Progressive Tandem Learning for Pattern Recognition With Deep Spiking Neural Networks

Jibin Wu, Chenglin Xu, Xiao Han, Daquan Zhou, Malu Zhang, Haizhou Li, *Fellow, IEEE*, and Kay Chen Tan, *Fellow, IEEE*

Abstract—Spiking neural networks (SNNs) have shown clear advantages over traditional artificial neural networks (ANNs) for low latency and high computational efficiency, due to their event-driven nature and sparse communication. However, the training of deep SNNs is not straightforward. In this paper, we propose a novel ANN-to-SNN conversion and layer-wise learning framework for rapid and efficient pattern recognition, which is referred to as *progressive tandem learning*. By studying the equivalence between ANNs and SNNs in the discrete representation space, a primitive network conversion method is introduced that takes full advantage of spike count to approximate the activation value of ANN neurons. To compensate for the approximation errors arising from the primitive network conversion, we further introduce a layer-wise learning method with an adaptive training scheduler to fine-tune the network weights. The progressive tandem learning framework also allows hardware constraints, such as limited weight precision and fan-in connections, to be progressively imposed during training. The SNNs thus trained have demonstrated remarkable classification and regression capabilities on large-scale object recognition, image reconstruction, and speech separation tasks, while requiring at least an order of magnitude reduced inference time and synaptic operations than other state-of-the-art SNN implementations. It, therefore, opens up a myriad of opportunities for pervasive mobile and embedded devices with a limited power budget.

Index Terms—Deep spiking neural network, ANN-to-SNN conversion, spike-based learning, large-scale object recognition, speech separation, efficient neuromorphic inference

1 Introduction

Human brains, after evolving for many hundreds of millions of years, are incredibly efficient and capable of performing complex pattern recognition tasks. In recent years, the deep artificial neural networks (ANNs) that are inspired by the hierarchically organized cortical networks have become the dominant approach for many pattern recognition tasks and achieved remarkable successes in a wide spectrum of application domains, instances include speech processing [1], [2], computer vision [3], [4], language understanding [5] and robotics [6]. The deep ANNs, however, are notoriously expensive to operate both in terms of computational cost and memory usage. Therefore, they are prohibited from large-scale deployments in pervasive mobile and Internet-of-Things (IoT) devices.

In contrast, the adult's brains only consume about 20 watts to perform complex perceptual and cognitive tasks that are only equivalent to the power consumption of a dim light bulb [7]. While many efforts are devoted to improving the memory and computational efficiency of deep ANNs, for example, network compression [8], network quantization [9] and knowledge distillation [10], it is more interesting to exploit the efficient computation paradigm inherent to the biological neural systems that are fundamentally different from and potentially integratable with the aforementioned strategies.

The spiking neural networks (SNNs) are initially introduced to study the functioning and organizing mechanisms of biological brains. Recent studies have shown that deep ANNs also benefit from biologically realistic implementation, such as event-driven computation and sparse communication [11], for computational efficiency. Neuromorphic computing (NC), as an emerging non-von Neumann computing paradigm, aims to mimic the biological neural systems with SNNs in silicon [12]. The novel neuromorphic computing architectures, including Tianjic [13], TrueNorth [14], and Loihi [15], have shown compelling throughput and energy-efficiency in pattern recognition tasks, crediting to their inherent event-driven computation and fine-grained

- *Jibin Wu, Xiao Han, Daquan Zhou, and Malu Zhang are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077, Singapore. E-mail: {jibin.wu, e0269084, daquan.zhou}@u.nus.edu, maluzhang@nus.edu.sg.*
- *Chenglin Xu is with the School of Computer Science and Engineering and Temasek Laboratories @ NTU, Nanyang Technological University, Singapore 639798, Singapore. E-mail: xuchenglin@ntu.edu.sg.*
- *Haizhou Li is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077, Singapore, and also with the School of Data Science, The Chinese University of Hong Kong, Shenzhen 518172, China. E-mail: haizhou.li@nus.edu.sg.*
- *Kay Chen Tan is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, Hong Kong. E-mail: kctan@polyu.edu.hk.*

Manuscript received 30 June 2020; revised 16 Aug. 2021; accepted 7 Sept. 2021. Date of publication 21 Sept. 2021; date of current version 3 Oct. 2022.
*This work was supported in part by A*STAR under RIE2020 Advanced Manufacturing and Engineering Domain (AME) Programmatic under Grant (A1687b0033, Project Title: Spiking Neural Networks), in part by the IAF, A*STAR, SOITEC, NXP, and by the National University of Singapore under FD-fAbrICS: Joint Lab for FD-SOI Always-on Intelligent & Connected Systems under Grant I2001E0053. The work of Jibin Wu was supported by the Zhejiang Lab under Grant 2019KC0AB02. The work of Malu Zhang was supported in part by the China Postdoctoral Science Foundation under Grant 2020M680148 and in part by the Zhejiang Lab's International Talent Found for Young Professionals.*
(Corresponding author: Malu Zhang.)
Recommended for acceptance by C. Wang.
Digital Object Identifier no. 10.1109/TPAMI.2021.3114196

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

parallelism of the computing units. Moreover, the co-located memory and computation can effectively mitigate the problem of low bandwidth between the computing units and memory (i.e., von Neumann bottleneck) in data-driven pattern recognition tasks.

While it remains a challenge to train large-scale spiking neural networks that can be deployed onto these NC chips for real-world pattern recognition tasks. Due to the discrete and hence non-differentiable nature of spiking neuronal function, the powerful back-propagation (BP) algorithm that is widely used for deep ANN training is not directly applicable to the SNN.

Recent studies suggest that the dynamical system formed by spiking neurons can be formulated as a recurrent ANN [16], whereby the subthreshold membrane potential dynamics of these leaky integrators (i.e., spiking neurons) can be effectively modeled. In addition, the discontinuity of the spike generation function can be circumvented with surrogate gradients that provide an unbiased estimation of the true gradients [17], [18], [19], [20], [21], [22]. In this way, the canonical error back-propagation through time algorithm (BPTT) can be applied to optimize the SNN. However, it is both computation- and memory-inefficient to optimize the SNN using the BPTT algorithm since spike trains are typically very sparse in both time and space. Therefore, the scalability of the technique remains to be improved, for instance, the size of SNNs is GPU memory bounded as demonstrated in a gesture classification task [19]. Furthermore, the vanishing and exploding gradient problem [23] of the BPTT algorithm adversely affects the learning in face of input spike trains of long temporal duration or low firing rate.

To address the aforementioned issues in surrogate gradient learning, a novel tandem learning framework [24] has been proposed. This learning framework consists of an ANN and an SNN coupled through weight sharing, wherein the SNN is used to derive the exact neural representation, while the ANN is designed to approximate the surrogate gradients at the spike-train level. The SNNs thus trained have demonstrated competitive classification and regression capabilities on a number of frame- and event-based benchmarks, with significantly reduced computational cost and memory usage. Despite the promising learning performance demonstrated by these spike-based learning methods, their applicability to deep SNNs with more than 10 hidden layers remains elusive.

Following the idea of rate-coding, recent studies have shown that SNNs can be effectively constructed from ANNs by approximating the activation value of ANN neurons with the firing rate of spiking neurons [25], [26], [27], [28], [29], [30]. This approach not only simplifies the training procedures of the aforementioned spike-based learning methods but also enable SNNs to achieve the best-reported results on a number of challenging tasks, including object recognition on the ImageNet-12 dataset [27], [28] and object detection on the PASCAL VOC and MS COCO datasets [29]. However, to reach a reliable firing rate approximation, it requires a notoriously large encoding time window with at least a few hundred time steps. Moreover, the total number of synaptic operations required to perform one classification usually increases with the size of the encoding time window, therefore, a large encoding time window will also adversely impact the computational efficiency. An ideal SNN model should not only perform pattern recognition tasks with high accuracy but also obtained the results rapidly with as few time steps as possible, and efficiently with a small number of synaptic operations. In this work, we introduce a novel ANN-to-SNN conversion and learning framework to progressively convert a pre-trained ANN into an SNN for accurate, rapid, and efficient pattern recognition.

To improve the inference speed and energy efficiency, we introduce a layer-wise threshold determination mechanism to make good use of the encoding time window of spiking neurons for information representation. To maintain a high pattern recognition accuracy, a layer-wise learning method with an adaptive training scheduler is further applied to fine-tune the network weights after each primitive layer conversion that compensates for the conversion errors. The proposed layer-wise conversion and learning framework also supports effective algorithm-hardware co-design by progressively imposing hardware constraints during the training process. To summarize, the main contributions of this work are in four aspects:

- *Rethinking ANN-to-SNN Conversion:* We introduce a new perspective to understand the neural discretization process of spiking neurons by comparing it to the activation quantization of ANN neurons, which offers a new angle to understand and perform network conversion. By making efficient use of the spike count that is upper bounded by the encoding time window size to represent the information of counterparts, the inference speed, and computational cost can be significantly reduced over other conversion methods grounded on a firing rate approximation.
- *Progressive Tandem Learning Framework:* We propose a novel layer-wise ANN-to-SNN conversion and learning framework with an adaptive training scheduler to support effortless and efficient conversion, which allows fast, accurate, and efficient pattern recognition with deep SNNs.[1] The proposed conversion framework also allows easy incorporation of hardware constraints into the training process, for instance, limited weight precision and fan-in connections, such that the optimal performance can be achieved when deploying onto the actual neuromorphic chips.
- *Rethinking Spike-based Learning Methods:* We conduct a comprehensive study on the scalability of both the time-based surrogate gradient learning and the spike count-based tandem learning methods, revealing that the accumulated gradient approximation errors may impede the training convergence in deep SNNs.
- *Solving Cocktail Party Problem with SNN:* To evaluate the proposed learning framework, we apply deep SNNs to separate high fidelity voices from a mixed multiple-talker speech, which effectively mimics the perceptual and cognitive ability of the human brain. To the best of our knowledge, this is the first work that successfully applied deep SNNs to solve the challenging cocktail party problem.

1. The source codes of the PTL framework are public available at: https://github.com/deepspike/progressive_tandem_learning

The rest of the paper is organized as follows. In Section 2, we first review the conventional ANN-to-SNN conversion methods and discuss the trade-off between accuracy and latency. In Section 3, we compare the neuronal functions between the spiking neurons and ANN neurons, and their discrete equivalents, which provide a new perspective to perform network conversion. With this, we propose to use the spike count of spiking neurons as the bridge between the spiking neurons and their ANN counterparts for network conversion. In Section 4, to minimize the conversion errors, we propose a novel layer-wise learning method with an adaptive training scheduler to fine-tune network weights. In Sections 5 and 6, we validate the proposed network conversion and learning framework, that is referred to as *progressive tandem learning* (PTL), through a set of classification and regression tasks, including the large-scale image classification, time-domain speech separation and image reconstruction. Finally, we conclude the paper in Section 7.

2 Related Work

Recently, many ANN-to-SNN conversion methods are proposed. Nearly all of these methods follow the idea of rate-coding, which approximates the activation value of ANN neurons with the firing rate of spiking neurons. In what follows, we will review the development of ANN-to-SNN conversion methods and highlight the issue of accuracy and latency trade-off in these methods.

The earliest attempt for ANN-to-SNN conversion was presented in [31], where Pérez-Carrasco *et al.* devised an approximation method for leaky integrate-and-fire (LIF) neurons using ANN neurons. The pre-trained weights of ANN neurons are rescaled by considering the leaky rate and other parameters of spiking neurons before copying into the SNN. This conversion method was proposed to handle event streams captured by the event-driven camera, whereby promising recognition results were demonstrated on the human silhouette orientation and poker card symbol recognition tasks. While this conversion method requires a large number of hyperparameters to be determined manually and the conversion process suffers from quantization and other approximation errors.

