Spiking-YOLO: Spiking Neural Network for Energy-Efficient Object Detection

Preprint · November 2019

CITATIONS
0

READS
1,104

4 authors:

Seijoon Kim
Seoul National University
11 PUBLICATIONS **28** CITATIONS
SEE PROFILE

Seongsik Park
Seoul National University
15 PUBLICATIONS **52** CITATIONS
SEE PROFILE

Byunggook Na
Seoul National University
5 PUBLICATIONS **27** CITATIONS
SEE PROFILE

Sungroh Yoon
Seoul National University
206 PUBLICATIONS **2,247** CITATIONS
SEE PROFILE

Spiking-YOLO: Spiking Neural Network for Energy-Efficient Object Detection

Seijoon Kim, Seongsik Park, Byunggook Na, Sungroh Yoon
Department of Electrical and Computer Engineering, ASRI, INMC, and Institute of Engineering Research
Seoul National University, Seoul 08826, Korea
sryoon@snu.ac.kr

Abstract

Over the past decade, deep neural networks (DNNs) have demonstrated remarkable performance in a variety of applications. As we try to solve more advanced problems, increasing demands for computing and power resources has become inevitable. Spiking neural networks (SNNs) have attracted widespread interest as the third-generation of neural networks due to their event-driven and low-powered nature. SNNs, however, are difficult to train, mainly owing to their complex dynamics of neurons and non-differentiable spike operations. Furthermore, their applications have been limited to relatively simple tasks such as image classification. In this study, we investigate the performance degradation of SNNs in a more challenging regression problem (i.e., object detection). Through our in-depth analysis, we introduce two novel methods: channel-wise normalization and signed neuron with imbalanced threshold, both of which provide fast and accurate information transmission for deep SNNs. Consequently, we present a first spiked-based object detection model, called Spiking-YOLO. Our experiments show that Spiking-YOLO achieves remarkable results that are comparable (up to 98%) to those of Tiny YOLO on non-trivial datasets, PASCAL VOC and MS COCO. Furthermore, Spiking-YOLO on a neuromorphic chip consumes approximately 280 times less energy than Tiny YOLO and converges 2.3 to 4 times faster than previous SNN conversion methods.

Introduction

One of the primary reasons behind the recent success of deep neural networks (DNNs) can be attributed to the development of high-performance computing systems and the availability of large amounts of data for model training. However, solving more intriguing and advanced problems in real-world applications requires more sophisticated models and training data, which results in significant increase in computational overhead and power consumption. To overcome these challenges, many researchers have attempted to design computationally- and energy-efficient DNNs using pruning (Guo, Yao, and Chen 2016; He, Zhang, and Sun 2017), compression (Han, Mao, and Dally 2016; Kim et al. 2015), and quantization (Gong et al. 2014; Park et al. 2018), some of which have shown promising results. Despite these efforts, the demand for computing and power resources will increase as deeper and more complicated neural networks achieve higher accuracy (Tan and Le 2019).

Spiking neural networks (SNNs), which are the third-generation neural networks, were introduced to mimic how information is encoded and processed in the human brain by employing spiking neurons as computation units (Maass 1997). Unlike conventional neural networks, SNNs transmit information via the precise timing (temporal) of spike trains consisting of a series of spikes (discrete), rather than a real value (continuous). That is, SNNs utilize temporal aspects in information transmission as in biological neural systems (Mainen and Sejnowski 1995), thus providing sparse yet powerful computing ability (Mostafa et al. 2017; Bellec et al. 2018). Moreover, the spiking neurons integrate inputs into a membrane potential when spikes are received and generate (fire) spikes when the membrane potential reaches a certain threshold, which enables event-driven computation. Driven by the sparse nature of spike events and event-driven computation, SNNs offer exceptional power efficiency and are the preferred neural networks in neuromorphic architectures (Merolla et al. 2014; Poon and Zhou 2011).

Despite their excellent potential, SNNs have been limited to relatively simple tasks (e.g., image classification) and small datasets (e.g., MNIST and CIFAR), on a rather shallow structure (Lee, Delbruck, and Pfeiffer 2016; Wu et al. 2019). One of the primary reasons for the limited application scope is the lack of scalable training algorithms due to complex dynamics and non-differentiable operations of spiking neurons. DNN-to-SNN conversion methods, as an alternative approach, have been studied widely in recent years (Cao, Chen, and Khosla 2015; Diehl et al. 2015; Sengupta et al. 2019). These methods are based on the idea of importing pre-trained parameters (e.g., weights and biases) from a DNN to an SNN. DNN-to-SNN conversion methods have achieved comparable results in deep SNNs to those of original DNNs (e.g., VGG and ResNet); however, results from MNIST and CIFAR datasets were competitive, while those of ImageNet dataset were unsatisfactory when compared with DNN's accuracy.

In this study, we investigate a more advanced machine learning problem in deep SNNs, namely object detection, using DNN-to-SNN conversion methods. Object detection is regarded as significantly more challenging as it involves both recognizing multiple overlapped objects and calculating precise coordinates for bounding boxes. Thus, it requires high numerical precision in predicting the output values of neural networks (i.e., regression problem) instead of selecting one class with the highest probability (i.e., argmax function) as performed in image classification. Based on our in-depth analysis, several issues arise when object detection is applied in deep SNNs: a) inefficiency of conventional normalization methods and b) absence of an efficient implementation method of leaky-ReLU in an SNN domain.

