Connection Pruning for Deep Spiking Neural Networks with On-Chip Learning

Thao N.N. Nguyen*, Bharadwaj Veeravalli†, and Xuanyao Fong‡

Department of Electrical and Computer Enginnering, National University of Singapore

Abstract

Long training time hinders the potential of the deep Spiking Neural Network (SNN) with the online learning capability to be realized on the embedded systems hardware. Our work proposes a novel connection pruning approach that can be applied during the online Spike Timing Dependent Plasticity (STDP)-based learning to optimize the learning time and the network connectivity of the SNN. Our connection pruning approach was evaluated on a deep SNN with the Time To First Spike (TTFS) coding and has successfully achieved 2.1x speed-up in the online learning and reduced the network connectivity by 92.83%. The energy consumption in the online learning was saved by 64%. Moreover, the connectivity reduction results in 2.83x speed-up and 78.24% energy saved in the inference. Meanwhile, the classification accuracy remains the same as our non-pruning baseline on the Caltech 101 dataset. In addition, we developed an event-driven hardware architecture on the Field Programmable Gate Array (FPGA) platform that efficiently incorporates our proposed connection pruning approach while incurring as little as 0.56% power overhead. Moreover, we performed a comparison between our work and the existing works on connection pruning for SNN to highlight the key features of each approach. To the best of our knowledge, our work is the first to propose a connection pruning algorithm that can be applied during the online STDP-based learning for a deep SNN with the TTFS coding.

*thaonnnguyen@u.nus.edu

†elebv@nus.edu.sg

‡kelvin.xy.fong@nus.edu.sg

1 Introduction

SNN has been increasingly used in the energy-aware real-time applications on the embedded systems platforms [1–3]. Due to its event-driven nature, SNN consumes less energy while performing as accurately as the Artificial Neural Network (ANN) [4]. However, the SNN architectures that achieve the high accuracy are usually deep and large, consisting of multiple layers and thousands of neurons and connections. For example, the SNN architecture proposed in [4] consists of 6 layers with more than 14,500 neurons and 29,000 connections, and achieved an accuracy of 99.14% on the MNIST dataset. The authors of [5] implemented an SNN based on the VGG-16 architecture [6], which consists of more than 138 millions of connections, and achieved an accuracy of 65.19% on the ImageNet dataset. These large-scale networks require long computation time, large hardware resources and high energy consumption. This limits the potential of these SNN architectures to be realized on the embedded systems platforms. Therefore, we are motivated to explore the techniques to compress the SNN to optimize the computation time, hardware resources, and energy consumption on the neuromorphic hardware. Our work aims to improve both the learning and the inference time of the SNN while minimizing the loss in the accuracy.

There are two common approaches to convert the continous inputs to the spike events: rate-based coding and temporal coding. In the rate-based coding, the input value is encoded into the spike frequency. An input that has a large value is converted to a series of spike events that occur frequently. On the other hand, in the temporal coding, the input value is encoded into the spike timing. Each neuron spikes at most once for every input. An input that has a large value is converted to a spike event that occurs early. The latter approach consumes less energy as a fewer number of spikes are generated [7]. In the temporal coding, the input information can be encoded into the relative order (rank order coding) or the latency (TTFS coding) of the spikes. The TTFS coding has been used in many existing works that achieved more than 95% accuracy in various applications [8, 9]. Therefore, our work focuses on developing the connection pruning algorithm for the SNN with the TTFS coding.

Connection pruning is a compression technique that has been applied to ANN [10, 11] and SNN [12–18] to reduce the network complexity and energy consumption. It has been shown that more than half of the connections in a well-performing neural network can be removed with minimal impact on the classification accuracy [10, 12]. The connection pruning can be performed either on a pre-trained network [13, 15] or during the network learning [14, 16–18]. The authors of [13] proposed a heuristic connection pruning algorithm for the pre-trained SNN, in which the connection weights are obtained by converting from those of an ANN. The connection pruning is triggered periodically during the inference stage, based on the neuron parameters such as spike rate, membrane potential, and connectivity. Although this approach has efficiently improved the number of operations and hardware energy during the inference stage, it cannot be applied to the applications that require the online learning capability on hardware [1, 2, 19]. In addition, the work in [13] forcused on the SNN with the rate-based coding, in which a neuron spikes multiple times during the feed-forward computation of an input image. The spike rate indicates the activity of the neurons and varies greatly among the neurons in the network. In this case, it can be used as a pruning parameter to remove a higher percentage of connections from the less active neurons. However, in the TTFS coding, each neuron spikes at most once in the feed-forward computation. Consequently, the spike rate does not vary as much as in the rate-based coding and cannot be used as a pruning parameter. Therefore, the works in [13] and [16–18], which depend on the spike rate of the neurons to prune the connections, are not applicable to the SNN with the energy-efficient TTFS coding. Our work is similar to the work in [13] in that (i) the pruning is performed periodically based on the network behaviors and the spike activities and (ii) a hardware architecture was implemented which efficiently incorporates our connection pruning algorithm to optimize the delay and energy consumption. However, our pruning parameters are applicable to the TTFS coding. In addition, our connection pruning approach can be applied during the STDP-based learning to support the real-life applications that require the online learning capability.

Prior to our work, the authors of [14] have proposed an algorithm that skips the membrane potential update when the connection weight is below a threshold during the STDP-based learning. The connection is not eliminated from the network; it can still participate in the STDP-based learning and has a chance to be strengthen in the future. Although this approach helps to reduce the number of membrane potential updates in the feed-forward computation, it does not reduce the number of weight updates in the STDP-based learning. On the other hand, our proposed connection pruning approach completely eliminates the connections during the STDP-based learning, which reduces the updates in both the membrane potentials and the synaptic weights. The highlights of our work are as follows:

- A novel connection pruning approach was proposed to prune the SNN with the TTFS coding during the online STDP-based learning. Our approach has successfully improved the learning time by 2.1x, reduced the network connectivity by 92.83%, and saved 64% energy consumption during the online learning without incurring any loss in the classification accuracy on the Caltech 101 dataset. In addition, the inference time was speeded-up by 2.83x and the energy consumption during the inference was reduced by 78.24%.
- A hardware architecture was implemented to efficiently perform our proposed connection pruning approach during the online STDP-based

learning. Our hardware implementation for the connection pruning incurs as little as 0.56% overhead in the power consumption.