There are recent studies on ANN-to-SNN conversion with applications to accurate object recognition and detection on frame-based images. Cao *et al.* [25] proposed a conversion framework by using the rectified linear unit (ReLU) as the activation function for ANN neurons and set the bias term to zero. The activation value of ANN neurons can thus be well approximated by the firing rate of integrate-and-fire (IF) neurons. Furthermore, the max-pooling operation, which is hard to determine in the temporal domain for a rate-based SNN, is replaced with the average pooling. Diehl *et al.* [26] further improved this conversion framework by analyzing the causes of performance degradation, which reveals the potential problems of over- and under-activation of spiking neurons. To address these problems, they proposed model- and data-based weight normalization schemes to rescale the SNN weights based on the maximum activation values of ANN neurons. These normalization schemes prevent the over- and under-activation of spiking neurons and strike a good balance between the firing threshold and the model weights. As a result, near-lossless classification accuracies were reported on the MNIST dataset with fully connected and convolutional spiking neural networks.

Rueckauer *et al.* [27] identified a quantization error caused by the reset-to-zero scheme of IF neurons, where surplus membrane potential over the firing threshold is discarded after firing. This quantization error tends to accumulate over layers and severely impacts the classification accuracy of converted deep SNNs. To address this problem, they propose a reset-by-subtraction scheme to preserve the surplus membrane potential after each firing. Moreover, a modified data-based weight normalization scheme is introduced to improve the robustness against outliers, which significantly improves the firing rate of spiking neurons and hence the inference speed of SNN. For the first time, they had demonstrated competitive results to the ANN counterparts on the challenging ImageNet-12 object recognition task.

In the same line of research, Hu *et al.* [30] provided a systematic approach to convert deep residual networks and propose an error compensation scheme to address the accumulated quantization errors. With these modifications, they achieved near-lossless conversion for spiking residual networks up to 110 layers. Kim *et al.* [29] extended the conversion framework by applying the weight normalization channel-wise for convolutional neural networks and propose an effective strategy for converting ANN neurons with both positive and negative activation values. The proposed channel-wise normalization scheme boosted the firing rate of neurons and hence improved the information transmission rate. Benefiting from these modifications, competitive results are demonstrated in the challenging objection detection task where the precise coordinate of bounding boxes is required to be predicted. Sengupta *et al.* [28] further optimized the weight normalization scheme by taking into consideration the behavior of spiking neurons at the run time, which achieved the best-reported result on the ImageNet-12 dataset. To improve the applicability of the aforementioned conversion methods to the pooling layer as well as to reduce the overall computational overhead, Xu *et al.* [32] and Wang *et al.* [33] proposed to normalize the firing threshold instead of the weights.

In these earlier studies, methods are proposed for the firing threshold determination or weight normalization so as to achieve a good firing rate approximation. Despite competitive results achieved by these conversion methods, the underlying firing-rate assumption has led to an inherent trade-off between accuracy and latency, which requires a few hundred to thousands of time steps to reach a stable firing rate. Rueckauer *et al.* [27] provided a theoretical analysis of this issue by analyzing the firing rate deviation of these ANN-to-SNN conversion methods. By assuming a constant input current to spiking neurons at the first layer, the actual firing rate of the first (Eq. (1)) and subsequent layers (Eq. (2)) can be summarised as follows

$$\mathrm{r}_i^1(t) = a_i^1 r_{\max} - \frac{V_i^1(t)}{t\vartheta} \tag{1}$$

$$\mathrm{r}_i^l(t) = \sum_j^{M^{l-1}} w_{ij}^l \mathrm{r}_j^{l-1}(t) + b_i^l \mathrm{r}_{\max} - \frac{V_i^l(t)}{t\vartheta}, \tag{2}$$

where $\mathrm{r}_i^l(t)$ denotes the firing rate of neuron i at layer l and $r_{\max}$ denotes the maximum firing rate that is determined by the time step size. a_i^1 is the activation value of ANN neuron i at the first layer, $V_i^1(t)$ is the membrane potential of the corresponding spiking neuron, and ϑ is the neuronal firing threshold. M^{l-1} is the total number of neurons in layer $l-1$ and b_i^l is the bias term of ANN neuron i at layer l. Ideally, the firing rate of spiking neurons should be proportional to the activation value of their ANN counterparts as per the first term of Eq. (1). While the surplus membrane potential that has not been discharged by the end of simulation will cause an approximation error as shown by the second term of Eq. (1), which can be counteracted with a large firing threshold or a large encoding time window. Since increasing the firing threshold will inevitably prolong the evidence accumulation time, a proper firing threshold that can prevent spiking neurons from either under- or over-activating is usually preferred and the encoding time window is extended to minimize such a firing rate approximation error.

Besides, this approximation error accumulates gradually while propagating over layers as shown in Eq. (2), thereby a further extension of the encoding time window is required to compensate. As such, a few thousand time steps are typically required to achieve a competitive accuracy for deep SNNs with more than 10 layers [28], [29]. From these formulations, it is clear that to approximate the continuous input-output representation of ANNs with the firing rate of spiking neurons will inevitably lead to the accuracy and latency trade-off. To overcome this issue, as will be introduced in the following sections, we propose a novel conversion method that is grounded on the discrete neural representation, whereby the spike count, upper bounded by the encoding time window size, is taken to approximate the discrete input-output representation of ANNs. To make efficient use of the spike count for information representation, we propose a novel firing threshold determination strategy such that rapid and efficient pattern recognition can be achieved with SNNs. To counteract the conversion errors and hence ensure high accuracies in pattern recognition tasks, a layer-wise learning method is further proposed to fine-tune the network.

3 Rethinking ANN-to-SNN Conversion

Over the years, many spiking neuron models are developed to describe the rich dynamical behavior of biological neurons. Most of them, however, are too complex for real-world pattern recognition tasks. As discussed in Section 2, for computational simplicity and ease of conversion, the IF neuron model is commonly used in ANN-to-SNN conversion works [26], [27], [28]. Although this simplified spiking neuron model does not emulate the rich sub-threshold dynamics of biological neurons, it preserves attractive properties of discrete and sparse communication, therefore, allows for efficient hardware implementation. In this section, we reinvestigate the approximation of input-output representation between a ReLU ANN neuron and an integrate-and-fire spiking neuron.

3.1 Spiking Neuron Versus ANN Neuron

Let us consider a discrete-time simulation of spiking neurons with an encoding time window of N_s that determines the inference speed of an SNN. At each time step t, the incoming spikes to the neuron i at layer l are transduced into synaptic current $z_i^l[t]$ according to

$$z_i^l[t] = \sum_j w_{ij}^{l-1} s_j^{l-1}[t] + b_i^l, \tag{3}$$

where $s_j^{l-1}[t]$ indicates the occurrence of an input spike at time step t, and w_{ij}^{l-1} is the synaptic weight between the pre-synaptic neuron j and the post-synaptic neuron i at layer l. b_i^l can be interpreted as a constant injecting current.

The synaptic current $z_i^l[t]$ is further integrated into the membrane potential $V_i^l[t]$ as per Eq. (4). Without loss of generality, a unitary membrane resistance is assumed in this work. The membrane potential is reset by subtracting the firing threshold after each firing as described by the last term of Eq. (4).

$$V_i^l[t] = V_i^l[t-1] + z_i^l[t] - \vartheta^l s_i^l[t-1]. \tag{4}$$

An output spike is generated whenever the $V_i^l[t]$ rises above the firing threshold ϑ^l (determined layer-wise) as follows

$$s_i^l[t] = \Theta(V_i^l[t] - \vartheta^l) \;\; with \;\; \Theta(x) = \begin{cases} 1, & if\ x \geq 0 \\ 0, & otherwise \end{cases}. \tag{5}$$

The spike train s_i^l and spike count c_i^l for a time window of N_s can thus be determined and represented as follows

$$\begin{aligned} s_i^l &= \{s_i^l[1], \ldots, s_i^l[N_s]\} \\ c_i^l &= \sum_{t=1}^{N_s} s_i^l[t]. \end{aligned} \tag{6}$$

For non-spiking ANN neurons, let us describe the neuronal function of neuron i at layer l as

$$a_i^l = f\left(\sum_j w_{ij}^{l-1} x_j^{l-1} + b_i^l\right), \tag{7}$$

which has w_{ij}^{l-1} and b_i^l as the weight and bias. x_j^{l-1} and a_i^l denote the input and output of the ANN neuron. $f(\cdot)$ denotes the activation function, which we use the ReLU in this work. For ANN-to-SNN conversion, an ANN with the ReLU neurons is first trained, that is called pre-training, before the conversion.

3.2 Neural Discretization Versus Activation Quantization

In the conventional ANN-to-SNN conversion studies, the firing rate of spiking neurons is usually taken to approximate the continuous input-output representation of the pre-trained ANN. As discussed in Section 2, a spiking neuron takes a notoriously long time window to reliably approximate a continuous value. Recent studies, however, suggest such a continuous neural representation may not be necessary for ANNs [34]. In fact, there could be little impact on the network performance when the activation value of ANN neurons are properly quantized to a low-precision

discrete representation [35], [36], which is known as activation quantization.

In ANNs, the activation quantization refers to the mapping of a floating-point activation value $a_i^{l,f}$ to a quantized value $a_i^{l,q}$. With a ReLU activaiton function, the activation quantization can be formulated as follows

$$\begin{aligned} \hat{a}_i^{l,f} &= \min(\max(a_i^{l,f}, 0), a_u^l) \\ \varphi^l &= \frac{a_u^l}{N_q} \\ a_i^{l,q} &= round\left(\frac{\hat{a}_i^{l,f}}{\varphi^l}\right) \cdot \varphi^l, \end{aligned} \tag{8}$$

where a_u^l refers to the upper bound of the quantization range at layer l, whose values are usually determined from the training data. N_q is the total number of quantization levels and φ^l is the quantization scale for layer l. With such a discrete neural representation, the computation and storage overheads during the training and inference of ANNs can be significantly reduced. The success of activation quantization can be explained by the fact that there is a high level of redundancy in the continuous neural representation.

In SNNs, the information is inherently discretized into spike trains according to the neuronal dynamics of spiking neurons, which is referred to as the neural discretization hereafter. It is worth noting that the size of the encoding time window determines the discrete representation space for SNNs. The activation quantization of ANNs leads to a reduction in data storage, which takes place in the spatial domain. By mapping the discrete neural representation of a good performing ANN to an SNN, it is expected that we translate the reduction of the data storage into the reduction of the encoding time window size, thus allowing rapid and efficient pattern recognition with SNNs.

The ANN neurons respond to the input stimuli instantly, while spiking neurons respond to the input spike trains through a temporal process within a time window. In order to establish a correspondence between the activation quantization of ANN neurons and the neural discretization of spiking neurons, we simplify the neural discretization process by assuming the preceding layer's spike trains and the constant injecting current are integrated and discharged instantly. The overall contributions from the preceding layer's spike trains and constant injecting current can be summarized by the free aggregate membrane potential (no firing) [24] defined as

$$V_i^l = \sum_j w_{ij}^{l-1} c_j^{l-1} + b_i^l N_s. \tag{9}$$

By considering $b_i^l N_s$ as the bias term and c_j^{l-1} as the input to ANN neurons that defined in Eq. (7), V_i^l is exactly the same as the pre-activation quantity of non-spiking ANN neurons. By considering the spike count of spiking neurons as the information carrier, the simplification of neural discretization provides the basis for mapping the discrete inputs of an ANN neuron to the discrete spike count inputs of a spiking neuron.

