Deep Spiking Neural Network: Energy Efficiency Through Time Based Coding

Bing Han(✉) and Kaushik Roy

Purdue University, West Lafayette, IN 47907, USA
{han183,kaushik}@purdue.edu

Abstract. Spiking Neural Networks (SNNs) are promising for enabling low-power event-driven data analytics. The best performing SNNs for image recognition tasks are obtained by converting a trained deep learning Analog Neural Network (ANN) composed of Rectified Linear Unit (ReLU) activation to SNN consisting of Integrate-and-Fire (IF) neurons with "proper" firing thresholds. However, this has come at the cost of accuracy loss and higher inference latency due to lack of a notion of time. In this work, we propose an ANN to SNN conversion methodology that uses a time-based coding scheme, named Temporal-Switch-Coding (TSC), and a corresponding TSC spiking neuron model. Each input image pixel is presented using two spikes and the timing between the two spiking instants is proportional to the pixel intensity. The real-valued ReLU activations in ANN are encoded using the spike-times of the TSC neurons in the converted TSC-SNN. At most two memory accesses and two addition operations are performed for each synapse during the whole inference, which significantly improves the SNN energy efficiency. We demonstrate the proposed TSC-SNN for VGG-16, ResNet-20 and ResNet-34 SNNs on datasets including CIFAR-10 (93.63% top-1), CIFAR-100 (70.97% top-1) and ImageNet (73.46% top-1 accuracy). It surpasses the best inference accuracy of the converted rate-encoded SNN with 7–14.5× lesser inference latency, and 30–60× fewer addition operations and memory accesses per inference across datasets.

Keywords: Spiking Neural Network · ANN-SNN conversion · Temporal coding · Energy efficiency · Deep learning · Machine learning

1 Introduction

Deep neural networks, referred to as Analog Neural Networks (ANNs) in this article (to distinguish them from the digital spiking counterpart), composed of several layers of interconnected neurons, have achieved state-of-the-art performance in various Artificial Intelligence (AI) tasks including image localization and recognition [18,33], video analytics [29], and natural language processing [16], among other tasks. For instance, ResNet [12] that won the ImageNet Large Scale Visual Recognition Challenge in 2015 consists of 152 layers with over 60 million parameters, and incurs 11.3 billion FLOPS per classification. In an effort to explore more power efficient neural architectures, recent research efforts have been directed towards devising computing models inspired from biological neurons that compute and communicate using spikes. These emerging class of networks with increased bio-fidelity are known as Spiking Neural Networks (SNNs) [22]. The intrinsic power-efficiency of SNNs stems from their sparse spike-based computation and communication capability, which can be exploited to achieve higher computational efficiency in specialized neuromorphic hardware [2,4,24].

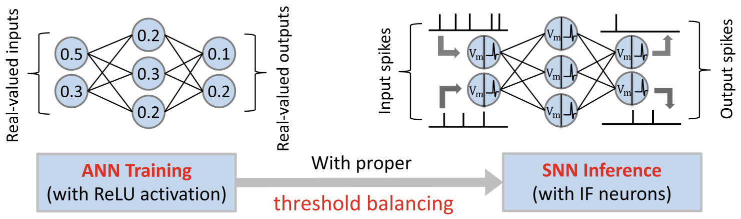


Fig. 1. Illustration of the ANN-SNN conversion methodology.

Considering the rapid strides in accuracy achieved by ANNs over the past few years, SNN training algorithms are much less mature and are an active field of research. The ANN to SNN conversion based training approaches have yielded the best performing SNNs (typically composed of Integrate-and-Fire (IF) neurons), which are converted from a trained non-spiking ANN (consisting of Rectified Linear Units (ReLUs) as the neural activation) [3,6,7,31,37] [47] as illustrated in Fig. 1. However, this has come at the cost of large inference latency (time-steps) and accuracy degradation. Recent work has shown that, the soft-reset membrane potential [35,36] in SNN can alleviate the information loss that occurs during ANN-SNN conversion by retaining the membrane potential above threshold at the firing instants. Near loss-less ANN-SNN conversion has been achieved using the "soft-reset" Residual-Membrane-Potential (RMP) spiking neuron and the required RMP-SNN firing threshold initialization [10]. The conversion schemes intelligently assign "appropriate" firing thresholds to the neurons at different layers of the network, thereby, ensuring that the IF spiking rates (number of spikes over large enough time interval) are proportional to the corresponding analog ReLU activations. However, it still requires sizeable number of inference time-steps, and the spiking activity is relatively high due to the rate-based neural coding for SNNs. The number of spikes for encoding a real-valued activation increases with both the ANN ReLU activation value and the SNN inference time-steps performed. A large number of spikes are fired to achieve accuracy comparable to the ANN accuracy, which leads to high computational cost [6,36,37]. Although several recent methods [34,45,46] reduced the number of spikes by employing more efficient neural coding, these methods relied on complex neuron models that continually perform expensive operations and the computational costs remain high (will be discussed in Sect. 2). The benefits of using spike-times as an additional dimension for computation has not been fully explored due to lack of general learning algorithms for SNNs. The Spike Timing Dependent Plasticity (STDP) based supervised [5,23,40,43] and semi-supervised learning algorithms [17,19,26,42] have thus far been restricted to shallow SNNs (with $\leq$5 layers) yielding considerably lower than acceptable accuracy on complex datasets like CIFAR-10 [9,41]. In order to scale the networks much deeper, the spike-based error backpropagation algorithms have been proposed for supervised training of SNNs [1,15,20,21,28,30,32,38,44]. However the training complexity incurred for performing backpropagation of the rate-encoded spikes (error) over time has limited their scalability for SNNs beyond 9–11 layers [20].

In this work, we propose an ANN to SNN conversion methodology that uses a time-based neural coding scheme, named Temporal-Switch-Coding (TSC), and a corresponding TSC spiking neural model. The proposed TSC encoding scheme is more energy efficient than the First-spike latency based encoding schemes such as Time-To-First-Spike (TTFS). In the converted SNN with TTFS encoding, a real-valued ReLU activation in ANN was approximated by the latency to the first spike of the corresponding spike train in the SNN, and at most one spike needs to be fired for each activation. Even though total spikes are reduced, the memory access and computational costs remain high, because the spiking neuron needs to keep track of a synapse, at every time-step afterwards, ever since the synapse received its first spike. In SNN with the proposed TSC encoding scheme, at most two spikes are fired for each activation, but the spiking neuron keeps track of a synapse only at the instant of an input-spike to reduce the memory access and computational costs (details discussed in Sect. 3).

