable parameter affecting both latency and accuracy. By increasing the value of K, one can increase the classification accuracy at the cost of increased latency. In case of an image classification, a pixel intensity of an image is normally quantized to 8 bits for each of RGB channels. Therefore, setting $K = 8$ is usually enough to accurately deliver the pixel information to the neurons in the input layer. $K < 8$ causes a loss of the input signal precision, but it can reduce the classification latency. On the other hand, $K > 8$ incurs a waste of time because there will be no input spike for the neurons in the input layer during the phases over 8. However, the hidden layer can have a precision higher than 8 bits.

Fig. 8a and b shows the classification error for various K parameters with Net2 in Table 1 on MNIST dataset. The membrane potentials of the output neurons are not updated for the first period in order to avoid initial noise spike. The time at which the classification error begins to fall is different depending on K because a period consists of K time steps. If we skip the update of membrane potentials of the output neurons for fixed time steps regardless of K, the input values fed into the SNN are distorted and it incurs accuracy loss. $K = 4$ results in the shortest classification latency on MNIST dataset under the assumption that 1%p accuracy loss is allowed. Even though the small K worsens the input signal precision, it does not result in a severe accuracy loss. It is since most pixels are biased toward either 0 (black) or 255 (white) in images of MNIST dataset, which makes the classification accuracy insensitive to the bits with lower weights.

Fig. 9a and b shows the classification error with Net4 in Table 1 on CIFAR-10 dataset. In this case, smaller K leads to higher classification error at the end of convergence. Images on CIFAR-10 dataset use RGB color format and pixel value in each of the RGB channels is quantized to 8 bits. Unlike the MNIST dataset, pixel values of CIFAR-10 image are not biased toward either of the edges 0 and 255, but rather distributed broadly in the whole range [0, 255]. Thus, small K leading to omission of less significant bits incurs severe degradation of converged classification accuracy. According to

[3] In this experiment, the number of phases in a period is set to 8. Thus, 1 period consists of 8 time steps.

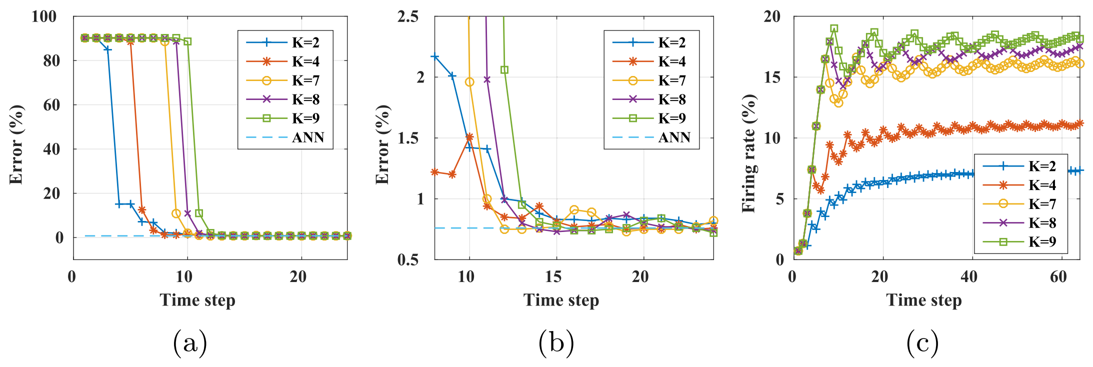


Fig. 8. Comparison among different number of phases (K) in Net2 on MNIST dataset. $V_i^L(t)$ update is skipped for the first period. The smaller K leads to earlier start of the error convergence, but it results in worse converged accuracy. The average firing rate fluctuates within a period and the smaller K leads to lower firing rate. (a) Classification error over time. (b) Classification error over the time range [8, 24]. (c) Average firing rate over time.

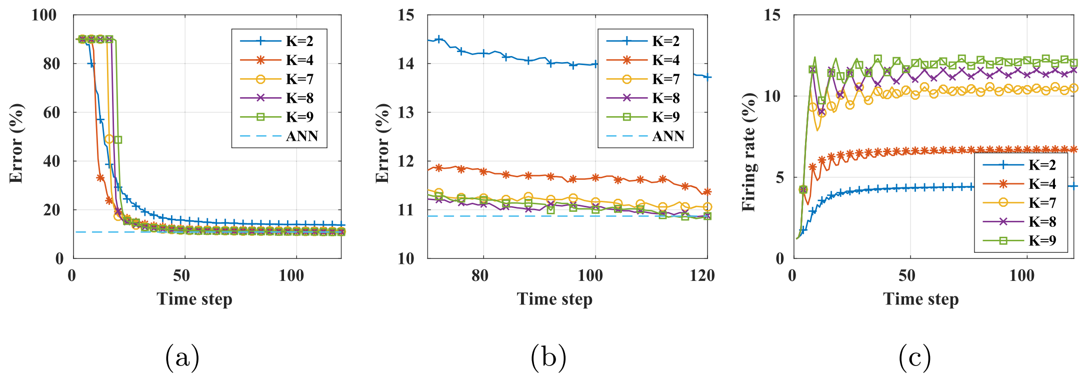


Fig. 9. Comparison among different number of phases (K) in Net4 on CIFAR-10 dataset. $V_i^L(t)$ update is skipped for the first 2 periods. It shows slower inference convergence when $K < 7$. (a) Classification error over time. (b) Classification error over the time range [70, 120]. (c) Average firing rate over time.

our experiments, $K = 8$ yields the best performance on SVHN and CIFAR datasets.

Figs. 8c and 9c compare the average firing rate over time among different K in Net2 and Net4. The average firing rate converges over time, but it shows a periodic oscillation within a period since near-0 activations are dominant in neural networks, which means the probability of generating a spike in a high-weight phase of a period is much lower than that in a low-weight phase in a period. Therefore, smaller K results in lower average firing rate.

