Temporal-Coded Deep Spiking Neural Network with Easy Training and Robust Performance

Shibo Zhou
Dept. of ECE, Binghamton Uiversity
Binghamton, USA
szhou19@binghamton.edu
Xiaohua Li
Dept. of ECE, Binghamton Uiversity
Binghamton, USA
xli@binghamton.edu
Ying Chen
(Corresponding author) Harbin Insitutde of Technology
Harbin, China
yingchen@hit.edu.cn
Sanjeev T. Chandrasekaran
Dept. of EE, University at Buffalo
Buffalo, USA
stannirk@buffalo.edu
Arindam Sanyal
Dept. of EE, University at Buffalo
Buffalo, USA
arindams@buffalo.edu

Abstract

Spiking neural network (SNN) is interesting both theoretically and practically because of its strong bio-plausibility and outstanding energy efficiency. Unfortunately, its development has fallen far behind conventional deep neural networks (DNNs) because of difficult training and lack of appropriate hardware support. In this paper, we show that deep SNNs constructed with single-spike temporal-coded non-leaky neurons can be trained easily and directly over the benchmark datasets such as ImageNet, with testing accuracy within 1% of DNNs of equivalent size and architecture. Training becomes identical to DNNs because closed-form solutions to spiking time are available for this type of neurons. To address the robustness concern over single-spike neurons, we develop a phase-domain signal processing circuit schematic for implementing the neuron and train our deep SNNs under weight quantization and noise perturbation. Our results showed a 45% gain on spiking energy efficiency and superior robustness to timing jitter, weight quantization and noise.

1 Introduction

Spiking neural networks (SNNs) are interesting both theoretically and practically. They have great theoretical significance because of their bio-inspiration nature. Neurons communicate via spike waveforms just as biological neurons. They work asynchronously, i.e., generate output spikes without waiting for all input neurons to spike. This leads to advantages such as spike sparsity, low latency and high energy efficiency that are attractive for practical applications [1].

Unfortunately, the performance of SNNs falls far behind the conventional deep neural networks (DNNs). One of the primary reasons is that SNNs are difficult to train. DNNs are formulated with the standard layer-by-layer linear mapping plus nonlinear activation whose gradients can be efficiently calculated. In contrast, for SNNs we have to deal with the temporal-domain spike waveforms that are formulated as differential equations with discrete impulses. Gradient evaluation is both difficult and time-consuming. Direct training of SNN has so far been limited to shallow network only. As an alternative, deep SNNs can be obtained without direct training by translating trained DNNs to SNNs. However, the translation sacrifices important SNN properties such as asynchronous processing, spike sparsity and energy efficiency.

Different from DNNs that can run on GPUs efficiently, SNNs should be implemented on appropriate neuromorphic hardware for full advantage. Unfortunately, there is a lack of widely accepted or available neuromorphic hardware platform for experimenting and implementing. Considering the challenge of gradient calculation, it is unclear whether the training can speed up even if such hardware is available. Therefore, a more valid approach is to develop deep SNNs that can be trained directly and efficiently in software platforms just like DNNs. Neuromorphic hardware can then be simplified to run the trained models only. This will lead to low-cost high-performance SNN devices that can potentially be more competitive to DNNs in practical applications.

In this paper, our first focus is to develop deep SNNs that can be directly trained over large datasets such as ImageNet. For this we extend the single-spike temporal-coded SNN model of [2] into deep SNNs. To address practical hardware implementation issues, especially the concern over the robustnesss of single-spike neurons, our second focus is to develop neuron circuit and investigate the robust performance of both the circuit and the deep SNNs.

Major contributions of this paper are listed as follows.

- Temporal-coded deep SNNs such as SpikingVGG16 and SpikingGoogleNet are developed and trained over the standard MNIST, CIFAR10 and ImageNet datasets. New benchmark results are obtained. To the best of our knowledge, this is the first time that direct trained SNN is reported for the ImageNet dataset.
- A phase-domain signal processing circuit schematic is designed to implement the SNN neuron with energy efficiency 45% higher than existing results. The neuron circuit is shown robust to input timing jitter and weight quantization.
- The temporal-coded deep SNNs are trained under weight quantization and noise perturbation to demonstrate their robustness in practical applications.

This paper is organized as follows. Related works are introduced in Section 2. The deep SNN is described in Section 3. Experiments are presented in Section 4. Conclusions are given in Section 5.

2 Related Works

SNN training methods can be categorized into three classes: unsupervised learning, supervised learning with indirect training, and supervised learning with direct training. For unsupervised learning, spiking timing dependent plasticity (STDP) is perhaps the most well-known one. As a biologically inspired local learning rule, STDP adjusts the weights connecting the pre- and post-synaptic neurons based on their relative spike times [3–6]. Even though it has achieved some success, the dependency on the local neuronal activities without global supervisor makes it have low performance in deep networks.

