frontiers in Neuroscience

ORIGINAL RESEARCH
published: 19 March 2019
doi: 10.3389/fnins.2019.00189

ReStoCNet: Residual Stochastic Binary Convolutional Spiking Neural Network for Memory-Efficient Neuromorphic Computing

Gopalakrishnan Srinivasan and Kaushik Roy*

Department of ECE, Purdue University, West Lafayette, IN, United States

In this work, we propose ReStoCNet, a residual stochastic multilayer convolutional Spiking Neural Network (SNN) composed of binary kernels, to reduce the synaptic memory footprint and enhance the computational efficiency of SNNs for complex pattern recognition tasks. ReStoCNet consists of an input layer followed by stacked convolutional layers for hierarchical input feature extraction, pooling layers for dimensionality reduction, and fully-connected layer for inference. In addition, we introduce residual connections between the stacked convolutional layers to improve the hierarchical feature learning capability of deep SNNs. We propose Spike Timing Dependent Plasticity (STDP) based probabilistic learning algorithm, referred to as Hybrid-STDP (HB-STDP), incorporating Hebbian and anti-Hebbian learning mechanisms, to train the binary kernels forming ReStoCNet in a layer-wise unsupervised manner. We demonstrate the efficacy of ReStoCNet and the presented HB-STDP based unsupervised training methodology on the MNIST and CIFAR-10 datasets. We show that residual connections enable the deeper convolutional layers to self-learn useful high-level input features and mitigate the accuracy loss observed in deep SNNs devoid of residual connections. The proposed ReStoCNet offers $>20\times$ kernel memory compression compared to full-precision (32-bit) SNN while yielding high enough classification accuracy on the chosen pattern recognition tasks.

Keywords: convolutional SNN, spiking ResNet, binary kernels, probabilistic STDP, unsupervised feature learning

OPEN ACCESS

Edited by:
*Yansong Chua, Institute for Infocomm Research (A*STAR), Singapore*

Reviewed by:
Timothée Masquelier, Centre National de la Recherche Scientifique (CNRS), France
Andrew Rowley, University of Manchester, United Kingdom

****Correspondence:***
Gopalakrishnan Srinivasan
srinivg@purdue.edu

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *29 November 2018*
Accepted: *18 February 2019*
Published: *19 March 2019*

Citation:
Srinivasan G and Roy K (2019) ReStoCNet: Residual Stochastic Binary Convolutional Spiking Neural Network for Memory-Efficient Neuromorphic Computing. Front. Neurosci. 13:189. doi: 10.3389/fnins.2019.00189

1. INTRODUCTION

The proliferation in real-time content generated by the ubiquitous battery-powered edge devices necessitates a paradigm shift in neural architectures to enable energy-efficient neuromorphic computing. Spiking Neural Networks (SNNs) offer a promising alternative toward realizing intelligent neuromorphic systems that require lower computational effort than the artificial neural networks. SNNs encode and communicate information in the form of sparse spiking events. The intrinsic sparse event-driven processing capability, which entails neuronal computations and synaptic weight updates only in the event of a spike fired by the constituting neurons, leads to improved energy efficiency in neuromorphic hardware implementations (Sengupta et al., 2019). Spike Timing Dependent Plasticity (STDP) (Bi and Poo, 1998) is a localized hardware-friendly plasticity mechanism used for unsupervised learning in SNNs. STDP-based learning rules (Song et al., 2000) modify the weight of a synapse interconnecting a pair of input (pre) and output (post) neurons depending on the degree of correlation between the respective spike times. The spike timing information is encoded in the bit-precision of the synaptic weight. In an effort to reduce the synaptic memory footprint, Suri et al. (2013), Querlioz et al. (2015), and Srinivasan et al. (2016) proposed two-layer fully-connected SNN composed of binary synaptic weights. The fully-connected SNN learns complete input representations rather than distinctive features making up the input patterns. As a result, it requires large number of trainable parameters to attain competitive classification accuracy (Diehl and Cook, 2015), which negatively impacts the scalability of such shallow SNNs for complex pattern recognition tasks.

We propose deep Residual Stochastic Binary Convolutional Spiking Neural Network, referred to as *ReStoCNet*, as a scalable architecture to achieve improved classification accuracy with compressed synaptic memory. ReStoCNet consists of an input layer followed by stacked convolutional layers with Leaky-Integrate-and-Fire (LIF) spiking non-linearity (Dayan and Abbott, 2001) for hierarchical input feature extraction, spatial pooling layers for dimensionality reduction, and one or more fully-connected layers for inference. We introduce residual or shortcut connections between the stacked convolutional layers, inspired by the organization of deep residual networks (He et al., 2016), in order to improve the representations learnt by the later convolutional layers. In addition, we enforce binary synaptic weights for the convolutional kernels during both training and inference. We propose STDP-based probabilistic learning rule, referred to as Hybrid-STDP (HB-STDP), incorporating Hebbian and anti-Hebbian learning mechanisms to train the binary kernels. Based on HB-STDP, a binary synaptic weight is probabilistically potentiated for small positive time difference between excitatory pre- and post-spikes, which is in agreement with the Hebbian learning theory (Hebb, 1949). On the other hand, it is probabilistically depressed for large positive time difference (anti-Hebbian in nature) or small negative time difference (Hebbian in nature) between the respective spikes. The spike timing information is essentially encoded in the synaptic switching probability, which is held constant within the Hebbian potentiation, Hebbian depression, and anti-Hebbian depression windows, and is zero elsewhere. We note that Suri et al. (2013) proposed an STDP-based learning rule employing constant switching probabilities, where the potentiation and depression windows extend over the entire STDP timing window. On the contrary, HB-STDP contains dead zone in the STDP timing window, where the switching probability is zero. We visually demonstrate the significance of dead zone for efficient feature learning using binary fully-connected SNN.

We present HB-STDP based layer-wise unsupervised training methodology for ReStoCNet, where we train the binary kernels interconnecting successive convolutional layers using HB-STDP. Once a given layer is trained, we forward propagate the spikes from the input through the trained layers and update the binary kernels of the following convolutional layer. After all the convolutional layers are trained, we feed the input dataset, estimate the spiking activations of the spatially pooled convolutional spike maps by accumulating the spikes at every time instant and decaying the resultant sum between successive spike timing instants, and pass them on to the fully-connected layer, trained using error backpropagation (Rumelhart et al., 1986), for inference. We validate the efficacy of ReStoCNet and the HB-STDP based unsupervised training methodology on the MNIST (LeCun et al., 1998) and CIFAR-10 datasets (Krizhevsky, 2009). We show that residual connections enable the deeper convolutional layers to extract useful high-level input features and effectively mitigate the accuracy degradation observed in deep SNNs devoid of residual connections (Lee et al., 2018b). We note that Masquelier and Thorpe (2007), Panda and Roy (2016), Lee et al. (2016), Stromatias et al. (2017), Srinivasan et al. (2018), Tavanaei et al. (2018), Kheradpisheh et al. (2018), Ferré et al. (2018), Thiele et al. (2018), Lee et al. (2018a,b), and Mozafari et al. (2018) have demonstrated convolutional SNNs composed of full-precision kernels. Recently, Sengupta et al. (2019) and Hu et al. (2018) presented residual SNNs, trained using error backpropagation with real-valued inputs and artificial ReLU neurons (Nair and Hinton, 2010), which are mapped to spiking neurons post training for energy-efficient inference. To the best of our knowledge, ReStoCNet is the first demonstration of STDP-trained deep residual convolutional SNN composed of binary kernels for complex pattern recognition tasks. We believe that ReStoCNet, with event-driven computing capability and memory-efficient learning with binary kernels trained using hardware-friendly probabilistic-STDP learning rule, offers a promising alternative for energy-efficient neuromorphic computing in battery-powered edge devices. Overall, the key contributions of our work are:

1. We propose ReStoCNet, a deep residual convolutional SNN composed of binary kernels, for memory-efficient neuromorphic computing.
2. We present HB-STDP, an STDP-based probabilistic learning rule incorporating Hebbian and anti-Hebbian learning mechanisms, for training the binary kernels constituting ReStoCNet in a layer-wise unsupervised manner for hierarchical input feature extraction.
3. We validate the efficacy of ReStoCNet on the MNIST and CIFAR-10 datasets, and show that residual connections enable the deeper convolutional layers to learn useful high-level input features and mitigate the accuracy loss incurred by STDP-trained deep SNNs without residual connections.

2. MATERIALS AND METHODS

2.1. ReStoCNet: Residual Stochastic Binary Convolutional Spiking Neural Network

ReStoCNet consists of an input layer followed by stacked convolutional layers for hierarchical input feature extraction, spatial pooling layers for dimensionality reduction, and one or more fully-connected layers for inference as illustrated in **Figure 1**. The pixels in the input image maps are converted to Poisson spike trains firing at a rate proportional to the corresponding pixel intensities. At any given time, the input spike maps are convolved with the binary kernels, which are constrained to logic states -1 (w_{low}) and $+1$ (w_{high}), to produce the convolutional output maps. The convolutional

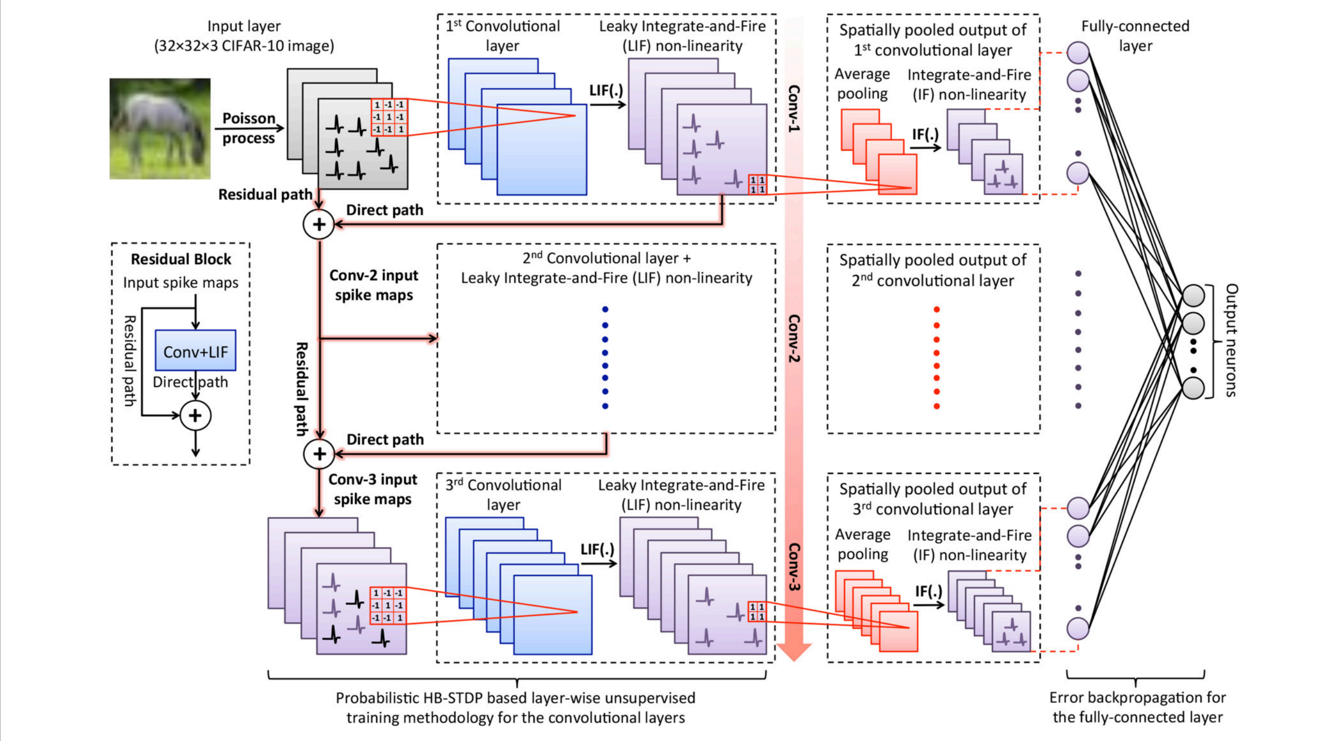
Input layer
(32×32×3 CIFAR-10 image)
Poisson process
Residual path
Direct path
1st Convolutional layer
Leaky Integrate-and-Fire (LIF) non-linearity
LIF(.)
Conv-1
Spatially pooled output of 1st convolutional layer
Average pooling
Integrate-and-Fire (IF) non-linearity
IF(.)
Fully-connected layer
Residual Block
Input spike maps
Conv+LIF
Residual path
Direct path
Conv-2 input spike maps
2nd Convolutional layer + Leaky Integrate-and-Fire (LIF) non-linearity
Conv-2
Spatially pooled output of 2nd convolutional layer
Output neurons
Conv-3 input spike maps
3rd Convolutional layer
Leaky Integrate-and-Fire (LIF) non-linearity
Conv-3
Spatially pooled output of 3rd convolutional layer
Probabilistic HB-STDP based layer-wise unsupervised training methodology for the convolutional layers
Error backpropagation for the fully-connected layer