5.3. Accuracy-energy trade-off by early decision

Among weighted spikes in a period, by transmitting the spike with the highest weight first, we allow the network to make a decision as early as possible (before the arrival of lower weight spikes) and thus terminate the classification earlier. For this purpose, we additionally define the “difficulty of classification” of an input image as the difference of top-2 activations among the output neurons. If the difference of the top-2 activations for an input image is small, it requires quite accurate computation to successfully distinguish the difference and make a correct decision. On the other hand, if the difference of top-2 activations is large enough, it can be distinguished even with relatively rough approximation. As mentioned in Section 4.4, the accuracy of SNN-WS increases over time, implying an accuracy-latency/energy trade-off.

Algorithm 1 demonstrates the procedure of early decision in this work. The early decision loop begins after T_1 in order to avoid the initial noise spikes. The current decision $D(t)$ is the class label of the output neuron that has the highest membrane potential (line 3). It computes the top-2 difference among the membrane potentials of the output neurons (line 4), which represents the classification difficulty of the input image. It takes the exponential moving average as a stabilizer since the top-2 difference fluctuates over time (line 5). If the decision is unchanged for θ/δ_{EMA} time steps (line 8), the early decision is made and the classification finishes (line 9). The stable decision threshold θ contributes to an accurate early decision in spite of the fluctuations of the membrane potentials in the output layer. By tuning θ, we can take the trade-off between classification accuracy and energy saving which can be

Algorithm 1 Early decision procedure

Input: a time range $[T_1, T_2]$, membrane potentials of output neurons $\vec{V}^L(t) = \{V_1^L(t), V_2^L(t), \ldots\}$ for $t \in [T_1, T_2]$, and stable decision threshold θ.

Output: class decision $D(t)$ and decision time t.

```
1:  $D(T_1-1) \leftarrow \emptyset,\ \delta_{EMA}(T_1-1) \leftarrow 0,\ count \leftarrow 0$   // initialization
2:  for $t \leftarrow T_1$ to $T_2$ do
3:    $D(t) \leftarrow arg\,max_i V_i^L(t)$   // decision $D(t)$ in the current time step
4:    $\delta(t) \leftarrow$ top-2 difference of $V_i^L(t)$   // $\delta$ represents classification difficulty
5:    $\delta_{EMA}(t) \leftarrow (\delta(t) + \delta_{EMA}(t-1))/2$   // exponential moving average of $\delta$
6:    if $D(t) = D(t-1)$ then
7:      $count \leftarrow count + 1$   // increase stable decision count
8:      if $count \geq \theta/\delta_{EMA}(t)$ then
9:        return   // return if current decision has been stable for $\theta/\delta_{EMA}(t)$ time steps
10:   else
11:     $count \leftarrow 0$
```

Table 1
Network configurations.

Dataset	Net	Configuration	ANN Acc. (ours)	SNN Acc. (ours)	SNN Acc. (prev)
MNIST	Net1	MLP: 784-1200-1200-10 [42]	98.6%	98.6%	98.6%
	Net2	CNN: 12c5-2s-64c5-2s-10 [42]	99.2%	99.2%	99.1%
SVHN	Net3	NiN: 3 Mlpconv layers [50]	95.2%	95.2%	–
CIFAR-10	Net4	CNN: 32c3-32c3-2s-64c3-64c3-2s-512-10 [12]	89.1%	89.2%	87.8%
	Net5	ResNet-20 [49]	91.4%	91.4%	–
CIFAR-100	Net6	ResNet-32 [49]	66.1%	66.2%	–
	Net7	Plain-32 [49]	64.3%	63.7%	–

estimated by the reduced number of spikes. For a given value of θ, smaller δ_{EMA} (implying that the input image is more difficult to classify) requires larger value of *count* and thus the decision takes longer to terminate.

5.4. Consideration on hardware implementation

The spike weight $\omega(t)$ is used for the calculations of the input current $z_i^l(t)$ and the output spike $s_i^l(t)$ as shown in Eqs. (8) and (9). If SNN-WS is implemented with hardware strictly following those equations, every input spike and the threshold should be scaled by $\omega(t)$, which requires quite frequent scaling since the number of all input spikes in a network is enormous. It can be avoided by changing (9) to

$$s_i^l(t) = U\left(V_i^l(t-1)/\omega(t) + z_i^l(t)/\omega(t) - \tau\right). \tag{18}$$

Then, $\omega(t)$ in $z_i^l(t)$ is cancelled out, so that the input spikes and the threshold no longer require the scaling. Only the membrane potential needs to be scaled by $1/\omega(t)$, which can be easily implemented by a bit-shift operation.

6. Experimental setup

Experiments were performed with seven different networks on four different datasets: MNIST [37], SVHN [15], CIFAR-10, and CIFAR-100 [16].

MNIST. It is a handwritten digit (0–9) image dataset containing 60,000 images for the training set and 10,000 images for the test set. Each image consists of 28 x 28 pixels, and each pixel is represented with 8-bit grayscale format.

SVHN. It is a real-world image dataset obtained from house numbers (0–9) in Google Street View images. It contains 73,257 images for the training set, 26,032 images for the test set, and additional 531,131 extra images. Each image consists of 32 x 32 pixels, and each pixel is represented with 24-bit RGB format. We do not use the extra images in the experiments.

CIFAR-10. It is a real-world image dataset with 10 classes containing 50,000 images for the training set and 10,000 images for the test set. Each image consists of 32 x 32 pixels, and each pixel is represented with 24-bit RGB format.

CIFAR-100. It has the same image format as CIFAR-10, but it consists of 100 classes instead of 10. Each class has 500 training images and 100 test images.

Table 1 shows all the network configurations (Net1 through Net7) with ANN and SNN accuracies of our approach and SNN accuracies of previous approaches. Net1 is a multi-layer perceptron (MLP) including an input layer of 784 neurons, two hidden layers of 1200 neurons each, and an output layer of 10 neurons in order to verify that the proposed idea works in a typical neural network. Net2 is a typical CNN including two convolutional layers with 12 and 64 5 $\times$ 5 kernels (12c5, 64c5), two 2 $\times$ 2 max subsampling layers (2s, 2s), and a fully connected layer between the vectorized final features of the second subsampling layer and 10 output neurons. Those two networks are made with reference to [42], and their SNN accuracies are reported as 98.6% and 99.1% respectively.