For the second class, the most successful approach is to translate trained DNNs to SNNs [1]. This was conducted by mapping DNN neuron values to SNN neuron spike rates in [7]. Rate coding is inefficient because either a long time duration or a high rate count is needed for precise rate estimation. Considering this, Zhang et al. [8] converted VGG16 to SNN with single-spike temporal coding. Although the translation approach can get the highest accuracy performance among SNNs so far, the loss of spike sparsity degrades energy efficiency. Theoretically, since DNN neuron values can be positive and negative, the use of neurons with both excitatory and inhibitory output synapses is not plausible from the biological perspective [5].

For the third class, SpikeProp [9] minimized the loss between the true firing time and the single desired firing time with gradient descent rule over soft nonlinearity models. Gardner et al. [10] applied a probability neuron model to calculate gradients. With spike rate coding, gradients were calculated after impulse spikes were approximated by smooth functions in [11, 12]. Single-spike temporal coding was adopted in [2] where direct training was demonstrated in shallow networks over the MNIST dataset. Wu et al. [13] conducted direct training based on spatio-temporal backpropagation, where "spatio" referred to layer-by-layer gradient propagation and "temporal" referred to gradient propagation through time-domain spike waveforms. In general, existing direct training approaches still fall short of efficiency and stability to deal with complicated datasets such as ImageNet.

For hardware support, neuromorphic platforms such as IBM's TrueNorth and Intel's Loihi are well known [14, 15]. Goltz et al. [16] tested shallow SNNs such as those of [2] in an ASIC neuromorphic platform. Mixed digital-analog implementations of spiking neurons were developed with memristive devices in [17, 18]. Zhou et al. [19] proposed an analog neuron circuit and used it to estimate the energy consumption of a YoLo-based deep SNN.

Mixed digital-analog hardware has the problem of noise, parameter drifting, as well as severe limitation on resources such as memory. Lathi et al.[20] studied pruning of unimportant weights and quantizing important weights to improve energy efficiency for shallow SNNs. The immunity to variations in SNNs with memristive nano-devices was studied in [21].

3 Deep SNN with Direct Training Capability

3.1 Neuron Model and Circuit Schematic

Following [2] we adopt the non-leaky integrate-and-fire (n-LIF) neuron model. The membrane potential $v_j(t)$ of the neuron j is described by

$$\frac{dv_j(t)}{dt} = \sum_i w_{ji}\kappa(t-t_i), \quad \text{where } \kappa(t) = u(t)e^{-\frac{t}{\tau}}, \tag{1}$$

w_{ji} is the weight of the synaptic connection from the input neuron i to the neuron j, t_i is the spiking time of the neuron i, $\kappa(t)$ is the synaptic current kernel function with delay exponent τ, and $u(t)$ is the unit step function. The value of neuron i is encoded in the spike time t_i. The function $\kappa(t)$ determines how the spike stimulation decays over time. A neuron is only allowed to spike once unless the network is reset or a new input pattern is presented. The bottom-left inset of Fig. 1 illustrates how this neuron works. Obviously, this neuron model supports asynchronous processing. As seen from the figure, the 3rd neuron will not spike since the output neuron spikes before it.

Analog circuit to implement such n-LIF neuron was developed in [19] based on analog voltage-domain (VD) signal processing. Prior works have also used VD signal processing to realize LIF neurons using CMOS circuits [22–24]. VD signal processing techniques require high-gain amplifiers which are energy inefficient to design in advanced CMOS nodes with low intrinsic gain. To reduce energy consumption, we propose phase-domain (PD) signal processing instead. The proposed PD signal processing is also more suitable for advanced CMOS technology and can maximally benefit from the advantages of the technology scaling.

Fig. 1 shows the circuit schematic of the n-LIF neuron designed with voltage-controlled ring oscillator (VCO) based integrators. If a voltage input $V_{in}(t)$ is applied to a ring VCO, its instantaneous phase is given by $\Phi(t) = \int 2\pi k_{vco}V_{in}(t)dt$, where k_{vco} is VCO tuning gain. Thus, a VCO acts as perfect PD integrator with infinite dc gain [25–27]. The VCO is built using a simple chain of inverters which are digital and can operate from very low supply voltages. Hence, VCO can be used as an integrator in advanced CMOS processes which allows area and power scaling unlike VD integrators.