To overcome these issues, we introduce two novel methods; channel-wise normalization and signed neuron with imbalanced threshold. Consequently, we present a spike-based object detection model, called Spiking-YOLO. As the first step towards object detection in SNNs, we implemented Spiking-YOLO based on Tiny YOLO (Redmon et al. 2016). To the best of our knowledge, this is the first deep SNN for object detection that achieves comparable results to those of DNNs on non-trivial datasets, PASCAL VOC and MS COCO. Our contributions can be summarized as follows:

- **First object detection model in deep SNNs** We present Spiking-YOLO, a model that enables energy-efficient object detection in deep SNNs, for the first time. Spiking-YOLO achieves comparable results to original DNNs on non-trivial datasets, i.e., 98%.
- **Channel-wise normalization** We developed a fine-grained normalization method for deep SNNs. The proposed method enables a higher, yet proper firing rate in multiple neurons, thus leads to fast and accurate information transmission in deep SNNs.
- **Signed neuron featuring imbalanced threshold** We proposed an accurate and efficient implementation method of leaky-ReLU in an SNN domain. The proposed method can easily be implemented in neuromorphic chips with minimum overheads.

Related work

DNN-to-SNN conversion

In contrary to DNNs, SNNs use spike trains consisting of a series of spikes to convey information between neurons. The integrate-and-fire neurons accumulate input z into a membrane potential V_{mem} as

$$V^l_{\text{mem},j}(t) = V^l_{\text{mem},j}(t-1) + z^l_j(t) - V_{\text{th}}\Theta^l_j(t), \quad (1)$$

where $\Theta^l_j(t)$ is a spike, and $z^l_j(t)$ is the input of jth neuron in the lth layer with a threshold voltage V_{th}. $z^l_j(t)$ can be described as

$$z^l_j(t) = \sum_i w^l_{i,j}\Theta^{l\text{-}1}_i(t) + b^l_j, \quad (2)$$

where w and b are weight and bias, respectively. A spike Θ is generated when the integrated value V_{mem} exceeds the threshold voltage V_{th} as

$$\Theta^l_i(t) = U(V^l_{\text{mem},i}(t) - V_{\text{th}}), \quad (3)$$

where $U(x)$ is a unit step function. Due to the event-driven nature, SNNs offer energy-efficient operations (Pfeiffer and Pfeil 2018). However, they are difficult to train which has been one of the major obstacles when deploying SNNs in various applications (Wu et al. 2019).

The training method of SNNs consists of unsupervised learning with spike-timing-dependent plasticity (STDP) (Diehl and Cook 2015) and supervised learning with gradient descent and error back-propagation (Lee, Delbruck, and Pfeiffer 2016). Although STDP is biologically more plausible, the learning performance is significantly lower than that of supervised learning. Recent works proposed a supervised learning algorithm with a function that approximates the non-differentiable portion (integrate-and-fire) of SNNs (Jin, Zhang, and Li 2018; Lee, Delbruck, and Pfeiffer 2016) to improve the learning performance. Despite these efforts, most previous works have been limited to the image classification task and MNIST dataset on shallow SNNs.

As an alternative approach, the conversion of DNNs to SNNs has been recently proposed. (Cao, Chen, and Khosla 2015) proposed a DNN-to-SNN conversion method that neglected bias and max-pooling. In subsequent work, (Diehl et al. 2015) proposed data-based normalization to improve the performance in deep SNNs. (Rueckauer et al. 2017) presented an implementation method of batch normalization and spike max-pooling. (Sengupta et al. 2019) expanded conversion methods to VGG and residual architectures. Nonetheless, most previous works have been limited to the image classification task (Park et al. 2019).

Object detection

Object detection locates a single or multiple object(s) in an image or video by drawing bounding boxes, then identifying their classes. Hence, an object detection model consists of not only a classifier that classifies objects but also a regressor that predicts the precise coordinates (x- and y-axis) and size (width and height) of the bounding boxes. Because predicting precise coordinates of bounding boxes is critical, object detection is considered to be a much more challenging task than image classification, where an argmax function is used to simply pick one class with the highest probability.

Region-based CNN (R-CNN) (Girshick et al. 2014) is considered to be one of the most significant advances in object detection. To improve detection performance and speed, various extended versions of R-CNN have been proposed, namely fast R-CNN (Girshick 2015), faster R-CNN (Ren et al. 2015), and Mask R-CNN (He et al. 2017). Nevertheless, R-CNN based networks suffer from a slow inference speed due to multiple-stage detection schemes and thus are not suitable for real-time object detection.