- Our proposed connection pruning approach and hardware implementation were evaluated on a deep, large-scale SNN consisting of three convolutional layers, 229,376 neurons, and 33,220 connections. To the best of our knowledge, our work is the first to implement a connection pruning approach that can be applied during the online STDP-based learning for the deep SNN augnmented with Convolutional Neural Networks (CNN). Note that it is more challenging to apply the connection pruning on a deep SNN than on a one-layer SNN as the learning errors may propagate through the network layers, causing more spuriousness in the deep layers.

The rest of our paper is structured as follows: Section 2 provides an overview of SNN and the STDP-based learning algorithm. Our connection pruning approach is proposed in Section 3, followed by our hardware architecture in Section 4. Section 5 describes our experimental setup and Section 6 analyses the experimental results of our proposed connection pruning approach. Finally, we conclude the paper and discuss the future directions in Section 6.

2 Preliminaries

In this section, we will present an overview of SNN, the neuron model, and the hardware-friendly STDP-based learning algorithm applied in our work. In SNN, the inputs are discrete spike events. When a spike event arrives at a neuron, its synaptic weight is accumulated in the membrane potential of the neuron. In our work, the potential accumulation follows the Integrate and Fire (IF) model [20], formulated in the following equation:

$$v_i(t) = v_i(t-1) + \sum_j w_{j,i} s_j(t-1) \tag{1}$$

where

$$s_i(t) = \begin{cases} 1 & \text{if neuron } i \text{ spikes at time } t \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

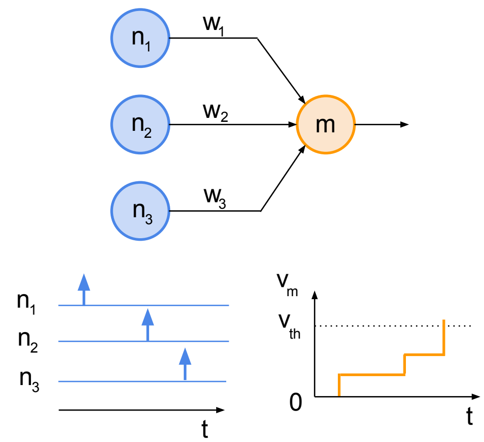


Figure 1: An example of the potential accumulation in the post-synaptic neuron m connected to three pre-synaptic neurons n_1, n_2, and n_3. The connection weights are w_1, w_2, and w_3 respectively. When there is a spike from one of the pre-synaptic neurons, the membrane potential V_m of neuron m is increased by the corresponding weight. When V_m exceeds the firing threshold V_{th}, neuron m emits a post-synaptic spike.

$v_i(t)$ and $s_i(t)$ are the membrane potential and post-synaptic spike of neuron i at time t, $w_{j,i}$ is the synaptic weight between neuron i and neuron j. As soon as the membrane potential exceeds the firing threshold, the neuron fires a post-synaptic spike and inhibits the other neurons that are close to it in the network. The overview of this feed-forward stage is illustrated in Figure 1.

The STDP-based learning algorithm in SNN is performed based on the correlation between the pre-synaptic spike and post-synaptic spike of a connection. If the post-synaptic spike occurs within a small time window from the pre-synaptic spike, the two spikes are considered correlated and the connection is strengthen. This process is called Long Term Potentiation (LTP). Otherwise, the two spikes are considered not correlated and the connection is weaken. This process is called Long Term Depression (LTD) [21].

In our work, we applied the hardware-friendly SNN

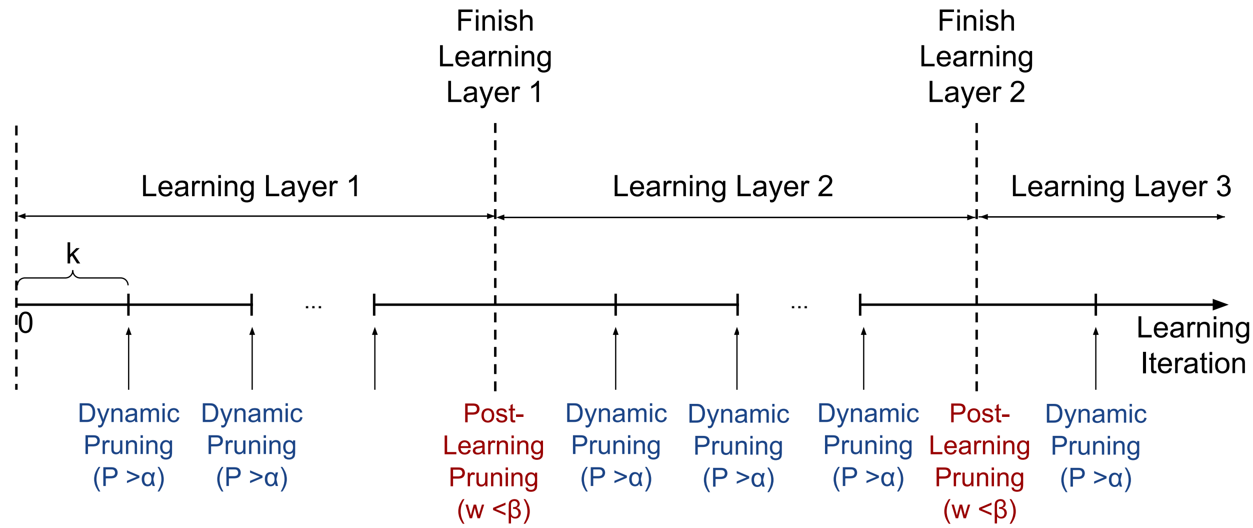


Figure 2: Overview of our two-stage connection pruning approach.

and STDP-based learning model proposed in [8] for image recognition tasks. The input images are pre-processed by a Difference of Gaussians (DoG) filter and converted to spike events using the TTFS coding. The weight update in the STDP-based learning is computed using the following equation:

$$\Delta w = \begin{cases} a^{+} w_{ij}(1 - w_{ij}) & \text{if } t_j - t_i \leq 0 \\ a^{-} w_{ij}(1 - w_{ij}) & \text{otherwise} \end{cases} \tag{3}$$

where a^{+} and a^{-} are the learning rates, t_j and t_i are the time of the pre-synaptic and post-synaptic spikes respectively. The values of the connection weights are in the range [0, 1]. The STDP-based learning is performed layer by layer: the learning of a layer only starts after the learning of its the preceeding layer finishes. In the next section, we will describe our connection pruning approach for this SNN model.