2 Related Work

Recent ANN to SNN conversion methods reduced the number of spikes used to encode activations by employing more efficient neural coding. In [45], an ANN was converted to an Adapting SNN (AdSNN) based on synchronous Pulsed Sigma-Delta coding. When driven by a strong stimulus, an Adaptive Spiking Neuron (ASN) adaptively raises its dynamic firing threshold every time it fires a spike, reducing its firing rate. However, an ASN has to perform four multiplications every time step to update its postsynaptic current, firing threshold, and refractory response. In [34], an ANN was converted to an SNN based on the Time-To-First-Spike (TTFS) temporal coding, where an activation in the ANN was approximated by the latency to the first spike of the corresponding spike train in the SNN. Thus, at most one spike needs to be fired for each activation, which reduces the number of spikes in inference. However, the spiking neuron needs to keep track of a synapse, at every time-step till the end of the inference, ever since this synapse received its first spike. At each time-step, a large number of synapses including those that receive an input-spike at the current time-step and those that have received an input-spike at any prior time-step are added to the membrane potential of the spiking neuron, which incurs expensive memory access and computational costs. In [25], each activation of an ANN was approximated with a power of two, where the exponents of the powers were constrained within a set of several consecutive integers. The error tolerance of an ANN allows it to compensate for approximation errors in the corresponding SNN during the training phase, which in turn helps close the performance gap between the SNN and the ANN. Authors in [46] proposed Logarithmic Temporal Coding (LTC), where the number of spikes used to encode an activation value grows logarithmically with activation. The Exponentiate-and-Fire (EF) spiking neuron only involves bit-shift and addition operations. However, the energy benefit comes at the cost of large accuracy loss due to the approximation error introduced by LTC, and requires constrained ANN training to compensate for the loss. The method is implemented for shallow network consisting of 2 convolutional layers and evaluated on MNIST dataset. The performance on deeper architectures such as VGG and more complex datasets such as CIFAR-10 are not clear.

3 Temporal Switch Coding Scheme

The real-valued pixel intensities of input image are mapped to the spike-times over a large enough time interval for SNN during inference. The time-step dt is used to keep track of the discrete time, and the total time-steps (latency) required are dictated by the desired inference accuracy. Note that the input images fed to ANN are typically normalized to zero mean and unit standard deviation, yielding pixel intensities between ±1, and bipolar spikes are used to represent the positive and negative pixels. In our proposed TSC time-based coding scheme, at most a pair of spikes with opposite signs are used to encode one real-valued pixel in time. Suppose $N (N \geq 2)$ time-steps are used to encode a pixel p $(-1 \leq p \leq 1)$. The magnitude of p is first quantized into N levels as described by Eq. 1, where $p^*(0 \leq p^* \leq N)$ is the pixel magnitude after quantization. As described by Eq. 2, two spikes will be produced at time $t = 1$ and $t = p^*$, if p^* is greater or equal to 2; no spike will be produced otherwise. As shown in Fig. 2(a), the real-valued pixel p $(p \neq 0)$ is encoded using the spike-times (t_s^+, t_s^-) of a pair of spikes. The first spike which occurs at $t_s^+ = 1$ always has the same sign as p, and the second spike that occurs at $t_s^- = p^*$ always has the opposite sign as p.

$$\begin{gathered} p^* = \lfloor |p|\ (N-1) \rfloor + 1 \\ (-1 \leq p \leq 1, 1 \leq p^* \leq N, \mathit{and}\ \lfloor\ \rfloor\ \mathit{is\ the\ floor\ operation}) \end{gathered} \tag{1}$$

$$T(t) = \begin{cases} sgn(p), & if\ (p^* \geq 2,\ t = 1) \\ -sgn(p), & if\ (p^* \geq\ 2,\ t = p^*) \\ 0, & else \end{cases} \tag{2}$$

$$(where\ \text{“}sgn\text{”}\ is\ the\ sign\ function,\ sgn(p) = \tfrac{|p|}{p})$$

We use time-based TTFS and rate-based Poisson coding schemes as benchmarks to evaluate the performance of our proposed TSC encoding. In time-based TTFS, the real-valued pixel p $(p \neq 0)$ is encoded using the spike time t_s of a

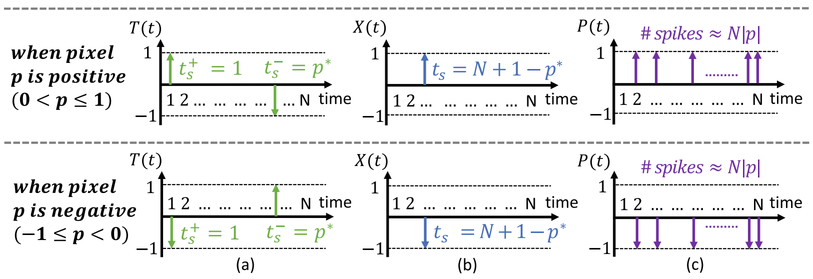
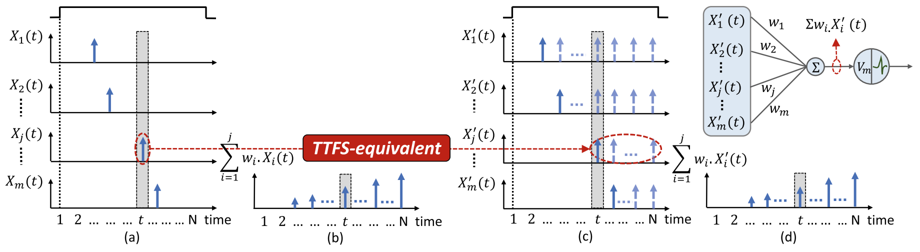


Fig. 2. Illustration of encoding a real-valued pixel p ($-1 \le p \le 1$) of static image to spike trains for SNN using: (a) Temporal-Switch-Coding (TSC), (b) Time-To-First-Spike (TTFS), and (c) Poisson rate-based encoding.

single spike, which always has the same sign as p. The spike-time $t_s = N+1-p^*$, where p^* is the quantized pixel intensity as described by Eq. 1. As shown in Fig. 2 (b), larger the quantized pixel intensity p^*, earlier the spike occurs. In Poisson rate coding, pixels are mapped to spike trains firing at a rate (number of spikes over time) proportional to the corresponding pixel intensities as shown in [13]. The pixel intensity is first mapped to instantaneous spiking probability of the corresponding input neuron. We use Poisson process to generate the input-spike in a stochastic manner as explained below. At every time-step of SNN operation, we generate a uniform random number between 0 and 1, which is compared against the magnitude of the pixel intensity $|p|$ ($|p| \le 1$). A spike is produced if the random number is lesser than $|p|$. As shown in Fig. 2(c), the spikes produced with Poisson rate coding always have the same sign as p, and total number of spikes over N time-steps is proportional to the magnitude $|p|$.