As an alternative approach, one-stage detection methods have been proposed, where the bounding box information is extracted, and the objects are classified in a unified network. In one-stage detection models, "Single-shot multi-box detector" (SSD) (Liu et al. 2016) and "You only look once"

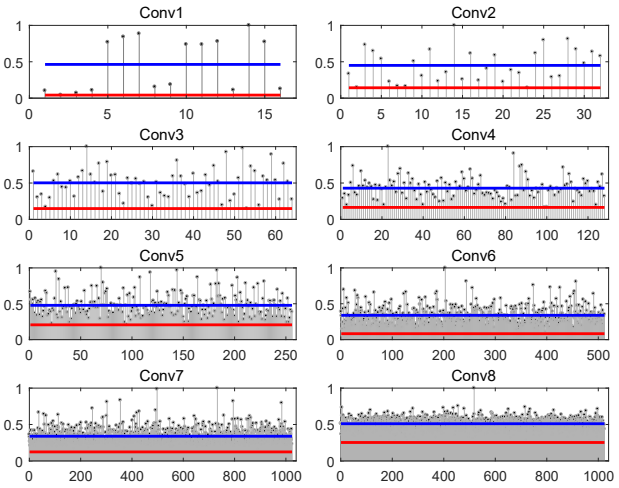

Figure 1: Normalized maximum activation via layer-wise normalization in each channel for eight convolutional layers in Tiny YOLO. Blue and red lines indicate the average and minimum of the normalized activations, respectively.

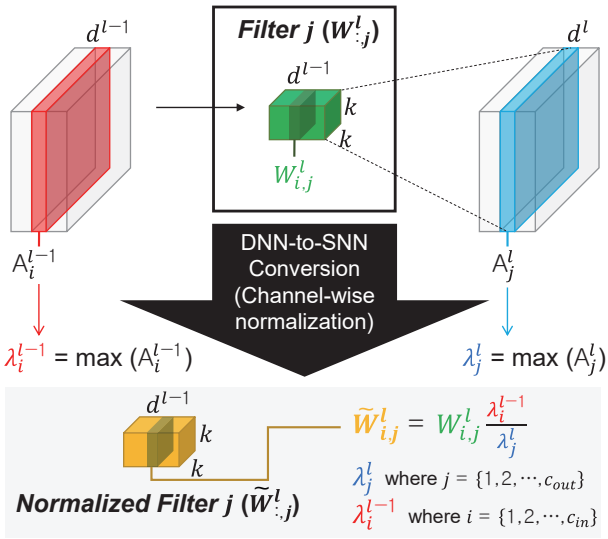

Figure 2: Proposed channel-wise normalization; A^l_j is jth activation matrix (i.e., feature map) in layer l.

(YOLO) (Redmon and Farhadi 2018) achieve the state-of-the-art performance. Particularly, the YOLO has superior inference speed (FPS) without a significant loss of accuracy, which is a critical factor in real-time object detection. Thus we selected Tiny YOLO as our object detection model.

Methods

In object detection, recognizing multiple objects and drawing bounding boxes around them (i.e., regression problem) poses great challenges: high numerical precision is required to predict the output value of the network. When object detection is applied in deep SNNs using conventional DNN-to-SNN conversion methods, it suffers from severe performance degradation and is unable to detect any objects. Our in-depth analysis highlights possible explanations for this performance degradation: a) an extremely low firing rate in numerous neurons and b) lack of an efficient implementation method of leaky-ReLU in SNNs. To overcome these complications, we propose two novel methods: channel-wise normalization and signed neuron with imbalanced threshold.

Channel-wise data-based normalization

Conventional normalization methods In a typical SNN, it is vital to ensure that a neuron generates spike trains according to the magnitude of the input and transmits those spike trains without any information loss. However, information loss can occur from under- or over-activation in the neurons given a fixed number of time steps. For instance, if a threshold voltage V_{th} is extremely large or the input is small, then a membrane potential V_{mem} will require a long time to reach V_{th}, thus resulting in a low firing rate (i.e., under-activation). Conversely, if V_{th} is extremely small or input is large, then V_{mem} will most likely exceed V_{th} and the neuron will generate spikes regardless of the input value (i.e., over-activation). It is noteworthy that the firing rate can be defined as $\frac{N}{T}$, where N is the total number of spikes in a given time step T. The maximum firing rate will be 100% since a spike can be generated at every time step.

To prevent under- or over-activation in the neurons, both the weights and the threshold voltage need to be carefully chosen for sufficient and balanced activation of the neuron (Diehl et al. 2015). Various data-based normalization methods (Diehl et al. 2015) have been proposed. Layer-wise normalization (Diehl et al. 2015) (abbreviated to layer-norm) is one of the most well-known normalization methods; layer-norm normalizes weights in a specific layer using the maximum activation of the corresponding layer, calculated from running the training dataset in a DNN. This is based on an assumption that the distributions of the training and test datasets are similar. In addition, note that normalizing the weights using the maximum activation will have the same effect as normalizing the output activation. Layer-norm can be calculated by

$$\tilde{w}^l = w^l \frac{\lambda^{l-1}}{\lambda^l} \quad \text{and} \quad \tilde{b}^l = \frac{b^l}{\lambda^l}, \tag{4}$$

where w, λ, and b are the weights, the maximum activations calculated from the training dataset, and bias in layer l, respectively. As an extended version of layer-norm, (Rueckauer et al. 2017) introduced an approach that normalizes the activations using the 99.9th percentile of the maximum activation; this increases the robustness to outliers and ensures sufficient firing of neurons. However, our experiments show that when object detection is applied in deep SNNs using the conventional normalization methods, the model suffers from significant performance degradation.