FIGURE 1 | Illustration of ReStoCNet consisting of an input layer followed by stacked convolutional layers with Leaky-Integrate-and-Fire (LIF) spiking non-linearity, which are interconnected via binary kernels. The deeper convolutional layers receive residual inputs that are summed up with direct inputs from the preceding convolutional layer as depicted in the inset. The binary kernels forming the convolutional layers are trained using probabilistic Hybrid-STDP (HB-STDP) based layer-wise unsupervised training methodology. After all the convolutional layers are trained, the respective spike maps are spatially pooled using average pooling with 2×2 unit-weight kernels followed by Integrate-and-Fire (IF) spiking non-linearity to produce the pooled spike maps. The spike trains of the pooling layers are low-pass filtered to obtain their spiking activations over the time period for which the input is presented, which are fed to the fully-connected layer, trained using error backpropagation, for inference.

outputs, referred to as post-synaptic currents, are fed to non-linear layer of Leaky-Integrate-and-Fire (LIF) spiking neurons (Dayan and Abbott, 2001). An LIF neuron integrates the post-synaptic current into its membrane potential, whose dynamics are described by

$$\tau_{mem}\frac{dV_{mem}}{dt} = -V_{mem} + I_{post} \quad (1)$$

where V_{mem} is the neuronal membrane potential, τ_{mem} is the membrane potential leak time constant, and I_{post} is the post-synaptic current. The LIF neuron emits a spike when its membrane potential exceeds a definite firing threshold after which the membrane potential is reset to zero. Every convolutional output map yields a corresponding spike map based on the LIF spiking neuronal dynamics, which is directly fed to the following convolutional layer. In addition, we introduce residual connections feeding into the deeper convolutional layers, which is inspired by the architecture of deep residual networks (He et al., 2016). The second convolutional layer receives residual connections from the input layer while the third convolutional layer receives residual connections from the input and first convolutional layer as shown in **Figure 1**. The residual connections feeding into a target convolutional layer perform identity mapping, i.e., the residual path spike maps are simply added to the direct path spike maps from the preceding convolutional layer and fed to the target convolutional layer. In the event of a mismatch in the number of spike maps (or channels) between the residual and direct paths, the spike maps in the residual path are replicated to be consistent with the number of channels in the direct path. Consider, for instance, the second convolutional layer that receives spike maps from the input layer via the residual path and the first convolutional layer via the direct path. Let us suppose that the input image pattern is stored in RGB colorspace. Consequently, each image pattern yields 3 input spike maps that needs to be summed up with the spike maps of the first convolutional layer, which typically contains more than 3 spike maps. Hence, the 3 input spike maps are replicated to match the number of spike maps in the first convolutional layer, summed up with the spike maps of the first convolutional layer, and fed to the second convolutional layer. Note that the summed spike maps from the residual and direct paths are constrained to unit magnitude to produce resultant spike maps feeding into the target convolutional layer. The binary kernels constituting the convolutional layers are trained using probabilistic Hybrid-STDP (HB-STDP) based layer-wise unsupervised training methodology. We find that the residual connections ensure rich and diverse inputs for deeper convolutional layers and enable them to self-learn useful high-level input features as shown in subsection 3.3. The improved feature learning capability mitigates the accuracy loss incurred by stacked convolutional layers without residual connections as experimentally validated in subsection 3.3 and enhances the scalability of deep SNNs.

After all the convolutional layers are trained, we feed the input dataset and spatially pool the spike maps of the convolutional layers. Spatial pooling is the mechanism used to suitably combine the neighboring pixels of a convolutional feature map to reduce the map size (height and width) while retaining the salient features. Spatial pooling also renders the network invariant to slight translations in the input features (Jaderberg et al., 2015). We perform a class of spatial pooling operation known as average pooling with 2×2 kernels composed of unit weights and stride length of 2 as detailed below. The spikes in every 2×2 non-overlapping region of the convolutional maps are summed up and normalized by the kernel size (4 for a 2×2 kernel) to produce the pooled output maps, which are then fed to a layer of Integrate-and-Fire (IF) spiking neurons to generate the pooled spike maps. An IF neuron integrates the input into its membrane potential and spikes if the membrane potential exceeds pre-specified threshold (θ_{pool}) after which the membrane potential is reset. The IF neurons, in effect, fire based on the average spiking activity of the spatially pooled convolutional spike maps. We low-pass filter the spike trains of the pooled maps by integrating the spikes at every time instant and decaying the resultant sum between successive spike timing instants to estimate their spiking activations over the time period for which the input is presented. The spiking activations of the pooled maps pertaining to all the convolutional layers are fed to the fully-connected layer composed of ReLU neurons (Nair and Hinton, 2010) for inference. This ensures that the input features learnt independently by the convolutional layers in an unsupervised manner are combined optimally by the fully-connected layer to yield the best accuracy. We note that LIF neurons can instead be used in the fully-connected layer, which can be trained using spike-based backpropagation algorithms (Lee et al., 2016, 2018a; Panda and Roy, 2016; Jin et al., 2018; Wu et al., 2018). In this work, we use fully-connected layer of ReLU neurons trained with backpropagation algorithm commonly used for deep learning networks since we are primarily interested in evaluating the efficacy of the proposed probabilistic HB-STDP based unsupervised training methodology for the convolutional layers that is detailed in the following subsection.

2.2. Hybrid-STDP (HB-STDP) for Binary Synaptic Weights

We propose STDP-based probabilistic learning rule, referred to as Hybrid-STDP (HB-STDP), integrating Hebbian and anti-Hebbian learning mechanisms to train the binary synaptic weights constituting an SNN. We present two versions of the HB-STDP learning rule, namely, excitatory HB-STDP (eHB-STDP) and inhibitory HB-STDP (iHB-STDP) to train the binary synaptic weights connecting excitatory and inhibitory pre-neurons, respectively, to excitatory post-neurons. An excitatory neuron is modeled as a neuron firing unit positive spikes while an inhibitory neuron fires unit negative spikes. Input image pixels with intensities ranging from 0 to 255 are mapped to excitatory pre-neurons firing unit positive spikes at a rate proportional to the respective pixel intensities. On the contrary, input images when pre-processed by normalizing the raw pixel intensities to zero mean and unit variance result in normalized images with positive and negative pixel intensities. The normalized pixels with negative intensities are mapped to inhibitory pre-neurons firing unit negative spikes. The normalized input maps containing excitatory and inhibitory pre-neurons offer richer spike-encoding of the image patterns, resulting in efficient STDP-based feature learning. We find that input normalization is critical for natural images like those from the CIFAR-10 dataset (Krizhevsky, 2009) that do not have clear separation between the region of interest and the background unlike digit patterns from the MNIST dataset (LeCun et al., 1998).

Binary synapses require a probabilistic learning rule to prevent rapid switching of the weights between the allowed levels, which could otherwise render the synapses memoryless. Both the proposed eHB-STDP and iHB-STDP learning rules map the time difference between a pair of pre- and post-spikes to the switching probability of the interconnecting binary synapse. We first detail the eHB-STDP learning rule for excitatory pre-neurons and subsequently discuss how the learning dynamics are adapted for inhibitory pre-neurons. According to eHB-STDP, if an excitatory pre-spike (at time instant, t_{pre}) triggers the post-neuron to fire (at time instant, t_{post}) and the difference between the respective spike times ($\Delta t = t_{post} - t_{pre}$) is smaller than a pre-specified time period (t_{Hebb_pot}), we switch the synapse from low to high ('L'→'H') state with a constant probability, p_{Hebb_pot}, as illustrated in **Figure 2A** and described by

$$P_{L\to H} = \begin{cases} p_{Hebb\_pot}, & \text{if } 0 < \Delta t \leq t_{Hebb\_pot} \\ 0, & \text{for all other } \Delta t \end{cases} \tag{2}$$

where $P_{L\to H}$ is the probability of synaptic potentiation. Probabilistic synaptic potentiation is carried out for small time difference between causally related pre- and post-spikes following the Hebbian learning principle that can be summarized as "*neurons that fire together, must wire together*" (Lowel and Singer, 1992). Hence, the corresponding timing window is designated as the *Hebbian potentiation* window. On the other hand, probabilistic synaptic depression is carried out for large positive or small negative time difference between the pre- and post-spikes as specified by

$$P_{H\to L} = \begin{cases} p_{antiHebb\_dep}, & \text{if } \Delta t > 0 \cap \Delta t \geq t_{antiHebb\_dep} \\ p_{Hebb\_dep}, & \text{if } t_{Hebb\_dep} \leq \Delta t \leq 0 \\ 0, & \text{for all other } \Delta t \end{cases} \tag{3}$$

where $P_{H\to L}$ is the probability of synaptic depression. We depress the synapse from high to low state with a constant probability, $p_{antiHebb\_dep}$, if the time difference between causally related pre- and post-spikes is larger than $t_{antiHebb\_dep}$, which is anti-Hebbian in nature. Hence, the corresponding STDP

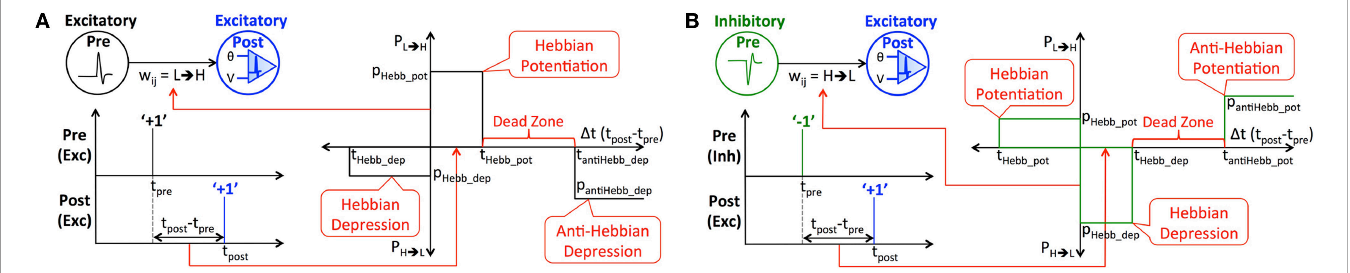

FIGURE 2 | (A) Illustration of eHB-STDP, an STDP-based probabilistic learning rule, incorporating Hebbian and anti-Hebbian learning mechanisms, for training the binary synaptic weights interconnecting excitatory pre- and post-neurons firing positive spikes. The synaptic weight is probabilistically potentiated for small positive time difference (Hebbian in nature) while it is probabilistically depressed for large positive (anti-Hebbian in nature) or small negative time difference (Hebbian in nature) between the pre- and post-spikes. The switching probability is held constant within the Hebbian potentiation, Hebbian depression, and anti-Hebbian depression windows, and is zero in the dead zone. **(B)** Illustration of iHB-STDP for binary synaptic weights connecting inhibitory pre-neurons firing negative spikes to excitatory post-neurons. The iHB-STDP dynamics are obtained by mirroring the eHB-STDP dynamics about the Δt ($t_{post} - t_{pre}$) axis.

timing window is referred to as the *anti-Hebbian depression* window. Anti-Hebbian depression enables the synapses to unlearn features lying outside the neuronal receptive field like noisy background in image patterns. Synaptic depression, in addition, is carried out with a probability, p_{Hebb_dep}, if a pre-spike follows a post-spike and the difference between the respective spike times lies within the negative *Hebbian depression* ($[t_{Hebb\_dep}, 0]$) window. It is important to note that eHB-STDP contains a dead zone in the STDP timing window, where the switching probability is zero, between the Hebbian potentiation and anti-Hebbian depression windows as depicted in **Figure 2A**. We find that expanding the anti-Hebbian depression window toward the Hebbian potentiation window leads to depression of moderately correlated features in addition to the weakly correlated ones. On the other hand, expanding the Hebbian potentiation window causes the synapses connecting a post-neuron to encode multiple overlapping input features, which negatively impacts the selectivity of the post-neuron and degrades the inference capability of the SNN. The dead zone, in effect, ensures that binary synapses learn and retain strongly correlated input features and unlearn only the weakly correlated ones by facilitating optimal balance between the potentiation and depression updates. We visually demonstrate the significance of dead zone for efficient feature learning using binary fully-connected SNN in subsection 3.1.