Net3 is Network-in-Network [50] with three Mlpconv layers followed by a global average pooling layer. The first Mlpconv layer consists of 192 5 $\times$ 5 convolutional layer followed by two 192 1 $\times$ 1 convolutional layers. The second Mlpconv layer consists of 192 3 $\times$ 3 convolutional layer followed by two 192 1 $\times$ 1 convolutional layers. The third Mlpconv layer is the same as the second Mlpconv layer except that its last 1 $\times$ 1 convolutional layer has 10 feature maps instead of 192. The global average pooling layer takes the average of each feature map and produces 10 outputs. The Net3 is designed to have a large number of feature maps in order to verify that SNN-WS works in a wide neural network.

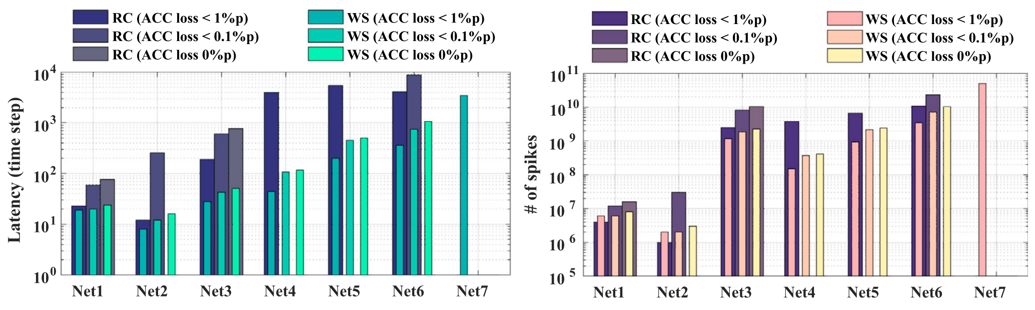
Net4 is a CNN with four 3 $\times$ 3 convolutional layers (2 $\times$ 32c3, 2 $\times$ 64c3), two 2 $\times$ 2 max subsampling layers (2 $\times$ 2s), and two fully connected layers originated from [12]. Its SNN accuracy is reported as 87.8% in [12]. Net5 and Net6 are residual networks [49]. To the best of our knowledge, this is the first work to implement SNN with ResNet topology. Net5 consists of 19 convolutional layers (7 $\times$ 16c3, 6 $\times$ 32c3, 6 $\times$ 32c3) followed by a global average pooling layer and a fully connected layer producing 10 outputs. Net6 consists of 31 convolutional layers (11 $\times$ 16c3, 10 $\times$ 32c3, 10 $\times$ 32c3) followed by a global average pooling layer and a fully connected layer producing 100 outputs. There is a shortcut connection for each of two convolutional layers as shown in Fig. 6. Net7 is a plain network obtained by removing the shortcut connections from Net6, which is added to check to see if the residual network shows faster inference convergence compared to the plain network.

The spiking neural networks cannot accept negative input signals, thus the image preprocessing techniques producing negative values such as subtracting the per-pixel mean [51] and whitening [52] cannot be used. Instead, we normalized the input values into the range $[0, 1-2^{-K}]$, which causes zero or marginal accuracy loss compared to the training results with the image preprocessing techniques. The batch normalization [48] is used to train all the networks. For classifiers learning CIFAR dataset, the simple data augmentation with 4-pixel zero padding followed by random crop and random horizontal flip [53] is applied to the training images. The networks are trained with Keras [54] and MatConvNet [55]. The inferences of the networks were performed on MatConvNet.

7. Results

7.1. Comparison between SNN-RC and SNN-WS

The experimental results with the seven networks on the four different datasets are summarized in Fig. 10. It compares the classification latency and number of spikes between SNN-RC and SNN-WS. The experiments were performed with three different stop conditions depending on the allowed accuracy loss from the orig-



(a) Classification latency (b) Number of spikes

Fig. 10. Experimental results for the networks. Some bars are omitted if they do not reach the target accuracy within 10,000 time steps. Note that the Y-axis is plotted on a log scale.

Table 2
Reduction of latency and # of spikes by SNN-WS compared to SNN-RC.

ACC	Latency reduction			# Spike reduction		
loss	<1%p	<0.1%p	0%p	<1%p	<0.1%p	0%p
Net1	1.2 ×	3.0 ×	3.2 ×	0.7 ×	2.0 ×	2.0 ×
Net2	1.5 ×	21.3 ×	–	0.5 ×	15.0 ×	–
Net3	6.8 ×	14.0 ×	15.1 ×	2.2 ×	4.4 ×	4.6 ×
Net4	91.1 ×	–	–	25.1 ×	–	–
Net5	17.2 ×	–	–	4.4 ×	–	–
Net6	11.4 ×	11.9 ×	–	3.1 ×	3.2 ×	–
Net7	–	–	–	–	–	–

inal ANN accuracy. In all the networks, SNN-WS shows much shorter classification latency compared to SNN-RC regardless of the network type, and the number of spikes is proportional to the latency. Note that the Y-axis of each chart is log scale.

For SNN-WS on MNIST dataset, the number of phases (K) in a period is set to 4, which means only the upper 4 bits of input pixel values are used in the inference. SNN-WS does not incur any accuracy loss in spite of the dropping of lower 4 bits, whereas SNN-RC cannot reach the no-accuracy-loss point within 10,000 time steps in Net2. We set $K = 8$ for other datasets because reducing K causes rather longer classification latency and accuracy loss.

