Training Spiking Neural Networks with Local Tandem Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Spiking neural networks (SNNs) have demonstrated great biologically plausibility and energy efficiency over their predecessors. However, there is a lack of an efficient and generalized training method for deep SNNs, especially for deployment on analog computing substrates. In this paper, we put forward a generalized learning rule, termed Local Tandem Learning (LTL). The LTL rule follows the teacher-student learning approach by mimicking the intermediate feature representations of a pre-trained ANN. By decoupling the learning of network layers and leveraging highly informative supervisor signals, we demonstrate rapid network convergence within five training epochs on the CIFAR-10 dataset while having low computational complexity. Our experimental results have also shown that the SNNs thus trained can achieve comparable accuracies to their teacher ANNs on CIFAR-10, CIFAR-100, and Tiny ImageNet datasets. Moreover, the proposed LTL rule is hardware friendly. It can be easily implemented on-chip to perform fast parameter calibration and provide robustness against the notorious device non-ideality issues, including device mismatch, quantization noise, thermal noise, and neuron silencing. It, therefore, opens up a myriad of opportunities for training and deployment of SNN on ultra-low-power mixed-signal neuromorphic computing chips.

1 Introduction

Over the last decade, artificial neural networks (ANNs) have improved the perceptual and cognitive capabilities of machines by leaps and bounds, and have become the de-facto standard for many pattern recognition tasks including computer vision [26, 47, 48], speech processing [55, 34], language understanding [3], and robotics [46]. Despite their superior performance, ANNs are computationally expensive to be deployed on ubiquitous mobile and edge computing devices due to high memory and computation requirements.

Spiking neural networks (SNNs), the third generation artificial neural networks, have gained growing research attention due to their greater biologically plausibility and potential to realize ultra-low-power computation as observed in biological neural networks. Leveraging the sparse, spike-driven computation and fine-grain parallelism, the fully digital neuromorphic computing (NC) chips like TrueNorth [2], Loihi [10], and Tianjic [37], that support the efficient inference of SNNs, have indeed demonstrated orders of magnitude improved power efficiency over GPU-based AI solutions. Moreover, the emerging in situ mixed-signal NC chips [49, 42], enabled by nascent non-volatile technologies, can further boost the hardware efficiency by a large margin over the aforementioned digital chips.

Despite remarkable progress in neuromorphic hardware development, how to efficiently and effectively train the core computational model, spiking neural network, remains a challenging research topic. It, therefore, impedes the development of efficient neuromorphic training chips as well as

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

the wide adoption of neuromorphic solutions in mainstream AI applications. The existing training algorithms for deep SNNs can be grouped into two categories: ANN-to-SNN conversion and gradient-based direct training.

For ANN-to-SNN conversion methods, they propose to reuse network weights from more easily trainable ANNs. By properly determining the neuronal firing threshold and initial membrane potentials for SNNs, recent studies show that the activation values of ANNs can be well approximated with the firing rate of spiking neurons, achieving near-lossless network conversion on a number of challenging AI benchmarks [7, 44, 12, 18, 17, 41, 30, 11, 4, 5]. Nevertheless, these network conversion methods are developed solely based on the non-leaky integrate-and-fire (IF) neuron model and typically require a large time window so as to reach a reliable firing rate approximation. It is, therefore, not straightforward and efficient to deploy these converted SNNs onto the existing neuromorphic chips [2, 10, 37].

In another vein of research, the gradient-based direct training methods explicitly model each spiking neuron as a self-recurrent neural network and leverage the canonical Backpropagation Through Time (BPTT) algorithm to optimize the network parameters. The non-differentiable spiking activation function is typically circumvented with continuous surrogate gradient (SG) functions during error backpropagation [45, 53, 54, 33, 8, 13, 40]. Despite their compatibility with event-based inputs and different spiking neuron models, they are computational and memory inefficient to operate in practice. Moreover, the gradient approximation error introduced by these SG functions tends to accumulate over layers, causing significant performance degradation in the face of the deep network structure and short time window [52].

In general, SNN learning algorithms can be categorized into off-chip learning [14, 56] and on-chip learning [9, 36, 38]. Almost all of the direct SNN training methods discussed above belong to the off-chip learning category. Due to the lack of effective ways to exploit the high level of sparsity in spiking activities and the requirement to store non-local information for credit assignment, these off-chip methods exhibit very low training efficiency. Moreover, due to notorious device non-ideality problems [6], the actual network dynamics will deviate from the off-chip simulated ones, causing the accuracy of off-chip trained SNNs degrades significantly when deployed onto the analog computing substrates [39, 20, 32, 1]. To address these problems, recent work proposes on-chip learning algorithms in the form of local Hebbian learning [10, 36, 24] and approximation of gradient-based learning [28, 35], while the effectiveness of these algorithms had only been demonstrated on simple benchmarks, such as MNIST and N-MNIST datasets.