Next, we discuss how the eHB-STDP dynamics are adapted for binary synapses connecting inhibitory pre-neurons firing negative spikes. The iHB-STDP dynamics (shown in **Figure 2B**) are obtained by symmetrically inverting the eHB-STDP dynamics (shown in **Figure 2A**) about the Δt ($t_{post} - t_{pre}$) axis. As a result, the erstwhile potentiation windows are converted to depression windows, and vice versa. According to iHB-STDP, if an inhibitory pre-spike causes the post-neuron to fire and the spike timing difference is smaller than a pre-specified time period, we probabilistically depress the binary synaptic weight. This ensures that the strongly correlated inhibitory (negative) pre-spike modulated by the depressed synaptic weight causes an effective increase in the post-neuronal membrane potential, thereby improving the chances of a post-spike at subsequent time instants. Probabilistic synaptic depression enables a post-neuron to integrate the small positive time difference between an inhibitory pre-spike and the ensuing post-spike, which conforms to the Hebbian learning theory. Probabilistic synaptic potentiation, on the other hand, causes an inhibitory pre-spike modulated by the synaptic weight to lower the post-neuronal membrane potential, thus reducing the chances of a post-spike at subsequent time instants. Hence, it is carried out for large positive time difference (anti-Hebbian in nature) or small negative time difference (Hebbian in nature) between the pre- and post-spikes. The iHB-STDP learning rule for inhibitory pre-neurons effectively incorporates the learning dynamics of eHB-STDP for excitatory pre-neurons by mirroring the potentiation and depression windows about the Δt axis.

In this work, we use trace-based technique to estimate spike timing differences as it is commonly adopted for efficient implementation of STDP learning rules (Diehl and Cook, 2015). For instance, the positive time difference between a pair of pre- and post-spikes is estimated by generating an exponentially decaying pre-trace (with time constant τ_{pre}) that is reset to unity at the time instant of a pre-spike, and sampling it in the event of a post-spike. Smaller the time difference between the pre- and post-spikes, larger is the sampled pre-trace, and vice versa. Every pre-neuron has a pre-trace that is sampled upon a post-spike to obtain the positive spike timing difference. Likewise, every post-neuron has a post-trace (with time constant τ_{post}) that is sampled upon a pre-spike to obtain the negative spike timing difference. As a result, the eHB-STDP (iHB-STDP) hyperparameters, namely, t_{Hebb_pot} (t_{Hebb_dep}), $t_{antiHebb\_dep}$ ($t_{antiHebb\_pot}$), and t_{Hebb_dep} (t_{Hebb_pot}) are mapped to pre_{Hebb_pot} (pre_{Hebb_dep}), $pre_{antiHebb\_dep}$ ($pre_{antiHebb\_pot}$), and $post_{Hebb\_dep}$ ($post_{Hebb\_pot}$), respectively.

2.3. Unsupervised Training Methodology for the Convolutional Layers

We train the binary kernels forming ReStoCNet in a layer-wise unsupervised manner using the proposed probabilistic e/iHB-STDP learning rule. Consider a $k \times k$ binary kernel ($kernel_{ij}^{l}$) connecting the i^{th} input spike map in layer "$l - 1$" (map_{i}^{l-1})

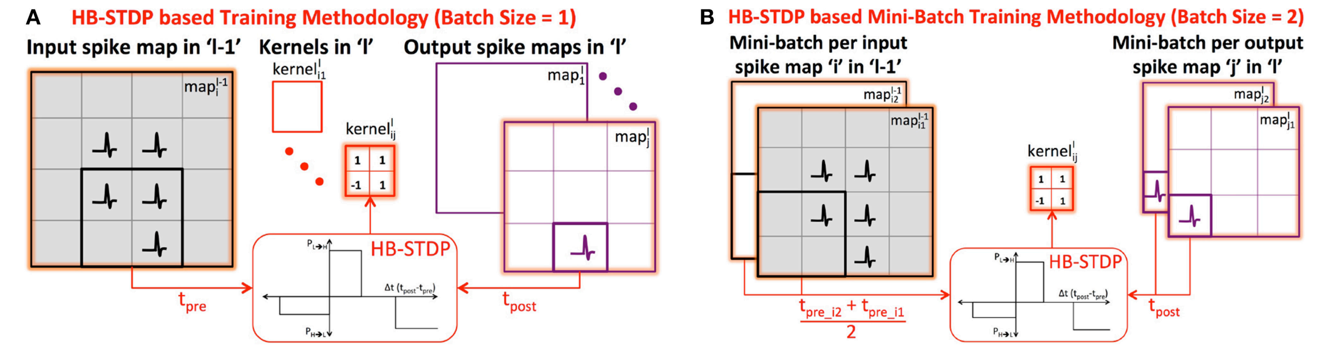
A HB-STDP based Training Methodology (Batch Size = 1)
Input spike map in 'l-1'
Kernels in 'l'
Output spike maps in 'l'
HB-STDP
t_pre
t_post
B HB-STDP based Mini-Batch Training Methodology (Batch Size = 2)
Mini-batch per input spike map 'i' in 'l-1'
Mini-batch per output spike map 'j' in 'l'
HB-STDP
(t_pre_i2 + t_pre_i1)/2
t_post

FIGURE 3 | (A) Illustration of HB-STDP based unsupervised training methodology for $kernel_{ij}^{l}$ connecting the i^{th} input spike map in layer "$l-1$" (map_i^{l-1}) to the j^{th} output spike map in layer "l" (map_j^l). The $kernel_{ij}^l$ is updated using HB-STDP based on the spike timing difference between the spiking post-neuron in output map_j^l and the corresponding pre-neurons in input map_i^{l-1}. The HB-STDP based weight updates are carried out on all the kernels in layer "l" based on the respective input and output maps. **(B)** Illustration of HB-STDP based mini-batch training methodology for mini-batch size of 2. The $kernel_{ij}^l$ is now updated based on the average spike timing difference between the spiking post-neurons in the output mini-batch (map_{j1}^l and map_{j2}^l) and the respective pre-neurons in the input mini-batch (map_{i1}^{l-1} and map_{i2}^{l-1}).

to the j^{th} output spike map in layer "l" (map_j^l) as shown in **Figure 3A**. Let us suppose that a post-neuron in the output map_j^l spikes at a particular time instant: the kernel weights are then probabilistically updated based on the time difference between the post-spike and the corresponding $k \times k$ pre-spikes in the input map_i^{l-1}. We use the eHB-STDP learning rule for excitatory pre-neurons and iHB-STDP learning rule for inhibitory pre-neurons as described in subsection 2.2. If multiple post-neurons in the output map_j^l spike, we update $kernel_{ij}^l$ based on the average spike timing difference between the spiking post-neurons and the respective pre-neurons, which leads to generalized feature learning. However, in order to achieve optimal generalization performance, we average the spike timing differences computed with fixed stride, known as $STDP_{stride}$, over the output map_j^l. As an example, for $STDP_{stride}$ of 2, we average the spike timing differences computed between every alternate spiking post-neuron in output map_j^l and the respective pre-neurons. Larger the $STDP_{stride}$, fewer is the number of post-neurons whose spike timing difference estimates are averaged to update the kernel. Consequently, there is loss of generality and added specificity in the features learnt by the kernel for larger $STDP_{stride}$. We experimentally determine the $STDP_{stride}$ for optimal generalization performance that yields the highest test accuracy for a given pattern recognition task.

STDP-based learning is typically performed in an online manner by feeding the input patterns sequentially. STDP-based online learning has been shown to work well particularly for two-layer fully-connected SNNs, where each output or excitatory neuron learns to spike exclusively for a unique class of input patterns by encoding a general input representation in the input to excitatory synaptic weights (Diehl and Cook, 2015). Convolutional SNNs, on the other hand, require each kernel to extract features shared across different input classes. In order to enable the kernel to extract general features characterizing different input classes, we perform mini-batch learning following recent works by Lee et al. (2018b) and Ferré et al. (2018). The proposed HB-STDP based mini-batch training methodology is illustrated in **Figure 3B**, where the $kernel_{ij}^l$ is now shared by a mini-batch of i^{th} input map in layer "$l-1$" (input mini-batch) and j^{th} output map in layer "l" (output mini-batch). We first average the spike timing differences between the spiking post-neurons and the respective pre-neurons, estimated using fixed $STDP_{stride}$, over each output map in the mini-batch to obtain the resultant spike timing difference per output map in the mini-batch. We subsequently average the resultant spike timing differences of the output maps across the mini-batch and probabilistically update $kernel_{ij}^l$ using HB-STDP as shown in **Figure 3B** for a specific post-neuron in the output mini-batch. At every time instant, the HB-STDP driven mini-batch weight updates are carried out on all the kernels in a given layer. This process is repeated over the entire time duration, T_{STDP}, for which the training patterns are presented.

Finally, in order to ensure that different kernels in a layer learn diverse input features, we incorporate the uniform firing threshold adaptation scheme proposed by Lee et al. (2018b) and dropout (Srivastava et al., 2014) for the output maps. In the beginning of training, the firing threshold of all the post-neurons in every output mini-batch is reset to zero. When a mini-batch of training patterns is presented, multiple post-neurons in an output mini-batch spike and encode definite input features in the kernel weights. We then increase the firing threshold of all the post-neurons in the output mini-batch by an amount $\Delta thresh$, which is specified by

$$\Delta thresh = \beta_{thresh} \times \frac{output\ spike\ count}{output\ map\ size} \tag{4}$$

where β_{thresh} is the rate of threshold increase, *output spike count* is the number of spikes per output map summed over the mini-batch, and *output map size* is the product of the height and width of the output maps. The amount of threshold increase depends on the *output spike count* normalized by the *output map size* to account for the drop in spiking activity of the output maps across successive convolutional layers due to gradual reduction in the respective sizes. Higher the normalized spiking activity of the output mini-batch, greater is the corresponding increase in its firing threshold, and vice versa. Firing threshold adaptation effectively regulates the spiking activity of the output mini-batch and provides an opportunity for the hitherto dormant output mini-batches to spike and learn, thereby ensuring that no single output mini-batch completely dominates the learning process during a mini-batch training iteration. In addition, we introduce dropout (Srivastava et al., 2014) for the output maps to achieve diversity in feature learning across successive mini-batch training iterations. At the beginning of every training iteration, we randomly drop a fraction of output mini-batches based on the dropout probability, p_{drop}, by forcing the respective spike outputs to zero. Dropout ensures that the same output mini-batch does not spike repeatedly for every training iteration, thereby promoting diversity in feature learning among the kernels in a layer. Once a layer is trained, we propagate the spikes from the input through the trained layers, and update the kernels and firing thresholds of the output maps in the following layer using the presented training methodology. The training process is repeated for all the convolutional layers in ReStoCNet.

2.4. Supervised Training Methodology for the Fully-Connected Layer

After all the convolutional layers are trained, we pool the respective spike maps using average pooling as detailed in subsection 2.1. We then low-pass filter the spike trains of the pooled maps, by integrating the spike outputs at every time instant and decaying the resultant sum between successive time instants, to obtain their spiking activations as described in Lee et al. (2016, 2018a) and specified by

$$\begin{aligned} pool^l_{lpf}(t) &= e^{-\frac{\Delta t_{sim}}{\tau_{lpf}}} \times pool^l_{lpf}(t - \Delta t_{sim}) + pool^l(t) \\ pool^l_{out} &= \frac{pool^l_{lpf}(T_{sim})}{T_{sim}} \end{aligned} \tag{5}$$

where $pool^l_{lpf}(t)$ is the low-pass filtered output of the pooled spike map $pool^l(t)$ in layer "l" at any given time t, τ_{lpf} is the low-pass filter time constant, Δt_{sim} is the simulation time-step, T_{sim} is the simulation period for which the input patterns are presented, and $pool^l_{out}$ is the spiking activation of the pooled map in layer "l" over the simulation period. The spiking activation thus obtained accounts for the highly non-linear leaky-integrate-and-fire and membrane potential reset dynamics of the spiking neurons in the convolutional layers. The spiking activations of the pooled maps of all the convolutional layers are concatenated and fed to the fully-connected layer, trained using error backpropagation (Rumelhart et al., 1986), for inference. We use full-precision synaptic weights in the fully-connected layer to comprehensively validate the efficacy of the proposed probabilistic HB-STDP learning rule for training the binary kernels in the convolutional layers. The full-precision synaptic weights can be binarized using algorithms proposed for training binary deep learning networks (Courbariaux et al., 2015; Rastegari et al., 2016; Hubara et al., 2017). It is important to note that the presented HB-STDP based learning methodology effectuates plasticity by probabilistically switching the binary weights, thereby precluding the need to store the full-precision weights during training. Binarization algorithms for deep learning networks, on the other hand, update the full-precision weights during training, which are subsequently binarized for forward propagation and computing the error gradients.