We use a type-I phase-locked loop (PLL) to perform the summation operation in (1). T_1 and T_2 represent 2 inputs to the neuron while the resistors R_{1j} and R_{2j} denote the weights of synaptic connections of the inputs to the neuron respectively. When the inputs to VCO1 spike, output phase of the VCO1 jumps abruptly. VCO1 acts as reference clock source for the PLL. Hence, once phase of VCO1 jumps, phase of VCO2 starts rising exponentially to follow VCO1 phase since in steady state the PLL ensures that change in phase of VCO1 is exactly same as change in phase of VCO2. Assuming VCO1 phase is denoted by Φ_{ref} and VCO2 phase is denoted by Φ_{out}, the closed-loop transfer function of the PLL can be written as

$$\frac{\Phi_{out}(s)}{\Phi_{ref}(s)} = \frac{k_{pd}k_{vco}}{\frac{s^2}{w_{lf}} + s + k_{pd}k_{vco}}, \tag{2}$$

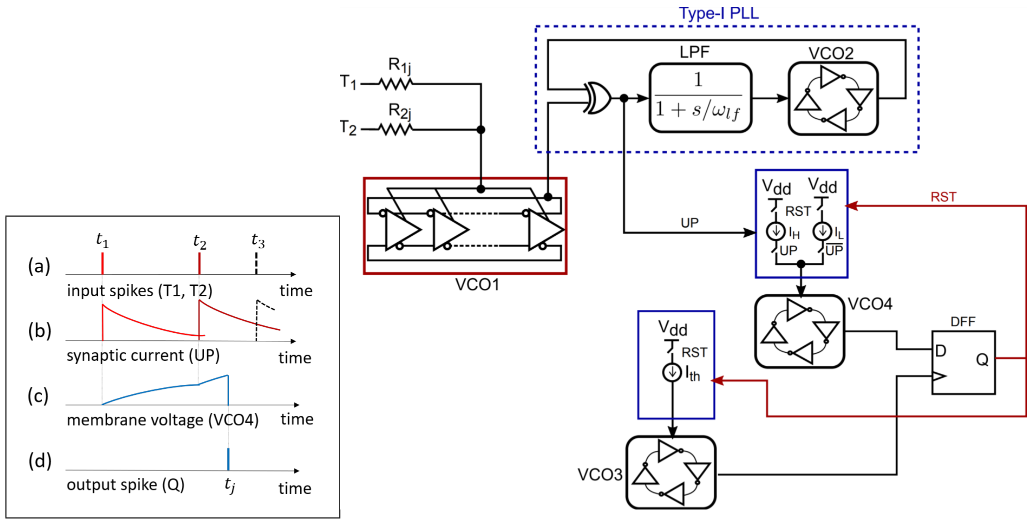

Figure 1: Circuit schematic for the n-LIF neuron. The bottom-left inset illustrates the working principle of the neuron: (a) Three input neurons spike at time t_1, t_2, t_3, respectively. (b) Synaptic current $\kappa(t - t_i)$ jumps and decays over time. (c) Membrane voltage potential $v_j(t)$ rises towards the firing threshold. (d) The neuron j emits a spike at time t_j when the threshold is crossed.

where k_{pd} is the gain of XOR phase-detector in the PLL. The PLL closed-loop transfer function is of the second order. Once VCO1 phase spikes, it acts as a step input to the transfer function in (2). Thus, Φ_{out} rises exponentially to catch up with Φ_{ref} and the difference signal $\Phi_{ref} - \Phi_{out}$ decays exponentially.

The phase difference signal is given as an input to VCO4 which acts as the second integrator. The difference signal, denoted by 'UP' in Fig. 1, is a pulse-width modulated (PWM) signal which encodes the $\Phi_{ref} - \Phi_{out}$ in the width of its pulses. When 'UP' is high, VCO4 is driven by a high current I_H, and when 'UP' is low, VCO4 is driven by a low current I_L. The phase of VCO4 is compared with the phase of a reference VCO, VCO3, which is driven by a constant current I^{th}. Once the phase of VCO4 exceeds the phase of VCO3, the 'Q' output of the DFF goes high which represents neuron firing. The current I^{th} can be changed to vary the threshold for the neuron firing. Once 'Q' goes high, both VCO3 and VCO4 phases are reset till the next input arrives.

The proposed neuron is highly scalable since it is built using digital CMOS circuits. The synaptic weights can be made tunable by using a digitally controllable resistor bank at the cost of increased area consumption. Due to its highly digital and simple architecture, the neuron consumes less than 10 pJ/spike in 65nm process and the energy consumption will reduce further with technology scaling. This leads to an energy efficiency gain of 45% over [19] as the latter had 19 pJ/spike energy consumption.