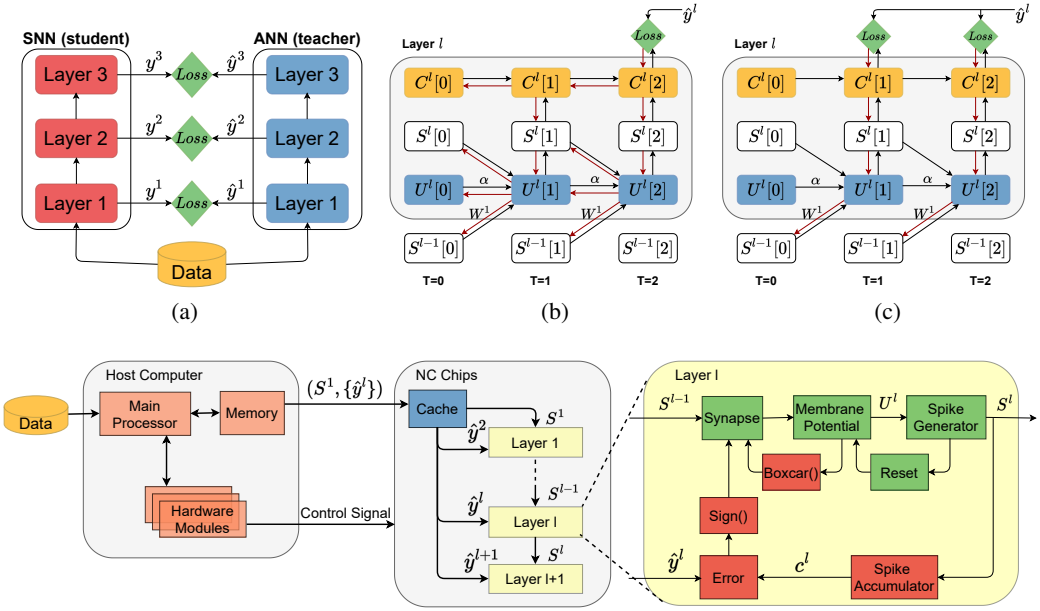
To address the aforementioned problems in SNNs training and hardware deployment, we put forward a generalized SNN learning rule in this paper, which we referred to as the Local Tandem Learning (LTL) rule. The LTL rule takes the best of both ANN-to-SNN conversion and gradient-based training methods. On one hand, it makes good use of highly effective intermediate feature representations of ANNs to supervise the training of SNNs. By doing so, we show that it can achieve rapid network convergence within five training epochs on the CIFAR-10 dataset with low computational complexity. On the other hand, the LTL rule adopts the gradient-based approach to perform knowledge transfer, which can support different neuron models and achieve rapid pattern recognition. By propagating gradient information locally within a layer, it can also alleviate the compounding gradient approximation errors of the SG method and lead to near lossless knowledge transfer on CIFAR-10, CIFAR-100, and Tiny ImageNet datasets. Moreover, the LTL rule is designed to be hardware friendly, which can perform efficient on-chip learning using only local information. Under this on-chip setting, we demonstrate that the LTL rule is capable of addressing the notorious device non-ideality issues of analog computing substrates, including device mismatch, quantization noise, thermal noise, and neuron silencing.

2 Methods

2.1 Spiking Neuron Model

To demonstrate the proposed LTL rule is compatible with different spiking neuron models, we base our study on both non-leaky integrate-and-fire (IF) [43] and leaky integrate-and-fire (LIF) neuron models [16], whose neuronal dynamics can be described by the following discrete-time formulation:

$$U_i^l[t] = \alpha U_i^l[t-1] + I_i^l[t] - \vartheta S_i^l[t-1] \tag{1}$$



(d)

Figure 1: Illustration of the proposed LTL rule and on-chip implementation. (a) The LTL rule follows the teacher-student learning approach, whereby the SNN tries to mimic the feature representation of a pre-trained ANN through local loss functions. (b) Computational graph of the offline LTL rule. (c) Computational graph of the online LTL rule. (d) Functional block diagram of the proposed on-chip implementation, where the host computer transfers the control signal and training data pairs (input spike train, hidden layer targets) from ANN to NC chips. The proposed SNN on-chip learning circuit consists of three parts: external input (blue), spiking neuron (green), and learning circuits (red).

with

$$I_i^l[t] = \sum_j w_{ij}^{l-1} S_j^{l-1}[t-1] + b_i^l \tag{2}$$

where $U_i^l[t]$ and $I_i^l[t]$ refer to the subthreshold membrane potential of and input current to neuron i at layer l, respectively. $\alpha \equiv exp(-dt/\tau_m)$ is the membrane potential decaying constant, wherein τ_m is the membrane time constant and dt is the simulation time step. For IF neuron, α takes a value of 1. ϑ denotes the neuronal firing threshold. w_{ij}^{l-1} represents the connection weight from neuron j of the preceding layer $l-1$ and b_i^l denotes the constant injecting current to neuron i. $S_i^l[t-1]$ indicates the occurrence of an output spike from neuron i at time step $t-1$, which is determined according to the spiking activation function as per

$$S_i^l[t] = \Theta\left(U_i^l[t] - \vartheta\right) \quad \text{with} \quad \Theta(x) = \begin{cases} 1, & \text{if} \quad x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

2.2 Local Tandem Learning

As illustrated in Figure 1(a), the LTL rule follows the Teacher-Student (T-S) learning approach, whereby the intermediate feature representation of a pre-trained ANN is transferred to SNN through layer-wise local loss functions. In contrast to the ANN-to-SNN conversion methods, we establish the feature representation equivalence at the neuron level rather than at the synapse level, which provides the flexibility for choosing any neuron model to be used in the SNN. On the other hand, with the proposed spatially local loss function, we simplify the spatial-temporal credit assignment required in

the end-to-end direct training methods, which can dramatically improve the network convergence speed and meanwhile reduce the computational complexity. In the following, we introduce two versions of the LTL rule, offline and online, depending on whether the temporal locality constraint is imposed.

Offline Learning Following the T-S learning approach, we consider the intermediate feature representation of a pre-trained ANN as the knowledge and train an SNN to reproduce an equivalent feature representation via layer-wise loss functions. In particular, we establish an equivalence between the normalized activation values of an ANN and the global average firing rates of an SNN. To reduce the discrepancy between these two quantities, we adopt the mean square error (MSE) loss function and apply it separately for each layer. Thus, for any layer l, the local loss function $\mathcal{L}^l$ is defined as follows

$$\mathcal{L}^l\left(\hat{y}^l, y^l[T_w]\right) = \left|\left|\frac{\hat{y}^l}{y_{norm}} - \frac{C^l[T_w]}{T_w}\right|\right|_2^2 \tag{4}$$

where $\hat{y}^l$ is the output of ANN layer l. y_{norm} is a normalization constant that takes the value of $99th$ or $99.9th$ percentile across all $\hat{y}_i^l$. This can alleviate the effect of outliers compared to use the maximum activation value [43]. T_w is the time window length. $C^l[T_w] = \Sigma_{t=1}^{T_w} S[t]$ is the total spike count.