3. RESULTS

We first validate the efficacy of HB-STDP, by visually demonstrating the significance of having distinct potentiation and depression windows separated by a dead zone for efficient feature learning, using two-layer binary fully-connected SNN trained on the MNIST dataset. We then comprehensively evaluate ReStoCNet and the presented HB-STDP based unsupervised mini-batch training methodology on the MNIST and CIFAR-10 datasets. We show that the residual connections are critical to achieving efficient unsupervised learning in deeper convolutional layers and minimizing the accuracy degradation incurred by STDP-trained deep SNNs without residual connections. We use the classification accuracy on the test set and the synaptic memory compression obtained by using binary kernels as the evaluation metrics for ReStoCNet compared to full-precision (32-bit) SNN under iso-accuracy conditions.

3.1. Two-Layer Binary Fully-Connected SNN for MNIST Digit Recognition

The binary fully-connected SNN (Diehl and Cook, 2015) consists of an input layer fully-connected via binary synapses to neurons in the excitatory layer, which are connected in a one-to-one manner to neurons in the subsequent inhibitory layer. Each inhibitory neuron laterally inhibits all the excitatory neurons except the one from which it receives a forward connection. Lateral inhibition facilitates competitive learning and enables each excitatory neuron to spike exclusively and recognize a unique class of input patterns. The input to excitatory synaptic weights are trained using three different configurations of the eHB-STDP learning rule that are enumerated below:

1. eHB-STDP – This is the proposed eHB-STDP learning rule containing distinct Hebbian potentiation and anti-Hebbian depression windows separated by a dead zone as shown in **Figure 4A**.
2. eHB-STDP2 – This is a variant of the eHB-STDP learning rule where the dead zone is replaced with a wider Hebbian potentiation window as depicted in **Figure 4B**.
3. eHB-STDP3 – This is an alternative variant of the eHB-STDP rule where the dead zone is replaced with a wider anti-Hebbian depression window as illustrated in **Figure 4C**.

Note that the excitatory↔inhibitory synaptic weights are fixed *a priori* and are not subjected to STDP-based learning. We simulated the fully-connected SNN using BRIAN (Goodman and Brette, 2008), which is an open-source SNN simulation framework, on the MNIST dataset. The input image pixels are converted to Poisson spike trains firing at a rate constrained between 0 and 63.75 Hz depending on

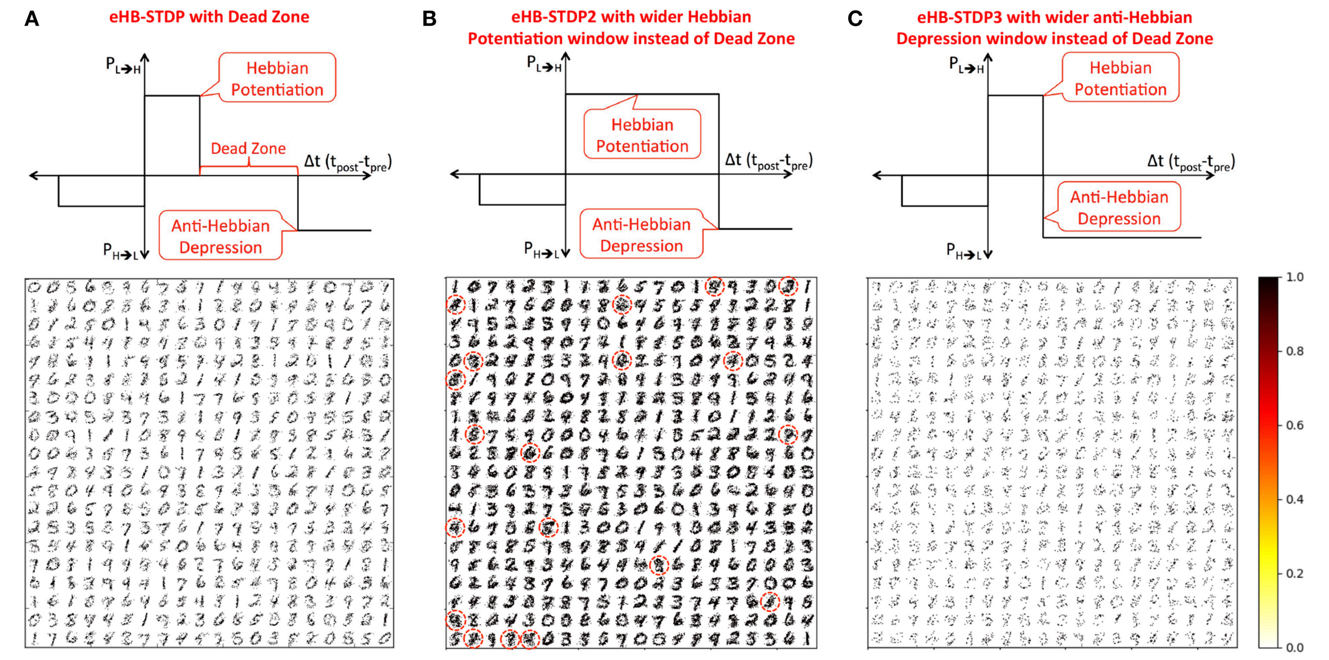

FIGURE 4 | MNIST digit representations (re-arranged in 28×28 format) learnt by the synapses connecting the input to each excitatory neuron in a binary fully-connected SNN of 400 neurons (arranged in 20×20 grid). The binary fully-connected SNN is trained using **(A)** the proposed eHB-STDP containing distinct Hebbian potentiation and anti-Hebbian depression windows separated by a dead zone, **(B)** eHB-STDP2 where the dead zone is replaced with a wider Hebbian potentiation window, and **(C)** eHB-STDP3 where the dead zone is replaced with a wider anti-Hebbian depression window.

the respective pixel intensities for a simulation period of 350 ms. Note that the simulation time-step is 0.5 ms. We use the spiking neuronal model detailed in Diehl and Cook (2015) whose parameters are adopted from Jug (2012). The eHB-STDP hyperparameters used in our simulations are listed in **Table 1**.

We first train a binary fully-connected SNN of 400 excitatory neurons using the three different eHB-STDP configurations on 3500 MNIST digit patterns. **Figure 4A** illustrates that eHB-STDP causes each excitatory neuron to self-learn general representation of a unique digit in the input to excitatory synaptic weights. On the other hand, eHB-STDP2, with a wider Hebbian potentiation window instead of the dead zone, causes certain excitatory neurons to self-learn overlapping input representations as highlighted in **Figure 4B**. Overlapping input representations negatively impact the selective spiking behavior of the excitatory neurons for specific input classes and degrade the recognition capability of the SNN. The final eHB-STDP configuration, eHB-STDP3, leads to insufficient representation learning as depicted in **Figure 4C** due to the dominance of synaptic depression over synaptic potentiation weight updates. Thus, the proposed eHB-STDP learning rule offers superior representation learning capability compared to the explored variants by maintaining optimal balance between the potentiation and depression weight updates. This is further corroborated by the accuracy results shown in **Figure 5A**, which is evaluated as explained below. At the end of eHB-STDP based training, each excitatory neuron is tagged as having learnt the class of input patterns for which it spiked the most during the training phase. A test pattern is predicted to belong to the class (or tag) represented by the group of neurons with the highest average spike count over the simulation period. The binary fully-connected SNN of 400 neurons trained using eHB-STDP yielded 79.94% accuracy on the MNIST test set, which is higher by >8% compared to that achieved using the remaining eHB-STDP variants. The accuracy can be further improved by increasing the number of excitatory neurons as shown in **Figure 5B**. We now estimate the *synaptic memory compression* offered by the binary SNN compared to full-precision (32-bit) SNN, which is specified by

$$\textit{synaptic memory compression} = \frac{\#\textit{input neurons} \times \#\textit{excitatory neurons}_{\textit{full-precisionSNN}} \times 32}{\#\textit{input neurons} \times \#\textit{excitatory neurons}_{\textit{binarySNN}} \times 1} \quad (6)$$

where *#input neurons* is 784 for the MNIST dataset. **Figure 5B** indicates that binary SNN of 6400 neurons offers comparable accuracy (∼92%) to that provided by full-precision (32-bit) SNN of 1600 neurons (Diehl and Cook, 2015), leading to 8× synaptic memory compression under iso-accuracy conditions. Note that the accuracy of ∼92% is higher than that reported in related works for binary fully-connected SNN, trained using probabilistic STDP-based learning rules, as shown in **Table 2**.

However, the fully-connected SNN introduces scalability issues as the network depth is increased due to explosion in the number of trainable parameters. We demonstrate ReStoCNet, which is a scalable multilayer convolutional SNN composed of binary kernels, trained using the optimal e/iHB-STDP based unsupervised mini-batch training methodology.

3.2. ReStoCNet for MNIST Digit Recognition

The MNIST dataset contains 60,000 training patterns and 10,000 test patterns of handwritten digits that are stored as 28×28 Grayscale images. In this work, we developed a custom simulation framework using Pytorch (Paszke et al., 2017) to evaluate ReStoCNet and the presented HB-STDP based unsupervised training methodology. The simulation parameters for the Leaky-Integrate-and-Fire (LIF) neuron in the convolutional layers and the Integrate-and-Fire (IF) neuron in the spatial pooling layers are shown in **Table 3**. The binary kernels in every convolutional layer are initialized to logic high state (w_{high}) with a probability, p_{high}, which is specified by

$$p_{high} = \sqrt{\frac{\alpha_{weight\_init}}{fan\_in + fan\_out}} \tag{7}$$

where α_{weight_init} is the proportionality constant controlling p_{high}, and fan_in and fan_out are the total number of input and output synaptic weights, respectively, for a given convolutional layer. The remaining kernel weights in the convolutional layer are initialized to logic low state (w_{low}). The firing threshold of the LIF neurons in every convolutional layer are initialized to zero.

We first simulated a 16C3-2P-10FC ReStoCNet, composed of single convolutional layer with 16 maps and 3×3 binary kernels followed by pooling layer whose spiking activations are directly fed to the final softmax layer. The input image pixels are mapped to excitatory pre-neurons firing at a rate constrained between 0 and 200 Hz depending on the corresponding pixel intensities. The eHB-STDP model parameters are provided in **Table 3**. We trained the convolutional layer in ReStoCNet using 2,000 MNIST digit patterns with a mini-batch size of 200. We thereafter fed the entire training dataset to ReStoCNet, spatially pooled the spike maps of the convolutional layer, and low-pass filtered the pooled spike trains over a simulation period of 100 ms to estimate their spiking activations. The pooling layer spiking activations are passed on to the fully-connected softmax layer, which is trained using the Adam optimizer (Kingma and Ba, 2014) and cross-entropy loss function for 100 epochs. The training parameters used for the fully-connected layer are mentioned in **Table 4**. The shallow ReStoCNet yielded an accuracy of 95.21% on the MNIST test set, which increased to 98.22% for a wider 36C3-2P-10FC ReStoCNet in which the convolutional layer is trained using 10,000 MNIST digit patterns. Further improvement in accuracy is obtained by augmenting the classifier in ReStoCNet with an additional fully-connected layer of 128 neurons prior to the softmax output layer as shown in **Figure 6**, which indicates that 36C3-2P-128FC-10FC ReStoCNet offers an improved accuracy

TABLE 1 | Simulation parameters for training the binary fully-connected SNN on the MNIST dataset.