Note that an IF neuron responds to the input spike trains by firing zero or a positive number of output spikes. It performs a non-linear transformation similar to that of the ReLU activation function of an ANN neuron. As defined in Eq. (8), the activation quantization discretizes the positive activation value of ReLU neurons, by a fixed quantization scale φ^l, into an integer. Similarly, the neural discretization of an IF neuron discretizes the positive-valued V_i^l by a fixed discretization scale, that is the firing threshold ϑ^l, into a discrete spike count, that can be formulated as follows

$$\begin{aligned} \hat{V}_i^l &= \min(\max(V_i^l, 0), V_u^l) \\ \vartheta^l &= \frac{V_u^l}{N_s} \\ V_i^{l,q} &= \underbrace{round\left(\frac{\hat{V}_i^l}{\vartheta^l}\right)}_{\approx c_i^l} \cdot \vartheta^l, \end{aligned} \tag{10}$$

where V_u^l refers to the free aggregated membrane potential upper bound of layer l. The Eqs. (8) and (10) establish a correspondence between the activation quantization of a ReLU neuron and the discrete neural representation of an IF neuron, thus provides the basis for mapping the discrete output of an ANN neuron to the spike count output of a spiking neuron. It is worth noting that the quantization scale φ^l is usually stored independently for ANNs and multiplied to the fixed point number during operations. However, the discretization scale ϑ^l is only stored at the spiking neuron and does not propagate together with output spike trains to the next layer. This issue can be easily counteracted by multiplying ϑ^l to the weights of neurons in the subsequent layer $l+1$.

With the simplification of neural discretization, we show that the discrete input-output representation of ANN neurons can be well approximated with spiking neurons. Following this formulation, an SNN can be constructed from the pre-trained ANN by directly copying its weights. The constant injecting current to spiking neurons can be determined by dividing the bias term of the corresponding ANN neuron over N_s. According to Eq. (10), the firing threshold ϑ^l of spiking neurons at layer l can be determined by dividing the upper bound V_u^l over N_s. From Eqs. (7) and (9), it clear that the upper bound V_u^l is equivalent to and hence can be directly taken from the maximum activation value a_u^l of the corresponding ANN layer.

However, two potential errors may arise from this formulation: a quantization error affected by the encoding time window size and a spike count approximation error arising from the temporal structure of input spike trains that may affect the actual discharging of V_i^l. These conversion errors, however, can be effectively mitigated by the threshold normalization mechanism and the layer-wise training method that will be introduced in the following sections.

3.3 Threshold LayerNorm

To better represent the quantization range of ANN neurons using spiking neurons that have a pre-defined encoding time window N_s, we introduce a novel threshold determination mechanism for spiking neurons. To properly define the quantization range of ANN neurons in a layer, we need to determine the activation value upper bound a_u^l.

As shown in Fig. 1 and also highlighted in [27], the a_u^l tends to be biased by the outlier samples, for instance, the

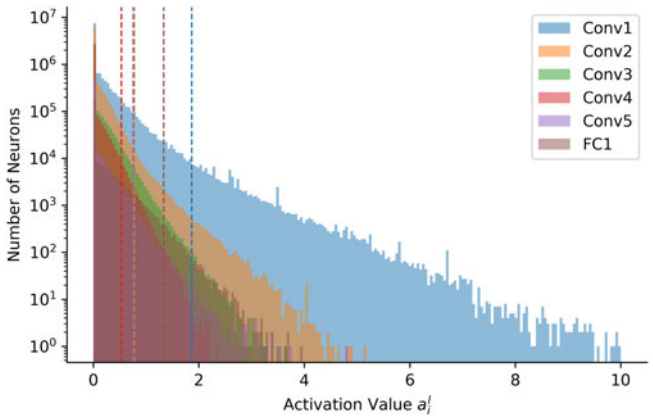


Fig. 1. Distribution of the activation value a_i^l of ReLU neurons in the pre-trained ANN layers. Here, the horizontal axis represents the activation values, while the vertical axis represents the number of neurons in a log scale. The majority of neurons output low activation values and the number of neurons decreases rapidly as the activation value increases. The dotted lines mark the 99th percentile of the number of neurons in each layer.

a_u^1 of Conv1 layer is five times larger than the 99th percentile (highlighted as the blue dotted line). To make efficient use of the available discrete representation space and reduce the quantization errors, we propose to use the 99th or 99.9th percentile of all a_i^l in a layer, determined from the training data, as the upper bound a_u^l such that the key information can be well-preserved. Given the equivalence of a_u^l and V_u^l established in the earlier section, the firing threshold ϑ^l of spiking neurons at layer l can hence be determined by dividing the value of a_u^l over N_s. For percentiles larger than 99.9th, the calculated firing threshold is prone to be affected by the outlier. On the other hand, a percentile below 99th will result in some informative activation range not represented in the SNN. In practice, we observe these two percentiles remain relatively stable across data batches with a sufficiently large batch size (e.g., 128 or 256). Therefore, the a_u^l can be effectively derived from a random training batch.

To further improve the numerical resolution for convolutional neural networks, the firing threshold can be determined independently for each channel similar to that proposed in [29]. While we did not notice significant improvements in the classification or regression performance in our experiments, probably due to the layer-wise learning method that we have applied counteracts the performance drop.

3.4 Neural Coding

A suitable neural encoding scheme is required to convert the static input feature tensors or images into spike trains for neural processing in SNNs. It was found that a direct discretization of the inputs introduces significant distortions to the underlying information. While discretizing the feature tensors derived from the first network layer can effectively preserve the information by leveraging the redundancies in the high-dimensional feature representation [37]. Following this approach, we interpret the activation value a_i^l of ANN neurons as the input current to the corresponding spiking neurons and add it to Eq. (4) at the first time step. The spike trains are generated by distributing this quantity over consecutive time steps according to the dynamic of IF neurons; the spiking output then starts from the first hidden layer. This neural encoding scheme effectively discretizes the feature tensor and represents it as spike counts.

The neural decoding determines the output class from the synaptic activity of spiking neurons. Instead of using the discrete spike counts, we suggest using the free aggregate membrane potential of neurons in the final SNN layer to determine the output class, which provides a much smoother learning curve over the discrete spike count due to the continuous error gradients derived at the output layer [24]. Moreover, this continuous quantity can also be directly considered as the outputs in regression tasks, such as image reconstruction and speech separation that will be presented later in this paper. As will be explained in the following section, the neural encoding and decoding layers are added into the hybrid network shown in Fig. 2B at the first and the last network conversion stage, respectively.

4 Progressive Tandem Learning

The primitive ANN-to-SNN conversion method introduced in the earlier section provides a more efficient way to approximate the input-output representation of ANNs. However, the conversion process inherently introduces quantization and spike count approximation errors as discussed in Section 3.2. Such errors tend to accumulate over layers and cause significant performance degradation especially with a small N_s. This therefore calls for a training scheme to fine-tune the network weights after the primitive conversion, so as to compensate for these conversion errors.

There have been spike-based learning schemes, such as time-based surrogate gradient learning [16] and spike count-based tandem learning methods [24], for SNN training in an end-to-end manner. However, they don't work the best for the required fine-tuning task. For example, the surrogate gradients approximated from these methods tend to be noisy for an extremely short encoding time window that we would like to have. As will be seen in Section 5.2, gradient approximation errors accumulate over layers with these end-to-end learning methods, which significantly degrade the learning performance for an SNN of over 10 layers.

To address this issue, we propose a layer-wise learning method, whereby ANN layers are converted into SNN layers one layer at a time to prevent the gradient approximation errors from accumulating. We define the conversion and weight fine-tuning of one SNN layer as one stage. Therefore, for an ANN network of L layers, as shown in Fig. 2A, it takes L stages to complete the entire conversion and fine-tuning process.

The details of each training stage are illustrated in Fig. 2C. All spiking neurons in the same SNN layer share the same firing threshold, which is first determined according to the proposed Threshold LayerNorm mechanism. Besides, the constant injecting current to spiking neurons is determined by dividing the corresponding bias term of ANN neurons over N_s. Following the tandem learning approach [24], a hybrid network is further constructed by coupling the converted SNN layer to the pre-trained ANN layer through weight sharing, thereafter the ANN layer becomes an auxiliary structure to facilitate the fine-tuning of the converted SNN layer. At each training stage, the PTL scheme follows

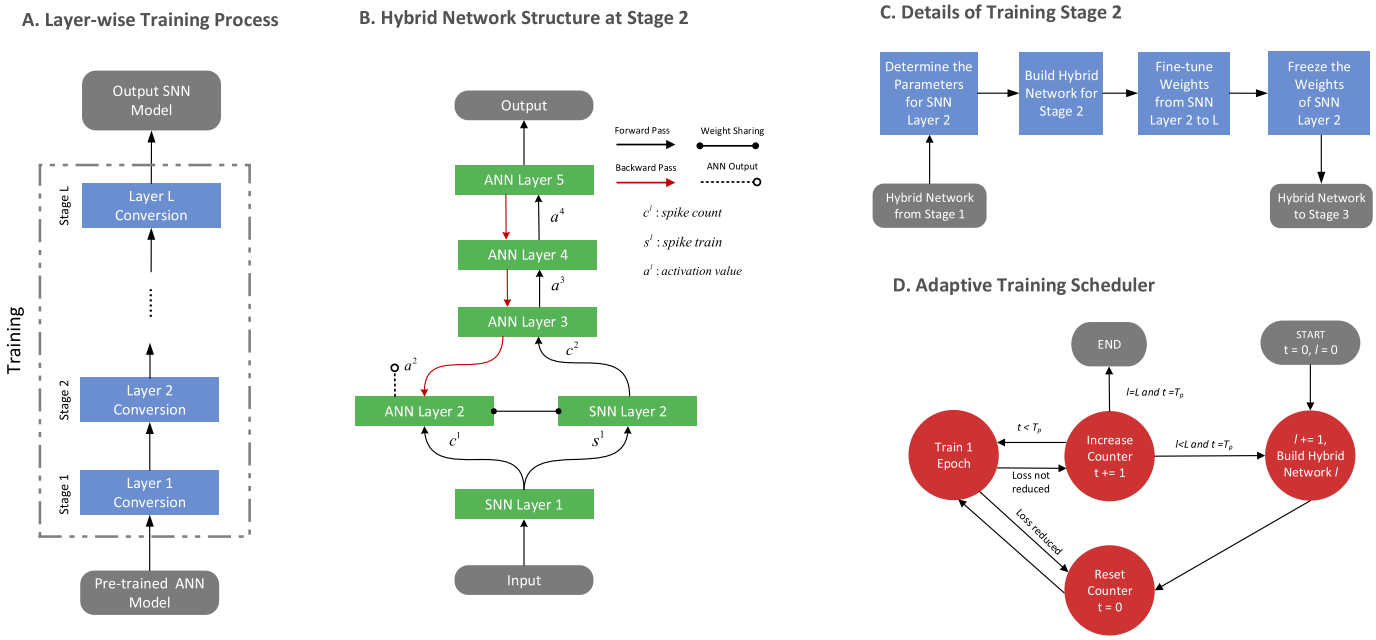


Fig. 2. Illustration of the proposed PTL framework. (A) The whole training process is organized into separate stages. (B) Details of the hybrid network at the training stage 2. Note that the SNN Layer 1 performs neural encoding following the process described in Section 3.4. (C) Details of the training processes at stage 2. (D) Illustration of the adaptive training scheduler.

the tandem learning idea except that 1) we fix the weights of the SNN layers in the previous stages; 2) we update only one SNN layer together with all ANN layers.