3.2 Deep SNN Formulation

For training or software implementations, we would better avoid the explicit calculation of the spike waveform $v_j(t)$. Fortunately, this can be realized by exploiting the closed-form solution of (1). Specifically, the member voltage at spiking time t_j can be solved as

$$v_j(t_j) = \sum_{\forall i \in C = \{k : t_k < t_j\}} w_{ji}\tau \left(1 - e^{-\frac{t_j - t_i}{\tau}}\right), \tag{3}$$

where the set C includes all (and only those) input neurons that spike before t_j. Obviously, larger τ leads to lower $v_j(t_j)$. For each τ, there exists an appropriate voltage threshold $v_j(t_j)$ for any fixed input neuron set C and output spike time t_j. Therefore, in software implementation, we can simply set voltage threshold $v_j(t_j) = 1$ and $\tau = 1$. The neuron j's spike time thus satisfies

$$e^{t_j} = \sum_{i \in C} e^{t_i} \frac{w_{ji}}{\sum_{\ell \in C} w_{j\ell} - 1}. \tag{4}$$

Table 1: Proposed models. SCNN(5,32,2) means SCNN layer with 32 5×5 kernels and stride 2. FC(10) means FC layer with 10 output neurons. MP(2) means 2×2 Max-Pooling layer.

MNIST	SCNN(5,32,2) $\to$ SCNN(5,16,2) $\to$ FC(10)
CIFAR10	**SpikingVGG16**: SCNN(3,64,1) $\to$ SCNN(3,64,1) $\to$ MP(2) $\to$ SCNN(3,128,1) $\to$ SCNN(3,128,1) $\to$ MP(2) $\to$ SCNN(3,256,1) $\to$ SCNN(3,256,1) $\to$ SCNN(3,256,1) $\to$ MP(2) $\to$ SCNN(3,512,1) $\to$ SCNN(3,512,1) $\to$ SCNN(3,512,1) $\to$ MP(2) $\to$ SCNN(3,1024,1) $\to$ SCNN(3,1024,1) $\to$ SCNN(3,1024,1) $\to$ MP(2) $\to$ FC(4096) $\to$ FC(4096) $\to$ FC(512) $\to$ FC(10)
ImageNet	**SpikingGoogleNet**: replace GoogleNet CNN/FC layers with SCNN/FC layers

Let e^{t_i} and e^{t_j} be the input and output neuron values in software implementation. Then (4) is the input-output mapping of a feed-forward fully connected neural network layer. We do not need other nonlinear activation because the composite weights $w_{ji}/\sum_{\ell \in C} w_{j\ell} - 1$ are nonlinear.

This SNN model was initially developed in [2]. Nevertheless, only some shallow networks with $2 \sim 3$ fully-connected layers were proposed and experimented over the simple XOR and MNIST dataset with mediocre performance, which was perhaps the reason that this work did not arouse too much interest. In this paper, by observing the similarity of (4) with the conventional DNN linear mapping $Y = X \cdot W$, we extend the work of [2] into deep SNNs and improve the training for better convergence. With thorough experiments, we find that deep SNNs can be constructed with the architectures of DNNs and can achieve classification accuracy within 1% of the latter or even better.

Our deep SNNs can be constructed with multiple convolutional layers, pooling layers and fully connected layers. With time-to-first-spike (TTFS) encoding, the classification result is made at the time of the first spike among output neurons. Based on the input-output expression (4), both spiking fully-connected (FC) layers and spiking convolutional neural network (SCNN) layers can be constructed just as conventional DNN FC and CNN layers. The Max-Pooling layer is much simpler because the output is just the first arrived input, i.e., the one with the smallest e^{t_i} among the inputs. Such asynchronous processing leads to reduction of latency and energy consumption. In contrast, Average-Pooling does not have this advantage.

For an L-layer deep SNN, define the input as $\mathbf{z}_0$ with elements $z_{0,i} = e^{t_{0,i}}$ and the final output as $\mathbf{z}_L$ with elements $z_{L,i} = e^{t_{L,i}}$. Then we have $\mathbf{z}_L = f(\mathbf{z}_0; \mathbf{w})$ with nonlinear mapping f and trainable weight $\mathbf{w}$ which includes all weights w_{ji}^{ℓ}. Let the targeting output be class c. We train the network with the loss function

$$\mathcal{L}(\mathbf{z}_L, c) = -\ln \frac{z_{L,c}^{-1}}{\sum_{i \neq c} z_{L,i}^{-1}} + K \sum_{\ell=1}^{L} \sum_{j} \max \left\{ 0, \beta - \sum_{i} w_{ji}^{\ell} \right\} + \lambda \sum_{\ell=1}^{L} \sum_{j,i} (w_{ji}^{\ell})^2. \quad (5)$$

The first term is to make $z_{L,c}$ the smallest (equivalently $t_{L,c}$ the smallest) one. The second term (with $\beta \geq 1$) is the weight sum cost, which enlarges each neuron's input weight summation to increase its firing probability. The third term is L_2 regularization to prevent weights from becoming too large. The parameters K and λ are weighting coefficients.