Parameters	Values
Simulation time-step, Δt_{sim}	0.5 ms
Simulation period, T_{sim}	350 ms
Maximum input spike rate	63.75 Hz
Pre-trace time constant, τ_{pre}	20 ms
Post-trace time constant, τ_{post}	20 ms
pre_{Hebb_pot} (eHB-STDP)	0.85
$pre_{antiHebb\_dep}$ (eHB-STDP)	0.10
$post_{Hebb\_dep}$ (eHB-STDP)	0.80
p_{Hebb_pot} (eHB-STDP)	0.08
$p_{antiHebb\_dep}$ (eHB-STDP)	0.06
p_{Hebb_dep} (eHB-STDP)	0.005
Maximum synaptic weight (w_{high})	1.0
Minimum synaptic weight (w_{low})	0.0

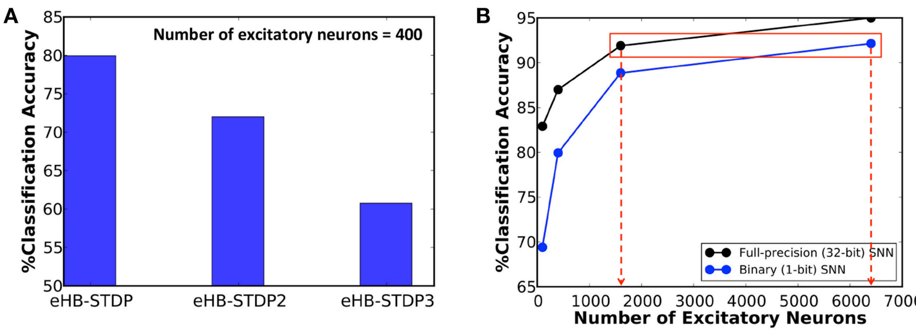

FIGURE 5 | (A) Classification accuracy of binary fully-connected SNN of 400 excitatory neurons trained using the three different eHB-STDP configurations illustrated in **Figure 4**. **(B)** Classification accuracy of binary fully-connected SNN, trained using the proposed eHB-STDP learning rule, compared to full-precision (32-bit) SNN (Diehl and Cook, 2015) for different network sizes.

TABLE 2 | Classification accuracy of binary fully-connected SNNs on the MNIST test set.

Model	#Excitatory neurons	Training methodology	Accuracy (%)
Binary SNN (Querlioz et al., 2015)	50	Probabilistic Rectangular STDP	60
Binary SNN (Srinivasan et al., 2016)	400	Probabilistic Exponential STDP	70.15
Binary SNN (our work)	400	Probabilistic eHB-STDP	79.94
Binary SNN (our work)	6400	Probabilistic eHB-STDP	92.14

TABLE 3 | Simulation parameters for training the convolutional layers in ReStoCNet.

Parameters	Values		
	C1	C1	C2/C3
Input dataset	MNIST	CIFAR-10	CIFAR-10
Maximum synaptic weight (w_{high})	+1.0	+1.0	+1.0
Minimum synaptic weight (w_{low})	−1.0	−1.0	−1.0
Weight initialization constant (α_{weight_init})	75	30	30
Simulation time-step, Δt_{sim}	1 ms	1 ms	1 ms
Simulation period for STDP, T_{STDP}	25 ms	25 ms	25 ms
Maximum input spike rate for STDP	200 Hz	200 Hz	500 Hz
Dropout probability for STDP, p_{drop}	0.5	0.5	0.5
$STDP_{stride}$	5	5	5
Pre-trace decay time constant, τ_{pre}	1.45 ms	1.45 ms	1.45 ms
pre_{Hebb_pot} (eHB-STDP)	0.50e-1	0.20e-1	0.20e-1
$pre_{antiHebb\_dep}$ (eHB-STDP)	0.50e-2	0.50e-2	0.50e-2
p_{Hebb_pot} (eHB-STDP)	0.01	0.05	0.05/25
$p_{antiHebb\_dep}$ (eHB-STDP)	0.01	0.01	0.01/25
p_{Hebb_dep} (eHB-STDP)	0	0	0
pre_{Hebb_dep} (iHB-STDP)	–	0.20e-1	0.20e-1
$pre_{antiHebb\_pot}$ (iHB-STDP)	–	0.50e-2	0.50e-2
p_{Hebb_dep} (iHB-STDP)	–	0.05	0.05/25
$p_{antiHebb\_pot}$ (iHB-STDP)	–	0.01	0.01/25
p_{Hebb_pot} (iHB-STDP)	–	0	0
Leaky-Integrate-and-Fire (LIF) neuron leak time constant, τ_{mem}	9.5 ms	9.5 ms	9.5 ms
Rate of increase of LIF neuronal firing threshold, β_{thresh}	6e-4	6e-4	6e-4 (C2) 8e-4 (C3)
Integrate-and-Fire (IF) neuron pooling threshold, θ_{pool}	0.80	0.80	0.80
Simulation period to estimate spiking activation, T_{sim}	100 ms	100 ms	100 ms
Maximum input spike rate to estimate spiking activation	500*Hz*	500*Hz*	500*Hz*
Low-pass filter time constant to estimate spiking activation, τ_{lpf}	99.5 ms	99.5 ms	99.5 ms

TABLE 4 | Simulation parameters for training the fully-connected layer in ReStoCNet.

Parameters	Values	
	MNIST	CIFAR-10
Batch size	256	256
Number of epochs	100	100
Learning rate (Adam)	1.5e-3	1.0e-4
betas (Adam)	(0.9, 0.999)	(0.9, 0.999)
eps (Adam)	1e-8	1e-8
Weight decay (Adam)	0	0
Dropout probability	0.5	0.5

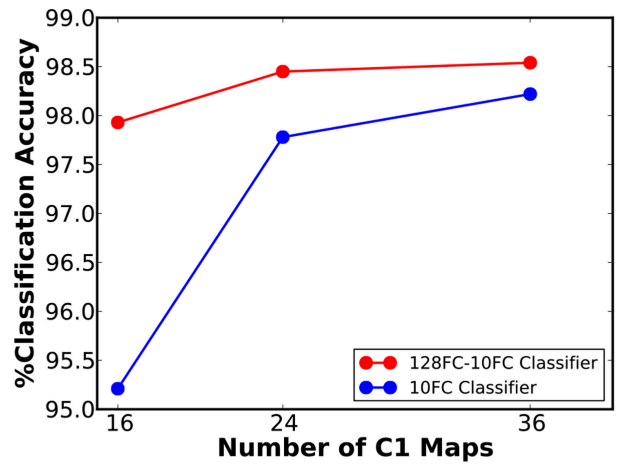

FIGURE 6 | Classification accuracy of ReStoCNet, composed of single convolutional layer followed by a pooling layer and one or more fully-connected layers, vs. the number of output (C1) maps, on the MNIST test set.

of 98.54% on the MNIST test set. Note that we did not simulate deep ReStoCNets for MNIST digit recognition since the shallow networks yield >98% accuracy, and that any further increase in the depth of STDP-trained convolutional layers would not provide commensurate improvements in the classification accuracy.

3.3. ReStoCNet for CIFAR-10 Image Recognition

The CIFAR-10 dataset contains 50,000 training images and 10,000 test images, 32×32×3 in dimension, spanning 10 output classes. We pre-processed the CIFAR-10 images using global contrast normalization followed by ZCA whitening (Krizhevsky, 2009). Global contrast normalization is performed by subtracting and scaling the pixel intensities of each input channel by the corresponding mean and standard deviation computed over the training set. The normalized image is then transformed by multiplying with whitening filters as explained in Krizhevsky (2009), which enables a network to learn higher-order pixel correlations. **Figure 7** illustrates a few original and pre-processed images from the CIFAR-10 dataset. The simulation parameters used for training the convolutional layers are provided in **Table 3** while those used for training the fully-connected layer are listed in **Table 4**. The binary kernels and firing thresholds of the convolutional layers are initialized as described in subsection 3.2.

In our first experiment, we simulated a 36C3-2P-1024FC-10FC ReStoCNet, designated as ReStoCNet-1, consisting of a single convolutional layer with 36 maps and 3×3 binary kernels followed by fully-connected layer containing 1024 ReLU neurons and a final softmax layer with 10 output neurons. The pre-processed CIFAR-10 images are composed of pixels with positive and negative intensities, which are, respectively, mapped to excitatory and inhibitory pre-neurons firing at a rate constrained between 0 and 200 Hz depending on the absolute value of the corresponding pixel intensities. The e/iHB-STDP model parameters are listed in **Table 3**. Note that the e/iHB-STDP switching probability is set to zero in the negative STDP timing window to facilitate optimal balance between the potentiation and depression updates for a smaller 3×3 kernel shared by 32×32 pre-neurons in the input map and 30×30 post-neurons in the convolutional map. The binary kernels in ReStoCNet-1 are trained using 5,000 images, with mini-batch size of 200, for simulation period of 25 ms per mini-batch training iteration. Note that we used a simulation time-step of 1 ms. **Figure 8A** illustrates the low-level input features self-learnt by the binary kernels, enabled by the e/iHB-STDP based unsupervised training methodology. The shallow ReStoCNet-1, wherein the fully-connected layer is trained on the entire dataset, yielded 64.31% test accuracy that is higher than an accuracy of 59.42% obtained using randomly initialized binary kernels and zero firing thresholds in the convolutional layer. In order to determine if accuracy loss is incurred as a result of using binary kernels, we trained ReStoCNet-1 composed of full-precision (32-bit) kernels using standard exponential STDP rule (Song et al., 2000) with learning rate of 0.01 for the positive STDP timing window and 0 for the negative STDP timing window. ReStoCNet-1 with full-precision kernels provided 64.30% test accuracy, which is comparable to that obtained using binary kernels. **Figure 8B** shows that the test accuracy improves with the number of maps in the convolutional layer. As explained in subsection 2.3, the classification accuracy of ReStoCNet has a strong dependence on the chosen $STDP_{stride}$ used for computing the average spike timing difference of the spiking post-neurons in the convolutional maps. **Figure 8C** indicates that the accuracy of ReStoCNet-1 degrades for $STDP_{stride}$ smaller than 4 or greater than 5. If the $STDP_{stride}$ is small, the binary kernels are updated based on the spike timing difference averaged over large number of spiking post-neurons in the convolutional maps, leading to degradation in the learnt features. On the contrary, if the $STDP_{stride}$ is large, the binary kernels are updated based on the spike timing difference estimates of few post-neurons, leading to loss of generality in the learnt features. We use the optimal $STDP_{stride}$ of 5 for all the ReStoCNet experiments presented in this work.

Next, we simulated a 36C3-36C3-2P-1024FC-10FC ReStoCNet, designated as ReStoCNet-2, composed of two convolutional layers, each with 36 maps and 3×3 binary kernels. The first convolutional layer is trained as described in the previous paragraph. The binary kernels and firing thresholds of the second convolutional layer are trained using a different subset of 5,000 CIFAR-10 images with a mini-batch size of 200. Note that the e/iHB-STDP hyperparameters are similar for both the convolutional layers except the synaptic switching probabilities, which are scaled down for the second convolutional layer as shown in **Table 3**. The lower switching probabilities for the second convolutional layer accounts for the fact that every constituting post-neuron receives weighted input from 36 maps each in the residual and direct paths while a post-neuron in the first convolutional layer receives weighted input from just the 3 maps in the input layer. We simulated two versions of ReStoCNet-2: one without residual connections and the other with residual connections from the input to second convolutional layer. **Figure 9** shows that ReStoCNet-2 with residual connections learns diverse high-level input features compared to the one without residual connections. As a result, ReStoCNet-2 with residual connections yielded 65.79% accuracy, which is roughly 1.5% higher than that provided by ReStoCNet-2 without residual connections as well as ReStoCNet-1. This begs the following question: *is ReStoCNet-2 yielding higher accuracy that ReStoCNet-1 just due to increased number of synaptic weights in the fully-connected layer as a consequence of concatenating the pooled spiking activations of both the convolutional layers?* To answer this question, we compare ReStoCNet-2, in which the spiking activations of the 72 pooled maps are fed to a fully-connected layer of 1024 neurons, with ReStoCNet-1 in which the spiking activations of the 36 pooled maps are fed to a larger fully-connected layer of 2048 neurons. **Figure 9C** indicates that ReStoCNet-2 offers higher accuracy than that provided by ReStoCNet-1 with 2048 neurons in the fully-connected layer, which is a testament to the improved feature learning capability of the second convolutional layer in the presence of residual inputs. **Figure 9D** shows that ReStoCNet-2 provides only modest improvement in accuracy as the number of output maps is increased in the second convolutional layer. The accuracy limitation is caused by the inability of the unsupervised training methodology to effectively optimize an over-parameterized network.