Analysis of layer-norm limitation Figure 1 represents the normalized maximum activation values in each channel obtained from layer-norm. Tiny YOLO consists of eight convolutional layers; the x-axis indicates the channel index

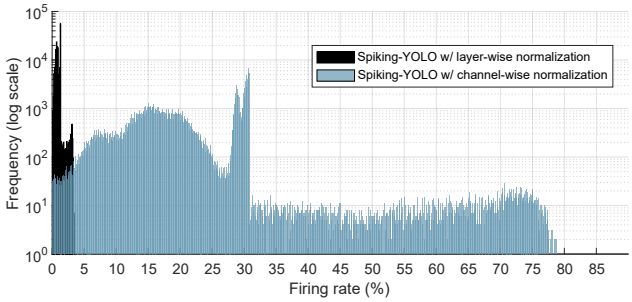

Figure 3: Firing rate distribution for layer-norm and channel-norm on channel 2 of Conv1 layer (Tiny YOLO)

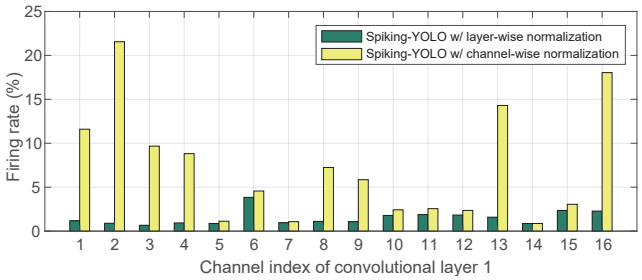

Figure 4: Firing rate of 16 channels in Conv1 layer for layer-norm and channel-norm of Tiny YOLO

and the y-axis represents the normalized maximum activation values. The blue and red lines indicate the average and minimum values of the normalized activations in each layer, respectively. As highlighted in Figure 1, for a specific convolutional layer, the deviation of the normalized activations on each channel is relatively large. For example, in the Conv1 layer, the normalized maximum activation is close to 1 for certain channels (e.g., channels 6, 7, and 14) and 0 for other channels (e.g., channels 1, 2, 3, 13, and 16). The same can be said for the other convolutional layers. Clearly, layer-norm yields exceptionally small normalized activations (i.e., under-activation) in numerous channels that had relatively small activation values prior to the normalization.

These extremely small normalized activations were undetected in image classification, but can be extremely problematic in solving regression problems in deep SNNs. For instance, to transmit 0.7, 7 spikes and 10 time steps are required. Applying the same logic, transmitting 0.007 would require 7 spikes and 1000 time steps without any loss of information. Hence, to send either extremely small (e.g., 0.007) or precise (e.g., 0.9007 vs. 0.9000) values without any loss, a large number of time steps is required. The number of time steps is considered as the resolution of the information being transmitted. Consequently, extremely small normalized activations yield low firing rates which results in information loss when the number of time steps is less than what it needs to be.

Proposed normalization method We propose a more fine-grained normalization method, called channel-wise normalization (abbreviated to channel-norm) to enable fast and efficient information transmission in deep SNNs. Our method normalizes the weights by the maximum possible activation (the 99.9th percentile) in a channel-wise manner instead of the conventional layer-wise manner. The proposed channel-norm can be expressed as

$$\tilde{w}^l_{i,j} = w^l_{i,j}\frac{\lambda^{l-1}_i}{\lambda^l_j} \quad \text{and} \quad \tilde{b}^l_j = \frac{b^l_j}{\lambda^l_j}, \tag{5}$$

where i and j are indices of channels. Weights w in a layer l are normalized (same effect as normalizing the output activation) by maximum activation λ^l_j in each channel. As mentioned before, the maximum activation is calculated from the training dataset. In the following layer, the normalized activations must be multiplied by λ^{l-1}_i to obtain the original activation prior to the normalization. The detailed method is depicted in Algorithm 1 and Figure 2.

Normalizing the activations in the channel-wise manner eliminates extremely small activations (i.e., under-activation), which had small activation values prior to the normalization. In other words, neurons are normalized to obtain a higher yet proper firing rate, which leads to accurate information transmission in a short period of time.

Algorithm 1: Channel-wise normalization

```
// Calculate maximum activation (λ) for each channel from
   training dataset
for l in layers do
    for j in output channels do
        λ^l_j = max(A^l_j)                    // A = activation matrix
// Apply channel-norm on inference (test dataset)
for l in layers do
    for j in output channels do
        b̃^l_j = b^l_j / λ^l_j                 // b = bias
        for i in input channels do
            if l = first layer then
                w̃^l_{i,j} = w^l_{i,j} / λ^l_j  // w = weight
            else
                w̃^l_{i,j} = w^l_{i,j} / λ^l_j λ^{l-1}_i
```

Analysis of the improved firing rate In Figure 3, the x- and y-axes indicate the firing rate and the number of neurons that produce a specific firing rate on a log scale, respectively. For channel-norm, numerous neurons generated a firing rate of up to 80%. In layer-norm, however, most of the neurons generated a firing rate in the range between 0% and 3.5%. This is a clear indication that channel-norm eliminates extremely small activations and that more neurons are producing a higher yet proper firing rate. In addition, Figure 4 presents the firing rate in each channel in the convolutional layer 1. Evidently, channel-norm produces a much higher firing rate in majority of the channels. Particularly in channel 2, channel-norm produces a firing rate that is 20 times higher than that of layer-norm. Furthermore, Figure 5 presents a raster plot of the spike activity from 20 sampled neurons. It can be seen that numerous neurons are firing more regularly when channel-norm is applied.