Thanks to the closed-form expression (4), gradient calculation is easy and gradient-based back-propagation can be used to train the weights. The training becomes nothing different from conventional DNNs. In contrast, SNNs made with LIF neurons usually do not have such easy training because no closed-form solutions exist. For example, even though the differential equation $dv/dt + bv = we^{-t}$ has solution $v(t) = (e^{-at} - e^{-(2a+1)t})w/(a+1)$, there is no closed-form solution of t_j to $v(t_j) = 1$ in general.

4 Experiments

In this section, we report our experiments over three standard datasets, namely, MNIST, CIFAR-10 and ImageNet. We designed three temporal-coded deep SNN models for them, which are shown in Table 1. We evaluated and compared their testing accuracy over existing state-of-the-art models. We also evaluated their robustness under weight quantization and noise perturbation.

Table 2: Classification accuracy comparison. DT: direct training. tran: translate SNN.

Dataset	Models	Method	Accuracy	Weights(million)	Sparsity
MNIST	Ciresan'11 [28]	CNN	99.73%	0.069	no
	Kheradpisheh'18 [5]	SNN+STDP	98.40%	0.076	no
	Lee'18 [6]	SNN+STDP	91.10%	0.025	-
	Mostafa'18 [2]	SNN+DT	97.55%	0.635	0.51
	Zhang'19 [8]	SNN+tran	99.08%	3.9	no
	Wu'19 [29]	SNN+DT	99.26%	0.051	-
	Our Model	**SNN+DT**	**99.33%**	**0.021**	**0.94**
CIFAR10	VGG16 [30]	CNN	91.55%	15	no
	Huang'19 [31]	CNN	99.00%	556	no
	Tan'19 [32]	CNN	98.90%	64	no
	Hunsberger'15 [11]	SNN+tran	82.95%	39	no
	Reuckauer'17 [7]	SNN+tran	90.85%	62	no
	Wu'19 [13]	SNN+DT	90.53%	45	-
	Our Model	**SNN+DT**	**92.68%**	**34**	**0.62**
ImageNet	AlexNet [33]	CNN	57.2%	60	no
	VGG16 [34]	CNN	71.5%	138	no
	GoogleNet [35]	CNN	69.8%	6.8	no
	MobileNet V2 [36]	CNN	70.6%	3.4	no
	Tan'19 [32]	CNN	84.4%	66	no
	Zhang'19 [8]	SNN+tran	80.4%	138+	no
	Our Model	**SNN+DT**	**68.8%**	**6.8**	**0.56**

4.1 Classification Accuracy

MNIST: The MNIST image pixels were normalized to $p_i \in [0, 1]$ and encoded into spiking time $t_i = \alpha(-p_i + 1)$. The parameter α was used to adjust spike temporal separation. We trained the network for 50 epochs with batch size 10 using the Adam optimizer. The learning rate started as 0.001 and gradually reduced to 0.0001 at the last epoch, with learning decay lr_decay = (learning_end -learning_start)/50. We set $K = 100$, $\lambda = 0.001$, and $\beta = 1$. According to (4), gradients could become very large in case $\sum_\ell w_{j\ell}$ is near 1, which is harmful to training. Therefore, we limited the maximum allowed row-normalized Frobenious norm of the gradient of each weight matrix to 10.

We trained this network over both noisy input spike times and non-noisy input spike times, and found that the former generally outperformed the latter. Therefore, we report only the results of noisy inputs. Classification accuracy is shown in Table 2. The proposed network had the highest accuracy (99.33%) among the SNNs listed in the table, yet had the smallest network size with the least number of trainable weights (21K). Because of the asynchronous operation of our SNN, on average 94% neurons spiked, which leaded to sparsity 0.94. The total consumed energy of the proposed network was 205 nJ assuming each spike cost 10 pJ based on the proposed neuron circuit and sparsity 0.94.

CIFAR-10: Our SNN model for CIFAR-10 was developed based on the VGG16 model [30][34], hence called SpikingVGG16 or sVGG16. To encode image pixels to spike time, we applied encoding rule $t_i = \alpha p_i$ and used a relative large α to enlarge spike time separation so that all pixel values could potentially be used. We exploited data augmentation such as crop, flip and whiten to considerably increase the diversity of data available for training. Learning rate started from 0.01 and ended at 0.0001. We ran 320 epochs and after 240 epochs the training tended to converge. Batch size was 128. The other hyper-parameters were the same as the MNIST experiments.