As the computational graph shown in Figure 1(b), we adopt the canonical BPTT algorithm to resolve the temporal credit assignment problem, and the weight gradients can be derived as

$$\frac{\partial \mathcal{L}^l}{\partial w_{ij}^l} = \sum_{t=1}^{T_w} \frac{\partial \mathcal{L}^l}{\partial U_i^l[t]} \frac{\partial U_i^l[t]}{\partial w_{ij}^l} = \sum_{t=1}^{T_w} \frac{\partial \mathcal{L}^l}{\partial U_i^l[t]} \frac{\partial U_i^l[t]}{\partial I_i^l[t]} \frac{\partial I_i^l[t]}{\partial w_{ij}^l} = \sum_{t=1}^{T_w} \frac{\partial \mathcal{L}^l}{\partial U_i^l[t]} S_j^{l-1}[t-1] \tag{5}$$

with

$$\frac{\partial \mathcal{L}^l}{\partial U_i^l[t]} = \begin{cases} \alpha \delta_i^l[t+1] \frac{\partial S_i^l[t+1]}{\partial U_i^l[t+1]} + \delta_i^l[t] \frac{\partial S_i^l[t]}{\partial U_i^l[t]} & \text{if} \quad t < T_w \\ \delta_i^l[T_w] \frac{\partial S_i^l[T_w]}{\partial U_i^l[T_w]} & \text{if} \quad t = T_w \end{cases} \tag{6}$$

where

$$\delta_i^l[t] = \frac{\partial \mathcal{L}^l}{\partial S_i^l[t]} = \begin{cases} -\vartheta \delta_i^l[t+1] \frac{\partial S_i^l[t+1]}{\partial U_i^l[t+1]} + \delta_i^l[T_w] & \text{if} \quad t < T_w \\ -\frac{2}{T_w}\left(\frac{\hat{y}_i^l}{y_{norm}} - \frac{1}{T_w}\Sigma_{t=1}^{T_w} S_i^l[t]\right) & \text{if} \quad t = T_w \end{cases} \tag{7}$$

To resolve the problem of the non-differentiable spiking activation function, we use the surrogate gradient method to replace it with a continuous function and apply it during the gradient calculation, i.e., $\Theta'(x) \approx \theta'(x)$. Specifically, we adopt the boxcar function for $\theta'(x)$ that supports convenient and efficient on-chip implementation.

$$\frac{\partial S_i^l[t]}{\partial U_i^l[t]} = \theta'(U_i^l[t] - \vartheta) = \frac{1}{p} sign\left(|U_i^l[t] - \vartheta| < \frac{p}{2}\right) \tag{8}$$

where p controls the permissible range of membrane potentials that allow gradients to pass through, and we tune this hyperparameter separately for each dataset. By substituting Eq. (8) into Eqs. (6) and (7), we can yield the ultimate form of weight gradients, and we can update the weights according to the stochastic gradient descent method or its adaptive variants. See Supplementary Materials Section A.1 for a more detailed derivation of gradients to the weight and bias terms.

Online Learning The offline LTL rule requires storing intermediate synaptic and membrane state variables so as to be used during error backpropagation, which is prohibited for on-chip learning where memory resources are limited. To address this problem, we introduce an online LTL rule, whose loss function is designed to be both spatially and temporally local. To achieve the temporal locality, we use the moving average firing rate, which can be calculated at each time step, to replace the global firing rate used in Eq. (4). It hence yields the following local loss function

$$\mathcal{L}^l[t] = \left|\left|\frac{\hat{y}^l}{y_{norm}} - \frac{C^l[t]}{t}\right|\right|_2^2 \tag{9}$$

Compared to the offline version, the gradient update is much simpler now:

$$\frac{\partial \mathcal{L}^l[t]}{\partial w_{ij}^l} = \frac{\partial \mathcal{L}^l[t]}{\partial S_i^l[t]} \frac{\partial S_i^l[t]}{\partial U_i^l[t]} \frac{\partial U_i^l[t]}{\partial w_{ij}^l} = \zeta_i^l[t] \frac{\partial S_i^l[t]}{\partial U_i^l[t]} S_j^{l-1}[t-1] \tag{10}$$

where $\zeta^l[t]$ can be directly computed from Eq. (9):

$$\zeta_i^l[t] = \frac{\partial \mathcal{L}^l[t]}{\partial S_i^l[t]} = -\frac{2}{t}\left(\frac{\hat{y}_i^l}{y_{norm}} - \frac{1}{t}\Sigma_{k=1}^{t} S_i^l[k]\right) \tag{11}$$

The computational graph of the online LTL rule is provided in Figure 1(c). As will be discussed in Sections 3.1 and 3.4, this online version can significantly reduce the computational complexity, while achieving a comparable test accuracy to that of the offline version.

On-chip Implementation To allow a convenient and efficient on-chip implementation of the proposed online LTL rule, we carefully designed the learning circuits as illustrated in Figure 1(d). The output spike count C^l is updated at the spike accumulator, and it is compared to the local target $\hat{y}^l$ following the layer-wise loss function defined in Eq. 9. This error term is further feedback to the neuron to update the synaptic parameters. Note that the synaptic updates are gated by $sign(\cdot)$ and $boxcar(\cdot)$ functions, which can greatly reduce the frequency of parameter updates. To improve the time synchronization between ANN and SNN during training, we can add a warm-up period T_{warm} to wait for the $\hat{y}^l$ signal to become available, before when the parameter update is frozen for SNN.