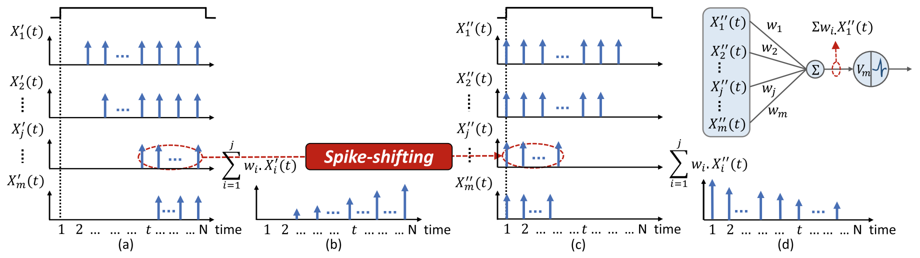
3.1 Computation Reduction

In this section, we first formalize the derivation of the proposed TSC coding scheme, which is obtained from modifying the TTFS, and explain reasons for the increased performances and decreased computation in SNN inference. Let us assume spike-trains $X_1(t)$ to $X_m(t)$ are encoded from an input image consisting of m pixels using TTFS coding as shown in Fig. 3(a). The spike-trains are fed to a spiking neuron through real-valued synaptic weights w_1 to w_m. The total number of inference time-steps is N. For simplicity, let us assume all pixels of the input image are of different values (between 0 and 1), and the m spike-trains from $X_1(t)$ to $X_m(t)$ have been sorted according to the spike-time (from early to late). As shown in Fig. 3(a), at time-step t, the spiking neuron receives only one spike from the input image through weight w_j. However, computing the weighted-sum of spike-input at t requires summing up synaptic weights from w_1 to w_j. Hence, both the TTFS spike-trains in Fig. 3(a) and the modified TTFS spike-trains in Fig. 3(c) incur the same amount of addition operations as shown in Fig. 3(b) and (d).



TTFS-equivalent
$X_1(t)$
$X_2(t)$
$X_j(t)$
$X_m(t)$
$\sum_{i=1}^{j} w_i.X_i(t)$
$\sum_{i=1}^{j} w_i.X_i'(t)$
$\Sigma w_i.X_i'(t)$
V_m
1 2 ... ... t ... ... N time
(a)
(b)
(c)
(d)

Fig. 3. (a) TTFS coding fires one spike for each pixel. (b) Modified TTFS fires multiple spike for one pixel. (c) Weighted-sum of the TTFS spikes. (d) Weighted-sum of the modified TTFS spikes.



Spike-shifting
$X_1'(t)$
$X_1''(t)$
$\sum_{i=1}^{j} w_i.X_i'(t)$
$\sum_{i=1}^{j} w_i.X_i''(t)$
$\Sigma w_i.X_i''(t)$
V_m
1 2 ... ... t ... ... N time
(a)
(b)
(c)
(d)

Fig. 4. (a) Modified TTFS fires "late" spikes. (b) Shifted TTFS fires "early" spikes. (c) Weighted-sum of the "late" spikes. (d) Weighted-sum of the "early" spikes.

Now let us transform the modified TTFS spike-trains in Fig. 3(c) to the proposed TSC spike-trains using the "spike-shifting" (shown in Fig. 4) and "frame-subtracting" (shown in Fig. 5) operations. First the "spike-shifting" operation is performed, in which, all "late" spikes in Fig. 4(a) are shifted in time to become the "early" spikes in Fig. 4(c), and the number of spikes for encoding the pixel does not change. The corresponding weighted-sum of input-spikes are shown in Fig. 4(b) and (d). Larger activation arrives earlier due to the "spike-shifting" operation. Next the "frame-subtracting" operation is performed to transform the shifted TTFS spike-trains in Fig. 5(a) to the proposed TSC spike-trains in Fig. 5(c). Suppose $X_j^{''}(t)$ is one of the shifted TTFS spike-trains that consists of t consecutive spikes as highlighted in Fig. 5(a). Every spike in the spike-train is subtracted by the previous spike (except for the first spike) as described by $X_j^{''}(t) - X_j^{''}(t-1)$. As shown in Fig. 5(c), only one positive spike (at time-step 1) and one negative spike (at time-step t+1) remain in the resulting TSC spike-train $T_j(t)$. All spikes from time-step 2 to time-step t become zeros. At each time-step, membrane potential $V_m(t)$ in the spiking neuron is updated as described by Eq. 3. Computing the weighted-sum of input-spike using the shifted TTFS spike-trains at t requires j addition operations as described by Eq. 4. However, by reusing the computation at $t-1$ as described by Eq. 5, the addition operations required for computing the weighted-sum of input-spike at time t is reduced to 1 using TSC spike-trains as described by Eq. 6.

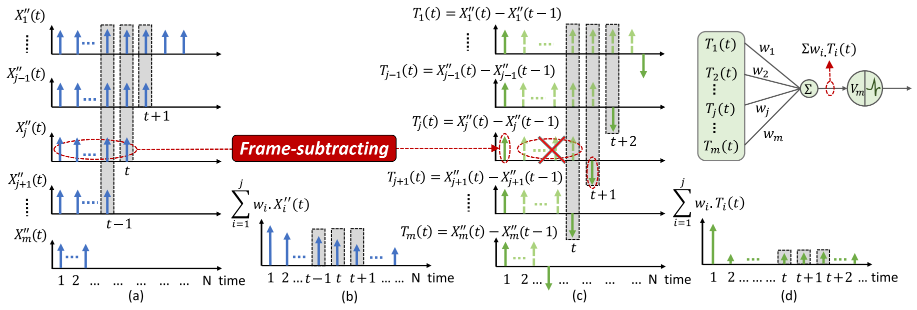


Fig. 5. (a) Shifted TTFS fires multiple consecutive spikes for one pixel. (b) Subtracted TTFS fires a pair of bipolar spikes for one pixel. (c) Weighted-sum of the shifted TTFS spikes. (d) Weight-sum of the TSC spikes.

$$V_m(t) = V_m(t-1) + \frac{dV_m}{dt}\bigg|_{(t)} \tag{3}$$

$$\frac{dV_m}{dt}\bigg|_{(t)} = \sum_{i=1}^{j} w_i.X_i^{''}(t) \tag{4}$$

$$\frac{dV_m}{dt}\bigg|_{(t)} = \frac{dV_m}{dt}\bigg|_{(t-1)} + \frac{d^2V_m}{dt^2}\bigg|_{(t)} \tag{5}$$

$$\frac{d^2V_m}{dt^2}\bigg|_{(t)} = \sum_{i=1}^{j} w_i.T_i(t) = w_j.T_j(t) \tag{6}$$

4 ANN to TSC-SNN Conversion

The fundamental distinction between ANN and SNN is the notion of time. In ANNs, input and output of neurons in all the layers are real-valued as shown in Fig. 6(a), and inference is performed with single feed-forward pass through the network. On the other hand, input and output of the TSC spiking neurons are encoded temporally using sparse spiking events over certain time period as shown in Fig. 6(b). Hence, inference in TSC-SNN is carried out over multiple feed-forward passes or time-steps (also known as inference latency), where each pass entails sparse spike-based computations. Achieving close to ANN accuracy with minimal inference latency is key to obtaining favorable trade-off between accuracy and computational efficiency. The proposed conversion methodology significantly advances the state-of-the-art in this regard as will be detailed in this section.