Net3 shows a relatively large number of spikes because it is a wide network with 192 feature maps for each convolution layer. When the network depth grows, SNN-RC suffers from the significant increase of the classification latency, which limits the use of SNN-RC in practice. In Net5 and Net6, SNN-RC does not reach no-accuracy-loss point in 10,000 time steps while SNN-WS reaches there in 502 and 1064 time steps, respectively even though SNN-WS also suffers from the superlinear increase of latency on the increase of network depth. In SNN-RC, Net5 takes more time steps than Net6, even though Net6 is deeper than Net5. It is because Net5 has higher ANN accuracy which requires more time to increase the accuracy. In SNN-WS, on the other hand, Net6 takes more steps than Net5 because it is deeper. The accuracy of SNN-WS makes the classification latency less sensitive to the level of ANN accuracy.

Interestingly, the residual network Net6 shows shorter classification latency compared to its plain counterpart Net7 that does not have any shortcut connection. The shortcut path in the residual network is intended to train deep layers better, but it also helps to boost the inference speed of SNN by propagating spikes faster to the deep layers through the shortcut path.

Table 2 represents the improvements of classification latency and number of spikes achieved by SNN-WS compared to SNN-RC. SNN-WS shows better performance in terms of latency and number of spikes except for the cases where 1%p accuracy loss is allowed on MNIST dataset. In a typical CNN such as Net2 and Net4, SNN-WS achieves significantly better performance than SNN-RC. The maximum latency reduction is 91.1x and the maximum spike reduction is 25.1x in the experiments of Net4 with 1%p accuracy loss allowed. The results of some conditions cannot be compared because SNN-RC cannot reach the target accuracy within 10,000 time steps. For the example of Net7, SNN-RC reaches 2.8%p accuracy loss point in 10,000 time steps while SNN-WS reaches 1%p accuracy loss point in 3,469 time steps.

Fig. 11 shows the detailed experimental results of Net1 through Net7. The changes of classification error over time are shown in Fig. 11a, d, g, and j. SNN-WS shows significantly faster inference convergence than SNN-RC after skipping the membrane potential updates of output neurons for initial periods. The skipping makes the SNN-WS reach no-accuracy-loss point much faster than SNN-RC although SNN-WS starts the inference convergence later than SNN-RC. SNN-RC shows comparable classification latency compared to SNN-WS in a shallow network (Net1 and Net2), but the classification latency of SNN-RC can not be compared to that of SNN-WS in deeper networks (Net3 through Net7). In very deep networks such as Net5, Net6, and Net7, the update of membrane potentials in the output layer should be skipped for many periods since it takes long time for the initial noise spikes to reach the output layer.

Fig. 11b, e, h, and k plot the average firing rates over time. The average firing rates of both SNN-RC and SNN-WS converge over time. But the firing rate of the SNN-WS oscillates within a period as explained in Section 5.2. The converged firing rate of SNN-WS is 2$\sim$4 times larger than that of SNN-RC. It is because, for the dominant near-0 activations, the spikes in SNN-WS are more densely packed within a period than those in SNN-RC. For example, if a neuron generates an output activation value of 3/256 in ANN, the corresponding neuron in SNN-RC fires spikes at the rate of 3/256 because the firing rate of SNN-RC is proportional to the activation value of ANN. On the other hand, the corresponding neuron in SNN-WS fires spikes for 2 phases out of 8 phases which means a firing rate of 2/8.

Fig. 11c, f, i, and l show the required number of spikes to achieve a specific accuracy. Even though the average firing rate of SNN-WS is larger than that of SNN-RC, the total number of generated spikes of SNN-WS is smaller due to the significantly shorter classification latency. In SNNs, the number of spikes is proportional to the dynamic energy consumption, which means that SNN-WS consumes much lower energy than SNN-RC.

7.2. Trade-off by early decision

Another advantage of SNN-WS is the early decision method that gives a chance for further energy reduction at the expense of small accuracy loss. Fig. 12 shows the trade-off between accuracy and energy when the early decision method is exploited. Net2 through Net5 achieve 50%–58% spike savings without accuracy loss, and