3 Experiments

In this section, we evaluate the effectiveness of the proposed LTL rule on the image classification task with CIFAR-10 [25], CIFAR-100 [25], and Tiny ImageNet [50] datasets. We perform a comprehensive study to demonstrate its superiority in: 1. accurate, rapid, and efficient pattern recognition; 2. rapid network convergence with low computational complexity; 3. provide robustness against hardware-related noises. More details about the experimental datasets and implementation details are provided in the Supplementary Materials Section B, and the source code will be released soon.

3.1 Accurate and Scalable Image Classification

Here, we report the classification results of LTL trained SNNs on CIFAR-10, CIFAR-100 and Tiny ImageNet datasets against other SNN learning rules, including ANN-to-SNN conversion [52, 44, 41, 30, 18, 17, 11, 4, 5, 27, 15] and direct SNN training [40] methods. Given the network architectures and data preparation processes vary slightly across different work, therefore, we focus our discussions on the conversion or transfer errors between the ANNs and SNNs whenever the data is available.

On the CIFAR-10 dataset, as reported in Table 1, our LTL offline trained spiking VGG-11 has surpassed the accuracy of teacher ANN at $T_w = 16$, this result suggests high effectiveness of the proposed teacher-student learning approach in transferring the knowledge through local objectives. Moreover, the LTL rule can take advantage of a larger time window that offers a greater representation power, and further improves the performance of SNNs at $T_w = 32$. It is also worth mentioning that by transferring the knowledge in a learnable way, the proposed LTL rule is compatible with different neuron models, manifested through high accuracies achieved by both IF and LIF neuron models.

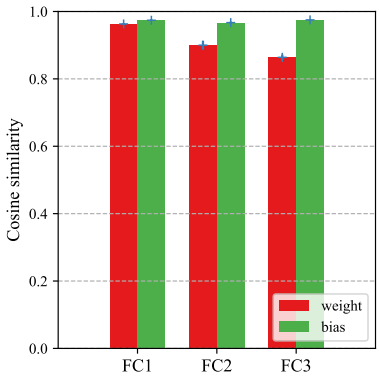


Figure 2: Cosine similarities between the online and offline calculated gradients on a MLP architecture. The error bar indicates the standard deviation across 50 random data batches.

Although slightly degrades from the offline version, the online trained SNNs can still achieve comparable results to their teacher models with accuracies dropping by less than 1%, which indicates that removing temporal dependency between time steps has little impact on the training of SNNs. To further shed light on why the online LTL rule can perform as effectively as the offline version, we conduct an experiment to analyze the degree of mismatch between their calculated gradients. In this experiment, we first pre-train a teacher ANN model on the MNIST dataset [29] with a 4-layer MLP architecture (i.e., 784-800-800-800-10). Then, we randomly initialize a student SNN model and draw 50 random batches of 128 samples to calculate the gradients at each layer. As shown in Figure 2, the cosine similarities between offline and online calculated gradients remain higher than 0.86 for all the hidden layers. According to the hyperdimensional computing theory [23], any two high dimensional random vectors are approximately

Table 1: Summary of classification accuracies on CIFAR-10, CIFAR-100 and Tiny ImageNet datasets. Δ Acc. = SNN Acc. - ANN Acc.; T_w: time window length.

Dataset	Method	Neuron	Architecture	ANN Acc. (%)	SNN Acc. (%)	T_w=16		T_w=32	
						SNN Acc. (%)	Δ Acc. (%)	SNN Acc. (%)	Δ Acc. (%)
CIFAR-10	PTL [52]	IF	VGG-11	90.59	-	91.24	0.65	-	-
	Offline (Online) LTL	LIF	VGG-11	92.89	-	93.20 (92.11)	0.31 (-0.78)	93.29 (92.75)	0.40 (-0.14)
	Offline (Online) LTL	IF	VGG-11	92.89	-	93.08 (92.03)	0.19 (-0.86)	93.25 (92.31)	0.36 (-0.58)
	SPIKE-NORM [44]	IF	VGG-16	91.70	91.55 (T_w=2500)	-	-	-	-
	Hybrid Train [41]	IF	VGG-16	92.81	91.13 (T_w=100)	-	-	-	-
	Calibration [30]	IF	VGG-16	95.72	-	-	-	93.71	-2.01
	RMP [18]	IF	VGG-16	93.63	93.63 (T_w=2048)	-	-	-	-
	TSC [17]	IF	VGG-16	93.63	93.63 (T_w=2048)	-	-	-	-
	Opt. [11]	IF	VGG-16	92.34	-	92.29	-0.05	92.29	-0.05
	Opt. mem. [4]	IF	VGG-16	94.57	-	93.38	-1.19	94.20	-0.37
	QCFS [5]	IF	VGG-16	95.52	-	95.40	-0.12	95.54	0.02
	Diet-SNN [40]	LIF	VGG-16	-	92.70 (T_w=5)	-	-	-	-
	Offline (Online) LTL	LIF	VGG-16	94.05	-	93.23 (92.85)	-0.82 (-1.20)	93.92 (93.52)	-0.13 (-0.53)
	Offline (Online) LTL	IF	VGG-16	94.05	-	93.04 (92.22)	-1.01 (-1.83)	93.72 (93.17)	-0.33 (-0.88)
CIFAR-100	Hybrid Train [41]	IF	VGG-11	71.21	67.87 (T_w=125)	-	-	-	-
	Offline (Online) LTL	LIF	VGG-11	71.59	-	72.63 (70.80)	1.04 (-0.79)	73.08 (72.45)	1.49 (0.86)
	Offline (Online) LTL	IF	VGG-11	71.59	-	72.74 (70.81)	1.15 (-0.78)	72.71 (72.29)	1.12 (0.70)
	SPIKE-NORM [44]	IF	VGG-16	71.22	70.77 (T_w=2500)	-	-	-	-
	Calibration [30]	IF	VGG-16	77.89	-	-	-	73.55	-4.34
	RMP [18]	IF	VGG-16	71.22	70.93 (T_w=2048)	-	-	-	-
	TSC [17]	IF	VGG-16	71.22	70.97 (T_w=2048)	-	-	-	-
	Opt. [11]	IF	VGG-16	70.49	-	65.94	-4.55	69.80	-0.69
	Opt. mem. [4]	IF	VGG-16	76.31	-	70.72	-5.59	74.82	-1.49
	QCFS [5]	IF	VGG-16	76.28	-	76.24	-0.04	77.01	0.73
	Diet-SNN [40]	LIF	VGG-16	–	69.67 (T_w=5)	-	-	-	-
	Offline (Online) LTL	LIF	VGG-16	74.42	-	74.19 (71.09)	-0.23 (-3.33)	74.92 (72.97)	0.50 (-1.45)
	Offline (Online) LTL	IF	VGG-16	74.42	-	73.67 (69.55)	-0.75 (-4.87)	74.56 (71.34)	0.14 (-3.08)
Tiny-ImageNet	DCT [15]	IF	VGG-13	56.90	52.43 (T_w=125)	-	-	-	-
	Offline (Online) LTL	LIF	VGG-13	56.16	-	55.37 (54.82)	-0.79 (-1.34)	55.85 (55.91)	-0.31 (-0.25)
	Offline (Online) LTL	IF	VGG-13	56.16	-	55.27 (54.28)	-0.89 (-1.88)	55.85 (55.61)	-0.31 (-0.55)
	Hybrid [27]	IF	VGG-16	56.56	51.92 (T_w=150)	-	-	-	-
	Offline (Online) LTL	LIF	VGG-16	57.39	-	56.85 (56.87)	-0.54 (-0.52)	57.73 (57.45)	0.34 (0.06)
	Offline (Online) LTL	IF	VGG-16	57.39	-	56.59 (56.39)	-0.80 (-1.00)	57.49 (57.00)	0.10 (-0.39)