Our detailed analysis verifies that the fine-grained channel-norm normalizes activations better, preventing insufficient activation that leads to a low firing rate. In other

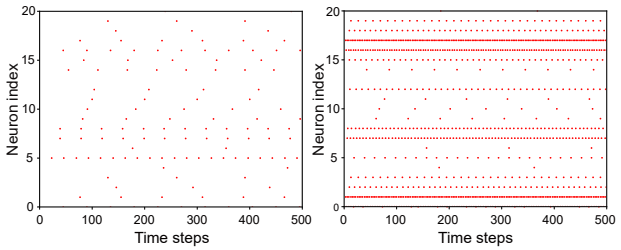

Figure 5: Raster plot of 20 sampled neurons' spike activity; layer-norm (left) vs. channel-norm (right)

words, extremely small activations are normalized properly such that neurons can transmit information accurately in a short period of time. These small activations may not be significant and have little impact on the final output of the network in simple applications such as image classification; however, they are critical in regression problems and significantly affect the model's accuracy. Thus, channel-norm is a viable solution for solving more advanced machine learning problems in deep SNNs.

Signed neuron featuring imbalanced threshold

Limitation of leaky-ReLU implementation in SNNs ReLU, one of the most commonly used activation functions, retains solely positive input values and discards all negative values; $f(x) = x$ when $x \geq 0$, otherwise $f(x) = 0$. Unlike ReLU, leaky-ReLU contains negative values with a leakage term, slope of α, which is typically set to 0.01; $f(x) = x$ when $x \geq 0$, otherwise $f(x) = \alpha x$ (Xu et al. 2015).

Most previous DNN-to-SNN conversion methods have focused on converting integrate-and-fire neurons to ReLU, while completely neglecting the leakage term in the negative region of the activation function. Note that negative activations account for over 51% in Tiny YOLO. To extend the activation function bound to the negative region in SNNs, (Rueckauer et al. 2017) added a second V_{th} term (-1). Their method successfully converted BinaryNet (Hubara et al. 2016) to SNNs, where the BinaryNet activations were constrained to $+1$ or -1 on CIFAR-10.

Currently, various DNNs use leaky-ReLU as an activation function, yet an accurate and efficient method of implementing leaky-ReLU in an SNN domain has not been proposed. Leaky-ReLU can be implemented in SNNs by simply multiplying negative activations by the slope α in addition to a second V_{th} term (-1). However, this is not biologically plausible (the spike is a discrete signal) and can be a formidable challenge when employed on neuromorphic chips. For instance, additional hardware would be required for the floating-point multiplication of the slope α.

The notion of imbalanced threshold We herein introduce a signed neuron featuring imbalanced threshold (hereinafter abbreviated as IBT) that can not only interpret both positive and negative activations, but also accurately and efficiently compensate for the leakage term in the negative regions of leaky-ReLU. The proposed method also retains the discrete characteristics of the spikes by introducing a different threshold voltage for the negative region, $V_{\text{th,neg}}$. The

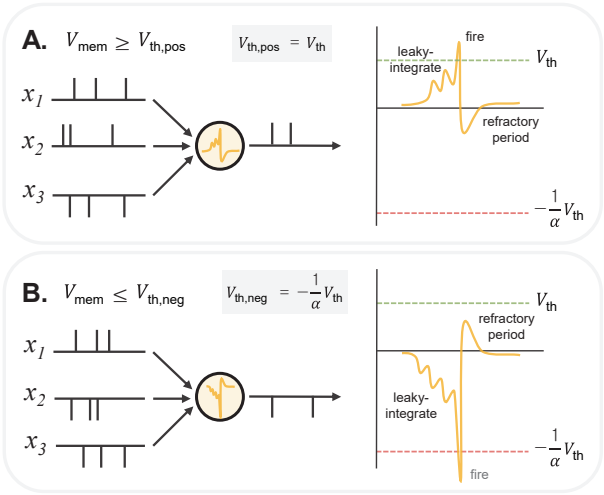

Figure 6: Overview of proposed signed neuron featuring imbalanced threshold; two possible cases for a spiking neuron

second threshold voltage $V_{\text{th,neg}}$ is equal to the V_{th} divided by the negative of the slope, $-\alpha$, and $V_{\text{th,pos}}$ is equal to V_{th} as before. This would replicate the leakage term (slope α) in the negative region of leaky-ReLU. The underlying dynamics of signed neuron with IBT are represented by

$$\text{fire}(V_{\text{mem}}) = \begin{cases} 1 & \text{if } V_{\text{mem}} \geq V_{\text{th,pos}}(V_{\text{th}}) \\ -1 & \text{if } V_{\text{mem}} \leq V_{\text{th,neg}}(-\frac{1}{\alpha}V_{\text{th}}) \\ 0 & \text{otherwise, no firing.} \end{cases} \quad (6)$$

As shown in Figure 6, if the slope $\alpha = 0.1$ then the threshold voltage responsible for a positive activation $V_{\text{th,pos}}$ is $1V$, and that for a negative activation, $V_{\text{th,neg}}$, is $-10V$; therefore, V_{mem} must be integrated ten times more to generate a spike for the negative activations in leaky-ReLU.