3 Proposed Connection Pruning Approach

In this section, we will propose our novel connection pruning approach that helps to compress the SNN and reduce the online STDP-based learning time. Our approach consists of two stages: (i) dynamic pruning during the online STDP-based learning in every layer and (ii) post-learning pruning after each layer is learned, as shown in Figure 2. Note that both of these stages are performed during the online STDP-based learning. We will describe each of these stages in detail in the following.

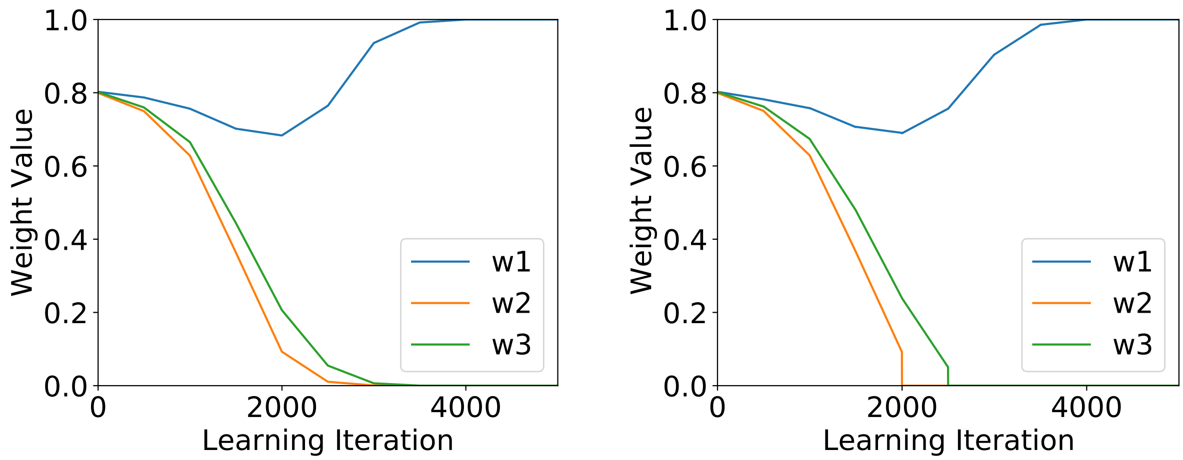
3.1 Dynamic Pruning During The Online STDP-Based Learning

In the dynamic pruning stage, the connection pruning is performed after every k iterations during the online STDP-based learning of each layer, based on two parameters: (i) weight update history h of the synaptic connection and (ii) time of post-synaptic spike t_{post}. In the following, we will explain these parameters and present our dynamic pruning approach in detail.

3.1.1 Weight Update History

Our weight update history parameter h is defined in the following equation:

$$h = \frac{d}{w} \tag{4}$$

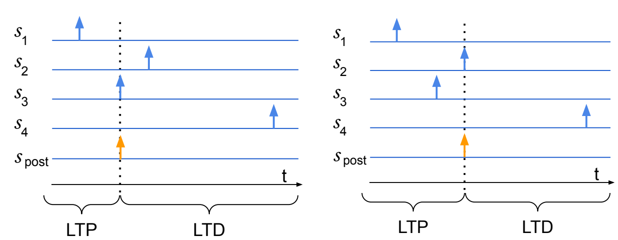


(a) (b)

Figure 3: Weight values of selected connections throughout the iterations of the online STDP-based learning performed on our hardware implementation (a) without connection pruning (b) with connection pruning.

where d is the number of times the LTD is performed on the connection, or the number of times the connection weight is decreased, in k learning iterations. w is the weight value at the time of pruning. The goal is to prune the connections having weights that are (i) small and (ii) decreasing steeply throughout a number of learning iterations. The intuition is that if a connection weight has been decreasing steeply in a time period and has reached a small value, it will likely continue to decrease until it approaches zero. Therefore, we propose to prune these connections early to reduce the number of membrane potential updates and learning computations in the future iterations. For example, in Figure 3a, which shows the learning progress of selected weights during the online STDP-based learning performed on our hardware implementation, weights w_2 and w_3 decrease steeply to a small value (less than 0.1) from iteration 2000 to 2500 and 1500 to 2000, respectively. In the later iterations, these weights contribute little to the spiking activities and will approach zero eventually. Consequently, w_2 can be pruned at iteration 2000 and w_3 can be pruned at iteration 2500, as shown in Figure 3b. Note that having the weight value in the denominator helps to reduce the chance of premature pruning, in which the connection is pruned when it still contributes non-negligibly to the spiking activities.

3.1.2 Time of Post-Synaptic Spike



(a) (b)

Figure 4: Examples of the spiking activities of four pre-synaptic neurons s_1, s_2, s_3, s_4 and a post-synaptic neuron s_{post}. (a) s_{post} spikes earlier, more connection weights are decreased (b) s_{post} spikes later, fewer connection weights are decreased.

In addition to the weight update history, our proposed pruning approach considers the synaptic competition during the online STDP-based learning to formulate the pruning decision. The connection weights that are decreased during a strong synaptic competition should be less likely to be pruned than those that are decreased during a less strong synaptic competition. The strength of the synaptic competition can be estimated using the timing of the post synaptic spikes. We will explain this relationship in the following.