4.1 Tandem Learning

As shown in Fig. 2B, the spike trains, derived from the preceding SNN layer, and their equivalent spike counts are forward propagated to the coupled layer. In the coupled layer, the spiking neurons take spike trains as input and generate spike counts as output, while the ANN neurons take spike counts as input and generate an output quantity that approximates the spike count of the coupled spiking neurons. To allow for weight sharing between the ANN and the SNN layers, we take the spike counts as the bridge. To this end, let us express the non-linear transformation of a spiking neuron as

$$c_i^l = g(s^{l-1}; w_i^{l-1}, b_i^l, \vartheta^l), \tag{11}$$

where $g(\cdot)$ denotes the effective transformation performed by spiking neurons. Given the state-dependent nature of spike generation, it is not feasible to directly determine an analytical expression from s^{l-1} to c_i^l. Here, we simplify the spike generation process by assuming the resulting synaptic currents from s^{l-1} are evenly distributed over time. We thus obtain the interspike interval of the output spike train as

$$\Delta_i^l = \rho\left(\frac{\vartheta^l N_s}{(\sum_j w_{ij}^{l-1} c_j^{l-1} + b_i^l N_s)}\right), \tag{12}$$

where $\rho(\cdot)$ denotes the ReLU non-linearity. The equivalent output spike count can be further determined as

$$c_i^l = \frac{N_s}{\Delta_i^l} = \frac{1}{\vartheta^l}\ \rho\left(\sum_j w_{ij}^{l-1} c_j^{l-1} + b_i^l N_s\right). \tag{13}$$

In practice, to reuse the original ANN layer for the fine-tuning purpose, we absorb the scaling factor $1/\vartheta^l$ into the learning rate. This configuration allows spike-train level error gradients to be effectively approximated from the ANN layer. It was shown that the ANN-SNN tandem learning method works more efficiently for rate-coded networks than other spike-based learning methods that update the weights for each time step [24].

In this paper, the tandem learning rule allows the spiking synaptic filters to be fine-tuned after the primitive conversion, which offers a good initialization for discrete neural representation. Along with the weights fine-tuning of subsequent ANN layers, the conversion errors can be effectively mitigated. Different from the end-to-end tandem learning framework introduced in [24], the tandem learning here is performed one layer at a time to prevent the gradient approximation error from accumulating across layers. The weights of the SNN layer are frozen after each training stage.

4.2 Scheduling of Progressive Tandem Learning

The PTL framework requires a schedule to be determined for each training stage. Inspired from [36], we propose an adaptive training scheduler to automate the PTL process. As shown in Fig. 2D, at the end of each training epoch we update the patience counter t based on the current validation loss and the best validation loss at the current training stage. The patience counter is reset to zero when the current validation loss improves, otherwise, the patience counter is increased by one. The patience counter serves a similar purpose to the patience parameter of an ANN learning rate scheduler. During ANN training, the patience parameter determines when the learning rate decay should happen. While the patience counter determines when the next layer should be converted. Once the patience counter reaches the pre-defined patience period T_p, the hybrid network parameters with the best validation loss are re-loaded to the network (i.e., the best model at

the current training stage) before the weights of the trained SNN layer are frozen. The training process terminates after the last ANN layer is replaced by the SNN layer. The pseudo codes of the proposed layer-wise ANN-to-SNN conversion framework are presented in Algorithm 1.

Algorithm 1. Pseudo Codes of the Progressive Tandem Learning Framework

Input: input sample X_{in}, target label Y, pre-trained ANN net_a, encoding time window size N_s, patience period T_p, number of network layers L
Output: converted SNN
// Network Initialization
$net = net_a$
for *layer* $l = 1$ *to* L **do**
  // Initialize the Training Scheduler
  $t = 1$
  $loss\_best = \infty$
  // Determine the Firing Threshold of Layer l
  $\vartheta^l = Threshold\_LayerNorm(net, N_s, X_{in})$
  // Build Hybrid Network for Training Stage l
  $net = Build\_Hybrid\_Network(net, \vartheta^l, l)$
  while $t < Tp$ **do**
    // Layer-wise Training for 1 Epoch on the Hybrid Network
    $[net, val\_loss] = Layer\_Wise\_Training(net, N_s, X_{in}, Y)$
    // Update the Training Scheduler
    $[t, loss\_best] = Update\_Training\_Scheduler(val\_loss, loss\_best)$
  // Freeze the Weights of SNN Layer l
  $net = Freeze\_Layer(net, l)$

4.3 Optimizing for Other Hardware Constraints

The PTL framework also allows other hardware constraints, such as the limited conductance states of non-volatile memory devices and limited fan-in connections in the neuromorphic architecture, to be incorporated easily during training. It hence greatly facilitates hardware-algorithm co-design and allows optimal performance to be achieved when deploying the trained SNN models onto the actual neuromorphic hardware.

To elucidate on this prospect, we focus on the constraint of limited conductance states that will lead to the limited weight precision for SNNs. Specifically, we explored the quantization-aware training [35] method whereby the low-precision weights are imposed progressively during training. As illustrated in Fig. 3, following the similar procedures that have been described for activation quantization in Eq. (8), the network weights and bias terms are quantized to a desirable precision before sharing to the SNN layer. While their full-precision copies are kept in the ANN layer to continue the learning with high precision. The flexibility provided by the PTL framework allows the SNN model to progressively navigate to a suitable parameter space to accommodate various hardware constraints.

5 Experiments on Pattern Classification

In this section, we first investigate the scalability of spike-based learning methods, which motivates the proposal of a layer-wise learning method in fine-tuning the converted

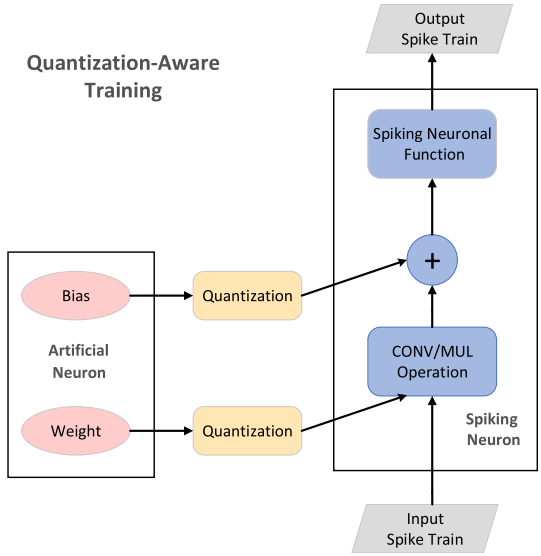


Fig. 3. Illustration of the quantization-aware training that can be incorporated into the proposed PTL framework. The full precision weight and bias terms of ANN neurons are quantized to the desired precision before sharing with the coupled spiking neurons.

SNN. Second, we demonstrate the learning effectiveness and scalability of the proposed PTL framework on large-scale object recognition tasks. Third, we investigate the effectiveness of the algorithm-hardware co-design methodology, that incorporates hardware constraints into the conversion process, with an example on the quantization-aware training for low precision neuromorphic hardware. Finally, we study the training efficiency of the proposed conversion framework as well as the improvements in the inference speed and energy efficiency of the trained SNN models.

5.1 Experimental Setup

We perform all experiments with PyTorch library that supports accelerated and memory-efficient training on multi-GPU machines. Under a discrete-time simulation, we implement the customized linear layer and convolution layer in Pytorch using IF neurons. We use the Adam optimizer [38] for all the experiments. To improve the training efficiency, we add batch normalization (BN) layer [39] after each convolution and linear layer. Following the approach introduced in [27], we integrate the parameters of BN layers into their preceding convolution or linear layers' weights before sharing them with the coupled SNN layers. We use this setup consistently for both the pattern classification tasks of this section and the signal reconstruction tasks that will be presented in the next section unless otherwise stated.

Dataset. We perform the object recognition experiments on the MNIST [40], Cifar-10 [41] and ImageNet-12 datasets [42], which are widely used in machine learning and neuromorphic computing communities to benchmark different learning algorithms. The MNIST handwritten digits dataset consists of grayscaled digits of 28×28 pixels that split into 60,000 training and 10,000 testing samples. The Cifar-10 dataset consists of 60,000 color images of size 32×32×3 from 10 classes, with a standard split of 50,000 and 10,000 for train and test, respectively. The large-scale ImageNet-12 dataset consists of over 1.2 million high-resolution images from 1,000 object categories. For Cifar-10 and MNIST datasets, we

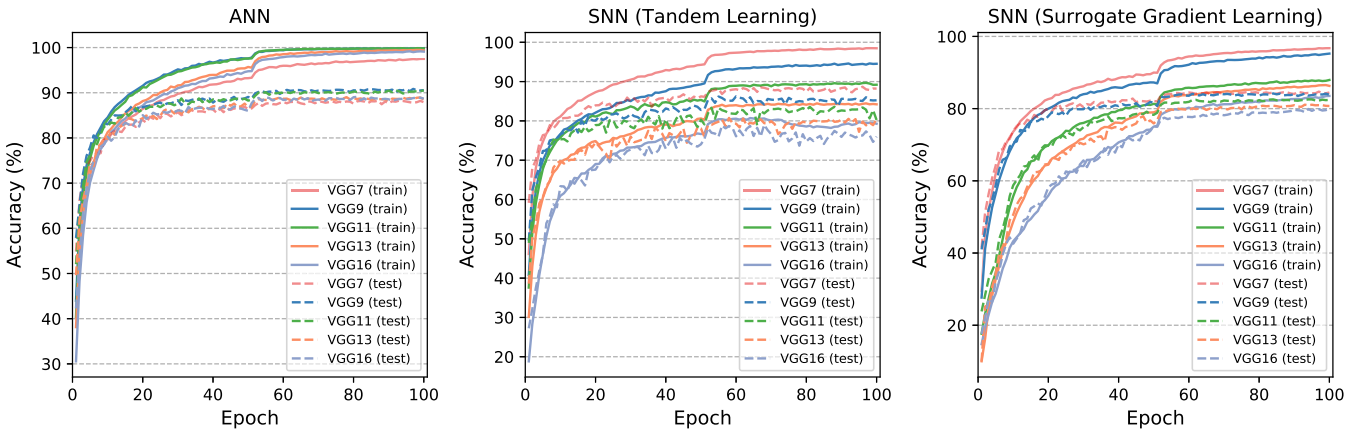


Fig. 4. Illustration of learning curves on the Cifar-10 dataset. (A) ANN models. (B) SNN models trained with spike count-based tandem learning [24]. (C) SNN models trained with time-based surrogate gradient learning [17]. It is worth noting that the jump of learning curves at Epoch 50 is due to the learning rate decay.

randomly split the original train set into train and validation sets with a split ratio of 9:1, which are fixed afterward for all the experiments. For ImageNet-12 dataset, the standard data split is followed for all experiments.

Network, Implementation and Evaluation Metric. Two classical CNN architectures are explored on the Cifar-10 dataset: AlexNet [3] and VGG-11 [43]. For the ImageNet-12 dataset, we performed experiments with AlexNet and VGG-16 [43] architectures to facilitate comparison with other existing ANN-to-SNN conversion works.

We also performed experiments with quantization-aware training of different weight precisions on the MNIST and Cifar-10 datasets. For MNIST dataset, the convolutional neural network with the structure of 28×28-c16s1-c32s2-c32s1-c64s2-800-10 is used, wherein the numbers after 'c' and 's' refer to the number of convolution filters and the stride of each convolution layer, respectively. The kernel size of 3 is used consistently for all convolution layers. For Cifar-10 dataset, we used AlexNet architecture.