It is noteworthy that a signed neuron also enables implementation of excitatory and inhibitory neurons, which is more biologically plausible (Dehghani et al. 2016; Wilson and Cowan 1972). Using signed neurons with IBT, leaky-ReLU can be implemented accurately in SNNs and can directly be mapped to the current neuromorphic architecture with minimum overhead. Moreover, the proposed method will create more opportunities for converting various DNN models to SNNs in a wide range of applications.

Evaluation

Experimental setup

As the first step towards object detection in deep SNNs, we used a real-time object detection model, Tiny YOLO, which is a simpler but efficient version of YOLO. We implemented max-pooling and batch-normalization in SNNs according to (Rueckauer et al. 2017). Tiny YOLO is tested on non-trivial datasets, PASCAL VOC and MS COCO. Our simulation is based on the TensorFlow Eager and we conducted all experiments on NVIDIA Tesla V100 GPUs.

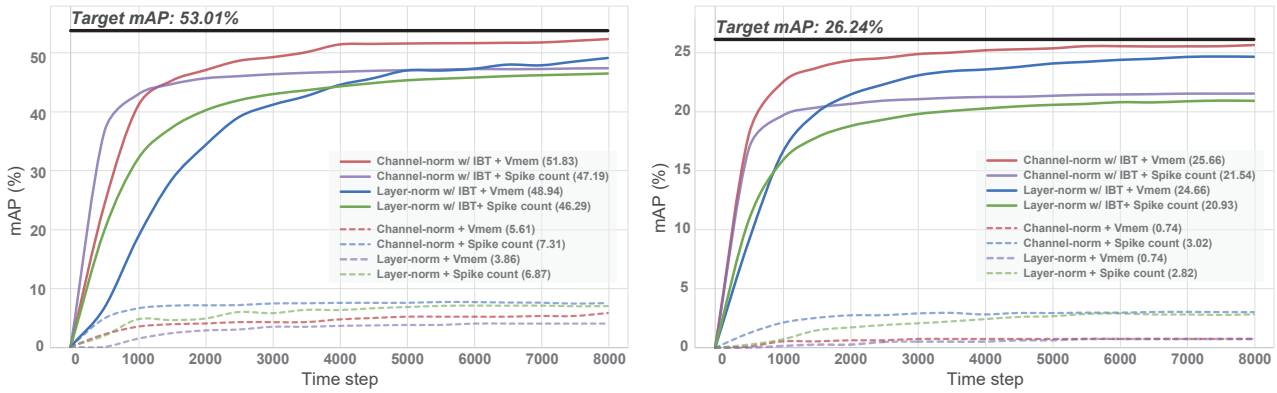

Figure 7: Experimental results of Spiking-YOLO on PASCAL VOC (left) and MS COCO (right) for various configurations (normalization methods + signed neuron w/ IBT + decoding scheme); maximum mAP is in parentheses.

Experimental results

Spiking-YOLO detection results To verify and analyze the functionalities of the proposed methods, we investigated the effects of the presence or absence of channel-norm and signed neuron with IBT. As depicted in Figure 7, when both channel-norm and signed neuron with IBT are applied, Spiking-YOLO achieves a remarkable performance of 51.83% and 25.66% on VOC PASCAL and MS COCO, respectively. The target mAP of Tiny YOLO is 53.01% (PASCAL VOC) and 26.24% (MS COCO). In fact, channel-norm outperforms layer-norm in detecting objects by a large margin, especially on PASCAL VOC (53.01% vs. 48.94%), and converges faster. For instance, to reach the maximum mAP of layer-norm (48.94), channel norm only requires approximately 3,500 time steps (2.3x faster). Similar results are observed in MS COCO where channel-norm converges even faster than the layer-norm (4x faster). Please refer to Table 1 for more detailed results.

Table 1: Experiment results for Spiking-YOLO (mAP %)

Signed neuron	Norm. method	PASCAL VOC (53.01)[a]		MS COCO (26.24)[a]	
		V_{mem}	Spike count	V_{mem}	Spike count
w/out IBT	Layer	3.86	6.87	0.74	2.82
	Channel	5.61	7.31	0.74	3.02
w/ IBT	Layer	48.94	46.29	24.66	20.93
	Channel	**51.83**	47.19	**25.66**	21.54

[a] Target mAP in parentheses

Notably, without the proposed methods, the model failed to detect objects, reporting 6.87% and 2.82% for VOC PASCAL and MS COCO, respectively. When channel-norm is applied, the model still struggles to detect objects, reporting approximately 7.31% and 3.02% at the best. This is a great indication that signed neuron with IBT accurately implements the leakage term in leaky-ReLU. Thus, the rest of the experiments were conducted using signed neuron with IBT as the default.