In our hardware implementation of the online STDP-based learning, the connection weights are increased if the pre-synaptic neurons spike before the post-synaptic neurons; otherwise, the connection weights are decreased, as described in Section 2. If there are many large connection weights, the post-synaptic neuron will be able to accumulate sufficient membrane potential to spike early. In this case, only the most competitive connections that have the largest weight values and carry the earliest pre-synaptic spikes will be able to contribute to the spiking of the post-synaptic neuron. Figure 4a shows an example to illustrate this scenario. When the post-synaptic spike s_{post} is emitted early, it is only the pre-synaptic spikes s_1 and s_3 that were early enough to contribute to s_{post} and their connection weights will be increased. Contrarily, although s_2 is relatively early as compared to s_4 and its connection weight may be large, it was not able to contribute to s_{post}. Consequently, the connection weight associated with s_2 will be decreased. On the other hand, if s_{post} is emitted later, as shown in Figure 4b, s_2 will be able to contribute to s_{post} and its connection weight will be increased. Therefore, the weight decrements in the former scenario (Figure 4a) should be weighted less than the weight decrements in the latter scenario (Figure 4b) in the connection pruning decision.

The strong synaptic competition, as explained above, may happen in the early iterations in the online STDP-based learning on our hardware, as shown in Figure 3a. We initialized the connection weights to follow the normal distribution in the range of (0, 1) with the mean being skewed at 0.8, following the work in [22]. This is to encourage the spiking activities and to emphasize the most dominant pre-synaptic spikes and connections in the early iterations of the learning. The connections that have smaller weights or carry less dominant pre-synaptic spikes will have their weights decreased. However, these connections still have a chance to have their weights increased in a later learning iteration, when many of the other connections are weakened and the post-synaptic neurons do not spike as early as before, as shown in weight w_1 in Figure 3a. Therefore, we introduced the time of post-synaptic spikes parameter t_{post} to help to avoid pruning these connections in the early learning iterations, when the synaptic competition is strong.

3.1.3 Our Dynamic Pruning Approach

Our dynamic pruning approach combines the weight update history h and the time of post-synaptic spike t into a pruning parameter P:

$$P = h * t_{post} \tag{5}$$

Combining equations (4) and (5), we have:

$$P = \frac{d}{w} * t_{post} \tag{6}$$

P is evaluated for every connection in every k iterations. If P is greater than a pre-defined thershold α, the connection is pruned. This dynamic pruning stage is performed during the online STDP-based learning of each layer. After the learning is finished, we proceed to the post-learning pruning stage, which we will describe next.

3.2 Post-Learning Pruning

Our post-learning pruning approach is performed after the learning of one layer is finished, before proceeding to the next layer, as shown in Figure 2. In the post-learning pruning stage, we eliminate all the connections that have the weights less than a threshold β. Note that although this stage is performed after the learning of a layer, it is still within the online learning process of the SNN. It is different from the existing pruning approaches on the pre-trained networks as it can affect the learning of the next layer. Overly aggressive pruning of the preceeding layer will lead to learning errors in the next, causing losses in the classification accuracy.

4 Proposed Hardware Architecture

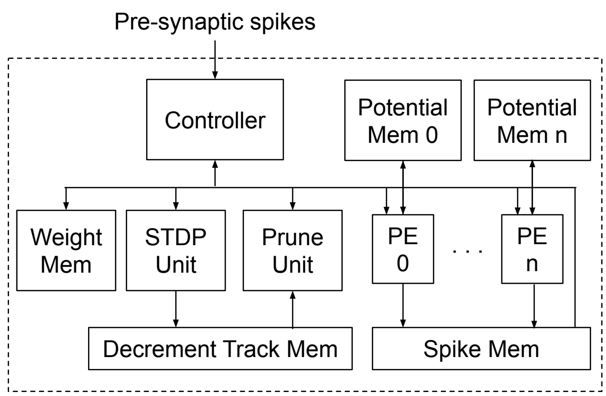


Figure 5: Our event-driven hardware architecture, which incorporates our connection pruning approach to reduce the online learning time and the network connectivity for SNN.

We developed an event-driven hardware architecture that incorporates our proposed connection pruning approach to reduce the online learning time and network connectivity for SNN, as shown in Figure 5. The overall flow of our hardware architecture is as follows. In every time step, the Controller receives the pre-synaptic spikes and forwards them to the Processing Elements (PEs). The PEs compute the membrane potentials and update the Potential memory. At the end of the time step, the PEs compares each membrane potential in the Potential memory with a pre-defined firing threshold. If the membrane potential of a neuron exceeds the firing threshold, the PE writes the neuron ID to the Spike memory. Finally, the Controller activates the STDP unit to perform the weight updates based on the STDP-based learning algorithm, as described in Section 2. During the learning, the STDP unit records the decrements of the weight values of each connection in the Decrement Track memory.

The connection pruning is performed in the Prune unit in every k iterations. After the STDP unit finishes updating the connection weights for the current time step, the Prune unit evaluates the pruning criteria for each connection weight as described in Section 3. If the pruning criteria is met, the connection weight is set to zero in the Weight memory. The pruning components incur little overheads in the resources and power consumption in our hardware implementation which we will discuss in the following sections.

5 Experimental Setup

This section describes our experimental setup to evaluate our proposed connection pruning approach and hardware architecture. We will first present the hardware platform on which our hardware architecture was implemented, followed by the dataset and the SNN network parameters used in our experiments.