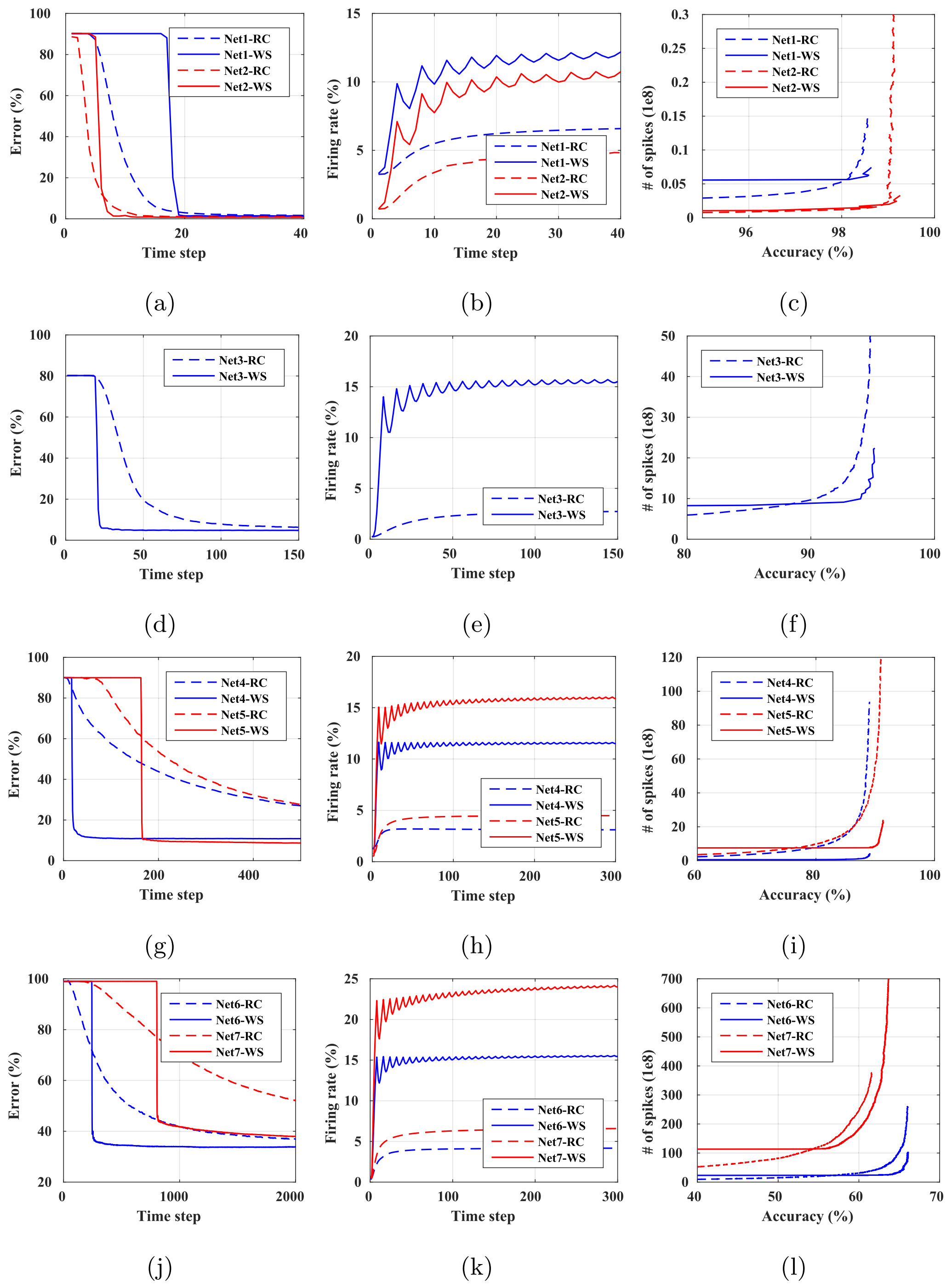


Fig. 11. Experimental results (classification error over time, average firing rate over time, and number of generated spikes over accuracy) of Net1 through Net7. The number of phases (K) in a period is set to 4 for Net1 and Net2, and 8 for Net3 through Net7. $V_i^L(t)$ updates are skipped for 4, 1, 2, 2, 20, 30, and 100 periods, respectively.

59%–77% spike savings within 1%p accuracy loss. Such huge savings are possible since most test images show relatively large differences of top-2 activations compared to the approximation errors, which lead to shorter classification latency by early decision. Net1 represents relatively low spike savings (22%–23%). In Net1, the earliest decision takes 17 time steps[4] and the latest decision takes 24 time steps. The small difference of the required time steps to classify between the least and the most difficult images diminishes the room of further improvement.

Fig. 13 shows the number of classified images and accuracy achieved until the corresponding time step when the early decision scheme is exploited in Net4 with 1%p allowed accuracy loss. The number of classified images increases steeply after 21 time steps (classification decision cannot be made during the initial 2 periods because the membrane potentials of output neurons are not updated for the initial 2 periods) and about 94% of test images are classified with 90.55% accuracy within only 40 time steps. The remaining 6% images, most of which belong to the leftmost

[4] In Net1, a classification decision cannot be made for the initial 16 time steps because K is set to 4 and the membrane potential updates of output neurons are skipped for the initial 4 periods.

Table 3
Comparison with other SNN algorithms.

Dataset	SNN algorithm	Spike coding	Network	Acc. (%)	# Neurons	# Params	Latency	# Spikes
Iris	SpikeProp [25]	Temporal	50-10-3	96.1	63	24,000	10	–
	NPBLR [35]	Temporal	48-4-3	95	55	60	400	–
	Dyn. threshold [20]	Temporal	9-42-3	96.0	54	168	340	–
	ROL [36]	Relative order	48–3	94.2	51	144	50	–
MNIST	ROL [36]	Relative order	784–10	90.3	884	50000	100	–
	Spike-DBN [38]	Rate	784-500-500-10	94.1	1794	0.6M	–	–
	STDP [22]	Rate	Two layers	95.0	7184	5.0M	350	–
	Weight norm. [42]	Rate	MLP (Net1)	98.6	3194	2.4M	20	1×10^6
	SNN-WS w/o ED[a]	Weighted spike	MLP (Net1)	98.6	3194	2.4M	24	8×10^6
	SNN-WS w/ ED	Weighted spike	MLP (Net1)	98.5	3194	2.4M	19	6×10^6
	Weight norm. [42]	Rate	CNN (Net2)	99.1	24784	29816	80	1×10^6
	SNN-WS w/o ED	Weighted spike	CNN (Net2)	99.2	24784	29816	16	3×10^6
	SNN-WS w/ ED	Weighted spike	CNN (Net2)	99.1	24784	29816	8	1×10^6
CIFAR-10	Spiking CNN [40]	Rate	CNN	77.4	38090	0.1M	400	2×10^7
	99.9% norm. [12]	Rate	CNN (Net4)	87.8	118282	2.2M	280	–
	SNN-WS w/o ED	Weighted spike	CNN (Net4)	89.2	118282	2.2M	117	4×10^8
	SNN-WS w/ ED	Weighted spike	CNN (Net4)	89.1	118282	2.2M	42	1×10^8

[a] ED means the early decision algorithm described in Section 5.3. 0.1%p accuracy loss is allowed here.

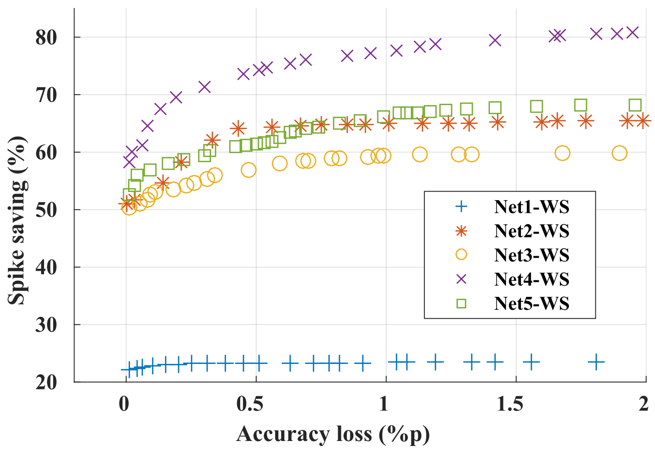


Fig. 12. Trade-off between accuracy and number of spikes by the early decision algorithm.