For further analysis, we performed additional experiments on two different output decoding schemes: one based on accumulated V_{mem}, and another based on spike count. The quotient from V_{mem} / V_{th} indicates the spike count, and the remainder is rounded off. This remainder will eventually become an error and lost information. Therefore, the V_{mem}-based output decoding scheme is more precise for interpreting spike trains; Figure 7 verifies this assertion. The V_{mem}-based output decoding scheme outperforms the spike-count-based scheme and converges faster in channel-norm.

Figure 8 illustrates the efficacy of Spiking-YOLO in detecting objects as the time step increases. For each example, the far-left image (Tiny YOLO) shows the ground truth label that Spiking-YOLO attempts to replicate. In the top-left example (three ships), after only 1000 time steps, Spiking-YOLO with channel-norm successfully detects all three objects. Meanwhile, Spiking YOLO with layer-norm failed to detect any objects. After 2000 time steps, it starts to draw bounding boxes around the objects, but there are multiple bounding boxes drawn over a single object, and their sizes are all inaccurate. The detection performance improves as the time steps increase but is still unsatisfactory; 5000 time steps are required to reach the detection performance of the proposed channel-norm. This remarkable performance of Spiking-YOLO is also shown in the other examples in Figure 8. The proposed channel-norm shows a clear advantage in detecting multiple and microscopic objects accurately in a shorter period of time.

Spiking-YOLO energy efficiency To investigate energy efficiency of Spiking-YOLO, we considered two different approaches: a) computing operations of Spiking-YOLO and Tiny YOLO in digital signal processing b) Spiking-YOLO on neuromorphic chips vs. Tiny YOLO on GPUs.

Firstly, most operations in DNNs occur in convolutional layers where the multiply-accumulate (MAC) operations are primarily responsible during execution. SNNs, however, perform accumulate (AC) operations because spike events are binary operations whose input is integrated (or accumulated) into a membrane potential only when spikes are received. For a fair comparison, we focused solely on the computational power (MAC and AC) used to execute object detection on a single image. According to (Horowitz 2014), a 32-bit floating-point (FL) MAC operation consumes 4.6 pJ (0.9 + 3.7 pJ) and 0.9 pJ for an AC operation. A 32-bit inte-

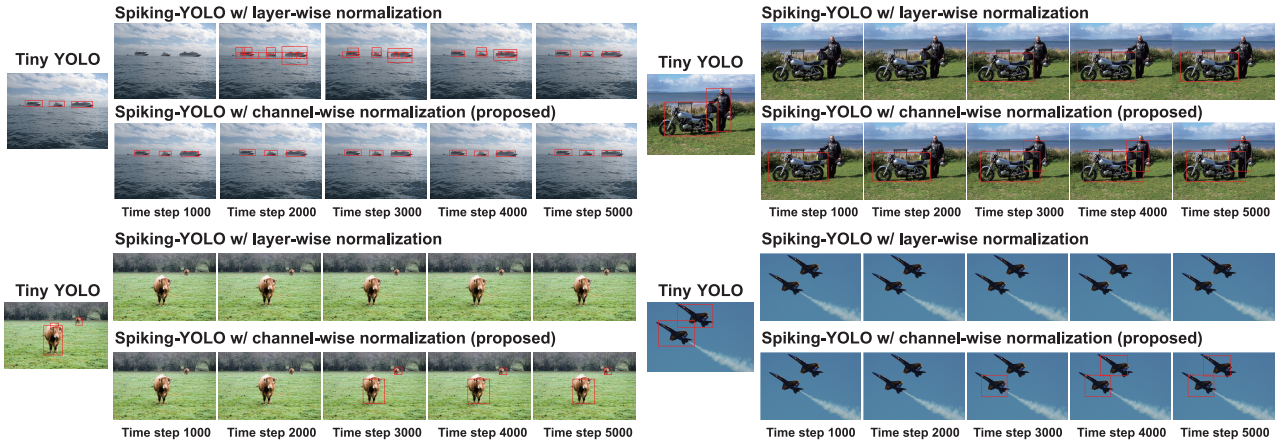

Figure 8: Object detection results (Tiny YOLO vs. Spiking-YOLO with layer-norm vs. Spiking-YOLO with channel-norm)

ger (INT) MAC operation consumes 3.2 pJ (0.1 + 3.1 pJ) and 0.1 pJ for an AC operation. Based on these measures, we calculated the energy consumption of Tiny YOLO and Spiking-YOLO by multiplying FLOPs (floating-point operations) and the energy consumption of MAC and AC operations calculated, as shown below. FLOPs of Tiny YOLO are reported on (Redmon 2016), and that for Spiking-YOLO are calculated during our simulation. Figure 9 shows that regardless of the normalization methods, Spiking-YOLO demonstrates exceptional energy efficiency, over 2,000 times better than Tiny-YOLO for 32-bit FL and INT operations.