5.1 Hardware Platform

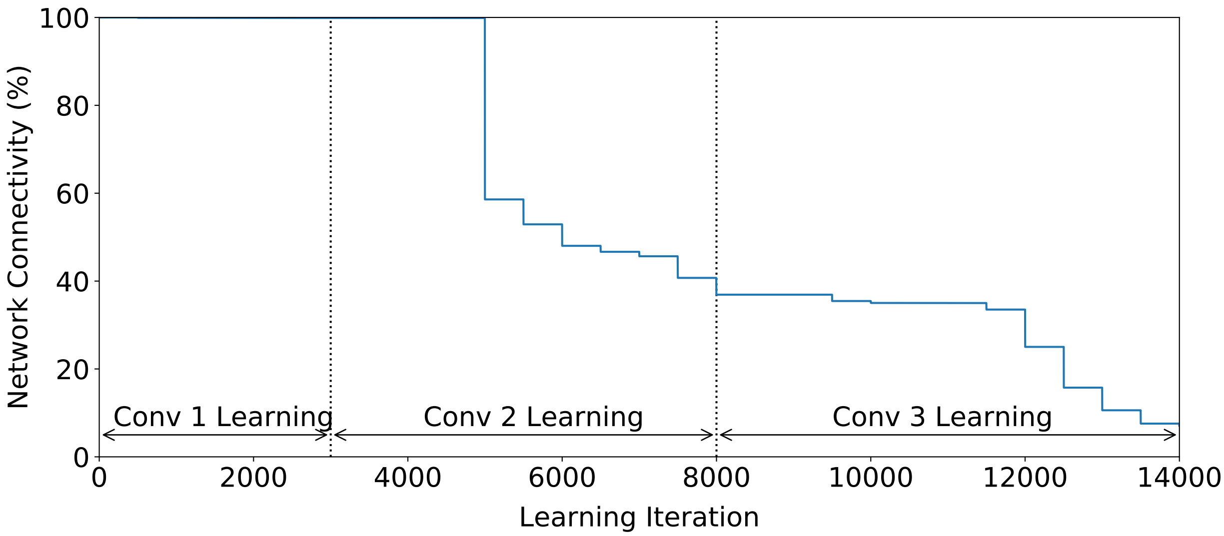
Our hardware architecture was implemented on the Zynq-7000 Zedboard (XC7Z020) using Verilog. The hardware resources and power consumption were estimated by Xilinx Vivado [23]. Our hardware implementation of the connection pruning incurs less than 3.52% overheads in the LUTs and FFs and approximately 8.7% overheads in the BRAM consumption. In addition, 11 DSP slices were used in the implementation of the connection pruning, which utilizes 5% of the DSP slices available on the board. The power consumption was increased by 0.56% as compared to our non-pruning baseline.

5.2 Dataset

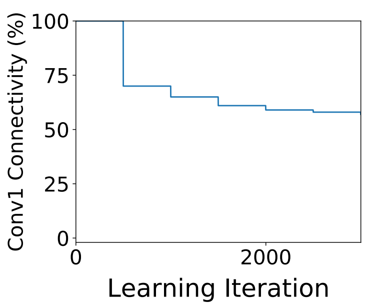
Table 1: Size of the convolutional layers in the SNN to evaluate our connection pruning approach.

Convolutional Layer	Kernel Size	# Neurons	# Connections
*conv*1	5 x 5 x 4	160,000	100
*conv*2	17 x 17 x 20	22,680	23,120
*conv*3	5 x 5 x 20	1,080	10,000

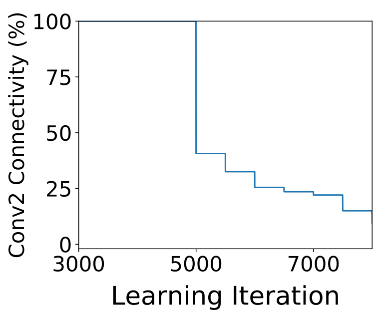
Our connection pruning approach was evaluated on the Caltech 101 dataset [24]. The training set con-



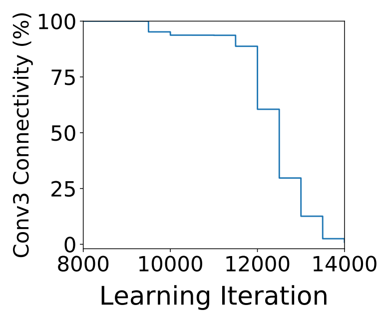
(a)



(b)



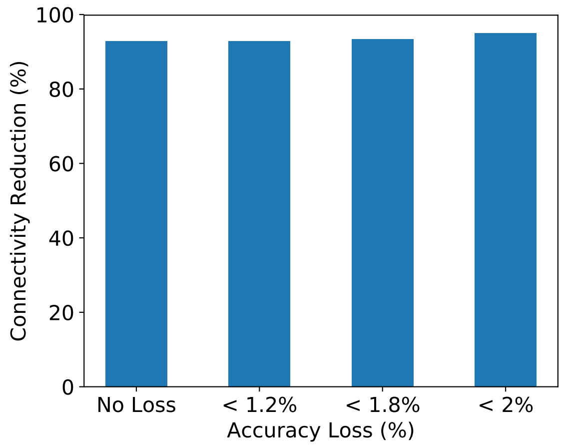
(c)



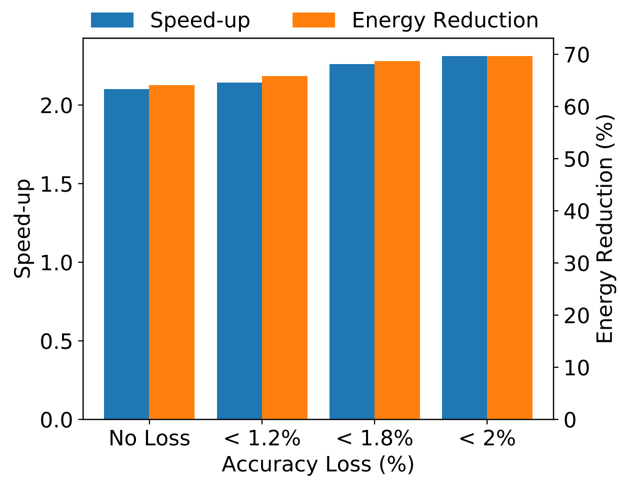
(d)

Figure 6: (a) Network connectivity throughout the online learning iterations in a sample run of the STDP-based learning with the connection pruning, which achieves the same accuracy as our non-pruning baseline. The network connectivity is computed as the percentage of the active connections out of the total number of connections in the network. (b-d) Connectivity reduction in each convolutional layer throughout the learning iterations in the same sample run. The connectivity of each layer is computed as the percentage of the active connections out of the total number of connections in the layer, as given in Table 1.

sists of 400 images of the two categories: (i) human face and (ii) motorbike in 160x250 pixels. The test set consists of 396 images. Our hardware implementation of the SNN was based on [8] and [22]. The SNN implemented on our hardware consists of three convolutional-pooling layers. Our connection pruning approach was applied to the convolutional layers, as shown in Table 1. The effects of our connection pruning approach on this SNN architecture will be discussed in the following section.