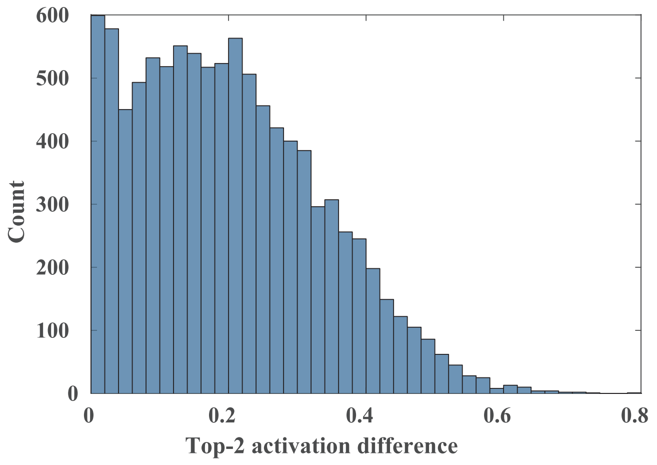


Fig. 14. Histogram of top-2 ReLU activation difference in the output layer of Net4 (ANN version) with 10,000 CIFAR-10 test images.

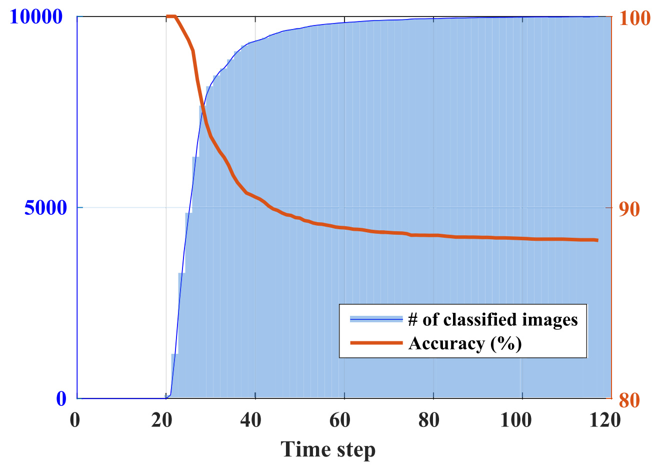


Fig. 13. Number of classified images (left) and classification accuracy (right) over time when the early decision scheme is applied in Net4 with 1%p allowed accuracy loss. The stable decision threshold (θ) is set to 0.9.

bar in Fig. 14, consume more time steps since they are difficult to classify correctly due to the small top-2 activation differences of output neurons. The last image (most difficult to classify) is classified in 117 time steps and the final accuracy of SNN-WS with early decision is 88.16% which is 0.94%p lower than the ANN accuracy (89.1%) of Net4. By the early decision scheme, the average classification latency is reduced from 117 time steps down to 27.6 time steps, which leads to 77% spike saving.

7.3. Comparison with other algorithms

Table 3 shows the comparison results of spike coding, classification accuracy, network size, classification latency in time steps, and number of generated spikes among different SNN algorithms. In the SNN algorithms based on temporal coding [20,25,35], information is packed in a spike timing, and it has higher information capacity than a rate coding and the proposed weighted spike. Thus they can perform classification tasks with less number of spikes theoretically. However, due to the difficulty of training based on spike timing, they are applied on a small dataset such as Iris (4 input features and 3 classes) [56], and there is no experimental result on larger dataset such as MNIST or CIFAR. To be applied on larger datasets, SNNs using temporal coding still require further researches.

The work on relative order learning [36], which encodes information on a relative order between spikes, shows that it is feasible on Iris and MNIST datasets. However, its classification accuracy on MNIST is just 90.3% which is far from the state-of-the-art accuracy 99%, and the method does not work on networks with more than two layers. Rate coding is used in most SNN algorithms [12,22,38,40,42] that show high classification accuracy on MNIST and CIFAR-10. The proposed SNN-WS achieves shorter classification latency than those with rate coding.

The number of spikes generated in a network is important since it is proportional to dynamic energy consumption. In SNNs, an incoming spike changes the membrane potential of a neuron and the process consumes dynamic energy and no dynamic energy is consumed if there is no spike. The SNN algorithms with temporal coding and relative order coding do not give the number of generated spikes since they are mainly focused on achieving high accuracy and training with deep and large networks.

In the work using rate coding [42], the number of generated spikes is less than or equal to that of SNN-WS. Since SNN-WS has higher average firing rate than SNN-RC, SNN-WS is not much competitive compared to SNN-RC when a small network is used because SNN-WS has higher average firing rate than SNN-RC and a small network does not have much room to reduce classification latency further by using SNN-WS. However, SNN-WS shows significant reduction in number of spikes for large networks as shown in Table 2. Although the work using rate coding [12] does not give the number of generated spikes for the case of Net4 on CIFAR-10[5], our experimental result of SNN-RC with Net4 on CIFAR-10 shows that more than 4×10^9 spikes are generated during an inference, which is much larger than that of SNN-WS. Moreover, exploiting the early decision algorithm enhances the classification latency and number of spikes further and makes SNN-WS faster and more energy-efficient.

8. Conclusion

We proposed a novel spiking neural network with weighted spikes to overcome the slow classification problem of the conventional SNNs with rate coding. The key idea is assigning different weights to spikes depending on the time phase within a period to encode more information within less number of spikes. The proposed SNN-WS reduces the classification latency as well as number of spikes significantly. The approach can be applied to various types of networks including deep residual networks and various datasets without accuracy loss. We also proposed an early decision algorithm to further reduce latency and number of spikes at the expense of small accuracy loss.