As shown in Table 2, the testing accuracy of our SpikingVGG16 reached 92.68%, higher than all the listed SNNs including the state-of-the-art of [13]. With a total of 34 million trainable weights, our network size was smaller that most others in the table. Especially, compared with the CNN-based VGG16 [30], our model was even better (It was within 1.0% of the best VGG16 result that we could find online, which was 93.56%). Our network enjoyed a sparsity of 0.62 which means only 62% neurons sent spikes. It consumed 0.336 mJ of energy for each image inference.

ImageNet: We built our deep SNN model based on the popular GoogleNet architecture [35], hence called SpikingGoogleNet or sGoogleNet. We simply replaced CNN layers with SCNN layers and FC layers with spiking FC layers. We used 1.2 million of images to train the network and 0.1 million of images to test the accuracy of the network. Training parameters were similar to the CIFAR10 experiments, and the training procedure followed that of GoogleNet. The input image was $224 \times 224 \times 3$ and was randomly cropped from a resized image using the scale and aspect ratio augmentation. We used SGD with a batch size of 256, learning rate decay 0.0001 and momentum 0.9. We started from a learning rate of 0.1. When the training error was no longer reducing, retraining and fine-tuning with very small learning rate were conducted until the test accuracy no longer increased.

Results of Top-1 testing accuracy are shown in Table 2, compared with some widely cited models. As can be seen, our SpikingGoogleNet achieved 68.8% accuracy, which is only 1.0% lower than the CNN-based GoogleNet. Our network enjoyed a sparsity of 0.56 which means only 56% of neurons were activated during image inference on average. The total energy consumption was thus 0.038 mJ based on our neuron circuit.

To the best of our knowledge, few work was reported over the CIFAR-10 dataset using direct training SNN (except [13]) and none was reported over the ImageNet dataset. Our models hence set up new benchmark accuracy in both cases. Obviously, there is still a big gap between SNN and DNN in classification accuracy. Nevertheless, our experiments showed that our proposed SNNs could achieve similar accuracy as the DNNs with similar network size and architecture. This fact was also demonstrated in many translate SNN works. New and high benchmark results of DNN were mostly obtained with neuro-architecture search and optimization techniques [32]. We expect that SNN performance could catch up rapidly by utilizing these new architectures or applying neuro-architecture search techniques.

4.2 Robustness

Neuron: For the neuron circuit shown in Fig. 1, while the VCO based PD signal processing is very energy efficient and suitable for design in advanced CMOS processes, a disadvantage is that it is very sensitive to changes in operating conditions such as supply voltage or temperature. We have used differential architecture in our proposed design in which VCO phase is always compared with phase from a reference VCO rather than using the absolute phase from a single VCO. While this architectural choice doubles area and energy consumption, it also increases robustness to changes in operating condition.

For temporal-coded neurons, an important distortion measure is spike timing jitter, which is defined as $|t_j^{\rm measured} - t_j^{\rm desired}|/t_j^{\rm desired}$. For mixed digital-analog circuits, there are quantization noise and circuit noise, which we simply combine into synaptic weight quantization noise. We conducted circuit simulation under various input timing jitters and weight quanizations, and the results are summarized in Table 3. Both input jitter and quantization of synaptic weights change temporal location of the output spikes. As the input jitter was swept from 1-9%, the output jitter varied from 0.024% to 0.997% only. Such a good inherent jitter suppression capability was due to the PLL which low-pass-filtered input jitter. It was also because the output spike time t_j was always bigger than the input spike time t_i in temporal coding.

Quantizing synaptic weights (for 2-input case) increased output jitter from 0.004% in case of 32-bit quantization to 0.957% in case of 4-bit quantization. Coarse quantization thus leaded to large jitter. One of the reasons was that in our neuron model (4) the denominator $\sum_\ell w_{j\ell} - 1$ could be small and a slight change of weights $w_{j\ell}$ might cause big change in t_j. To mitigate this problem, we can set $\beta > 1$ in (5) during training. Nevertheless, examining the numbers in Table 3 more clearly, 4-bit quantization has quantization SNR (signal to noise ratio) $6.02 \times 4 \approx 24$ dB, while output jitter of 0.957% means SNR $20\log_{10}(1/.00957) \approx 40$ dB. This shows that the jitter was small and the neuron was robust to weight quantization.