(a)



(b)

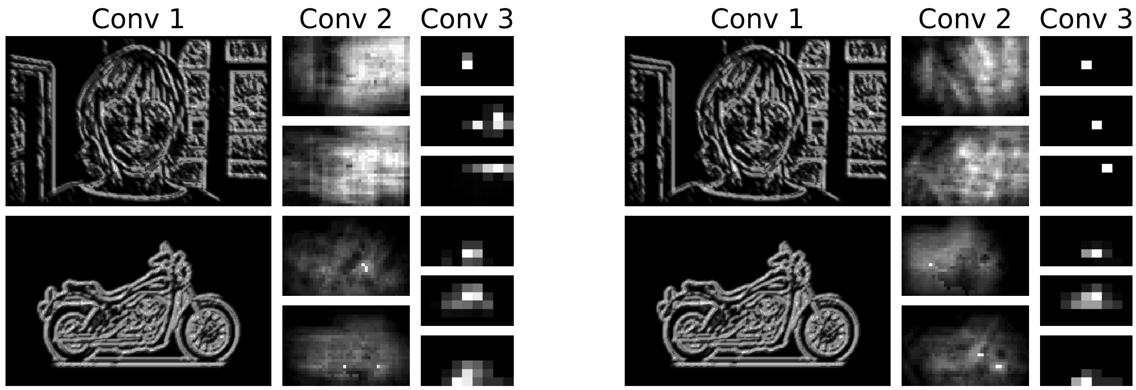
Figure 7: (a) Connectivity reduction and accuracy trade-offs. The connectivity reduction was computed as n_{pruned}/n_{total}, where n_{pruned} is the number of connections being pruned and n_{total} is the total number of connections in the network. (b) Online learning time, energy consumption, and accuracy trade-offs. The learning time speed-up and energy reduction are computed as $t_{noprune}/t_{pruned}$ and $(1 - E_{pruned}/E_{noprune}) * 100$, respectively. t_{pruned} and $t_{noprune}$ are the learning time with and without the connection pruning, E_{pruned} and $E_{noprune}$ are the energy consumption in the learning with and without the connection pruning

6 Experimental Results

The performance of our connection pruning approach was evaluated based on the following four metrics: (i) the number of connections reduced in the network, (ii) the time speed-up, (iii) the energy saved, and (iv) the classification accuracy. In this section, we will discuss the trade-offs between these metrics. In addition, the individual effects achieved by each of the stages: (i) dynamic pruning and (ii) post-learning pruning in our two-staged connection pruning approach will be analyzed. Next, we will present the effects of connection pruning on the network behaviors in the inference stage. Finally, the key features of our approach will be highlighted in comparison with the existing approaches. We will demonstrate that our connection pruning approach successfully eliminates a significant number of connections during the online STDP-based learning, which helps to reduce the time and energy in both the learning and the inference stages, without incurring any accuracy loss.

6.1 Connectivity Reduction vs. Accuracy

Our proposed connection pruning approach helps to eliminate 92.83% connections in the network without incurring any loss in the classification accuracy, as presented in Figure 6a. This is consistent with the result in [12] that more than 90% of the connections can be eliminated after one million iterations of the STDP-based learning, regardless of the network size. However, in the work in [12], the connections are eliminated after a large number of iterations (one million iterations). In contrary, our connection pruning approach eliminates the connections early to save the online learning time while reserving the accuracy. Figure 6 shows the connectivity reduction in a sample run on our hardware implementation, which achieves the same accuracy as our non-pruning baseline. More than 60% of the connections were reduced after the learning of the second convolutional layer, as shown in Figure 6a. At the end of the third con-



(a) (b)

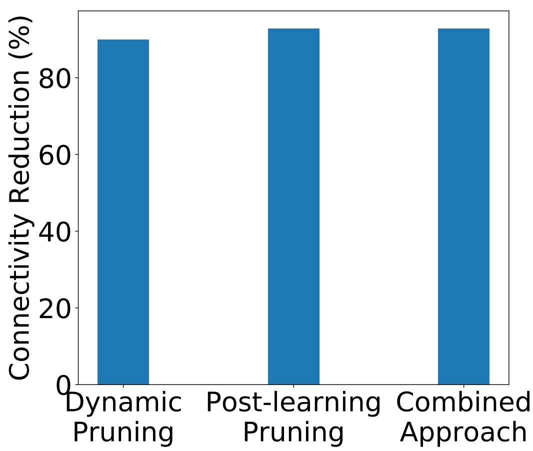
Figure 8: Selected feature maps in different convolutional layers after the online STDP-learning (a) without connection pruning and (b) with connection pruning on our hardware implementation.

volutional layer, more than 92% of the connections were reduced. The number of connections eliminated in the first convolutional layer is not observable in this figure as it is too small (approximately 0.13%). The reason is that the first convolutional layer consists of a small number of connections (100 connections, as compared to 33,220 connections in the whole network, as shown in Table 1). However, as shown in Figure 6b, the connection pruning still helps to reduce the connectivity of this layer, leading to the speed-up on its learning time, as shown in Figure 7b.

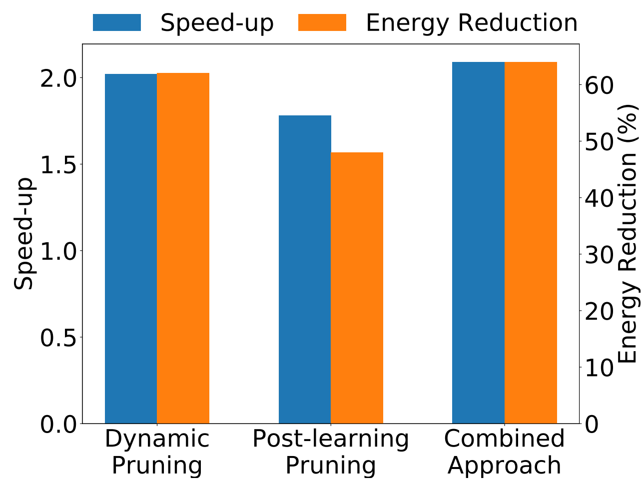
Different trade-offs between the number of connections eliminated and the accuracy loss are presented in Figure 7a. The number of connections in the network can be reduced by 92.83% without incurring any loss in the accuracy. Moreover, when the accuracy is allowed to fall within 1.2%, the network can be compressed by 92.9%. When up to 1.8% loss in the accuracy is accepted, the network connectivity can be reduced by 93.35%. In addition, when the accuracy loss is maintained within 2%, the connectivity can be reduced by up to 95.07%. The connectivity reduction during the STDP-based learning results in learning time speed-up and the energy saving on our hardware implementation, which we will discuss next.