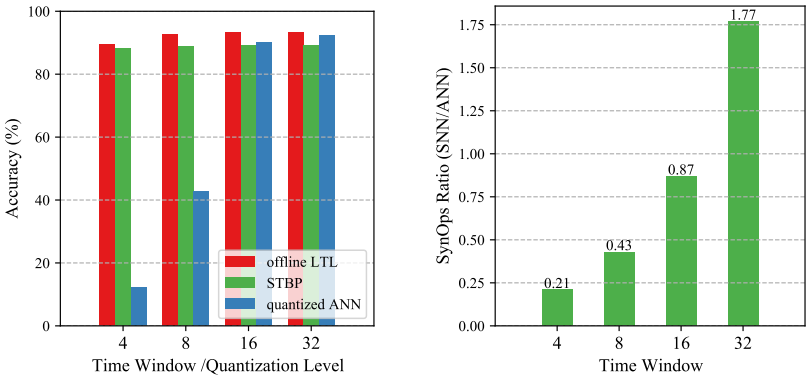
orthogonal. It suggests the online estimated gradients are very close to the offline calculated ground truth values and guarantees that the desired learning dynamics can be well approximated.

For VGG-16 model, we notice a modest accuracy drop that is more than 1% from the teacher model, which can be explained by the compounding layerwise representation errors. Fortunately, such errors can be well addressed with a larger time window. Specifically, at $T_w = 32$, the accuracy gap to the teacher ANN narrows down to only 0.13% and 0.33% respectively for LIF and IF neuron models under the offline training setting. When compared against other SOTA learning algorithms, the proposed LTL rule can achieve comparable performance in terms of both the final SNN model accuracies and the accuracy drop from ANN models. The same conclusions can also be drawn from the results of CIFAR-100 and Tiny ImageNet datasets, it hence suggests the proposed LTL rule is a generalized, scalable, and highly effective method for constructing high performance SNNs.

3.2 Rapid and Efficient Pattern Recognition

To realize the goal as a power-efficient alternative to the mainstream ANNs in solving pattern recognition tasks, SNNs should make accurate and rapid predictions with a low firing rate. Here, we study whether the proposed LTL rule is able to train SNNs that can cope with a short time window. Specifically, we design experiments that progressively increase the time window T_w from 4 to 32 during training, and we compare the offline LTL rule against the STBP [53] learning rule on the CIFAR-10 dataset. To ensure a fair comparison, we use the VGG-11 architecture with the same experimental settings, except that the LTL rule uses local objectives and the STBP rule uses a global objective. We note that the discrete neural representation of SNNs is equivalent to ANNs with quantized activation values [52]. Hence, we also compare our results to quantized ANNs obtained through quantization aware training [21].

As shown in Figure 3(a), we observe that the accuracy of the quantized ANN degrades rapidly when the total quantization level is lower than 16, while both LTL and STBP trained SNNs can maintain a high accuracy even at $T_w = 4$. It suggests the surrogate gradient approach adopted in SNN training can outperform the naive straight-through estimator [21] in addressing the discontinuity in the activation function. Moreover, the offline LTL rule performs better than the STBP rule, which can be explained by the fact that the gradient approximation errors of the surrogate gradient function will not compound across layers as in the STBP training [52]. Similar results have also been observed in the VGG-16 architecture (See Supplementary Materials Section C).



(a) (b)

Figure 3: (a) Comparison of the network performance under different time window lengths (SNN) and quantization levels (ANN). The results are obtained on the CIFAR-10 dataset using the VGG-11 architecture. (b) The ratio of total synaptic operations between SNN and ANN as a function of the time window length.