ANNs used for conversion to SNNs are typically trained with ReLU non-linearity [27] as shown in Fig. 6(a), where Y is the output of ReLU-based artificial neruon, $\sum_i w_i.p_i + b$ is the weighted sum of input p_i with weight w_i and bias

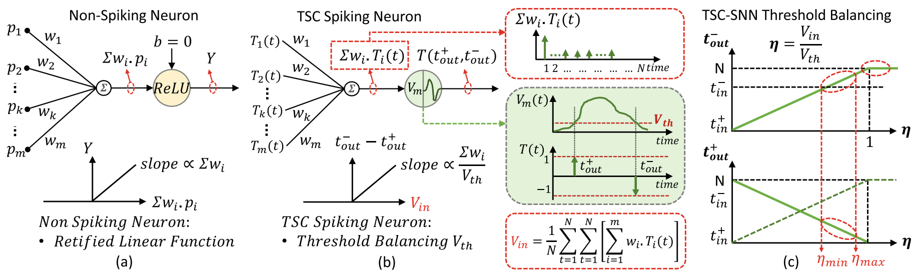

Non-Spiking Neuron
TSC Spiking Neuron
TSC-SNN Threshold Balancing
$b = 0$
$\Sigma w_i . p_i$
ReLU
Y
slope ∝ Σw_i
Non Spiking Neuron:
• Retified Linear Function
(a)
$\Sigma w_i . T_i(t)$
$T(t^+_{out}, t^-_{out})$
$t^-_{out} - t^+_{out}$
slope ∝ $\frac{\Sigma w_i}{V_{th}}$
V_{in}
TSC Spiking Neuron:
• Threshold Balancing V_{th}
(b)
$V_m(t)$
V_{th}
time
$T(t)$
$V_{in} = \frac{1}{N}\sum_{t=1}^{N}\sum_{t=1}^{N}\left[\sum_{i=1}^{m} w_i . T_i(t)\right]$
$\eta = \frac{V_{in}}{V_{th}}$
η_{min} η_{max}
(c)

Fig. 6. (a) Non-spiking ReLU neuron driven by a set of analog input pixels via weights (w). (b) TSC spiking neuron driven by a set of time-encoded spikes via weights (w). (c) TSC-SNN requires finding the appropriate firing threshold V_{th} to balance the spike-times across layers to achieve the best performance.

b. The bias is usually set to zero for effective ANN-SNN conversion [37]. The ReLU ouput varies linearly with the input for positive inputs. On the other hand, $\sum_i w_i.T_i(t)$ is the weighted-sum of spike-input received by the TSC spiking neuron as shown in Fig. 6(b). The TSC spiking neuron integrates the weighted-sum of spike-input into the membrane potential as described by Eq. 5 and Eq. 6. The average input V_{in} to the spiking neuron equals to the total amount of potential integrated by the TSC spiking neuron divided by the inference time steps N as described by Eq. 7. The membrane potential update mechanism in the TSC spiking neuron is similar to the "soft-reset" IF neuron. At any time-step, if the membrane potential $V_m(t)$ is higher than the firing threshold V_{th} ($V_{th} > 0$), the firing threshold is subtracted from the membrane potential $V_m(t) = V_m(t) - V_{th}$. However, a positive spike is produced only at time t^+_{out} when its membrane potential exceeds the firing threshold V_{th} ($V_{th} > 0$), and a negative spike is produced only at time t^-_{out} when its membrane potential drops below the firing threshold V_{th} ($V_{th} > 0$); otherwise, no spike is fired during the whole inference. The linear ReLU input-output dynamics are roughly mimicked using the input potential V_{in} and the output spike-times $T(t^+_{out}, t^-_{out})$ of the proposed TSC spiking neuron as illustrated in Fig. 6(b).

$$V_{in} = \frac{1}{N}\sum_{t=1}^{N}\sum_{t=1}^{N}\sum_{i=1}^{m} w_i.T_i(t) \tag{7}$$

4.1 Threshold Balancing for TSC-SNN

Setting threshold too high causes no spike to be fired in the spiking neuron, whereas setting threshold too low causes the spiking neuron to operate in the non-linear regime, both lead to significant accuracy loss in ANN-SNN conversion. The extended linear input-output relationship of TSC spiking neuron (see Fig. 6(b)) provides "wider operating range" for the neuronal firing threshold compared to that for the IF neuron which "hard reset" the membrane potential to 0 irrespective of the amount by which the membrane potential exceeds the threshold. This begets the following couple of questions that need to be answered to ensure appropriate threshold balancing for the TSC spiking neuron.

1. For any given V_{in}, what is the desired operating range for the TSC neuron firing threshold to ensure loss-less ANN-SNN conversion?
2. How should the absolute value of threshold be determined so that the TSC neuron operates in the desired range?

We determine the upper and lower bounds for the TSC neuron firing threshold based on the desired output spike-time $T(t^+_{out}, t^-_{out})$. Figure 6(c) indicates that the output spike time t^-_{out} must be larger than the input spike-time t^+_{in}, and smaller than the total inference time steps N. This is because the membrane potential integration does not start until it receives the first input spike at t^+_{in}, and the negative spike can not be fired if the membrane potential does not drop below the firing threshold by the end of time-step N. The desirable range for t^-_{out} is $[t^-_{in}, N)$. Satisfying $t^-_{out} \leq N$ requires $\eta = \frac{V_{in}}{V_{th}} \leq 1$ or $V_{th} \geq V_{in}$ as highlighted in Fig. 6(c), which ensures the linearity in TSC activation. Using smaller η or larger V_{th} helps to reduce the output spike-time t^-_{out}, which intends to improve the inference latency. However, it also delays the spike time t^+_{out}, which causes more delay to the successive layers. Hence, setting appropriate firing thresholds to balance the spike-times t^+_{out} and t^-_{out} across layers is the key to achieve the best latency in TSC-SNN.