Finally, we evaluated a deeper 36C3-36C3-36C3-2P-1024FC-10FC ReStoCNet, referred to as ReStoCNet-3, composed of three convolutional layers as depicted in **Figure 1**. We inverted the residual inputs to the third convolutional layer to ensure diversity in the residual maps received by the second and third layers from the input layer. We trained the third convolutional layer with the same hyperparameters (shown in **Table 3**) as those used for training the second convolutional layer, albeit on a different subset of 5,000 images from the CIFAR-10 dataset. In addition to ReStoCNet-3 (with residual connections), wherein the pooled spiking activations of all the convolutional layers are used for

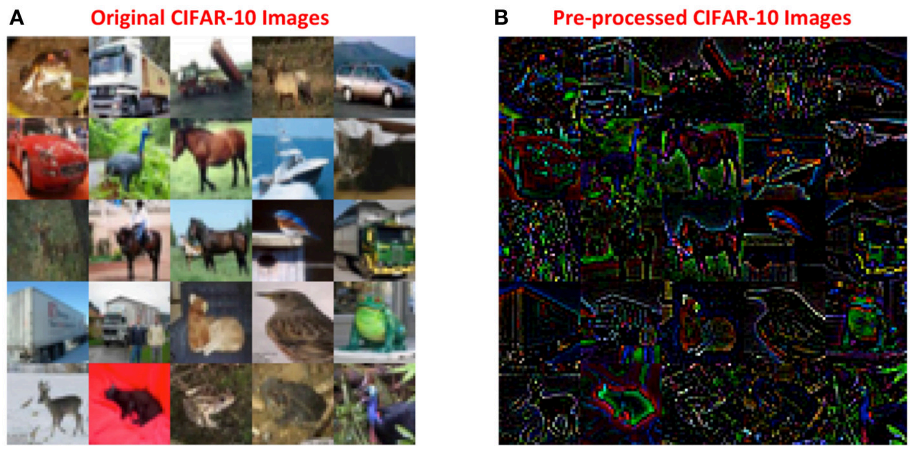

FIGURE 7 | (A) Original 32×32×3 CIFAR-10 images. **(B)** CIFAR-10 images pre-processed using global contrast normalization followed by ZCA whitening (Krizhevsky, 2009).

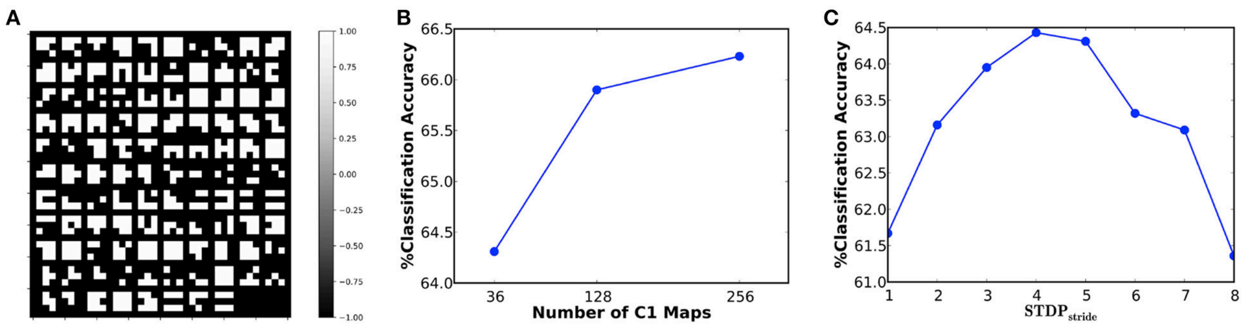

FIGURE 8 | (A) Binary kernels (3×3 in size) of ReStoCNet-1 (36C3-2P-1024FC-10FC ReStoCNet), trained using e/iHB-STDP based unsupervised training methodology, on 5,000 images from the CIFAR-10 dataset. **(B)** Classification accuracy of ReStoCNet vs. the number of convolutional maps. **(C)** Classification accuracy of ReStoCNet-1 vs. the $STDP_{stride}$ used to compute the average spike timing difference of the spiking post-neurons in the convolutional maps.

inference, we simulated the following variants to demonstrate the significance of residual connections for the scalability of deep SNNs:

1. ReStoCNet-3a – This is a variant of ReStoCNet-3 without residual inputs to the third convolutional layer. In addition, the pooled spiking activations of only the third convolutional layer are fed to the fully-connected layer for inference.
2. ReStoCNet-3b – This is a variant of ReStoCNet-3 with residual inputs to the third convolutional layer, wherein the pooled spiking activations of only the third convolutional layer are used for inference.

ReStoCNet-3a, devoid of residual connections, yielded 44.75% accuracy on the CIFAR-10 test set, which is 17.5% lower compared to an accuracy of 62.26% provided by ReStoCNet-3b with residual connections as shown in **Figure 10A**. The higher accuracy of ReStoCNet-3b can be directly attributed to its improved feature learning capability, rendered possible by the residual inputs feeding into the third convolutional layer. The optimal ReStoCNet-3 configuration (with residual connections), wherein the pooled spiking activations of all the convolutional layers are used for inference, offered 65.25% accuracy, which is only comparable to an accuracy of 65.79% provided by ReStoCNet-2 as shown in **Figure 10B**.

Our analysis on ReStoCNet, trained using the e/iHB-STDP based unsupervised training methodology, offers the following key insights. First, it shows that the residual connections are critical for the scalability of deep SNNs. Second, it reveals that the maximum achievable accuracy is limited by the STDP-based unsupervised training methodology as further corroborated by **Figure 11**, which illustrates the unsupervised clustering capability of ReStoCNet-3 for different training images from the CIFAR-10 dataset. In order to visualize the efficiency of unsupervised clustering offered by ReStoCNet-3, we reduce the dimension of the pooled spiking activations of the convolutional layers using Principal Component Analysis (PCA) followed by t-Distributed Stochastic Neighbor Embedding (t-SNE) (Maaten and Hinton, 2008), and plot the first two t-SNE components for the training images. The t-SNE dimensionality reduction technique computes pair-wise similarities between the data points (images) in the high-dimensional space and projects

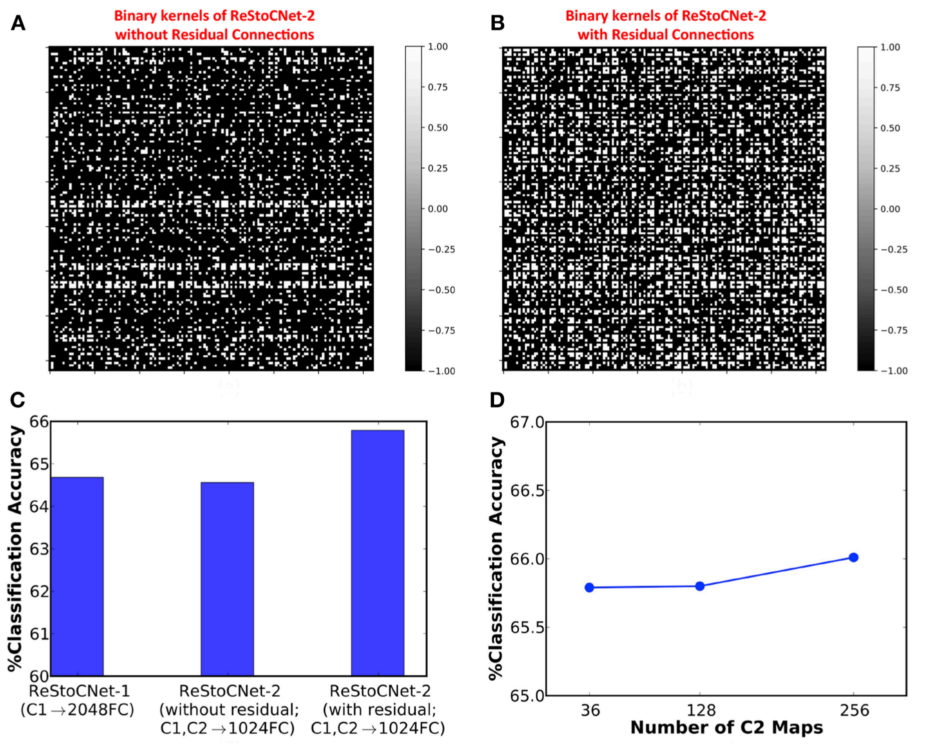

FIGURE 9 | Binary kernels (in second convolutional layer) of ReStoCNet-2 (36C3-36C3-2P-1024FC-10FC ReStoCNet) **(A)** without residual connections, and **(B)** with residual connections from the input to second convolutional layer. **(C)** Classification accuracy of ReStoCNet-2 with and without residual connections compared to that provided by ReStoCNet-1. **(D)** Classification accuracy of ReStoCNet-2 (with residual connections) vs. the number of output maps in the second convolutional layer.

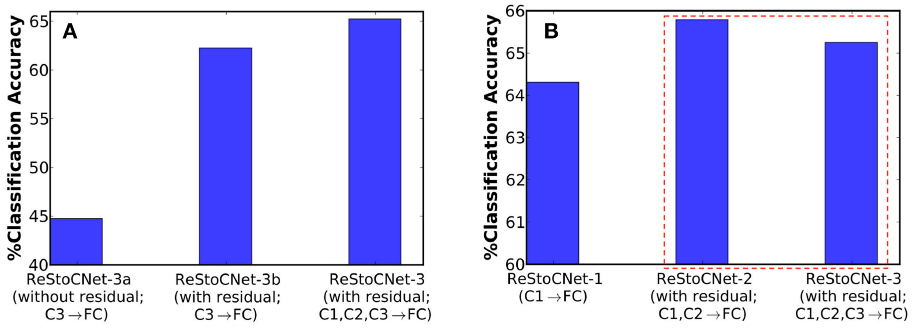

FIGURE 10 | (A) Classification accuracy of three different ReStoCNet-3 (36C3-36C3-36C3-2P-1024FC-10FC) configurations on the CIFAR-10 test set. **(B)** Comparison between the classification accuracy of different ReStoCNet configurations presented in this work.

them to a low-dimensional space that preserves the measured similarities. We refer the readers to Maaten and Hinton (2008) for a review of the t-SNE algorithm for visualizing high-dimensional input data. **Figure 11A** shows the t-SNE scatter plot for 15,000 training images spanning three different classes from the CIFAR-10 dataset, namely, airplane, bird, and frog. The primary objective of any machine learning model is to cluster the images per class together while ensuring sufficient separation among different classes. The t-SNE scatter plot of the pooled spiking activations of ReStoCNet-3 (shown in **Figure 11B**) indicates that, although distinct clusters are formed for the images in each class, there exists considerable overlap among different image clusters.

4. DISCUSSION

4.1. Comparison With Related Works

We compare ReStoCNet with convolutional SNNs, which employ unsupervised training methodology for the convolutional layers and supervised training algorithms like error backpropagation

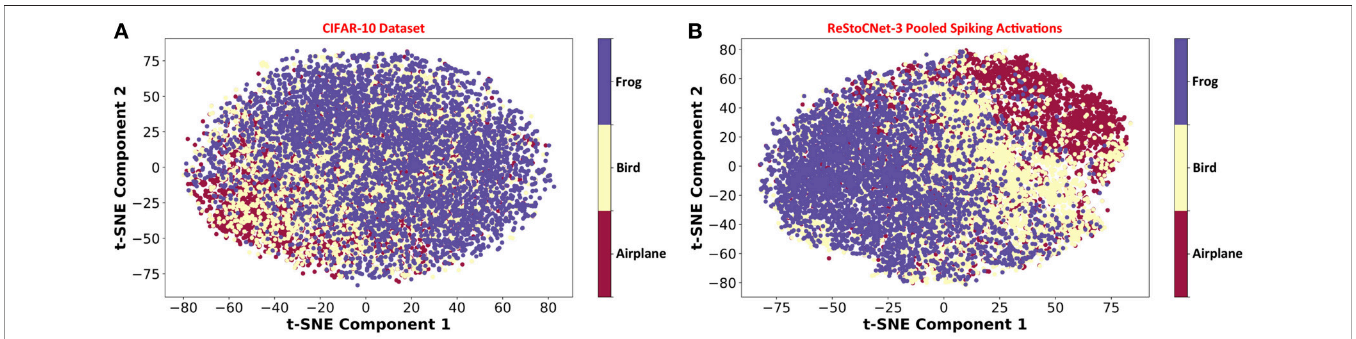

FIGURE 11 | (A) Scatter plot of the first two t-SNE components for 15,000 training images from the CIFAR-10 dataset from the classes: airplane, bird, and frog. **(B)** t-SNE scatter plot of the pooled spiking activations of the convolutional layers in ReStoCNet-3 for the corresponding training images.