To shed light on the energy efficiency of LTL trained SNN models, we follow the common practice by computing the total synaptic operations (SynOps) ratio of SNN to ANN under different T_w [51]. In general, the total SynOps (i.e., FLOPs) required by an ANN is a constant number that depends on the specific network architecture in use, while the total computations typically increase linearly with T_w for rate-based SNNs. As shown in Figure 3(b), to achieve a comparable accuracy to the ANN counterpart, our SNN only requires $0.43\times$ total SynOps with $T_w = 8$. It is worth noting that the multiply-accumulate (MAC) operation required by ANNs is $5\times$ more expensive than the accumulate (AC) operations used in SNNs under 45 nm CMOS technology [19]. As such, our SNNs can yield an order of magnitude energy saving over the mainstream ANNs, and this value can be further boosted when deploy our SNNs onto the emerging mixed-signal NC chips.

3.3 Rapid Learning Convergence

By decoupling the learning of each layer through local objectives, the proposed LTL rule cleverly circumvents the spatial credit assignment and reduces the credit assignment to only the temporal domain. Moreover, the gradient approximation errors of the surrogate gradient learning method are alleviated as the effective gradient transportation distance is reduced. Taking benefits from these, the network convergence speed can be improved significantly compared to those trained with the STBP and TET rule. As the learning curves are shown in Figure 4, both the offline and online LTL rules can converge rapidly within five epochs on the CIFAR-10 dataset. In contrast, the accuracy is still increasing at Epoch 50 for the STBP and TET learning rule. Note that we follow the same settings for all these experiments, except for the learning objectives. Similar results have also been observed in the VGG-16 architecture (see Supplementary Materials Section D).

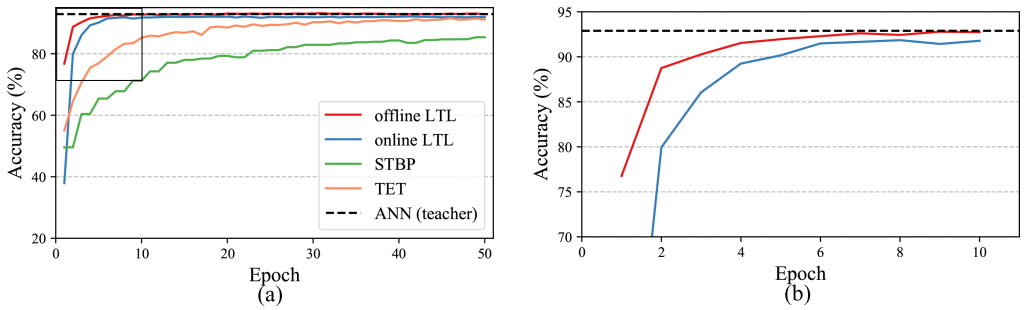


Figure 4: Comparison of the learning curves of offline LTL, online LTL, STBP and TET learning rules. The experiments are performed on the CIFAR-10 dataset with the VGG-11 architecture.

3.4 Low Memory and Time Complexity

In this section, we study the memory and time complexity of the proposed LTL rule and compare it against other popular learning rules in Table 2. The STBP [53] rule requires storing the intermediate state of each neuron for all time steps and layers so as to be used during backpropagation, resulting in $\mathcal{O}(MT_wL)$ memory complexity, where M is the number of neurons in a layer, T_w is the time window length, and L is the total number of layers. The offline LTL rule has the same memory complexity as the STBP rule if all layers are trained in parallel. However, as will be discussed soon, we can reduce the memory complexity by training all layers in sequence. Notably, by updating the network parameters using only local information, the online LTL and DECOLLE [22] learning rules do not require storing any intermediate state, resulting in $\mathcal{O}(1)$ memory complexity.

In terms of the time complexity, each parameter update of the STBP rule requires NMT_w multiplications, where N is the number of input neurons to a layer. Due to layer-wise interlocking, all layers need to be updated in sequence from top to bottom, requiring $\mathcal{O}(NMT_wL)$ time complexity to update all network parameters once. For offline LTL, by decoupling the dependency between layers and training all layers in parallel, the time complexity can be reduced to $\mathcal{O}(NMT_w)$. As mentioned earlier, we can trade off the memory complexity with the time complexity by training the network layers progressively one after the other. In this way, the memory usage of the offline LTL rule will be reduced to $\mathcal{O}(MT_w)$, while the time consumption will be increased to $\mathcal{O}(NMT_wL)$. To validate the effectiveness of this training strategy, we perform an experiment on the CIFAR-10 dataset using VGG-11 architecture and we obtain a comparable result (92.98%) to that obtained when all layers are trained concurrently (93.20%). In this work, to achieve the shortest training time, we use the parallel training strategy as the default for all the other reported experiments. Other users can flexibly select the training strategy according to their available computing resources.

For online LTL, by leveraging the temporal local errors, the time complexity can be further reduced to $\mathcal{O}(NM)$. Note that the DECOLLE requires an additional readout layer to calculate local errors, thereby requiring more time than online LTL. It is also worth mentioning that by gating the parameter updates with the proposed efficient on-chip implementation, we can further reduce the time complexity for online LTL. In sum, by coupling the low computational complexity with the rapid learning convergence discussed in Section 3.3, the proposed LTL rule can dramatically improve the training efficiency of SNNs.

Table 2: Comparison of the memory and time complexity of different learning rules. N: number of input neurons to a layer, M: number of neurons in a layer, N_r: number of readout neurons in a layer, L: number of layers, T_w: time window length

Method	Memory	Time
STBP [53]	$\mathcal{O}(MT_wL)$	$\mathcal{O}(NMT_wL)$
DECOLLE [22]	$\mathcal{O}(1)$	$\mathcal{O}(NM + MN_r)$
Offline LTL	$\mathcal{O}(MT_wL)$	$\mathcal{O}(NMT_w)$
Online LTL	$\mathcal{O}(1)$	$\mathcal{O}(NM)$

3.5 Robust to Hardware-related Noises

Device non-ideality issues of analog computing substrates remain a major hurdle for deploying SNNs on ultra-low-power mixed-signal NC chips. The proposed LTL rule can, however, address this challenge by enabling on-chip training in a noise-aware manner. Here, we examine the effectiveness of the on-chip LTL rule in addressing hardware-related noises and we focus on four typical noise sources: device mismatch, quantization noise, thermal noise, and neuron silencing [6]. In particular, we use the noise models introduced by Julian et al. [6], which were measured on DYNAP-SE [31] mixed-signal neuromorphic processor. Therefore, it can faithfully reflect the actual hardware noises that one can expect in the real scenario. To understand the effect of these noises, we first take an LTL pre-trained SNN and apply noise at different levels to it. Then, we use the online LTL rule to calibrate the pre-trained SNN model under the given noise.