Let us now address the second question concerning the precise V_{th} estimation methodology. In our analysis thus far, we estimated V_{th} using V_{in} as described by Eq. 7, which is the average weighted input sum to the TSC neuron over time. Prior works proposed setting V_{th} to the maximum weighted input sum to the neuron across time-steps [6,37]. Note that, rate encoding was used for the estimation. In this work, we followed a similar procedure to estimate the V^{max}_{in} for each layer of the TSC-SNN. The maximum estimate V^{max}_{in} can enable the TSC neuron to operate in the linear region (where $t^-_{in} < t^-_{out} < N$ as highlighted in Fig. 6(c)). We initialize the thresholds of each layer to the estimated V^{max}_{in} times a scaling factor $\alpha (\alpha \in [0, 1])$. According to our simulation, setting $\alpha \approx 0.8$ helps the TSC-SNN to achieve the best accuracy with minimal latency. Before presenting the results, we describe the methodology, originally proposed in [37], used to initialize the layer-wise threshold of deep SNN using the ANN-trained weights and SNN spiking statistics. We transfer the trained weights from ANN to SNN, and feed the TSC spike-inputs (for the entire training set) to the first layer of the SNN. We record the weighted input sum to all the neurons in the first layer across time-steps. We set the threshold of TSC neurons in the first layer to the maximum weighted input sum, across neurons and time-steps, over the training dataset. We then freeze the threshold of the first layer, and estimate the threshold of the second layer using the same procedure outlined previously. The threshold estimation process is carried out sequentially in a layer-wise manner for all the layers.

ResNet-20 SNN, with its layer-wise threshold assigned to V^{max}_{in}, achieved 91.36% on CIFAR-10 using 1024 time-steps, which is comparable to that

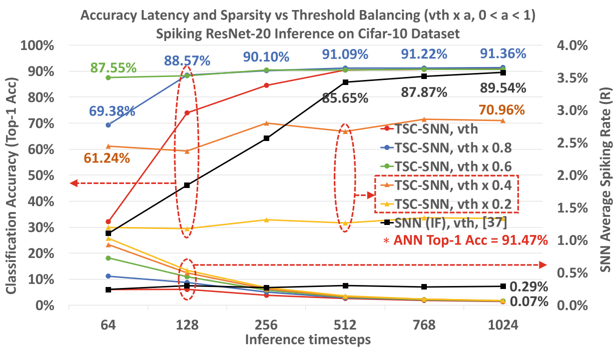


Fig. 7. Threshold balancing for ResNet-20 TSC-SNN inference on CIFAR-10 dataset.

(91.47%) achieved by the corresponding ANN as illustrated in Fig. 7. The x-axis is SNN inference latency, the y-axis on the left measures the SNN top-1 inference accuracy, and the y-axis on the right measures the percentage of neurons those fire spikes per time-step. We thereafter scaled the threshold by a factor of up to 0.6× and found that the TSC-SNN, with scaled threshold, converged to the same accuracy obtained using V_{th}=V_{in}^{max}. This corroborates our hypothesis that the TSC neuron operates in the linear region for a wide range of firing thresholds, thereby, causing the TSC-SNN to yield higher accuracy using fewer time-steps as depicted in Fig. 7. As the threshold is scaled further by up to 0.2×, we notice significant drop in accuracy. At such low thresholds, the TSC neuron operates in the non-linear regime, leading to higher accuracy loss during inference. We propose initializing the threshold of TSC-SNN with scaled version of V_{in}^{max} (scaling factor $\alpha \approx 0.8$ in this example) to achieve the optimal accuracy-latency trade-off. We validate the presented threshold initialization scheme across different SNN architectures and datasets. Improving the inference latency by reducing the firing threshold do not increase the spiking activity in TSC-SNN. In an effort to quantify the spiking activity of TSC-SNN for different thresholds, we measure the average spike rate as defined by the following equation.

$$R = \frac{total\ spikes}{total\ neurons \times\ inference\ time\text{-}steps} \times 100\% \tag{8}$$

The spike rate R in (8) indicates the average percentage of neurons that spike per time-step. Our analysis indicates that the TSC-SNN, with scaled thresholds, provides significant benefits in accuracy and latency with no increase in spiking activity as shown in Fig. 7.

5 Results

We evaluated TSC-SNNs on standard visual object recognition benchmarks, namely the CIFAR-10, CIFAR-100 and ImageNet datasets. We use VGG-16

Table 1. Accuracy loss due to ANN-SNN conversion of the state-of-the-art SNNs on CIFAR-10 dataset

Network Architecture	Spiking Neuron Model	ANN (Top-1 Acc)	SNN (Top-1 Acc)	Accuracy Loss
8-layered [14]	LIF (rate-based)	83.72%	83.54%	0.18%
3-layered [8]	LIF (rate-based)	–	89.32%	–
6-layered [36]	IF (rate-based)	91.91%	90.85%	1.06%
ResNet-20 [37]	IF (rate-based)	89.1%	87.46%	1.64%
ResNet-20 [10]	RMP (rate-based)	91.47%	91.36%	0.11%
ResNet-20 [This work]	**TSC (time-based)**	**91.47%**	**91.42%**	**0.05%**
VGG-16 [37]	IF (rate-based)	91.7%	91.55%	0.15%
VGG-16 [10]	RMP (rate-based)	93.63%	93.63%	<0.01%
VGG-16 [This work]	**TSC (time-based)**	**93.63%**	**93.63%**	**<0.01%**

architecture [39] for all three datasets. ResNet-20 configuration outlined in [12] is used for the CIFAR-10 and CIFAR-100 datasets while ResNet-34 is used for experiments on the ImageNet dataset. Our implementation is derived from the Facebook ResNet implementation code for CIFAR and ImageNet datasets. The code can be found online at https://github.com/facebookarchive/fb.resnet.torch. Proper weight initialization is crucial to achieve convergence in such deep networks without batch-normalization. Similar weights initialization was done as outlined in [11] although their networks were trained without both dropout and batch-normalization. For VGG networks, a dropout layer is used after every ReLU layer except for those layers which are followed by a pooling layer. For Residual networks, we use dropout only for the ReLUs at the non-identity parallel paths but not at the junction layers. We found this to be crucial for achieving training convergence. We found this to be crucial for achieving training convergence. The most recent state-of-the-art ANN-SNN conversion works are provided for comparison as shown in Tables 1, 2 and 3. Note that authors in [36] reported a top-1 SNN error rate of 25.04% for an Inception-V3 network, with their ANN trained to an error rate of 23.88%. The resulting accuracy loss is 1.52% which is much higher than our proposal. The Inception-V3 network conversion was also optimised by a voltage clamping method, that was found to be specific for the Inception network and did not apply to the VGG network [36]. In addition, the results reported on ImageNet in [36] are on a subset of 1382 image samples for Inception-V3 network and 2570 samples for VGG-16 network. Hence, the performance on the entire dataset is unclear. Our proposed TSC-SNN achieved not only the best SNN inference accuracy but also the lowest accuracy loss in ANN-SNN conversion across all network architectures and datasets we evaluated.