for the fully-connected layer, using classification accuracy (on the test set) and kernel memory compression as the evaluation metrics. The memory compression offered by ReStoCNet as a result of using binary kernels in the convolutional layers, referred to as kernel memory compression, is computed as specified by

$$\begin{aligned} &\textit{kernel memory compression} \\ &= \frac{N_{baseline} \times ksize_{baseline} \times ksize_{baseline} \times nbits_{full\_precision}}{N_{ReStoCNet} \times ksize_{ReStoCNet} \times ksize_{ReStoCNet} \times nbits_{binary}} \end{aligned} \tag{8}$$

where $N_{ReStoCNet}$ ($N_{baseline}$) and $ksize_{ReStoCNet}$ ($ksize_{baseline}$) are the number of kernels and kernel size, respectively, in ReStoCNet (baseline convolutional SNN used for comparison), and $nbits_{binary}$ and $nbits_{full\_precision}$ are the hardware bit-precision required for storing the binary and full-precision kernels, which are set to 2-bits and 32-bits, respectively. Note that the binary kernels in ReStoCNet require storage capacity of 2-bits per synaptic weight since they are constrained to binary states -1 and $+1$. **Table 5** shows that the classification accuracy offered by ReStoCNet for MNIST digit recognition is comparable to that reported for convolutional SNNs composed of full-precision kernels trained using unsupervised learning methodologies. Specifically, a 36C3-2P-128FC-10FC ReStoCNet offers 98.54% accuracy on the MNIST test set, which compares favorably with that (98.36%) provided by the convolutional SNN presented in Tavanaei and Maida (2017), composed of single convolutional layer with 32 maps and 5×5 full-precision kernels trained using STDP. The proposed ReStoCNet offers 39.5× kernel memory compression by virtue of using smaller 3×3 binary kernels under iso-accuracy conditions for MNIST digit recognition. On the contrary, very few works have benchmarked convolutional SNNs, trained using unsupervised learning algorithms, on the CIFAR-10 dataset. Panda and Roy (2016) proposed spike-based convolutional Auto-Encoders, where the kernels in every convolutional layer are trained in an unsupervised manner using error backpropagation to regenerate the input spike patterns. Ferré et al. (2018) presented convolutional SNN (without residual connections), where the kernels are trained using a simple Hebbian STDP learning rule. **Table 6** shows that ReStoCNet provides 4–5% lower accuracy than that reported in both the related works. In particular, a 256C3-2P-1024FC-10FC ReStoCNet yields 4.97% lower accuracy than that provided by the 64C7-8P-512FC-512FC-10FC convolutional SNN (Ferré et al., 2018) while offering 21.7× kernel memory compression. Note that the convolutional SNN presented in Ferré et al. (2018) is simulated by single-step forward propagation using input rates while ReStoCNet is simulated using input spike trains over multiple time-steps.

Finally, we note that deep learning Binary Neural Networks (BNNs) (Courbariaux et al., 2015; Rastegari et al., 2016; Hubara et al., 2017), which use binary activations for the neurons in every layer except the input and output layers and binary weights, have been demonstrated to yield superior classification accuracy than that provided by ReStoCNet. Nevertheless, ReStoCNet offers the following advantages over BNNs. First, ReStoCNet is inherently suited for processing spatiotemporal spike trains from event-based audio and vision sensors as shown by Stromatias et al. (2017) for convolutional SNNs with full-precision weights since it computes with static image pixels mapped to spike trains. BNNs, on the contrary, use real-valued pixel intensities for the input layer. Second, ReStoCNet is amenable for efficient implementation in event-driven asynchronous neuromorphic hardware platforms like IBM *TrueNorth* (Merolla et al., 2014) and Intel *Loihi* (Davies et al., 2018) since it uses {0, 1} for the outputs of the spiking neurons in every convolutional layer. The weighted sum of the input spikes with the synaptic weights in the convolutional layers needs to be computed only in the event of a spike fired by the corresponding input neurons. In addition, only the sparse spiking events need to be transmitted between the layers. The event-driven computing capability offered by ReStoCNet can be exploited to achieve higher energy efficiency in neuromorphic hardware implementations by minimizing the computation and communication energy in the absence of spiking events. BNNs, on the other hand, use {1, -1} for the neuronal activations and either {1, -1} (Courbariaux et al., 2015) or {α, $-\alpha$} (Rastegari et al., 2016) where α is a layer-wise scaling factor for the weights to achieve good accuracy and stable training convergence (Pfeiffer and Pfeil, 2018). Hence, the computation of the weighted input sum and communication of the binarized neuronal activations need to be carried out for all the neurons in every layer in a synchronous manner, which is in contrast to the event-based asynchronous computing capability provided by ReStoCNet.

TABLE 5 | Classification accuracy of SNN models, which use unsupervised training methodology for the hidden/convolutional layers and supervised training algorithm for the output (classification) layer, on the MNIST test set.

Model	Size	Training methodology	Accuracy (%)
FC_SNN (Yousefzadeh et al., 2018)	6400FC-10FC	Probabilistic STDP + ANN backpropagation	95.70
ConvSNN (Panda and Roy, 2016)	12C5-2P-64C5-2P-10FC	SNN backpropagation	99.08
ConvSNN (Stromatias et al., 2017)	18C7-2P-10FC	Fixed Gabor kernels + ANN backpropagation	98.20
ConvSNN (Lee et al., 2018b)	16C3-16C3-2P-10FC	STDP	91.10
ConvSNN (Ferré et al., 2018)	8C5-2P-16C5-2P-120FC-60FC-10FC	STDP + ANN backpropagation	98.49
ConvSNN (Kheradpisheh et al., 2018)	30C5-2P-100C5-2P-10FC	STDP + Support Vector Machine	98.40
ConvSNN (Tavanaei et al., 2018)	64C5-2P-1500FC-10FC	STDP	98.61
ConvSNN (Mozafari et al., 2018)	30C5-2P-250C3-3P-200C5-5P	Reward-modulated STDP	97.20
ConvSNN (Tavanaei and Maida, 2017)	32C5-2P-128FC-10FC	STDP + Support Vector Machine	98.36
ReStoCNet (our work)	36C3-2P-128FC-10FC	Probabilistic eHB-STDP + ANN backpropagation	98.54

TABLE 6 | Classification accuracy of SNN models, which use unsupervised training methodology for the hidden/convolutional layers and supervised training algorithm for the output (classification) layer, on the CIFAR-10 test set.

Model	Size	Training methodology	Accuracy (%)
ConvSNN (Panda and Roy, 2016)	32C5-2P-32C5-2P-64C4-10FC	SNN backpropagation	70.16
ConvSNN (Ferré et al., 2018)	64C7-8P-512FC-512FC-10FC	STDP + ANN backpropagation	71.20
ReStoCNet (our work)	256C3-2P-1024FC-10FC	Probabilistic e/iHB-STDP + ANN backpropagation	66.23

Last, ReStoCNet offers a memory-efficient solution for enabling on-chip intelligence in resource-constrained battery-powered Internet of Things (IoT) edge devices since the binary kernels are trained using probabilistic-STDP based local learning rule that can be efficiently implemented on-chip. Learning is achieved by probabilistically switching the binary kernel weights between the allowed states based on spike timing, which precludes the need for storing the full-precision weights and enhances the memory efficiency during training. BNNs, on the other hand, are trained using error backpropagation algorithms that update the full-precision weights based on the backpropagated error gradients and binarize the modified weights for forward propagation and computing the error gradients. Thus, ReStoCNet provides a promising alternative for energy- and memory-efficient computing during both training and inference in IoT edge devices, for instance, surveillance cameras, which produce large volumes of real-time data. It is inefficient for these devices to continuously offload raw/compressed data to the cloud for training. This is because the sheer volume of generated data could exceed the bandwidth available for transmitting them to the cloud. Alternatively, there could be connectivity issues restricting communication between the edge and the cloud. In addition, there are also security and data privacy issues that need to be addressed while sending (receiving) data to (from) the cloud. Hence, it is highly desirable to equip the edge devices with on-chip intelligence so that they can learn from real-time input data and invoke the cloud occasionally to update the on-chip trained weights using more complex algorithms. The proposed approach is also suited for building intelligent autonomous systems like robots and self-flying drones. For example, it is beneficial to embed on-chip learning in autonomous robots used for disaster relief operations that enables them to navigate obstacles and scour the disaster site for survivors. In the instance of self-flying drones used for reconnaissance operations, on-chip intelligence can enable them to effectively navigate the enemy territory and improve the chances of a successful mission.

The classification accuracy of ReStoCNet for complex applications could be improved by augmenting the layer-wise unsupervised training methodology with a global supervised training mechanism. Recent works have proposed error backpropagation algorithms for the supervised training of SNNs (Lee et al., 2016, 2018a; Panda and Roy, 2016; Jin et al., 2018; Mostafa, 2018; Wu et al., 2018). However, the backpropagation algorithms for SNNs, some of which backpropagate errors at multiple time-steps, are computationally prohibitive and prone to unstable convergence behaviors (Lee et al., 2018a). In this regard, Neftci et al. (2017) proposed event-driven random backpropagation that prevents the need for calculating and backpropagating precise error gradients. Future works could explore a hybrid unsupervised (local) and supervised (global) training methodology for ReStoCNet to obtain favorable trade-offs between classification accuracy and training effort as was shown by Lee et al. (2018a) for full-precision convolutional SNNs without residual connections. Such a hybrid approach would also preclude the need for using the pooled spiking activations of all the convolutional layers for inference, thereby enhancing the scalability of deep ReStoCNets.

4.2. Applicability of ReStoCNet for Neuromorphic Hardware Implementations

Together with research efforts that are geared toward the exploration of bio-plausible SNN algorithms (architectures and learning methodologies), parallel efforts are underway to develop neuromorphic hardware implementations with on-chip intelligence, which can exploit the inherent computational efficiency offered by the SNN algorithms. IBM *TrueNorth* (Merolla et al., 2014) and Intel *Loihi* (Davies et al., 2018) are recent demonstrations of event-driven neuromorphic hardware that were realized using the conventional CMOS technology. CMOS-based neuromorphic hardware implementations are area- and power-intensive because of the mismatch between the spiking neuronal/synaptic circuits and the neuroscience processes governing their dynamics. In this regard, nanoelectronic devices such as Ag-Si memristor (Jo et al., 2010), Phase-Change Memory (PCM) (Suri et al., 2011), Resistive Random Access Memory (Rajendran et al., 2013) and domain-wall Magnetic Tunnel Junctions (MTJs) (Sengupta et al., 2016a) that are capable of naturally mimicking multilevel synaptic dynamics have been proposed as potential candidates for achieving improved energy efficiency compared to CMOS-only realizations. However, as the technology is scaled, the multilevel memristive and spintronic devices suffer from limited bit-precision and exhibit stochastic behavior in the presence of thermal noise. The proposed ReStoCNet, which is composed of binary kernels trained using probabilistic HB-STDP, is naturally suited for neuromorphic hardware implementations based on stochastic device technologies as elaborated in the following paragraph.

Stochastic device technologies such as Conductive-Bridge Random Access Memory (CBRAM) (Suri et al., 2013), RRAM (Kavehei and Skafidas, 2014), MTJ (Vincent et al., 2015; Sengupta et al., 2016b; Srinivasan et al., 2016), and PCM (Tuma et al., 2016) have been shown to efficiently implement stochastic neuronal and synaptic models. The intrinsic stochastic switching behavior of these devices can be exploited to realize the probabilistic switching of a binary synapse during training without the need for costly random number generators to implement the stochastic operations as illustrated with MTJ-based synapse. An MTJ is composed of two ferromagnetic layers, namely, a pinned layer whose magnetization is fixed and a free layer whose magnetization can be switched, separated by a tunneling oxide barrier. It exhibits two stable conductance states based on the relative orientation of the pinned layer and free layer magnetizations, which can be switched probabilistically by passing charge current through a Heavy Metal (HM) located underneath the MTJ structure. Srinivasan et al. (2016) showed that the MTJ-HM heterostructure, with independent spike-transmission and programming current paths, can efficiently realize a stochastic binary synapse. During training, the MTJ is switched probabilistically based on the time difference between pre- and post-spikes by passing the appropriate current through the HM. During inference, an input pre-spike gets modulated with the trained MTJ conductance to produce resultant current into the post-neuron. Srinivasan et al. (2016) also presented peripheral circuits required to implement an exponential probabilistic-STDP rule, which needs to be modified for realizing the proposed HB-STDP rule. We note that CBRAM, RRAM, and PCM devices can similarly be used to realize a stochastic binary synapse during training by modulating the input voltage based on spike timing (Suri et al., 2013; Kavehei and Skafidas, 2014). Crossbar-based hardware implementations based on these stochastic device technologies with on-chip learning capability have been demonstrated for efficiently realizing binary fully-connected SNNs (Suri et al., 2013; Srinivasan et al., 2016), which consists of a unique synaptic weight connecting every pair of pre- and post-neurons. Recently Wijesinghe et al. (2018) showed that weight-shared convolutional SNNs such as ReStoCNet can be mapped to crossbar-based hardware implementations. However, large-scale networks with increased number of neurons and synapses cannot be mapped to a single large crossbar due to non-idealities that could result in erroneous computations. Hardware architectures composed of multiple smaller crossbars can be used to efficiently realize large-scale networks (Shafiee et al., 2016; Ankit et al., 2017; Song et al., 2017). Finally, we note that the fully-connected classification layer in ReStoCNet, which is composed of artificial ReLU neurons, cannot be directly implemented in event-driven asynchronous neuromorphic hardware platforms. The fully-connected layer of ReLU neurons could be mapped to Integrate-and-Fire neurons post training for inference within the neuromorphic fabric as shown by Diehl et al. (2015). Alternatively, fully-connected layer of Leaky-Integrate-and-Fire neurons can be trained using spike-based backpropagation algorithms for training and/or inference within the neuromorphic fabric.