Device Mismatch Device mismatch causes neuronal and synaptic parameters to vary from the desired ones. To simulate the parameter mismatch, we add Gaussian noise to both the initial network parameters and their parameter updates following $\theta' \sim \mathcal{N}(\theta, \sigma\theta)$. In this model, σ controls the level of mismatch, and we vary it from 0.05 to 0.4 to simulate different levels of noise. As shown in Figure 5(a), the accuracy of the pre-trained SNN degrades by more than 20% under the mismatch level $\sigma = 0.4$. Notably, the proposed LTL rule can perform fast calibration and restore the original

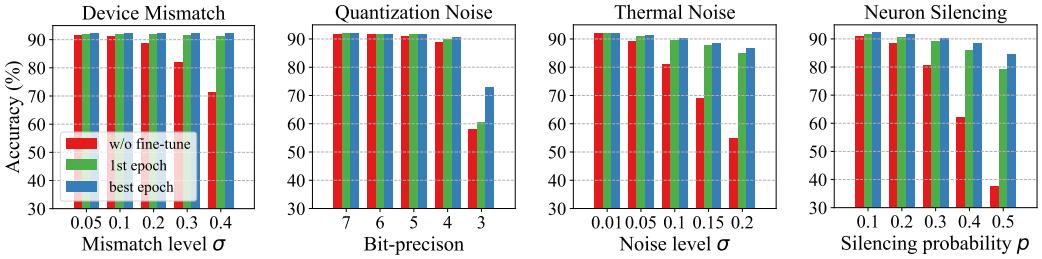


Figure 5: The classification accuracy is a function of the level of device noises for (a) device mismatch, (b) quantization noise, (c) thermal noise, and (d) neuron silencing. w/o fine-tune: without any noise-aware fine-tuning, 1st epoch: the result after training with the online LTL rule for 1 epoch, best epoch: the best result obtained when training with the online LTL rule for 50 epochs. The experiments are performed on the CIFAR-10 dataset with the VGG-11 architecture.

accuracy with only one training epoch. This will be advantageous for those devices that have poor endurance.

Quantization Noise The non-volatile memory devices typically have a limited number of analog states, thereby limiting the usable number of bits per network parameter. To simulate such a quantization noise, we perform post-training-quantization to the bit-width of $7, 6, 5, 4, 3$. As shown in Figure 5(b), the SNN performs robustly to the quantization noise for bit-precision above 3. After which, the accuracy drops sharply by more than 30%. Following the quantization aware training [21], the proposed LTL rule can alleviate the effect of this type of noise and successfully recover the accuracy by around 17%. Nevertheless, due to the reduced representation power, the resulted 3-bit precision model is still lagging behind the full precision one by about 15%.

Thermal Noise Thermal noise is intrinsic to the neuromorphic devices, which can be modeled as Gaussian noise on the input currents $I' \sim \mathcal{N}(I, \sigma I)$. Similar to the device mismatch noise, we simulate different levels of thermal noise by varying the σ from 0.01 to 0.2. As illustrated in Figure 5(c), the model is highly sensitive to the thermal noise and the accuracy drops steadily with a growing level of noise. We notice that the LTL rule can address such noise and quickly recover the accuracy in large part with only one training epoch.

Neuron Silencing Neuron silencing noise corresponds to the situation when a portion of spiking neurons fails to respond, and hence disturbs the network dynamics. We simulate neuron silencing noise by randomly masking the neuron outputs with a failure rate varying from 10% to 50%. As shown in Figure 5(d), the accuracy drops steadily with a growing amount of failed neurons. Similar to other noise types, the accuracies are quickly restored with the on-chip LTL training and maintained above 80% even under the challenging case when 50% of the neurons are failed.

4 Conclusion

In this paper, we propose the Local Tandem Learning (LTL) rule for SNNs. Taking inspiration from the teacher-student learning approach, we leverage the highly effective feature representations of easily trainable ANN to guide the training of SNNs. This simplifies the credit assignment problem and leads to rapid network convergence with significantly reduced computational complexity. We also demonstrate the required temporal credit assignment can be well approximated with locally available information, and we provide an efficient on-chip implementation of the proposed LTL rule. Under this on-chip setting, we demonstrate that the LTL rule can provide robustness against various device non-ideality issues. It, therefore, opens up myriad opportunities for rapid and efficient training and deployment of SNNs on ultra-low-power mix-signal NC platforms. For future work, we will explore the proposed teacher-student learning approach on the recurrent neural networks and transformer architectures so as to handle those temporally rich signals, such as speech, video, and text.

References

[1] Syed Ahmed Aamir, Yannik Stradmann, Paul Müller, Christian Pehle, Andreas Hartel, Andreas Grübl, Johannes Schemmel, and Karlheinz Meier. An accelerated lif neuronal network array for a large-scale mixed-signal neuromorphic architecture. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(12):4299–4312, 2018.

[2] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. *arXiv preprint arXiv:2202.01440*, 2022.

[5] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2021.

[6] Julian Büchel, Dmitrii Zendrikov, Sergio Solinas, Giacomo Indiveri, and Dylan R Muir. Supervised training of spiking neural networks for robust deployment on mixed-signal neuromorphic processors. *Scientific reports*, 11(1):1–12, 2021.