For all experiments, the networks are trained for 100 epochs using the cross-entropy loss function. The patience period T_p is fine-tuned by progressively increasing it to match the number of available training epochs. The learning rate is initialized at 10^{-3} and decayed to 10^{-4} at Epoch 50. The 99th percentile of all a_i^l in a randomly selected training batch is used to determine the firing threshold. The best test accuracy across 5 independent runs is reported for the Cifar-10 dataset. While only a single run is performed for the ImageNet-12 dataset.

To evaluate the energy efficiency of the converted SNN models to their ANN counterparts, we follow the convention of neuromorphic computing community by counting the total synaptic operations [27]. For SNN, as defined below, the total synaptic operations (SynOps, AC operations) correlate with the neurons' firing rate, fan-out f_{out} (number of outgoing connections to the subsequent layer), and time window size N_s.

$$SynOps = \sum_{t=1}^{N_s} \sum_{l=1}^{L-1} \sum_{j=1}^{Q^l} f_{out,j}^l s_j^l[t], \tag{14}$$

where L is the total number of layers and Q^l denotes the total number of neurons in layer l.

In contrast, the total synaptic operations (MAC operations) that are required to classify one image in the ANN is given as follows

$$SynOps = \sum_{l=1}^{L} f_{in}^l Q^l, \tag{15}$$

where f_{in}^l denotes the number of incoming connections to each neuron in layer l.

5.2 End-to-End Spike-Based Learning Leads to Accumulated Gradient Approximation Errors

As discussed in Section 4, to compensate for the errors arising from the primitive ANN-to-SNN conversion, a training method is required to fine-tune the network weights. Here, we take the object recognition task on the Cifar-10 dataset as an example to study the scalability of spike-based learning methods in training deep SNNs to perform rapid pattern recognition. Specifically, we implemented the surrogate gradient learning method and tandem learning method proposed in [16] and [24], respectively. The network structures employed in this study are taken from the famous VGGNet [43].

With an encoding time window N_s of 8, the learning curves for ANN and SNN models with different network depths are presented in Fig. 4. As shown in Fig. 4A, the training converges easily for all ANN models, despite slight overfitting observed for the VGG13 and VGG16 models. In contrast, the training convergence is difficult for the spiking counterparts that have a network depth of over 10 layers as shown in Figs. 4B and 4C. This observation suggests the gradient approximation error tends to accumulate over layers with the spike-based learning methods and significantly degrades the learning performance for deep SNNs over 10 layers. Therefore, these end-to-end learning methods would not work well for the fine-tuning task required after the primitive network conversion. In the following sections, we will show that the proposed PTL framework that performs fine-tuning one layer at a time can effectively overcome the accumulated gradient approximation errors and scale up freely to deep SNNs with 16 layers.

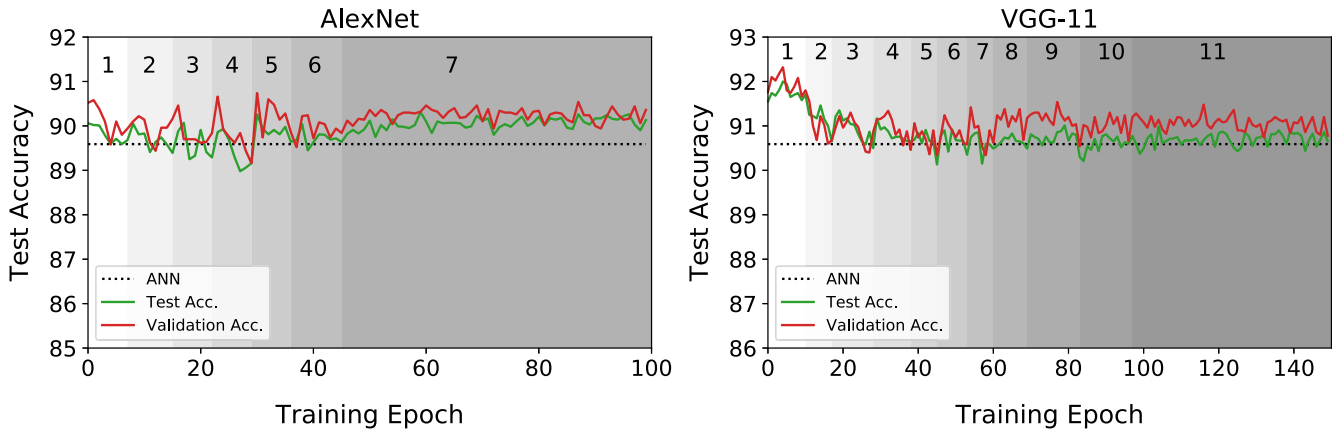


Fig. 5. Illustration of the training progresses of the AlexNet and VGG-11 on the Cifar-10 dataset ($N_s = 16$, $T_p = 6$). The shaded regions correspond to different training stages. After replacing each ANN layer with an equivalent SNN layer at the beginning of each training stage, the validation and test accuracies can be quickly restored with the proposed PTL framework. In these experiments, to allow searching for a better SNN model, the early termination did not apply during the last conversion stage.

5.3 Object Recognition on Cifar-10 and ImageNet-12

As shown in Fig. 5, we plot the training progress of the AlexNet and VGG-11 models on the Cifar-10 dataset, to illustrate the effectiveness of the proposed PTL framework. As expected, the validation accuracy drops mostly at the beginning of each conversion stage due to the conversion errors introduced. Notably, these errors are counteracted by the proposed layer-wise learning method, whereby the test and validation accuracies are restored quickly with only a few training epochs. Overall, the validation and test accuracies remain relatively stable during the whole training progress and can even surpass those of the pre-trained ANNs after training. It suggests that the proposed conversion framework can significantly reduce the representation space N_s by exploiting the redundancies that existed in the high-dimensional feature representation of the ANN.

As reported in Table 1, the trained deep SNNs achieve state-of-the-art classification accuracies over other existing SNN implementations with similar network architecture, with a test accuracy of 90.86% and 91.24% for AlexNet and VGG-11 respectively on the Cifar-10 dataset. It is worth mentioning that these SNN models even outperform their pre-trained ANN baselines by 1.27% and 0.65%. In comparison with a recently introduced binary neural network training method for neuromorphic implementation [36], which achieved a classification accuracy of 84.67%, the results suggest that the larger encoding time window $N_s = 16$ contributes to the higher accuracy.

To study the scalability of the proposed PTL framework on more complex datasets and network architectures, we conduct experiments on the challenging ImageNet-12 dataset. Due to the high computational complexity of modeling deep SNNs and the huge memory demand to store their intermediate states, only a limited number of ANN-to-SNN conversion methods have achieved some promising results on this dataset.

As reported in Table 1, the spiking AlexNet and VGG-16 models trained with the proposed PTL framework achieve promising results on the ImageNet-12 dataset. For the spiking AlexNet, the top-1 (top-5) accuracy improves by 3.39% (2.21%) over the early work that takes a constrain-then-train approach [44]. Meanwhile, the total number of time steps required is reduced by more than one order from 200 to 16. For the spiking VGG-16, despite the total number of time steps reduced by at least 25 times, our result is as competitive as those achieved with the state-of-the-art ANN-to-SNN conversion approaches [27], [28].

Nitin *et al.* [45] recently apply a spike-based learning method to fine-tune the weights of the converted SNN end-to-end, so as to speed up the model at run time. This method successfully reduces the total time steps from 2,500 to 250, with accuracy drops by about 3% on the ImageNet-12 dataset. In contrast, the discrete neural representation proposed in this work provides an improved network initialization that allows for a more radical reduction in the encoding time window. Notably, the classification accuracy of our system is on par with theirs, while requiring only a total of 16-time steps. Although our SNN models drop from the pre-trained AlexNet and VGG-16 models by about 3% and 6% respectively, it is much better than that obtained from the ANN-to-SNN conversion which is reported to have an accuracy drop of 16.6% [46]. Moreover, it is expected that our accuracy drop could be closed by providing a larger representation space N_s.

5.4 Quantization-Aware Training for Low Precision Neuromorphic Hardware

Table 2 provides the object recognition results with the quantization-aware training. On the MNIST and Cifar-10 datasets, the low-precision SNN models perform exceedingly well regardless of the reduced bit-width and the limited representation space (i.e., $N_s = 16$). Specifically, when the weights are quantized to 4-bit, the classification accuracy drops by only 0.03% and 0.85% on the MNIST and Cifar-10 datasets, respectively. Therefore, the proposed PTL framework offers immense opportunities for implementing SNNs on the low-precision neuromorphic hardware, for instance with emerging non-volatile memory devices that suffering from limited conductance states.

5.5 Rapid and Efficient Classification With SNNs

When implemented on the neuromorphic chips, the SNNs have great potential to improve the real-time performance and energy efficiency over ANNs. However, the learning

TABLE 1
Comparison of Classification Accuracy of Different SNN Implementations on the Cifar-10 and ImageNet-12 Test Sets

	Model	Network	Method	Accuracy (%)	Time Steps
Cifar-10	Wu *et al.* (2019) [17]	AlexNet (SNN)	Surrogate Gradient Learning	85.24	-
	Hunsberger and Eliasmith (2016)[44]	AlexNet (SNN)	Constrain-then-Train	83.54	200
	This work	**AlexNet (ANN)**	**Error Back-propagation**	**89.59**	**16**
	This work	**AlexNet (SNN)**	**Progressive Tandem Learning**	**90.86**	**16**
	Rueckauer *et al.* (2017)[27]	VGG-like (SNN)	ANN-to-SNN conversion	88.82	-
	Severa, William, *et al.* (2019)[36]	VGG-like (SNN)	Binary Neural Network	84.67	1
	Nitin *et al.* (2020)[45]	VGG-16 (SNN)	ANN-to-SNN conversion	90.20	100
	Nitin *et al.* (2020)[45]	VGG-16 (SNN)	ANN-to-SNN conversion + STDB	91.13	100
	This work	**VGG-11 (ANN)**	**Error Back-propagation**	**90.59**	**16**
	This work	**VGG-11 (SNN)**	**Progressive Tandem Learning**	**91.24**	**16**
ImageNet	Hunsberger and Eliasmith (2016)[44]	AlexNet (SNN)	Constrain-then-Train	51.80 (76.20)	200
	Wu *et al.* (2019)[24]	AlexNet (SNN)	Tandem Learning	50.22 (73.60)	10
	This work	**AlexNet (ANN)**	**Error Back-propagation**	**58.53 (81.07)**	**16**
	This work	**AlexNet (SNN)**	**Progressive Tandem Learning**	**55.19 (78.41)**	**16**
	Rueckauer *et al.* (2017)[27]	VGG-16 (SNN)	ANN-to-SNN conversion	49.61 (81.63)	400
	Sengupta *et al.* (2019)[28]	VGG-16 (SNN)	ANN-to-SNN conversion	69.96 (89.01)	2500
	Nitin *et al.* (2020)[45]	VGG-16 (SNN)	ANN-to-SNN conversion	68.12 (-)	2500
	Nitin *et al.* (2020)[45]	VGG-16 (SNN)	ANN-to-SNN conversion + STDB	65.19 (-)	250
	Deng and Gu (2021) [46]	VGG-16 (SNN)	ANN-to-SNN conversion	55.80 (-)	16
	This work	**VGG-16 (ANN)**	**Error Back-propagation**	**71.65 (90.37)**	**16**
	This work	**VGG-16 (SNN)**	**Progressive Tandem Learning**	**65.08 (85.25)**	**16**

The numbers inside and outside the round bracket of the 'Accuracy' column refer to the top-1 and top-5 accuracy, respectively.

methods grounded on the firing rate assumption require long inference time, typically a few hundred to thousands of time steps, to reach a stable network firing state. They diminish the latency advantages that can be obtained from the asynchronous operation of SNNs. In contrast, the proposed conversion framework allows making efficient use of the available time steps, such that rapid inference can be performed with only 16 time steps on the ImageNet-12 dataset. As shown in Fig. 6A, we notice a clear positive correlation between the encoding time window size and the classification accuracy on the Cifar-10 dataset. Notably, a reliable prediction can still be made with only a single time step when SNN is trained to utilize this limited amount of information as in the scenario of binary neural networks, while the performance can be further improved when larger encoding time windows are provided.