Acknowledgments

This work was supported by the KIST Institutional Program (Project No. 2E27330-17-P026).

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Proceedings of the NIPS, 2012, pp. 1097–1105.
[2] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, Signal Process. Mag. 29 (6) (2012) 82–97, doi:10.1109/MSP.2012.2205597.
[3] R. Collobert, J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning, in: Proceedings of the ICML, 2008, pp. 160–167, doi:10.1145/1390156.1390177.
[4] GPU-based deep learning inference: A performance and power analysis, 2015, (Nvidia White paper).
[5] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, D.S. Modha, A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm, in: Proceedings of the CICC, 2011, pp. 1–4, doi:10.1109/CICC.2011.6055294.
[6] J.M. Cruz-Albrecht, M.W. Yung, N. Srinivasa, Energy-efficient neuron, synapse and STDP integrated circuits, IEEE Trans. Biomed. Circ. Syst. 6 (3) (2012) 246–256, doi:10.1109/TBCAS.2011.2174152.
[7] P.A. Merolla, J.V. Arthur, R. Alvarez-Icaza, A.S. Cassidy, J. Sawada, F. Akopyan, B.L. Jackson, N. Imam, C. Guo, Y. Nakamura, et al., A million spiking-neuron integrated circuit with a scalable communication network and interface, Science 345 (6197) (2014) 668–673, doi:10.1126/science.1254642.
[8] B.V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A.R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J.V. Arthur, P.A. Merolla, K. Boahen, Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations, Proc. IEEE 102 (5) (2014) 699–716, doi:10.1109/JPROC.2014.2313565.
[9] S.B. Furber, F. Galluppi, S. Temple, L.A. Plana, The spinnaker project, Proc. IEEE 102 (5) (2014) 652–665, doi:10.1109/JPROC.2014.2304638.
[10] J. Vreeken, et al., Spiking Neural Networks, An Introduction, 2002, Institute for Information and Computing Sciences, Utrecht University Technical Report.
[11] D. Heeger, Poisson model of spike generation, 2000, (http://www.cns.nyu.edu/~david/handouts/poisson.pdf).
[12] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, Theory and tools for the conversion of analog to spiking convolutional neural networks, in: Proceedings of the NIPS Workshop on Computing with Spikes, 2016.
[13] J. Lisman, G. Buzsáki, A neural coding scheme formed by the combined function of gamma and theta oscillations, Schizophrenia Bull. 34 (5) (2008) 974–980, doi:10.1093/schbul/sbn060.
[14] C. Kayser, M.A. Montemurro, N.K. Logothetis, S. Panzeri, Spike-phase coding boosts and stabilizes information carried by spatial and temporal spike patterns, Neuron 61 (4) (2009) 597–608, doi:10.1016/j.neuron.2009.01.008.
[15] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in: Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.
[16] A. Krizhevsky, Learning multiple layers of features from tiny images, 2009, (Tech Report).
[17] W.M. Kistler, Spike-timing dependent synaptic plasticity: a phenomenological framework, Biolog. Cybern. 87 (5) (2002) 416–427, doi:10.1007/s00422-002-0359-5.
[18] Y. Dan, M.-m. Poo, Spike timing-dependent plasticity of neural circuits, Neuron 44 (1) (2004) 23–30, doi:10.1016/j.neuron.2004.09.007.
[19] T. Masquelier, S.J. Thorpe, Unsupervised learning of visual features through spike timing dependent plasticity, PLoS Computat. Biol. 3 (2) (2007) e31, doi:10.1371/journal.pcbi.0030031.
[20] T.J. Strain, L. McDaid, T.M. McGinnity, L.P. Maguire, H.M. Sayers, An STDP training algorithm for a spiking neural network with dynamic threshold neurons, Int. J. Neural Syst. 20 (06) (2010) 463–480, doi:10.1142/S0129065710002553.
[21] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, G. Cauwenberghs, Event-driven contrastive divergence for spiking neuromorphic systems, Front. Neurosci. 7 (2014) 272, doi:10.3389/fnins.2013.00272.
[22] P.U. Diehl, M. Cook, Unsupervised learning of digit recognition using spike-timing-dependent plasticity, Front. Comput. Neurosci. 9 (2015) 99, doi:10.3389/fncom.2015.00099.
[23] T. Iakymchuk, A. Rosado-Muñoz, J.F. Guerrero-Martínez, M. Bataller-Mompeán, J.V. Francés-Víllora, Simplified spiking neural network architecture and STDP learning algorithm applied to image classification, EURASIP J. Image Video Process. 2015 (1) (2015) 1–11, doi:10.1186/s13640-015-0059-4.
[24] Y.A. LeCun, L. Bottou, G.B. Orr, K.-R. Müller, Efficient backprop, in: Neural Networks: Tricks of the Trade, Springer, 2012, pp. 9–48, doi:10.1007/3-540-49430-8_2.
[25] S.M. Bohte, J.N. Kok, H. La Poutre, Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing 48 (1) (2002) 17–37, doi:10.1016/S0925-2312(01)00658-0.
[26] B. Schrauwen, J. Van Campenhout, Extending spikeprop, in: Proceedings of the IJCNN, 2004, pp. 471–475, doi:10.1109/IJCNN.2004.1379954.
[27] S.M. Silva, A.E. Ruano, Application of levenberg-marquardt method to the training of spiking neural networks, in: Proceedings of the ICNNB, 2005, pp. 1354–1358, doi:10.1109/ICNNB.2005.1614882.
[28] S. McKennoch, D. Liu, L.G. Bushnell, Fast modifications of the spikeprop algorithm, in: Proceedings of the IJCNN, 2006, pp. 3970–3977, doi:10.1109/IJCNN.2006.246918.
[29] S. Ghosh-Dastidar, H. Adeli, Improved spiking neural networks for eeg classification and epilepsy and seizure detection, Integr. Comput.-Aided Eng. 14 (3) (2007) 187–212.
[30] F. Ponulak, A. Kasiński, Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting, Neural Comput. 22 (2) (2010) 467–510, doi:10.1162/neco.2009.11-08-901.
[31] B. Widrow, M.E. Hoff, Adaptive switching circuits, in: IRE WESCON Convention Record, 1960, pp. 96–104.
[32] B. Widrow, Generalization and information storage in network of adaline neurons, Self-Organ. Syst. (1962) 435–462.
[33] A. Taherkhani, A. Belatreche, Y. Li, L.P. Maguire, DL-ReSuMe: a delay learning-based remote supervised method for spiking neurons, Trans. Neural Netw. Learn. Syst. 26 (12) (2015) 3137–3149, doi:10.1109/TNNLS.2015.2404938.
[34] Y. Xu, X. Zeng, S. Zhong, A new supervised learning algorithm for spiking neurons, Neural Comput. 25 (6) (2013) 1472–1511, doi:10.1162/NECO_a_00450.
[35] X. Xie, H. Qu, G. Liu, M. Zhang, Efficient training of supervised spiking neural networks via the normalized perceptron based learning rule, Neurocomputing 241 (2017) 152–163, doi:10.1016/j.neucom.2017.01.086.
[36] Z. Lin, D. Ma, J. Meng, L. Chen, Relative ordering learning in spiking neural network for pattern recognition, Neurocomputing (2017), doi:10.1016/j.neucom.2017.05.009.
[37] Y. LeCun, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324, doi:10.1109/5.726791.