The VGG-16 TSC-SNN inference performance on CIFAR-10, CIFAR-100 and ImageNet datasets are shown in Fig. 8(a) (c) and (e). In each figure, x-axis is the SNN inference latency, the y-axis on the left measures the SNN top-1 inference accuracy, and the y-axis on the right measures the percentage of

Table 2. Accuracy loss due to ANN-SNN conversion of the state-of-the-art SNNs on CIFAR-100 dataset

Network Architecture	Spiking Neuron Model	ANN (Top-1 Acc)	SNN (Top-1 Acc)	Accuracy Loss
ResNet-20 [37]	IF (rate-based)	68.72%	64.09%	4.63%
ResNet-20 [10]	RMP (rate-based)	68.72%	67.82%	0.9%
ResNet-20 [This work]	**TSC (time-based)**	**68.72%**	**68.18%**	**0.54%**
VGG-16 [37]	IF (rate-based)	71.22%	70.77%	0.45%
VGG-16 [10]	RMP (rate-based)	71.22%	70.93%	0.29%
VGG-16 [This work]	**TSC (time-based)**	**71.22%**	**70.97%**	**0.25%**

Table 3. Accuracy loss due to ANN-SNN conversion of the state-of-the-art SNNs on ImageNet dataset

Network Architecture	Spiking Neuron Model	ANN (Top-1 Acc)	SNN (Top-1 Acc)	Accuracy Loss
ResNet-34 [37]	IF (rate-based)	70.69%	65.47%	5.22%
ResNet-34 [10]	RMP (rate-based)	70.64%	69.89%	0.75%
ResNet-34 [This work]	**TSC (time-based)**	**70.64%**	**69.93%**	**0.71%**
VGG-16 [36]	RMP (rate-based)	63.89%	49.61%	14.28%
VGG-16 [37]	IF (rate-based)	70.52%	69.96%	0.56%
VGG-16 [10]	RMP (rate-based)	73.49%	73.09%	0.4%
VGG-16 [This work]	**TSC (time-based)**	**73.49%**	**73.46%**	**0.03%**

neurons those fire spikes per time-step. As shown in Fig. 8 (a), TSC-SNN (green curve) achieved the same accuracy 93.63% as the trained ANN, whereas the SNN with IF neurons achieved 93.50% using 2048 time-steps. TSC-SNN reaches an accuracy 92.79% using only 64 time-steps, which is 3 times faster than the RMP-SNN (blue curve) that uses about 200 time-steps, and 10 times faster than the SNN with IF neurons (black curve) that uses about 640 time-steps. The TSC-SNN attains a spike rate around 0.03%, which is 66.3 times lower than the RMP-SNN, and 20.3 times lower than the SNN with IF neuron. As mentioned above, lower spiking rate does not guarantee low computation in temporal SNNs. Hence, the number of addition operations performed in SNNs inference are also provided in Fig. 8(b) (d) and (f). In each figure, x-axis is the SNN inference latency, the y-axis measures the number of addition operations performed for computing the weighted sum of spike-input and updating the membrane potential in SNN inference. As shown in Fig. 8(b). The proposed TSC-SNN reduces the number of addition computation by one order of magnitude than SNN with IF neuron in [37], and two orders of magnitude than the RMP-SNN in [10]. The VGG-16 TSC-SNN inference performance on CIFAR-100 and ImageNet datasets are shown in Fig. 8 (c) (d) and Fig. 8 (e) (f) respectively. Note, no VGG-16 SNN

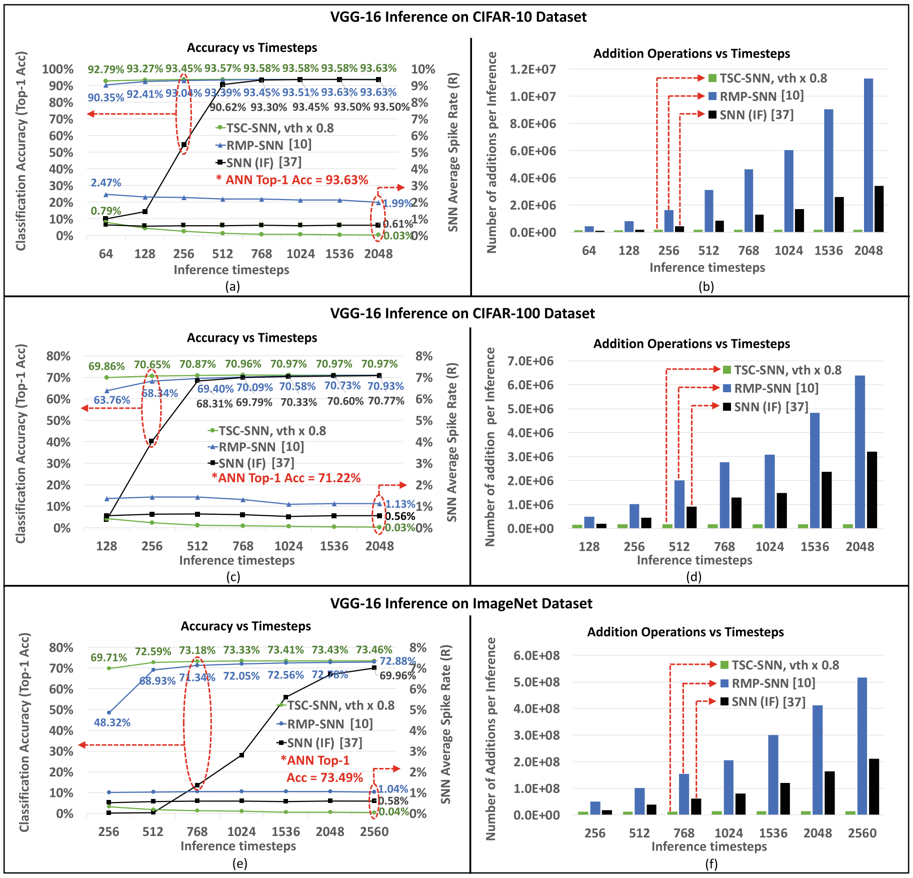


Fig. 8. Inference accuracy ((a) (c) and (e)) and computational cost ((b) (d) and (f)) comparisons between TSC-SNN and the two baseline SNNs (RMP-SNN [10] and SNN(IF) [37]) using VGG-16 architecture on CIFAR-10, CIFAR-100 and ImageNet datasets.

was evaluated on CIFAR-100 dataset in [37]. In this work, the results of VGG-16 on CIFAR-100 using the TSC-SNN, RMP-SNN and the baseline SNN with IF neurons were converted from our trained ANN with top-1 inference accuracy of 71.22%. The ResNet-20 and ResNet-34 TSC-SNNs inference performance on the CIFAR-10, CIFAR-100 and ImageNet datasets are also provided in Fig. 9.