TABLE 2
Comparison of the Classification Results as a Function of Weight Precision

Benchmark	Bit Width	Acc. (%)	Change of Acc. (%)
MNIST	Float32	99.32	0
	8-bit	99.32	0
	7-bit	99.30	-0.02
	6-bit	99.29	-0.03
	5-bit	99.30	-0.02
	4-bit	99.29	-0.03
Cifar-10	Float32	90.33	0
	8-bit	90.11	-0.22
	7-bit	90.06	-0.27
	6-bit	90.07	-0.26
	5-bit	90.04	-0.29
	4-bit	89.48	-0.85

The result of SNN models is obtained through quantization-aware training. The average results across 5 independent runs are reported.

To further study the energy efficiency of trained SNN models, we follow the convention by counting the synaptic operations per inference and calculating the ratio to the corresponding ANN models [24], [27]. In general, the total synaptic operations required by the ANN is a constant number depending on the network architecture, while it positively correlates with the encoding time window and the firing rate for SNNs. As shown in Fig. 6B, under the iso-accuracy setting, when the ANN and SNN models achieve an equal accuracy, the SNN ($N_s = 8$) consumes only around 0.315 times total synaptic operations over the ANN counterpart. In contrast, the state-of-the-art SNN implementations with the ANN-to-SNN conversion and spike-based learning methods have reported a SynOps ratio of 25.60 and 3.61 respectively on a similar VGGNet-9 network [47]. It suggests our SNN implementation is 81.27 and 11.46 times more efficient at run-time respectively.

It is worth noting that SNNs perform mostly accumulate (AC) operations to integrate the membrane potential contributions from incoming spikes. In contrast, multiply-accumulate (MAC) operations are used in ANN which is significantly more expensive in terms of energy consumption and chip area usage. For instance, the simulations in a Global Foundry 28 nm process report the MAC operation is 14x costly than the AC operation and requires 21x chip area [27]. Therefore, over 40 times cost savings can be received from our SNN models by taking the sparse and cheap AC operations over the ANN counterparts, and the cost savings can be further boosted from efficient neuromorphic chip architecture design and emerging ultra-low-power devices implementation. It is worth mentioning that low precision networks supported by the quantization-aware training strategy proposed in Section 4.3 can further reduce the computing cost and memory footprint.

Figs. 6C and 6D present the classification results and the required training epochs as a function of the patience period in the adaptive training scheduler. As shown in Fig. 6C, a

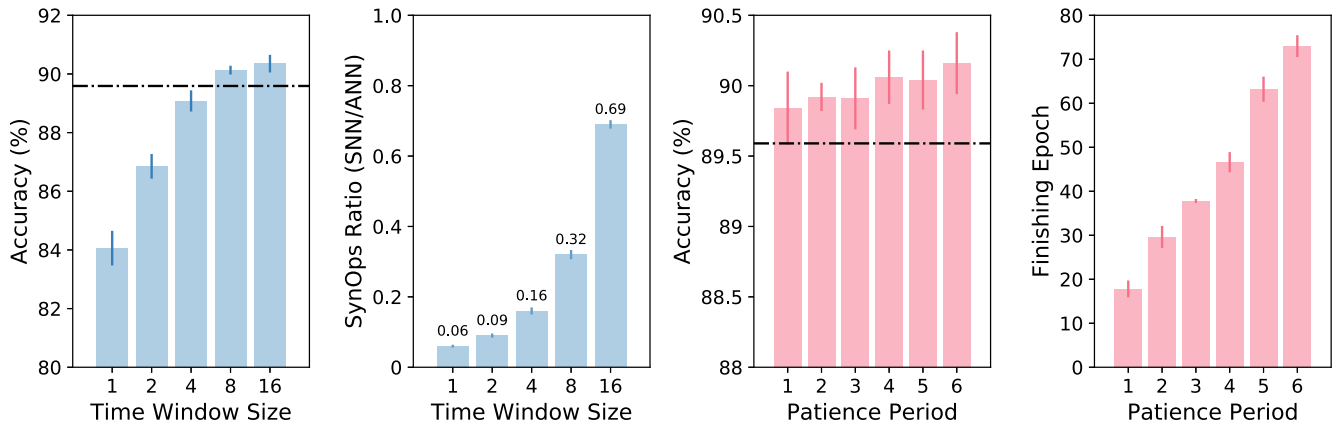


Fig. 6. (A) Classification accuracy as a function of the encoding time window on the Cifar-10 dataset. The horizontal dashed line refers to the accuracy of the pre-trained ANN. (B) The ratio of total synaptic operations between SNN and ANN as a function of encoding time window on the Cifar-10 dataset. (C) Classification accuracy as a function of the patience period defined in the adaptive scheduler. (D) Finishing epoch as a function of the patience period. All experimental results are summarized over 5 independent runs with spiking AlexNet. The error bars represent one standard deviation across the 5 runs.

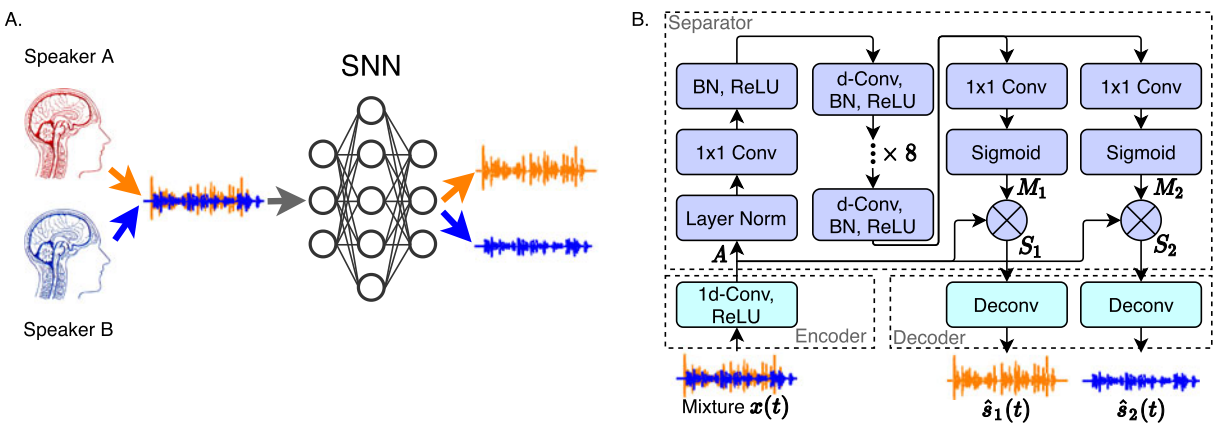


Fig. 7. (A) Illustration of the SNN-based speech separation approach to solving the cocktail party problem. (B) Illustration of the proposed SNN-based speech separation network. It takes two speakers mixture as input and outputs two independent streams for each individual speaker. "1d-Conv" indicates a 1-dimensional convolution. "1×1 Conv" is a convolution with a 1×1 kernel. "d-Conv" is a dilated convolution. "Deconv" is a deconvolution (also known as transposed convolution). "ReLU" is a rectified linear unit function. "BN" represents batch normalization. ⊗ refers to the element-wise multiplication.

competitive classification accuracy that surpasses the pre-trained ANN model can be achieved even with a patience period of only 1, which requires an average epoch of only 18 as shown in Fig. 6D. The accuracy can be further improved if a longer patience period is given.

6 Experiments on Signal Reconstruction

In Section 5, we demonstrate superior learning capability and scalability of the proposed PTL framework on pattern classification tasks. The existing ANN-to-SNN conversion works mainly focus on the pattern classification tasks, where a high-precision output is not required. The regression tasks like signal reconstruction however require the SNN model to predict high precision outputs using spikes, which have not been well explored. In this section, we further apply SNNs to solve pattern regression tasks that are known to be challenging for SNNs. Specifically, we perform experiments on the image reconstruction and speech separation tasks, both of which require reconstructing high-fidelity signals.

6.1 Image Reconstruction With Autoencoder

An autoencoder is a type of neural network that learns to decompose input signals into a compact latent representation, and then use that representation to reconstruct the original signals as closely as possible [48]. Typically an autoencoder learns compact latent representations through a bottleneck layer that has a reduced dimensionality over the input. In this way, it ignores the variation, removes the noise, and disentangles a mixture of information. Here, we investigate the compact latent representation extraction and reconstruction for static images using spike counts.

6.2 Time-Domain Speech Separation

Speech separation is one of the solutions for the cocktail party problem, where one is expected to selectively listen to a particular speaker in a multi-talker scenario [49]. Physiological studies reveal that selective auditory attention takes place both locally by transforming the receptive field properties of individual neurons and globally throughout the auditory cortex by rapid neural adaptation, or plasticity, of the cortical circuits [50], [51]. However, machines have yet to achieve the same attention ability as humans in segregating mixed stimuli into different streams. Such auditory attention capability is highly demanded in real-world applications, such as, hearing aids [52], speech recognition [53], speaker verification [54], and speaker diarization [55].

Inspired by the recent progress in deep ANN approaches to time-domain speech separation and extraction [56], [57], we propose and implement a deep SNN-based solution for speech separation. As shown in Fig. 7, the SNN takes the mixture speech as input and generates individual speech into separate streams. With a stack of dilated convolutional layers, the SNN captures the long-range dependency of speech signals with a manageable number of parameters. It is optimized to maximize a scale-invariant signal-to-distortion ratio (SI-SDR) [58] loss for high fidelity speech reconstruction.

The proposed SNN-based speech separation framework consists of three components: an encoder, a separator, and a decoder, as shown in Fig. 7. The encoder transforms the time-domain mixture signal into a high-dimensional representation, which is then taken as the input to the separator. The separator estimates a mask for each speaker at each time step. After that, a suitable representation for every individual speaker is extracted by filtering the encoded representation of the input mixture with the estimated mask for that speaker. Finally, the time-domain signal of each speaker is reconstructed using a decoder.

6.3 Experimental Setup

In the following, we will present the experiments designed for image reconstruction and speech separation tasks. By applying the PTL framework, the pre-trained ANNs are converted into SNNs for high-fidelity signal reconstruction in these tasks.

6.3.1 Image Reconstruction

6.3.1.1 Dataset. The MNIST dataset [40] is used for the image reconstruction task, which consists of 60,000 training and 10,000 test samples. These samples are directly used for training and testing without applying any data pre-processing steps.