[5] We set the network configurations of Net1 and Net2 with reference to [42] and that of Net4 to [12] for comparison.

[38] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, M. Pfeiffer, Real-time classification and sensor fusion with a spiking deep belief network, Front. Neurosci. 7 (2013), doi:10.3389/fnins.2013.00178.
[39] J.A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, B. Linares-Barranco, Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets, IEEE Trans. Pattern Anal. Mach. Intel. 35 (11) (2013) 2706–2719, doi:10.1109/TPAMI.2013.71.
[40] Y. Cao, Y. Chen, D. Khosla, Spiking deep convolutional neural networks for energy-efficient object recognition, Int. J. Comput. Vis. 113 (1) (2015) 54–66, doi:10.1007/s11263-014-0788-3.
[41] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the AISTATS, 2011, pp. 315–323.
[42] P.U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, M. Pfeiffer, Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing, in: Proceedings of the IJCNN, 2015, pp. 1–8, doi:10.1109/IJCNN.2015.7280696.
[43] E.D. Adrian, The impulses produced by sensory nerve endings, J. Physiol. 61 (1) (1926) 49–72, doi:10.1113/jphysiol.1926.sp002281.
[44] S. Thorpe, A. Delorme, R. Van Rullen, Spike-based strategies for rapid processing, Neural Netw. 14 (6) (2001) 715–725, doi:10.1016/S0893-6080(01)00083-1.
[45] A. Belatreche, L. Maguire, M. McGinnity, Q. Wu, A method for supervised training of spiking neural networks, in: Proceedings of the CICA, 2003, pp. 39–44.
[46] S.J. Thorpe, Spike arrival times: a highly efficient coding scheme for neural networks, Parallel Process. Neural Syst. (1990) 91–94.
[47] E.M. Izhikevich, Resonance and selective communication via bursts in neurons having subthreshold oscillations, BioSystems 67 (1) (2002) 95–102.
[48] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the ICML, 2015, pp. 448–456.
[49] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the CVPR, 2016, pp. 770–778, doi:10.1109/CVPR.2016.90.
[50] M. Lin, Q. Chen, S. Yan, Network in network, in: Proceedings of the ICLR, 2014.
[51] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv preprint arXiv:1207.0580 (2012).
[52] A. Hyvärinen, J. Karhunen, E. Oja, Independent component analysis, John Wiley & Sons, 2004.
[53] C.-Y. Lee, S. Xie, P.W. Gallagher, Z. Zhang, Z. Tu, Deeply-supervised nets, in: Proceedings of the AISTAT, 2015, pp. 562–570.
[54] F. Chollet, Keras, 2015, (https://github.com/fchollet/keras).
[55] A. Vedaldi, K. Lenc, MatConvNet: Convolutional neural networks for MATLAB, in: Proceedings of the ACMMM, 2015, doi:10.1145/2733373.2807412.
[56] E. Anderson, The species problem in iris, Annals Missouri Botan. Garden 23 (3) (1936) 457–509.

Jaehyun Kim received the B.S. degree in electrical and computer engineering from Yonsei University, Seoul, Korea, in 2006, and the M.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2008. He is currently pursuing the Ph.D. degree in Seoul National University, Seoul, Korea. From 2008 to 2014, he was with Samsung Electronics Co, Ltd., Hwaseong, Korea. His research interests include learning algorithms and accelerator architectures for deep learning.

Heesu Kim received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, Korea in 2015. After 2015, he is a Ph.D. candidate (integrated with M.S. degree) at Seoul National University. He has been interested in deep learning research fields which include spiking neural network, meta-learning, and continual-learning.

Subin Huh is expected to receive the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, Korea in 2018. She is an undergraduate visitor in Perception and Intelligence Lab, Seoul National University. Her research primarily focuses on spiking neural networks and stochastic computing.

Jinho Lee received the B.S. degree in electrical engineering from Seoul National University, Seoul, Korea, in 2009. He received the M.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University in 2011 and 2016, respectively. He is currently working at IBM research as a postdoctoral researcher. His research interests include memory system, graph databases and domain specific accelerators.

Kiyoung Choi received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1978, the M.S. degree in electrical and electronics engineering from Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 1980, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 1989. From 1989 to 1991, he was with Cadence Design Systems, Inc. In 1991, he joined the faculty of the Department of Electrical and Computer Engineering, Seoul National University. His primary research interests include computer-aided electronic systems design, low-power systems design, computer architecture, and machine learning.