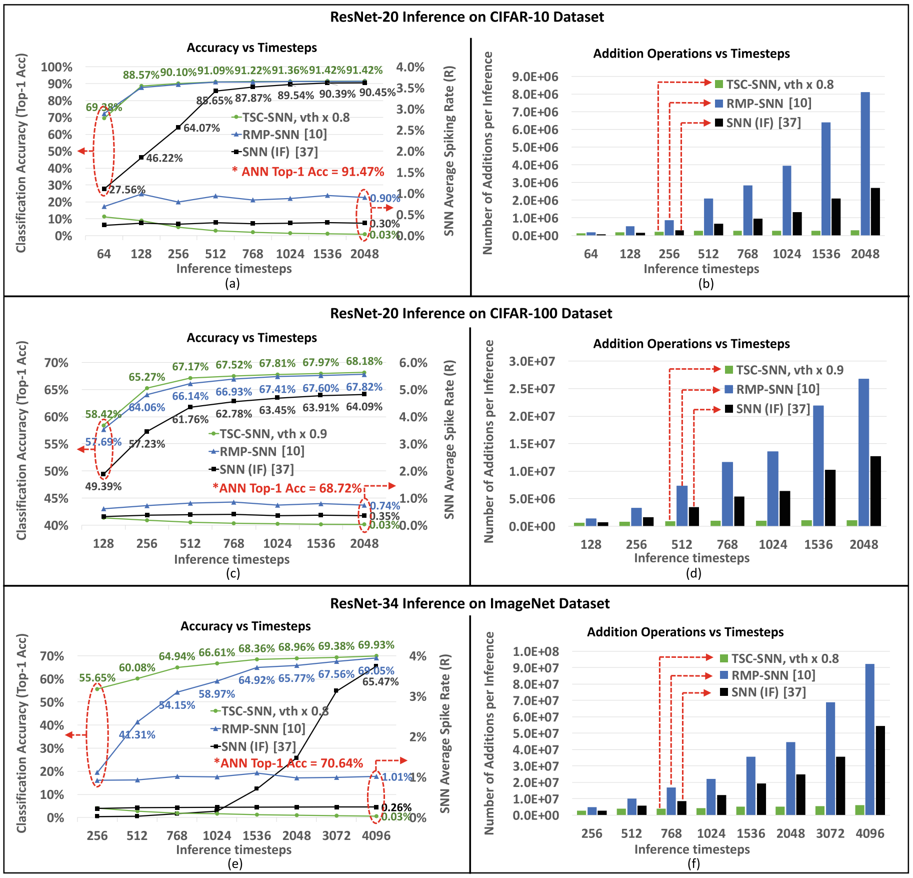


Fig. 9. Inference accuracy ((a) (c) and (e)) and computational cost ((b) (d) and (f)) comparisons between TSC-SNN and the two baseline SNNs (RMP-SNN [10] and SNN(IF) [37]) using ResNet architectures on CIFAR-10, CIFAR-100 and ImageNet datasets.

6 Conclusion and Discussion

In this work, we propose an ANN to SNN conversion technique. It uses a novel time-based coding scheme (TSC) and TSC spiking neuron model. We also propose a threshold balancing technique which alleviates the ANN-SNN conversion accuracy loss and significantly improved the latency and scalability of TSC-SNNs to deep architectures. We implemented large scale deep network architectures such as VGG and Residual networks using the proposed conversion based training and evaluated performance on cifar-10, cifar-100 and ImageNet datasets. Our proposed TSC-SNNs achieve the best accuracies and latencies, the lowest accuracy loss and the lowest computational cost than the state-of-the-art across all network architectures and datasets we tested.

Acknowledgment. This work was supported in part by Center for Brain-Inspired Computing (C-BRIC), a MARCO and DARPA sponsored StarNet center, by the Semiconductor Research Corporation, National Science Foundation, Sandia National Laboratories, Vannevar Bush Faculty Fellowship and by the US Army Research Laboratory and the UK Ministry of Defense under Agreement Number W911NF-16-3-0001.

References

1. Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., Maass, W.: Long short-term memory and learning-to-learn in networks of spiking neurons. In: Advances in Neural Information Processing Systems, pp. 787–797. Montréal, Quebec, Canada (2018)
2. Blouw, P., Choo, X., Hunsberger, E., Eliasmith, C.: Benchmarking keyword spotting efficiency on neuromorphic hardware. In: Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop, p. 1. ACM (2019)
3. Cao, Y., Chen, Y., Khosla, D.: Spiking deep convolutional neural networks for energy-efficient object recognition. Int. J. Comput. Vis. **113**(1), 54–66 (2015). https://doi.org/10.1007/s11263-014-0788-3
4. Davies, M., et al.: Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro **38**(1), 82–99 (2018)
5. Diehl, P.U., Cook, M.: Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Front. Comput. Neurosci. **9**, 99 (2015)
6. Diehl, P.U., Neil, D., Binas, J., Cook, M., Liu, S.C., Pfeiffer, M.: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2015)
7. Diehl, P.U., Zarrella, G., Cassidy, A., Pedroni, B.U., Neftci, E.: Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware. In: 2016 IEEE International Conference on Rebooting Computing (ICRC), pp. 1–8. IEEE (2016)
8. Esser, S.K., et al.: Convolutional networks for fast, energy-efficient neuromorphic computing. CoRR abs/1603.08270 (2016). http://arxiv.org/abs/1603.08270
9. Ferré, P., Mamalet, F., Thorpe, S.J.: Unsupervised feature learning with winner-takes-all based STDP. Front. Comput. Neurosci. **12**, 24 (2018)
10. Han, B., Srinivasan, G., Roy, K.: 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 (CVPR), June 2020
11. Hardt, M., Ma, T.: Identity matters in deep learning. CoRR abs/1611.04231. http://arxiv.org/abs/1611.04231 (2016)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR abs/1512.03385. http://arxiv.org/abs/1512.03385 (2015)
13. Heeger, D.: Poisson model of spike generation. Stanford Univ. Handout **5**, 1–13 (2000)
14. Hunsberger, E., Eliasmith, C.: Training spiking deep networks for neuromorphic hardware. CoRR abs/1611.05141. http://arxiv.org/abs/1611.05141 (2016)
15. Jin, Y., Zhang, W., Li, P.: Hybrid macro/micro level backpropagation for training deep spiking neural networks. In: Advances in Neural Information Processing Systems, pp. 7005–7015. Montréal, Quebec, Canada (2018)