6.3.1.2 Network, Implementation and Evaluation Metric. We evaluate a fully-connected autoencoder that has an architecture of 784-128-64-32-64-128-784, wherein the numbers refer to the number of neurons at each layer [36]. The sigmoid activation function is used in the output layer to normalize the output so as to match to the input range, while the rest of the layers use a ReLU activation function. Following the neural coding scheme introduced in Section 3.4, instead of using the spike count, the free aggregate membrane potential of spiking neurons in the final SNN layer is considered as the pre-activation quantity to the sigmoid activation function, which provides a high-resolution reconstruction. The networks are trained for 100 epochs using the mean square error (MSE) loss function, and the patience period T_p of the training scheduler is set to 6. We report the peak signal-to-noise ratio (PSNR) and Structural Similarity (SSIM) of reconstructed images on the MNIST test set with different encoding time window size. The rest of the training configurations follow those used in pattern classification tasks as presented in Section 5.1.

6.3.2 Time-Domain Speech Separation

6.3.2.1 Dataset. We evaluated the methods on the two-talker mixed WSJ0-2mix dataset[2] [59] with a sampling rate of 8 kHz, which was mixed by randomly choosing utterances of two speakers from the WSJ0 corpus [60]. The WSJ0-2mix corpus consists of three sets: training set (20,000 utterances $\approx 30h$), development set (5,000 utterances $\approx 8h$), and test set (3,000 utterances $\approx 5h$). Specifically, the utterances from 50 male and 51 female speakers in the WSJ0 training set (si_tr_s) were randomly selected to generate the training and development set in WSJ0-2mix at various signal-to-noise (SNR) ratios that uniformly chosen between 0dB and 5dB. Similarly, the test set was created by randomly mixing the utterances from 10 male and 8 female speakers in the WSJ0 development set (si_dt_05) and evaluation set (si_et_05). The test set was considered as the open condition evaluation because the speakers in the test set were different from those in the training and development sets. We used the development set to tune parameters and considered it as the closed condition evaluation because the speakers are seen during training. The utterances in the training and development set were broken into 4s segments.

6.3.2.2 Network and Implementation. Inspired by the Conv-TasNet speech separation system [56], the proposed SNN-based speech separation system first encodes the mixture input $x(t) \in \mathbb{R}^{1\times T}$ by a 1d-convolution with $N(=512)$ filters followed by the ReLU activation function. Each filter has a window of $L(=20)$ samples with a stride of $L/2(=10)$ samples. In the separator part, a mean and variance normalization with trainable gain and bias parameters is applied to the encoded representations $A \in \mathbb{R}^{K\times N}$ on the channel dimension, where K is equal to $2(T-L)/L+1$. A 1×1 convolution together with batch normalization and ReLU activation is applied to the normalized encoded representations. The dilated convolutions with 512 filters are repeated 10 times with dilations ratios of $[2^0, 2^1, \ldots, 2^9]$. These dilated convolution filters have a kernel size of 1×3 and a stride of 1. The batch normalization and ReLU activation function are also applied to the dilated convolutions layers. A mask (M_1, M_2) for each speaker is then estimated by a 1×1 convolution with a sigmoid activation function. The modulated representation (S_1, S_2) for each speaker is obtained by filtering the encoded representation A with the estimated mask (M_1, M_2). Finally, the time-domain signal (s_1, s_2) for each speaker is reconstructed by the decoder, which acts as the inverse process of the encoder.

The ANN-based system is optimized with the learning rate started from 0.001 and is halved when the loss increased on the development set for at least 3 epochs. Then, we take the pre-trained ANN model and convert the separator into an SNN. It is worth mentioning that the aggregate membrane potential is applied as the inputs to the last 1×1 convolution layer where a float-point representation is required to generate high-resolution auditory masks. The encoding time window N_s and patience period T_p are set to 32 and 3 for SNNs, respectively. Both ANN and

2. Available at: http://www.merl.com/demos/deep-clustering. The database used in this work is simulated with the released script and configuration in [59].

TABLE 3
Comparison of the Image Reconstruction Results as a Function of the Encoding Time Window Size N_s

Model	N_s	PSNR	SSIM
ANN	-	21.24	0.84
SNN	32	20.76	0.84
	16	18.00	0.76
	8	16.63	0.67
	4	15.93	0.60
	2	15.26	0.53
	1	14.06	0.41

The average results across 5 independent runs are reported.

SNN models are trained for 100 epochs, and an early stopping scheme is applied when the loss does not improve on the development set for 10 epochs.

6.3.2.3 Training Objective and Evaluation Metric.

The speech separation system is optimized by maximizing the scale-invariant signal-to-distortion ratio (SI-SDR) [58], that is defined as

$$\text{SI-SDR} = 10\log_{10}\left(\frac{||\frac{\langle \hat{s}, s\rangle}{\langle s, s\rangle} s||^2}{||\frac{\langle \hat{s}, s\rangle}{\langle s, s\rangle} s - \hat{s}||^2}\right), \tag{16}$$

where $\hat{s}$ and s are separated and target clean signals, respectively. $\langle \cdot, \cdot \rangle$ denotes the inner product. To ensure scale invariance, the signals $\hat{s}$ and s are normalized to zero-mean prior to the SI-SDR calculation. Since we don't know which speaker the separated stream belongs to (permutation problem), we adopt permutation invariant training to find the best permutation by maximizing the SI-SDR performance among all the permutations. The SI-SDR is used as the evaluation metric to compare the performances of the original ANN-based and the converted SNN-based speech separation systems. We also evaluate the systems with Perceptual Evaluation of Speech Quality (PESQ) [61], [62], which is recommended as the ITU-T P.862 standard to automatically assess the speech quality instead of the subjective Mean Opinion Score (MOS). During the evaluation, the permutation problem between the separated streams and the corresponding target clean signals are decided following the permutation invariant training during the training phase.

Fig. 8. Illustration of the reconstructed images from spiking autoencoder ($N_s = 32$) on the MNIST dataset. For each pair of digits, the left side is the original image and the right side is the reconstruction by SNN.

TABLE 4
Comparative Study Between ANN and SNN on Speech Separation Tasks Under Both Closed and Open Condition

Cond.	Methods	SI-SDR (dB)			PESQ		
		Diff.	Same	Overall	Diff.	Same	Overall
Closed	ANN	15.2	11.7	13.5	3.12	2.83	2.97
	SNN	14.5	11.0	12.8	3.03	2.75	2.89
Open	ANN	14.9	10.4	12.8	3.11	2.74	2.94
	SNN	14.2	9.8	12.2	3.02	2.66	2.85

The closed condition is on the development set, where the speakers are seen during training. The open condition is on the test set, where the speakers are unseen during training. "Diff." refers to the different gender mixture. "Same" refers to the same gender mixture. "Overall" refers to the combination of both different and same gender mixtures.

6.4 Experimental Results

6.4.1 Image Reconstruction With Autoencoder

Table 3 provides the image reconstructions results. As expected, a clear positive correlation between the encoding time window size N_s and the image reconstruction quality has been observed. Notably, with an encoding time window of 32, the spiking autoencoder achieves a comparable performance to the pre-trained ANN, in terms of the PSNR and SSIM metrics. As also shown in Fig. 8, this spiking autoencoder ($N_s = 32$) can effectively reconstruct images with high quality. In contrast to the object recognition results shown in Fig. 6 A, the results on the image reconstruction suggest regression tasks may require a larger discrete representation space or encoding time window to match the performance of the pre-trained ANN.

6.4.2 Time-Domain Speech Separation

Table 4 summarizes the comparative study between the original ANN-based and the converted SNN-based speech separation systems. The ANN- and SNN-based systems achieve an SI-SDR of 12.8 dB and 12.2 dB under the open condition evaluation, respectively. In terms of the perceptual quality, we observe that the ANN and SNN have a very close PESQ score of 2.94 and 2.85, respectively. The open condition evaluation results suggest that the SNN can achieve comparable performance to the ANN in this challenging speech separation task, while the SNN can take additional benefits of rapid inference and energy efficiency at test time. The same conclusion could also be drawn for the closed condition evaluation.

By listening to the separated examples generated by both ANN and SNN, we observe that the separated examples by SNN are very similar to those generated by ANN with high-fidelity. We publish some examples from the testing set (open condition) online to demonstrate our system performance.[3] We randomly select a speech sample under the male-male mixture condition from the test set and show

3. More listening examples are available at https://xuchenglin28.github.io/files/iccbc2019/index.html

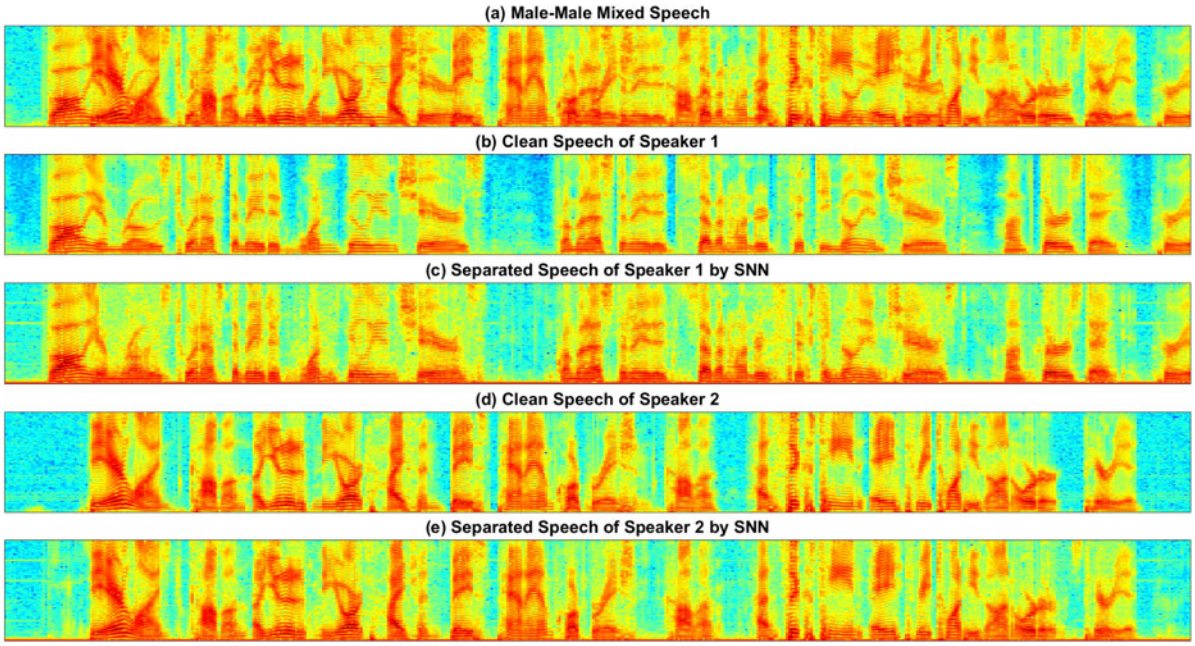


Fig. 9. The example of male-male mixture speech separated by SNN-based speech separation network.

their magnitude spectra in Fig. 9. We observe that the SNN obtains a similar spectrum as the ground truth clean spectrum even under the challenging condition of the same gender, where the multi-talkers have similar acoustic characteristics, i.e., pitch, hence less information is available to discriminate them from each other.

7 Conclusion

In this work, we reinvestigate the conventional ANN-to-SNN conversion approach and identify the accuracy and latency trade-off with the adopted firing rate assumption. Taking inspiration from the activation quantization works, we further propose a novel network conversion method, whereby spike count is utilized to represent the activation space of ANN neurons. This configuration allows better exploitation of the limited representation space and improves the inference speed. Furthermore, we introduce a layer-wise learning method to counteract the errors resulted from the primitive network conversion. The proposed conversion and learning framework, that is called *progressive tandem learning* (PTL), is highly automated with the proposed adaptive training scheduler, which supports flexible and efficient training. Benefiting from the proposed PTL framework, the algorithm-hardware co-design can also be effectively accomplished by imposing the hardware constraints progressively during training.