[7] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015.

[8] Yanqi Chen, Zhaofei Yu, Wei Fang, Tiejun Huang, and Yonghong Tian. Pruning of deep spiking neural networks through gradient rewiring. *arXiv preprint arXiv:2105.04916*, 2021.

[9] Erika Covi, Stefano Brivio, Alexander Serb, Themis Prodromakis, Marco Fanciulli, and Sabina Spiga. Analog memristive synapse in spiking networks implementing unsupervised learning. *Frontiers in neuroscience*, 10:482, 2016.

[10] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

[11] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations*, 2021.

[12] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International joint conference on neural networks (IJCNN)*, pages 1–8. ieee, 2015.

[13] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[14] Simon Friedmann, Johannes Schemmel, Andreas Grübl, Andreas Hartel, Matthias Hock, and Karlheinz Meier. Demonstrating hybrid learning in a flexible neuromorphic hardware system. *IEEE transactions on biomedical circuits and systems*, 11(1):128–142, 2016.

[15] Isha Garg, Sayeed Shafayet Chowdhury, and Kaushik Roy. Dct-snn: Using dct to distribute spatial information over time for learning low-latency spiking neural networks. *arXiv preprint arXiv:2010.01795*, 2020.

[16] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[17] Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In *European Conference on Computer Vision*, pages 388–404. Springer, 2020.

[18] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13558–13567, 2020.

[19] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[20] Giacomo Indiveri and Yulia Sandamirskaya. The importance of space and time for signal processing in neuromorphic agents: the challenge of developing low-power, autonomous agents that interact with the environment. *IEEE Signal Processing Magazine*, 36(6):16–28, 2019.

[21] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[22] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14:424, 2020.

[23] P. Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation*, 1(2):139–159, 2009.

[24] Chul-Heung Kim, Soochang Lee, Sung Yun Woo, Won-Mook Kang, Suhwan Lim, Jong-Ho Bae, Jaeha Kim, and Jong-Ho Lee. Demonstration of unsupervised learning with spike-timing-dependent plasticity using a tft-type nor flash memory array. *IEEE Transactions on Electron Devices*, 65(5):1774–1780, 2018.

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[27] Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A Beerel. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3953–3962, 2021.

[28] Dongseok Kwon, Suhwan Lim, Jong-Ho Bae, Sung-Tae Lee, Hyeongsu Kim, Young-Tak Seo, Seongbin Oh, Jangsaeng Kim, Kyuho Yeom, Byung-Gook Park, et al. On-chip training spiking neural networks using approximated backpropagation with analog synaptic devices. *Frontiers in neuroscience*, page 423, 2020.

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[30] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, pages 6316–6325. PMLR, 2021.

[31] Saber Moradi, Ning Qiao, Fabio Stefanini, and Giacomo Indiveri. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE transactions on biomedical circuits and systems*, 12(1):106–122, 2017.

[32] Emre Neftci and Giacomo Indiveri. A device mismatch compensation method for vlsi neural networks. In *2010 Biomedical Circuits and Systems Conference (BioCAS)*, pages 262–265. IEEE, 2010.

[33] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[34] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[35] Melika Payvand, Mohammed E Fouda, Fadi Kurdahi, Ahmed M Eltawil, and Emre O Neftci. On-chip error-triggered learning of multi-layer memristive spiking neural networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 10(4):522–535, 2020.

[36] G Pedretti, V Milo, S Ambrogio, R Carboni, S Bianchi, A Calderoni, N Ramaswamy, AS Spinelli, and D Ielmini. Memristive neural network for on-line learning and tracking with brain-inspired spike timing dependent plasticity. *Scientific reports*, 7(1):1–10, 2017.

[37] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

[38] M Prezioso, MR Mahmoodi, F Merrikh Bayat, H Nili, H Kim, A Vincent, and DB Strukov. Spike-timing-dependent plasticity learning of coincidence detection with passively integrated memristive circuits. *Nature communications*, 9(1):1–8, 2018.

[39] Ning Qiao and Giacomo Indiveri. Scaling mixed-signal neuromorphic processors to 28 nm fd-soi technologies. In *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 552–555. IEEE, 2016.

[40] Nitin Rathi and Kaushik Roy. Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–9, 2021.

[41] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv preprint arXiv:2005.01807*, 2020.

[42] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[43] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

[44] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

[45] Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31, 2018.

[46] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[47] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[49] Stanisław Woźniak, Angeliki Pantazi, Thomas Bohnstingl, and Evangelos Eleftheriou. Deep learning incorporating biologically inspired neural dynamics and in-memory computing. *Nature Machine Intelligence*, 2(6):325–336, 2020.

[50] Jiayu Wu, Qixiang Zhang, and Guoxi Xu. Tiny imagenet challenge. *Technical report*, 2017.

[51] Jibin Wu, Yansong Chua, Malu Zhang, Guoqi Li, Haizhou Li, and Kay Chen Tan. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[52] Jibin Wu, Chenglin Xu, Xiao Han, Daquan Zhou, Malu Zhang, Haizhou Li, and Kay Chen Tan. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[53] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

[54] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1311–1318, 2019.

[55] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Michael L Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. Toward human parity in conversational speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(12):2410–2423, 2017.

[56] Friedemann Zenke and Emre O Neftci. Brain-inspired learning on neuromorphic substrates. *Proceedings of the IEEE*, 109(5):935–950, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 4
   (c) Did you discuss any potential negative societal impacts of your work? [No] Not applicable to our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Supplementary Materials Section A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Supplementary Materials Section B
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Supplementary Materials Section B
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report the error bar for MNIST experiments. Due to time constraint, we not run for other datasets and we may update in future.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Supplementary Materials Section B
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 3
   (b) Did you mention the license of the assets? [N/A] We use public datasets.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]