16. Johnson, M., et al.: Google's multilingual neural machine translation system: enabling zero-shot translation. Trans. Assoc. Comput. Linguit. **5**, 339–351 (2017)
17. Kheradpisheh, S.R., Ganjtabesh, M., Thorpe, S.J., Masquelier, T.: STDP-based spiking deep convolutional neural networks for object recognition. Neural Netw. **99**, 56–67 (2018). https://doi.org/10.1016/j.neunet.2017.12.005. http://www.sciencedirect.com/science/article/pii/S0893608017302903
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
19. Lee, C., Srinivasan, G., Panda, P., Roy, K.: Deep spiking convolutional neural network trained with unsupervised spike timing dependent plasticity. IEEE Trans. Cogn. Dev. Syst. pp. 1–1 (2018). https://doi.org/10.1109/TCDS.2018.2833071
20. Lee, C., Sarwar, S.S., Panda, P., Srinivasan, G., Roy, K.: Enabling spike-based backpropagation for training deep neural network architectures. Front. Neurosci. **14**, 119 (2020). https://doi.org/10.3389/fnins.2020.00119
21. Lee, J.H., Delbruck, T., Pfeiffer, M.: Training deep spiking neural networks using backpropagation. Front. Neurosci. **10**, 508 (2016)
22. Maass, W.: Networks of spiking neurons: the third generation of neural network models. Neural Netw. **10**(9), 1659–1671 (1997)
23. Masquelier, T., Thorpe, S.J.: Unsupervised learning of visual features through spike timing dependent plasticity. PLoS Comput. Biol. **3**(2), e31 (2007)
24. Merolla, P.A., et al.: A million spiking-neuron integrated circuit with a scalable communication network and interface. Science **345**(6197), 668–673 (2014)
25. Miyashita, D., Lee, E.H., Murmann, B.: Convolutional neural networks using logarithmic data representation. CoRR abs/1603.01025. http://arxiv.org/abs/1603.01025 (2016)
26. Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., Thorpe, S.J., Masquelier, T.: Combining STDP and reward-modulated STDP in deep convolutional spiking neural networks for digit recognition. arXiv preprint arXiv:1804.00227 (2018)
27. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 807–814 (2010)
28. Neftci, E.O., Mostafa, H., Zenke, F.: Surrogate gradient learning in spiking neural networks. arXiv preprint arXiv:1901.09948 (2019)
29. Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., Ng, A.Y.: Multimodal deep learning. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 689–696 (2011)
30. Panda, P., Roy, K.: Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition. In: 2016 International Joint Conference on Neural Networks (IJCNN), pp. 299–306. IEEE, Vancouver, British Columbia, Canada (2016)
31. Pérez-Carrasco, J.A., 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. **35**(11), 2706–2719 (2013)
32. Rathi, N., Srinivasan, G., Panda, P., Roy, K.: Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In: International Conference on Learning Representations. https://openreview.net/forum?id=B1xSperKvH (2020)

33. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
34. Rueckauer, B., Liu, S.: Conversion of analog to spiking neural networks using sparse temporal coding. In: 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5 (2018)
35. Rueckauer, B., Lungu, I.A., Hu, Y., Pfeiffer, M.: Theory and tools for the conversion of analog to spiking convolutional neural networks. arXiv preprint arXiv:1612.04052 (2016)
36. Rueckauer, B., Lungu, I.A., Hu, Y., Pfeiffer, M., Liu, S.C.: Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. Front. Neurosci. **11**, 682 (2017)
37. Sengupta, A., Ye, Y., Wang, R., Liu, C., Roy, K.: Going deeper in spiking neural networks: VGG and residual architectures. Front. Neurosci. **13**, 95 (2019)
38. Shrestha, S.B., Orchard, G.: Slayer: spike layer error reassignment in time. In: Advances in Neural Information Processing Systems, pp. 1412–1421. Montréal, Quebec, Canada (2018)
39. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
40. Srinivasan, G., Panda, P., Roy, K.: STDP-based unsupervised feature learning using convolution-over-time in spiking neural networks for energy-efficient neuromorphic computing. J. Emerg. Technol. Comput. Syst. **14**(4), 1–12 (2018). https://doi.org/10.1145/3266229. https://doi.org/10.1145/3266229
41. Srinivasan, G., Roy, K.: ReStoCNet: residual stochastic binary convolutional spiking neural network for memory-efficient neuromorphic computing. Front. Neurosci. **13**, 189 (2019)
42. Tavanaei, A., Kirby, Z., Maida, A.S.: Training spiking convnets by STDP and gradient descent. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. Rio de Janeiro, Brazil, July 2018. https://doi.org/10.1109/IJCNN.2018.8489104
43. Thiele, J.C., Bichler, O., Dupret, A.: Event-based, timescale invariant unsupervised online deep learning with STDP. Front. Comput. Neurosci. **12**, 46 (2018). https://doi.org/10.3389/fncom.2018.00046. https://www.frontiersin.org/article/10.3389/fncom.2018.00046
44. Wu, Y., Deng, L., Li, G., Zhu, J., Shi, L.: Spatio-temporal backpropagation for training high-performance spiking neural networks. Front. Neurosci. **12**, 331 (2018)
45. Zambrano, D., Nusselder, R., Scholte, H.S., Bohte, S.M.: Efficient computation in adaptive artificial spiking neural networks. CoRR abs/1710.04838. http://arxiv.org/abs/1710.04838 (2017)
46. Zhang, M., Zheng, N., Ma, D., Pan, G., Gu, Z.: Efficient spiking neural networks with logarithmic temporal coding. CoRR abs/1811.04233. http://arxiv.org/abs/1811.04233 (2018)
47. Zhao, B., Ding, R., Chen, S., Linares-Barranco, B., Tang, H.: Feedforward categorization on AER motion events using cortex-like features in a spiking neural network. IEEE Trans. Neural Netw. Learn. Syst. **26**(9), 1963–1978 (2014)