Deep SNN: To evaluate the robustness of deep SNNs, we experimented both weight quantization and noise perturbation. For weight quantization, the weights were quantized to 32-bit, 8-bit, 4-bit and 2-bit words. We retrained the deep SNNs following the procedure introduced in conventional CNN quantization [37][38]. Specifically, the forward inference used quantized weights while the backward gradient propagation used full-precision weights. We also gradually reduced the weight quantization level from 32-bit to 2-bit during training.

Table 3: Jitter in output spike versus input jitter and synaptic weight quantization.

input jitter (%)	output jitter (%)
1	0.024
3	0.189
5	0.397
7	0.789
9	0.997

quantization level	output jitter (%)
4-bit	0.957
8-bit	0.268
16-bit	0.008
24-bit	0.004
32-bit	0.004

Table 4: Accuracy versus weight quantization. Note: 8,4,2-bit quantizations correspond to quantization SNR 48, 24, 12dB.

Model	Bits	MNIST	CIFAR10	ImageNet
Our Models	32	99.33%	92.68%	68.8%
	8	99.32%	91.87%	66.1%
	4	99.21%	91.38%	65.2%
	2	99.11%	90.93%	60.0%
Chen'18 [39]	32	98.66%	84.80%	-
	8	98.48%	84.07%	-
	2	96.34%	81.56%	-
Zhang'18 [40]	32	-	92.1%	70.3%
	4	-	-	70.0%
	2	-	91.8%	68.0%

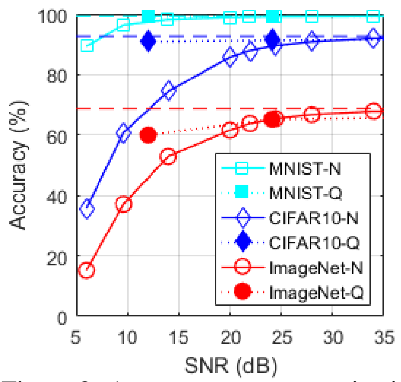

Figure 2: Accuracy versus quantization (Q) and noise (N). Dashed Lines: 32-bit quantization or noiseless.

Table 4 shows the testing accuracy under weight quantization. For the models over MNIST, CIFAR10 and ImageNet, weight quantization caused worst accuracy loss of 0.22%, 1.75%, and 8.8%, respectively. As comparison, we listed two typical CNN weight quantitation results, which indicated similar performance degradation pace. The results demonstrated that the SNN models were robust to weight quantization for relatively small datasets and small networks. For larger networks, the weights would better be encoded in 4-bit or over.

For noise perturbation, we added random noise to the trained weights. Retraining was not applied because during previous experiments we found that noisy inputs could enhance performance. Experiment results are shown in Fig. 2. Because Table 3 indicated that neuron noise (explained as jitter) was usually very small, we just need to pay attention to high SNR scenarios. With noise at SNR 25dB or quantization at 4-bit (24 dB) or over, accuracy reduction was negligible. For example, 4-bit quantization reduced ImageNet accuracy to 65.2% while noise at 24dB SNR reduced ImageNet accuracy to 65.43%, about 3.5% performance loss. Such results indicated that the deep SNNs were robust to both weight quantization and noise perturbation.

5 Conclusion

In this paper we develop temporal-coded deep SNNs that can be easily trained over large datasets such as ImageNet with classification accuracy within 1% of the corresponding DNNs of similar size and architecture. New benchmark accuracy results for direct train SNNs are obtained. A circuit schematic of the n-LIF neuron is designed with phase-domain signal processing, which shows 45% energy efficiency gain over existing state of the art. Both the neuron and the deep SNNs are demonstrated as robust to timing jitter, weight quantization and noise. The easy training, high performance and robustness indicate that SNNs can be competitive to DNNs and are promising for practical applications.

Broader Impact

This paper set up new benchmark accuracy records of deep spiking neural networks (SNNs) over large datasets such as CIFAR10 and ImageNet. The novel deep SNN can be easily trained like conventional

DNN, which can stimulate the future development of SNN. Our results indicate that SNN can potentially be as better as conventional DNN. Considering the fact that SNN can be implemented over energy efficient and fast neuromorphic hardware, this paper can promote the practical application of SNN technology. The entire AI research community will benefit from this research.

References

[1] Tavanaei, A., M. Ghodrati, S. R. Kheradpisheh, et al. Deep learning in spiking neural networks. *Neural Networks*, 111:47–63, 2019.

[2] Mostafa, H. Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7):3227–3235, 2017.

[3] Caporale, N., Y. Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

[4] Diehl, P. U., M. Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

[5] Kheradpisheh, S. R., M. Ganjtabesh, S. J. Thorpe, et al. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018.