5. CONCLUSION

In this work, we proposed ReStoCNet, a residual stochastic multilayer convolutional SNN composed of binary kernels, for memory-efficient neuromorphic computing. We presented probabilistic Hybrid-STDP (HB-STDP) learning rule, integrating Hebbian and anti-Hebbian learning mechanisms, for training the binary kernels constituting ReStoCNet in a layer-wise unsupervised manner. We demonstrated up to 3-layer deep ReStoCNet and showed that residual connections are critical to enabling the deeper convolutional layers to self-learn useful high-level input features and improving the scalability of deep SNNs. ReStoCNet offered 98.54% accuracy and 39.5× kernel memory compression compared to full-precision (32-bit) convolutional SNN under iso-accuracy conditions for MNIST digit recognition.

On the CIFAR-10 dataset, ReStoCNet provided 66.23% accuracy and 21.7× kernel memory compression, albeit with 5% accuracy degradation compared to full-precision convolutional SNN. We believe that ReStoCNet, with event-driven computing capability and memory-efficient probabilistic learning with binary kernels, is ideally suited for neuromorphic hardware implementations based on CMOS and stochastic emerging device technologies like Resistive Random Access Memory, Phase-Change Memory, and Magnetic Tunnel Junctions that can potentially lead to much improved energy efficiency in battery-powered IoT edge devices.

DATA AVAILABILITY

Publicly available datasets were analyzed in this study. This data can be found here: https://www.cs.toronto.edu/~kriz/cifar.html.

AUTHOR CONTRIBUTIONS

GS wrote the paper and performed the simulations. All authors helped with developing the concepts, conceiving the experiments, and writing the paper.

FUNDING

This work was supported in part by the Center for Brain Inspired Computing (C-BRIC), one of the six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, by the Semiconductor Research Corporation, the National Science Foundation, Intel Corporation, the DoD Vannevar Bush Fellowship, and by the U.S. Army Research Laboratory and the U.K. Ministry of Defense under Agreement Number W911NF-16-3-0001.

REFERENCES

Ankit, A., Sengupta, A., Panda, P., and Roy, K. (2017). "Resparc: a reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks," in *Proceedings of the 54th Annual Design Automation Conference 2017* (Austin, TX: ACM), 27.

Bi, G.-Q., and Poo, M.-m. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci.* 18, 10464–10472. doi: 10.1523/JNEUROSCI.18-24-10464.1998

Courbariaux, M., Bengio, Y., and David, J.-P. (2015). "Binaryconnect: training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems* (Montréal, QC), 3123–3131.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 82–99. doi: 10.1109/MM.2018.112130359

Dayan, P., and Abbott, L. F. (2001). *Theoretical Neuroscience*, Vol. 806. Cambridge, MA: MIT Press.

Diehl, P. U., and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9:99. doi: 10.3389/fncom.2015.00099

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. (2015). "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Neural Networks (IJCNN), 2015 International Joint Conference on* (Killarney: IEEE), 1–8.

Ferré, P., Mamalet, F., and Thorpe, S. J. (2018). Unsupervised feature learning with winner-takes-all based stdp. *Front. Comput. Neurosci.* 12:24. doi: 10.3389/fncom.2018.00024

Goodman, D. F., and Brette, R. (2008). Brian: a simulator for spiking neural networks in python. *Front. Neuroinformatics* 2:5. doi: 10.3389/neuro.11.005.2008

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Las Vegas, NV), 770–778.

Hebb, D. (1949). The organization of behavior. New York, NY: Wiley.

Hu, Y., Tang, H., Wang, Y., and Pan, G. (2018). Spiking deep residual network. arXiv:1805.01352v1.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2017). Quantized neural networks: training neural networks with low precision weights and activations. *J. Mach. Learn. Res.* 18, 6869–6898. Available online at: http://jmlr.org/papers/v18/16-456.html

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). "Spatial transformer networks," in *Advances in Neural Information Processing Systems* (Montreal, QC), 2017–2025.

Jin, Y., Zhang, W., and Li, P. (2018). "Hybrid macro/micro level backpropagation for training deep spiking neural networks," in *Advances in Neural Information Processing Systems* (Montréal, QC), 7005–7015.

Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., and Lu, W. (2010). Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett.* 10, 1297–1301. doi: 10.1021/nl904092h

Jug, F. (2012). *On Competition and Learning in Cortical Structures*. Doctoral thesis, ETH Zurich.

Kavehei, O., and Skafidas, E. (2014). "Highly scalable neuromorphic hardware with 1-bit stochastic nano-synapses," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on* (Melbourne, VIC: IEEE), 1648–1651. doi: 10.1109/ISCAS.2014.6865468

Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., and Masquelier, T. (2018). Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Netw.* 99, 56–67. doi: 10.1016/j.neunet.2017.12.005

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

Krizhevsky, A., and Hinton, G. (2009). *Learning Multiple Layers of Features From Tiny Images*, Vol. 1. Technical report, University of Toronto, 7.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Lee, C., Panda, P., Srinivasan, G., and Roy, K. (2018a). Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning. *Front. Neurosci.* 12:435. doi: 10.3389/fnins.2018.00435

Lee, C., Srinivasan, G., Panda, P., and Roy, K. (2018b). "Deep spiking convolutional neural network trained with unsupervised spike timing dependent plasticity," in *IEEE Trans. Cogn. Dev. Syst.* doi: 10.1109/TCDS.2018.2833071. [Epub ahead of print].

Lee, J. H., Delbruck, T., and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Front. Neurosci.* 10:508. doi: 10.3389/fnins.2016.00508

Lowel, S., and Singer, W. (1992). Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity. *Science* 255, 209–212. doi: 10.1126/science.1372754

Maaten, L. v. d., and Hinton, G. (2008). Visualizing data using t-sne. *J. Mach. Learn. Res.* 9, 2579–2605. Available online at: http://www.jmlr.org/papers/v9/vandermaaten08a.html

Masquelier, T., and Thorpe, S. J. (2007). Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Comput. Biol.* 3:e31. doi: 10.1371/journal.pcbi.0030031

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 668–673. doi: 10.1126/science.1254642

Mostafa, H. (2018). Supervised learning based on temporal coding in spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 29, 3227–3235. doi: 10.1109/TNNLS.2017.2726060

Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., Thorpe, S. J., and Masquelier, T. (2018). Combining stdp and reward-modulated stdp in deep convolutional spiking neural networks for digit recognition. arXiv: 1804.00227.

Nair, V., and Hinton, G. E. (2010). "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (Haifa), 807–814.

Neftci, E. O., Augustine, C., Paul, S., and Detorakis, G. (2017). Event-driven random back-propagation: enabling neuromorphic deep learning machines. *Front. Neurosci.* 11:324. doi: 10.3389/fnins.2017.00324

Panda, P., and Roy, K. (2016). "Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition," in *2016 International Joint Conference on Neural Networks (IJCNN)* (Vancouver, BC: IEEE), 299–306.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., et al. (2017). "Automatic differentiation in pytorch," in *NIPS Workshop* (Long Beach, CA).

Pfeiffer, M., and Pfeil, T. (2018). Deep learning with spiking neurons: opportunities and challenges. *Front. Neurosci.* 12:774. doi: 10.3389/fnins.2018.00774

Querlioz, D., Bichler, O., Vincent, A. F., and Gamrat, C. (2015). Bioinspired programming of memory devices for implementing an inference engine. *Proc. IEEE* 103, 1398–1416. doi: 10.1109/JPROC.2015.2437616

Rajendran, B., Liu, Y., Seo, J.-S., Gopalakrishnan, K., Chang, L., Friedman, D. J., et al. (2013). Specifications of nanoscale devices and circuits for neuromorphic computational systems. *IEEE Trans. Electron Devices* 60, 246–253. doi: 10.1109/TED.2012.2227969

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). "Xnor-net: imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision* (Amsterdam: Springer), 525–542.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature* 323:533. doi: 10.1038/323533a0

Sengupta, A., Banerjee, A., and Roy, K. (2016a). Hybrid spintronic-cmos spiking neural network with on-chip learning: devices, circuits, and systems. *Phys. Rev. Appl.* 6:064003. doi: 10.1103/PhysRevApplied.6.064003

Sengupta, A., Panda, P., Wijesinghe, P., Kim, Y., and Roy, K. (2016b). Magnetic tunnel junction mimics stochastic cortical spiking neurons. *Sci. Rep.* 6:30039. doi: 10.1038/srep30039

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

Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J. P., Hu, M., et al. (2016). Isaac: a convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars. *ACM SIGARCH Comput. Architect. News* 44, 14–26. doi: 10.1145/3007787.3001139

Song, L., Qian, X., Li, H., and Chen, Y. (2017). "Pipelayer: a pipelined reram-based accelerator for deep learning," in *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on* (Austin, TX: IEEE), 541–552.

Song, S., Miller, K. D., and Abbott, L. F. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci.* 3:919. doi: 10.1038/78829

Srinivasan, G., Panda, P., and Roy, K. (2018). Stdp-based unsupervised feature learning using convolution-over-time in spiking neural networks for energy-efficient neuromorphic computing. *J. Emerg. Technol. Comput. Syst.* 14, 44:1–44:12. doi: 10.1145/3266229

Srinivasan, G., Sengupta, A., and Roy, K. (2016). Magnetic tunnel junction based long-term short-term stochastic synapse for a spiking neural network with on-chip stdp learning. *Sci. Rep.* 6:29545. doi: 10.1038/srep29545

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1929–1958.

Stromatias, E., Soto, M., Serrano-Gotarredona, T., and Linares-Barranco, B. (2017). An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data. *Front. Neurosci.* 11:350. doi: 10.3389/fnins.2017.00350

Suri, M., Bichler, O., Querlioz, D., Cueto, O., Perniola, L., Sousa, V., et al. (2011). "Phase change memory as synapse for ultra-dense neuromorphic systems: application to complex visual pattern extraction," in *2011 IEEE International Electron Devices Meeting (IEDM)*, (Washington, DC: IEEE), 4.

Suri, M., Querlioz, D., Bichler, O., Palma, G., Vianello, E., Vuillaume, D., et al. (2013). Bio-inspired stochastic computing using binary cbram synapses. *IEEE Trans. Electron Devices* 60, 2402–2409. doi: 10.1109/TED.2013.2263000

Tavanaei, A., Kirby, Z., and Maida, A. S. (2018). "Training spiking convnets by stdp and gradient descent," in *2018 International Joint Conference on Neural Networks (IJCNN)* (Rio de Janeiro), 1–8.

Tavanaei, A., and Maida, A. S. (2017). "Multi-layer unsupervised learning in a spiking convolutional neural network," in *2017 International Joint Conference on Neural Networks (IJCNN)* (Anchorage, AK: IEEE), 2023–2030.

Thiele, J. C., Bichler, O., and Dupret, A. (2018). Event-based, timescale invariant unsupervised online deep learning with STDP. *Front. Comput. Neurosci.* 12:46. doi: 10.3389/fncom.2018.00046

Tuma, T., Pantazi, A., Le Gallo, M., Sebastian, A., and Eleftheriou, E. (2016). Stochastic phase-change neurons. *Nat. Nanotechnol.* 11, 693–699. doi: 10.1038/nnano.2016.70

Vincent, A. F., Larroque, J., Locatelli, N., Romdhane, N. B., Bichler, O., Gamrat, C., et al. (2015). Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems. *IEEE Trans. Biomed. Circ. Syst.* 9, 166–174. doi: 10.1109/TBCAS.2015.2414423

Wijesinghe, P., Ankit, A., Sengupta, A., and Roy, K. (2018). An all-memristor deep spiking neural computing system: a step toward realizing the low-power stochastic brain. *IEEE Trans. Emerging Top. Comput. Intell.* 2, 345–358. doi: 10.1109/TETCI.2018.2829924

Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. (2018). Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.* 12:331. doi: 10.3389/fnins.2018.00331

Yousefzadeh, A., Stromatias, E., Soto, M., Serrano-Gotarredona, T., and Linares-Barranco, B. (2018). On practical issues for stochastic stdp hardware with 1-bit synaptic weights. *Front. Neurosci.* 12:665. doi: 10.3389/fnins.2018.00665

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.