6.2 Online Learning Time Speed-up and Energy Reduction

The speed-up on the online STDP-based learning time depends on (i) the number of connections being pruned in the network and (ii) the earliness of the prunings during the online learning. Our connection pruning approach helps to speed-up the online learning time by 2.1x without incurring any loss in the classification accuracy, as shown in Figure 7b. In addition, when the accuracy loss is up to 1.2%, the learning time is improved by 2.14x. Furthermore, the learning time can be improved by up to 2.26x and 2.31x when the accuracy is allowed to fall within 1.8% and 2%, respectively. In addition to the learning time, the energy consumption during the online STDP-based learning is also improved as a result of the network compression. In our experiments, the energy saving is estimated based on the number of operations reduced, similar to the related work in [14]. As shown in Figure 7b, the energy consumption is reduced by 64-69.62% with 0-2% loss in the accuracy. The improvements on the learning time and the en-



(a)



(b)

Figure 9: (a) Connectivity reduction and (b) online learning time speed-up and energy reduction achieved by the dynamic pruning and post-learning pruning when applied individually and when combined together in our two-staged connection pruning approach.

ergy consumption were attributed to the two stages in our connection pruning approach: dynamic pruning and post-learning pruning, which we will analyze in the following.

6.3 Dynamic Pruning vs. Post-Learning Pruning

Each of the two stages: (i) dynamic pruning and (ii) post-learning pruning in our connection pruning approach has its own effects on the network connectivity and the learning time. While the dynamic pruning is performed periodically during the learning of each layer, the post-learning pruning is performed after the learning of the layer is finished, as illustrated in Figure 2. Therefore, the dynamic pruning can eliminate the connections earlier than the post-learning pruning. When the two approaches are applied seperately, the dynamic pruning achieves a higher learning time speed-up and energy reduction, as shown in Figure 9b. However, the connectivity reduction achieved by the dynamic pruning is less than the post-learning pruning, as shown in Figure 9a. The reason is that during the learning, the network needs to maintain sufficient connections to generate the spike events in order to stimulate the STDP-based weight updates. On the other hand, after the learning of the layer is finished, the weights can be pruned without the concerns of the learning activities. In our experiments, the threshold β of the post-learning pruning can be set as high as 0.9 for the first and second convolutional layer and 0.7 for the third convolutional layer without causing any loss in the accuracy. Our connection pruning approach combines the dynamic pruning and the post-learning pruning to achieve the high learning time speed-up, high energy saving and significant connectivity reduction.

6.4 Effects on Inference Stage

The network compression during the STDP-based learning significantly reduces the response time and energy consumption in the inference stage. As shown in Figure 10, the inference time on our hardware im-

Table 2: Highlights of the existing works on connection pruning for SNN.

	[15]	[14]	[17]	Our Work
SNN Architecture	VGG19	Fully Connected	Fully Connected	CNN
Neuron Coding	Rate-Based	Rate-Based	Rate-Based	TTFS
# Learnable Layers	19	1	1	3
Learning Algorithm	Convert ANN to SNN	STDP-Based	STDP-Based	STDP-Based
Dataset	CIFAR-10	MNIST, Caltech 101	MNIST	Caltech 101
Pruning during Learning	No	Yes	Yes	Yes
Hardware Implementation	No	No	Yes	Yes
% Connections Reduced	89%	MNIST: 92%, Caltech 101: 88%	75%	92.83%
Accuracy after Pruning	94.01%	MNIST: 91.5%, Caltech 101: 92.8%	>90%	95.7%

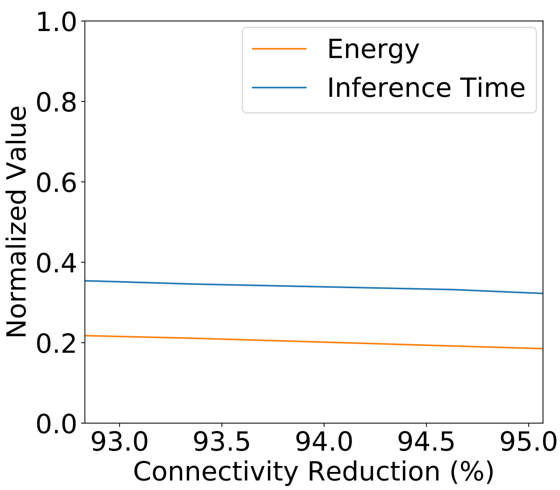


Figure 10: Normalized energy consumption and time in the inference, computed as $E_{pruned}/E_{noprune}$ and $t_{pruned}/t_{noprune}$, respectively. E_{pruned} and $E_{noprune}$ are the energy consumption in the inference with and without pruning, t_{pruned} and $t_{noprune}$ are the inference time with and without pruning.

plementation is speeded-up by 2.83-3.1x when 92.83-95% of the connections in the network are eliminated. In addition, the energy consumption is reduced by 78.24-81.47% as compared to our non-pruning baseline. Meanwhile, the number of spikes generated in the inference remains within 5% off as compared to the SNN trained without the connection pruning. Figure 8 shows the feature maps of different images in the Caltech 101 dataset in different convolutional layers. The first convolutional layer extracts the edges in the images while the deeper layers observe more abstract features. The feature maps obtained in the SNN trained with the connection pruning (Figure 8b) are similar to the ones obtained in the SNN trained without the connection pruning (Figure 8a). This demonstrates that our connection pruning approach does not cause significant effects on the network behaviors in the inference. Therefore, the classification accuracy was reserved.