The SNNs thus trained have demonstrated competitive classification and regression capabilities on the challenging ImageNet-12 object recognition, image reconstruction, and speech separation tasks. Moreover, the proposed PTL framework allows making efficient use of the available encoding time window, such that rapid and efficient pattern recognition can be achieved with deep SNNs. Taking the quantization-aware training as an example, we illustrate how the hardware constraint, limited weight precision, can be effectively introduced during training, such that the optimal performance can be achieved on the actual neuromorphic hardware. By integrating the algorithmic power of deep SNNs and energy-efficient neuromorphic computing architecture, it opens up a myriad of opportunities for rapid and efficient inference on the pervasive low-power devices.

Acknowledgments

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the supporting institutions and companies.

References

[1] W. Xiong *et al.*, "Toward human parity in conversational speech recognition," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 25, no. 12, pp. 2410–2423, Dec. 2017.
[2] A. Van Den Oord *et al.*, "Wavenet: A generative model for raw audio," 2016, *arXiv:1609.03499*.
[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[5] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015.
[6] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[7] S. B. Laughlin and T. J. Sejnowski, "Communication in neuronal networks," *Science*, vol. 301, no. 5641, pp. 1870–1874, 2003.
[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2015, *arXiv:1510.00149*.
[9] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," 2016, *arXiv:1602.02830*.
[10] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[11] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities & challenges," *Front. Neurosci.*, vol. 12, 2018, Art. no. 774.

[12] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[13] J. Pei *et al.*, "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.

[14] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[15] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan./Feb. 2018.

[16] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 51–63, Nov. 2019.

[17] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1311–1318.

[18] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Front. Neurosci.*, vol. 12, 2018, Art. no. 331.

[19] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Proc. Neural Inf. Process. Syst.*, S. Bengio *et al.* Eds., 2018, pp. 1412–1421.

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

[21] P. Gu, R. Xiao, G. Pan, and H. Tang, "STCA: Spatio-temporal credit assignment with delayed feedback in deep spiking neural networks," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 1366–1372.

[22] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Proc. Neural Inf. Process. Syst.*, 2018, pp. 795–805.

[23] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertainty Fuzziness Knowl. Based Syst.*, vol. 6, no. 02, pp. 107–116, 1998.

[24] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, "A tandem learning rule for effective training and rapid inference of deep spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jul. 21, 2021, doi: 10.1109/TNNLS.2021.3095724.

[25] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *Int. J. Comput. Vis.*, vol. 113, no. 1, pp. 54–66, 2015.

[26] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proc. Int. Joint Conf. Neural Netw.*, 2015, pp. 1–8.

[27] B. Rueckauer, I. A. Lungu, Y. Hu, M. Pfeiffer, and S. C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Front. Neurosci.*, vol. 11, 2017, Art. no. 682.

[28] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Front. Neurosci.*, vol. 13, 2019, Art. no. 95.

[29] S. Kim, S. Park, B. Na, and S. Yoon, "Spiking-YOLO: Spiking neural network for real-time object detection," 2019, *arXiv:1903.06530*.

[30] Y. Hu, H. Tang, Y. Wang, and G. Pan, "Spiking deep residual network," 2018, *arXiv:1805.01352*.

[31] J. A. Pérez-Carrasco *et al.*, "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. Intell.*, vol. 35, no. 11, pp. 2706–2719, Nov. 2013.

[32] Y. Xu, H. Tang, J. Xing, and H. Li, "Spike trains encoding and threshold rescaling method for deep spiking neural networks," in *Proc. IEEE Symp. Ser. Comput. Intell.*, 2017, pp. 1–6.

[33] Y. Wang, Y. Xu, R. Yan, and H. Tang, "Deep spiking neural networks with binary weights for object recognition," *IEEE Trans. Cogn. Develop. Syst.*, 2020.

[34] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6306–6315.

[35] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2704–2713.

[36] W. Severa, C. M. Vineyard, R. Dellana, S. J. Verzi, and J. B. Aimone, "Training deep neural networks for binary communication with the whetstone method," *Nat. Mach. Intell.*, vol. 1, no. 2, pp. 86–94, 2019.

[37] A. G. Anderson and C. P. Berg, "The high-dimensional geometry of binary neural networks," 2017, *arXiv:1705.07199*.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[39] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*.

[40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[41] A. Krizhevsky and G. E. Hinton, "Learning multiple layers of features from tiny images," Comput., Sci. Dept., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., Apr. 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[42] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[44] E. Hunsberger and C. Eliasmith, "Training spiking deep networks for neuromorphic hardware," 2016, *arXiv:1611.05141*.

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

[46] S. Deng and S. Gu, "Optimal conversion of conventional artificial neural networks to spiking neural networks," in *Proc. Int. Conf. Learn. Representations*, 2021.

[47] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Front. Neurosci.*, vol. 14, 2020, Art. no. 119.

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[49] N. Mesgarani and E. Chang, "Selective cortical representation of attended speaker in multi-talker speech perception," *Nature*, vol. 485, pp. 233–236, 2012.

[50] T. Isomura, K. Kotani, and Y. Jimbo, "Cultured cortical neurons can perform blind source separation according to the free-energy principle," *PloS Comput. Biol.*, vol. 11, no. 12, 2015, Art. no. e1004643.

[51] E. Kaya and M. Elhilali, "Modelling auditory attention," *Philos. Trans. Roy. Soc. B Biol. Sci.*, vol. 372, 2017, Art. No. 20160101.

[52] D. Wang, "Deep learning reinvents the hearing aid," *IEEE Spectr.*, vol. 54, no. 3, pp. 32–37, Mar. 2017.

[53] J. Li, L. Deng, R. Haeb-Umbach, and Y. Gong, *Robust Automatic Speech Recognition: A Bridge to Practical Applications*. Amsterdam, The Netherlands: Academic Press, 2015.

[54] W. Rao, C. Xu, E. S. Chng, and H. Li, "Target speaker extraction for multi-talker speaker verification," in *Proc. Interspeech*, 2019, pp. 1273–1277.

[55] G. Sell *et al.*, "Diarization is hard: Some experiences and lessons learned for the JHU team in the inaugural DIHARD challenge," in *Proc. Interspeech*, 2018, pp. 2808–2812.

[56] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 27, no. 8, pp. 1256–1266, Aug. 2019.

[57] C. Xu, W. Rao, E. Chng, and H. Li, "SpEX: Multi-scale time domain speaker extraction network," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 1370–1384, 2020, doi: 10.1109/TASLP.2020.2987429.

[58] J. Le Roux , S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR–Half-baked or well done?," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2019, pp. 626–630.

[59] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2016, pp. 31–35.

[60] J. Garofolo, D. Graff, D. Paul, and D. Pallett, "Csr-i (wsj0) complete,": Linguistic Data Consortium, Philadelphia, PA, USA, 1993. [Online]. Available: https://doi.org/10.35111/ewkm-cg47

[61] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2001, pp. 749–752.

[62] Y. Hu and P. C. Loizou, "Evaluation of objective quality measures for speech enhancement," *IEEE Trans. Audio Speech Lang. Process.*, vol. 16, no. 1, pp. 229–238, Jan. 2008.

Jibin Wu received the BE and PhD degrees in electrical engineering from the National University of Singapore, Singapore, in 2016 and 2020, respectively. He is currently a research fellow with HLT Lab, Department of Electrical and Computer Engineering, National University of Singapore. His research interests include spiking neural network, neuromorphic computing, auditory modelling, and automatic speech recognition.

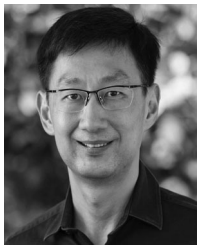
Chenglin Xu received the BEng and MSc degrees from Northwestern Polytechnical University, China, in 2012 and 2015, respectively, and the PhD degree from Nanyang Technological University, Singapore, in 2020. He is currently a research fellow with HLT lab, Department of Electrical and Computer Engineering, National University of Singapore. His research interests include speech extraction, speech enhancement, source separation, multi-talker speaker verification, and robust speech recognition.

Xiao Han received the BEng and MSc degrees from the National University of Singapore. Her research interests include neuromorphic computing and brain-inspired computing.

Daquan Zhou is currently working toward the PhD degree with the Institute of Data Science, National University of Singapore, under the supervision of professor Jiashi Feng. His research interests include deep learning, neural network compression, neural network structure design, and AutoML.

Malu Zhang received the PhD degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2019. He is currently a research fellow with HLT Lab, Department of Electrical and Computer Engineering, National University of Singapore. His research interests include spiking neural networks, neural spike encoding, and the neuromorphic applications of speech recognition and sound source localization. He is a guest associate editor for the *Frontiers in Neuroscience* (Neuromorphic Engineering) and a reviewer for several international journals, including the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Neural Networks and Learning Systems*, and *IEEE Transactions on Cybernetics*.

Haizhou Li (Fellow, IEEE) received the BSc, MSc, and PhD degrees in electrical and electronic engineering from the South China University of Technology, Guangzhou, China, in 1984, 1987, and 1990, respectively. He is currently a professor with the Department of Electrical and Computer Engineering, National University of Singapore and a presidential chair professor with the School of Data Science, The Chinese University of Hong Kong, Shenzhen. He was teaching with the University of Hong Kong from 1988 to 1990 and the South China University of Technology from 1990 to 1994. He was a visiting professor with CRIN, France, from 1994 to 1995, a research manager with Apple-ISS Research Centre from 1996 to 1998, the research director with Lernout and Hauspie Asia Pacific from 1999 to 2001, the vice president with InfoTalk Corp. Ltd. from 2001 to 2003, and the principal scientist and the department head of human language technology with the Institute for Infocomm Research, Singapore from 2003 to 2016. His research interests include automatic speech recognition, speaker and language recognition, natural language processing, and neuromorphic computing. He was the editor-in-chief of the *IEEE/ACM Transactions on Audio, Speech and Language Processing* from 2015 to 2018, has been a member of the editorial board of *Computer Speech and Language* since 2012, a member of IEEE Speech and Language Processing Technical Committee from 2013 to 2015, the president of the International Speech Communication Association from 2015 to 2017, the president of Asia Pacific Signal and Information Processing Association from 2015 to 2016, and the president of Asian Federation of Natural Language Processing from 2017 to 2018. He was the general chair of ACL 2012, INTERSPEECH 2014, and ASRU 2019. He was the recipient of the National Infocomm Award 2002 and the President's Technology Award 2013 in Singapore. He was named Nokia visiting professor in 2009 by the Nokia Foundation and U Bremen Excellence chair professor in 2019 by the University of Bremen, Germany. He is a fellow of the ISCA.



Kay Chen Tan (Fellow, IEEE) received the BEng degree (First Class Hons.) and the PhD degree from the University of Glasgow, U.K., in 1994 and 1997, respectively. He is currently a chair professor of computational intelligence with the Department of Computing, The Hong Kong Polytechnic University. He has authored or coauthored more than 300 refereed articles and seven books. He is currently the vice-president (Publications) of the IEEE Computational Intelligence Society, USA. He was the editor-in-chief of the *IEEE Computational Intelligence Magazine* from 2010 to 2013 and the *IEEE Transactions on Evolutionary Computation* from 2015 to 2020. He is currently the editorial board member for more than ten journals. He is also an IEEE Distinguished Lecturer Program (DLP) speaker and the chief co-editor of the Springer Book Series on *Machine Learning: Foundations, Methodologies, and Applications*.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.