[6] Lee, C., G. Srinivasan, P. Panda, et al. Deep spiking convolutional neural network trained with unsupervised spike-timing-dependent plasticity. *IEEE Transactions on Cognitive and Developmental Systems*, 11(3):384–394, 2018.

[7] Rueckauer, B., I.-A. Lungu, Y. Hu, et al. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

[8] Zhang, L., S. Zhou, T. Zhi, et al. Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pages 1319–1326. 2019.

[9] Bohte, S. M., J. N. Kok, H. La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002.

[10] Gardner, B., I. Sporea, A. Grüning. Learning spatiotemporally encoded pattern transformations in structured spiking neural networks. *Neural computation*, 27(12):2548–2586, 2015.

[11] Hunsberger, E., C. Eliasmith. Spiking deep networks with lif neurons. *arXiv preprint arXiv:1510.08829*, 2015.

[12] Lee, J. H., T. Delbruck, M. Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

[13] Wu, Y., L. Deng, G. Li, et al. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pages 1311–1318. 2019.

[14] Merolla, P. A., J. V. Arthur, R. Alvarez-Icaza, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[15] Davies, M., N. Srinivasa, T.-H. Lin, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[16] Göltz, J., A. Baumbach, S. Billaudelle, et al. Fast and deep neuromorphic learning with time-to-first-spike coding. *arXiv preprint arXiv:1912.11443*, 2019.

[17] Kuzum, D., S. Yu, H. P. Wong. Synaptic electronics: materials, devices and applications. *Nanotechnology*, 24(38):382001, 2013.

[18] Tuma, T., A. Pantazi, M. Le Gallo, et al. Stochastic phase-change neurons. *Nature nanotechnology*, 11(8):693, 2016.

[19] Zhou, S., Y. Chen, X. Li, et al. Deep scnn-based real-time object detection for self-driving vehicles using lidar temporal data. *IEEE Access*, 2020.

[20] Rathi, N., P. Panda, K. Roy. Stdp-based pruning of connections and weight quantization in spiking neural networks for energy-efficient recognition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(4):668–677, 2018.

[21] Querlioz, D., O. Bichler, P. Dollfus, et al. Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Transactions on Nanotechnology*, 12(3):288–295, 2013.

[22] Indiveri, G. A low-power adaptive integrate-and-fire neuron circuit. In *IEEE International Symposium on Circuits and Systems*, vol. 4, pages IV–IV. 2003.

[23] Wijekoon, J. H., P. Dudek. A CMOS circuit implementation of a spiking neuron with bursting and adaptation on a biological timescale. In *IEEE Biomedical Circuits and Systems Conference*, pages 193–196. 2009.

[24] Wu, X., V. Saxena, K. Zhu, et al. A CMOS spiking neuron for brain-inspired neural networks with resistive synapses and in-situ learning. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(11):1088–1092, 2015.

[25] Sanyal, A., N. Sun. A 18.5-fj/step vco-based 0–1 mash $\delta - \sigma$ adc with digital background calibration. In *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, pages 1–2. IEEE, 2016.

[26] Taylor, G., I. Galton. A mostly-digital variable-rate continuous-time delta-sigma modulator adc. *IEEE Journal of Solid-State Circuits*, 45(12):2634–2646, 2010.

[27] Perrott, M., et al. A 12-bit 10-mhz bandwidth continuous-time adc with a 5-bit 950-ms/s vco-based quantizer. *IEEE J. Solid-State Circuits*, 2008.

[28] Ciresan, D. C., U. Meier, L. M. Gambardella, et al. Convolutional neural network committees for handwritten character classification. In *2011 International Conference on Document Analysis and Recognition*, pages 1135–1139. IEEE, 2011.

[29] Wu, J., Y. Chua, M. Zhang, et al. Deep spiking neural network with spike count based learning rule. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2019.

[30] Liu, S., W. Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian conference on pattern recognition (ACPR)*, pages 730–734. IEEE, 2015.

[31] Huang, Y., Y. Cheng, A. Bapna, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*, pages 103–112. 2019.

[32] Tan, M., Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[33] Krizhevsky, A., I. Sutskever, G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. 2012.

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

[35] Szegedy, C., W. Liu, Y. Jia, et al. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9. 2015.

[36] Sandler, M., A. Howard, M. Zhu, et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520. 2018.

[37] Li, F., B. Zhang, B. Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[38] Rastegari, M., V. Ordonez, J. Redmon, et al. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[39] Cheng, H.-P., Y. Huang, X. Guo, et al. Differentiable fine-grained quantization for deep neural network compression. *arXiv preprint arXiv:1810.10351*, 2018.

[40] Zhang, D., J. Yang, D. Ye, et al. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382. 2018.