6.5 Comparison with Existing Works

Prior to our work, there have been approaches that eliminate the weak connections and dormnant neurons in the well-trained networks [15], as shown in Table 2. The work in [15] helps to reduce the number of connections by 89% during the inference. However, it is not applicable during the learning when the weight values and the neuron behaviors are not stable. This limits the potential of the work in [15] to be applied to the applications that require frequent online learning [1, 2, 19]. On the other hand, the approaches in [14, 17] and our work eliminate the connections during the STDP-based learning. This helps to reduce the time and energy in both the inference and the learning stages. While the works in [14] and [17] focus on the SNN with the rate-based coding, our work proposes a connection pruning approach for the SNN with the TTFS coding, which is more energy-efficient [7]. In addition, our connection pruning approach was evaluated on a deep SNN consisting of three convolutional layers. Note that it is more challenging to eliminate the connections during the learning of a deep SNN as compared to an SNN with a single layer. In the deep SNN, the errors caused by the overly aggressive pruning in a layer will be carried to the following layers and eventually cause losses in the accuracy.

7 Conclusions

In this paper, we have proposed a novel connection pruning approach to be applied during the online STDP-based learning for SNN, in which the spikes are encoded using TTFS. Our main contributions are three-folded. First, we proposed a novel connection pruning approach that helps to compress the network and speed-up the online learning time on embeded systems hardware. Our connection pruning approach was evaluated on a deep network and achieved the compression of 92.83%, online learning time speed-up by 2.1x, and inference time speed-up by 2.83x on the Caltech 101 dataset. The energy consumption was reduced by 64% in the online learning and 78.24% in the inference. Meanwhile, the connection pruning does not incur any loss in the classification accuracy. Second, we implemented a hardware architecture on the FPGA-based platform that efficiently utilizes our connection pruning approach to speed-up the online learning time. Our hardware implementation incurs 0.56% power overhead. Third, we performed a comparison on the key features of our work and the existing works and highlighted the strengths and limitations of each approach. To the best of our knowledge, our work is the first to propose a connection pruning approach to improve the online learning time of a deep SNN that uses the TTFS coding. In the future, our approach can be combined with various device-level optimizations to further reduce the delay, resources, and energy consumption of the SNN to support a wider range of real-life applications.

Acknowledgement

This work is supported in part by the NUS Start-up grant.

References

[1] Anton Akusok, Kaj-Mikael Björk, Leonardo Espinosa Leal, Yoan Miche, Renjie Hu, and Amaury Lendasse. Spiking networks for improved cognitive abilities of edge computing devices. In *Proc. PETRA*, pages 307–308. ACM, 2019.

[2] Nicolas Skatchkovsky, Hyeryung Jang, and Osvaldo Simeone. Federated neuromorphic learning of spiking neural networks for low-power edge intelligence. In *Proc. ICASSP*, pages 8524–8528. IEEE, 2020.

[3] Xuchen Guan and Lingfei Mo. Unsupervised conditional reflex learning based on convolutional spiking neural network and reward modulation. *IEEE Access*, 8:17673–17690, 2020.

[4] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *Proc. IJCNN*, pages 1–8. IEEE, 2015.

[5] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In *Proc. ICLR*, 2020.

[6] Karen Simonyan and Andrew Zisserman". "very deep convolutional networks for large-scale image recognition". In *Proc. ICLR*, 2015.

[7] Rufin Van Rullen and Simon J Thorpe. Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural Comput.*, 13(6):1255–1283, 2001.

[8] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J Thorpe, and Timothée Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Netw.*, 99:56–67, 2018.

[9] Saeed Reza Kheradpisheh and Timothee Masquelier. Temporal backpropagation for

spiking neural networks with one spike per neuron. *Int. J. Neural Syst.*, 13(6):1255–1283, 2001.

[10] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Proc. NIPS*, pages 598–605, 1990.

[11] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Proc. NIPS*, pages 164–171, 1993.

[12] Javier Iglesias, Jan Eriksson, François Grize, Marco Tomassini, and Alessandro EP Villa. Dynamics of pruning in simulated large-scale spiking neural networks. *Biosystems*, 79(1-3):11–20, 2005.

[13] Sanchari Sen, Swagath Venkataramani, and Anand Raghunathan. Approximate computing for spiking neural networks. In *Proc. DATE*, pages 193–198. IEEE, 2017.

[14] Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. Stdp-based pruning of connections and weight quantization in spiking neural networks for energy-efficient recognition. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 38 (4):668–677, 2018.

[15] Ruizhi Chen, Hong Ma, Shaolin Xie, Peng Guo, Pin Li, and Donglin Wang. Fast and efficient deep sparse multi-strength spiking neural networks with dynamic pruning. In *Proc. IJCNN*, pages 1–8. IEEE, 2018.

[16] Yuhan Shi, Leon Nguyen, Sangheon Oh, Xin Liu, Foroozan Koushan, John R Jameson, and Duygu Kuzum. Neuroinspired unsupervised learning and pruning with subquantum cbram arrays. *Nat. Commun.*, 9(1):1–11, 2018.

[17] Yuhan Shi, Leon Nguyen, Sangheon Oh, Xin Liu, and Duygu Kuzum. A soft-pruning method applied during training of spiking neural networks for in-memory computing applications. *Front. Neurosci.*, 13:405, 2019.

[18] Hoyoung Tang, Heetak Kim, Hyeonseong Kim, and Jongsun Park. Spike counts based low complexity snn architecture with binary synapse. *IEEE Trans. Biomed. Circuits Syst.*, 13(6): 1664–1677, 2019.

[19] Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH Hoi. Online deep learning: learning deep neural networks on the fly. In *Proc. IJCAI*, pages 2660–2666, 2018.

[20] Bard Ermentrout. Type i membranes, phase resetting curves, and synchrony. *Neural Comp.*, 8 (5):979–1001, 1996.

[21] Sen Song, Kenneth D Miller, and Larry F Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci.*, 3(9):919–926, 2000.

[22] SDNN Python. `https://github.com/npvoid/SDNN_python`, 2019. Accessed: 2019-07-15.

[23] Tom Feist. Vivado design suite. *White Paper*, 5: 30, 2012.

[24] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Proc. CVPR*, pages 178–178, 2004.