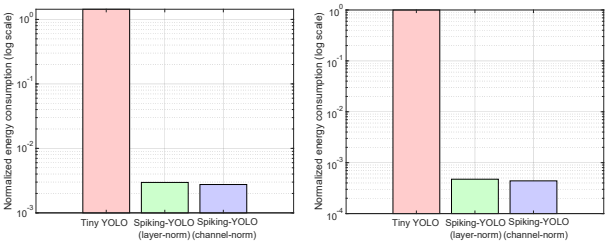

Figure 9: Energy comparison of Tiny YOLO and Spiking-YOLO for MAC and AC operations; 32-bit FL (left) and 32-bit INT (right)

Secondly, SNNs on neuromorphic chips offer excellent energy efficiency, which is an important and desirable aspect of neural networks (Pfeiffer and Pfeil 2018). We compare the energy consumption of Tiny YOLO and Spiking-YOLO when each ran on the latest GPU (Titan V100) and neuromorphic chip (TrueNorth), respectively. The power and GFLOPS (Giga floating-point operation per second) of Titan V100 were obtained from (NVIDIA 2017), and GFLOPS/W for TrueNorth is reported on (Merolla et al. 2014). We define one time step as equal to 1ms (1 kHz synchronization signal in (Merolla et al. 2014)).

Based on our calculations shown in Table 2, Spiking-YOLO consumes approximately 280 times less energy than Tiny YOLO when ran on TrueNorth. As mentioned in the experimental results, the proposed channel-norm converges much faster than layer-norm; therefore, the energy consumption of Spiking-YOLO with channel-norm is approximately four times less than that with layer-norm as they have similar power consumption. Note that contemporary GPUs are far more advanced computing technology, and the TrueNorth chip was first introduced in 2014. As neuromorphic chips continue to develop and have better performance, we can expect even higher energy and computational efficiency.

Table 2: Energy comparison of Tiny YOLO (GPU) and Spiking-YOLO (neuromorphic chips)

Tiny YOLO					
	Power (W)	GFLOPS	FLOPs		Energy (J)
	250	14,000	6.97E+09		0.12
Spiking-YOLO					
Norm. methods	GFLOPS / W	FLOPs	Power (W)	Time steps	Energy (J)
Layer	400	5.28E+07	1.320E-04	8,000	1.06E-03
Channel	400	4.90E+07	1.225E-04	**3,500**	**4.29E-04**

Conclusion

In this paper, we have presented Spiking-YOLO, the first SNN model that successfully performs object detection by achieving comparable results to those of the original DNNs on non-trivial datasets, PASCAL VOC and MS COCO. In doing so, we proposed two novel methods. We believe that this study represents the first step towards solving more advanced machine learning problems in deep SNNs.

Acknowledgements

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean government (Ministry of Science and ICT) [2016M3A7B4911115, 2018R1A2B3001628], the Brain Korea 21 Plus Project in 2019, Samsung Electronics (DS and Foundry), and AIR Lab (AI Research Lab) in Hyundai Motor Company through HMC-SNU AI Consortium Fund.

References

Bellec, G.; Salaj, D.; Subramoney, A.; Legenstein, R.; and Maass, W. 2018. Long short-term memory and learning-to-learn in networks of spiking neurons. In *NIPS*.

Cao, Y.; Chen, Y.; and Khosla, D. 2015. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision* 113(1):54–66.

Dehghani, N.; Peyrache, A.; Telenczuk, B.; Le Van Quyen, M.; Halgren, E.; Cash, S. S.; Hatsopoulos, N. G.; and Destexhe, A. 2016. Dynamic balance of excitation and inhibition in human and monkey neocortex. *Scientific reports* 6:23176.

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

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 *IJCNN*.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Girshick, R. 2015. Fast r-cnn. In *ICCV*.

Gong, Y.; Liu, L.; Yang, M.; and Bourdev, L. 2014. Compressing deep convolutional networks using vector quantization. In *CVPR*.

Guo, Y.; Yao, A.; and Chen, Y. 2016. Dynamic network surgery for efficient dnns. In *NIPS*.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *ICCV*.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel pruning for accelerating very deep neural networks. In *ICCV*.

Horowitz, M. 2014. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 10–14. IEEE.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks. In *NIPS*.

Jin, Y.; Zhang, W.; and Li, P. 2018. Hybrid macro/micro level backpropagation for training deep spiking neural networks. In *NIPS*.

Kim, Y.-D.; Park, E.; Yoo, S.; Choi, T.; Yang, L.; and Shin, D. 2015. Compression of deep convolutional neural networks for fast and low power mobile applications. In *CVPR*.

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

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. Ssd: Single shot multibox detector. In *ECCV*.

Maass, W. 1997. Networks of spiking neurons: the third generation of neural network models. *Neural networks* 10(9):1659–1671.

Mainen, Z. F., and Sejnowski, T. J. 1995. Reliability of spike timing in neocortical neurons. *Science* 268(5216):1503–1506.

Merolla, P. A.; Arthur, J. V.; Alvarez-Icaza, R.; Cassidy, A. S.; Sawada, J.; Akopyan, F.; Jackson, B. L.; et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345(6197):668–673.

Mostafa, H.; Pedroni, B. U.; Sheik, S.; and Cauwenberghs, G. 2017. Fast classification using sparsely active spiking networks. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–4. IEEE.

NVIDIA, T. 2017. V100 gpu architecture.

Park, S.; Kim, S.; Lee, S.; Bae, H.; and Yoon, S. 2018. Quantized memory-augmented neural networks. In *AAAI*.

Park, S.; Kim, S.; Choe, H.; and Yoon, S. 2019. Fast and efficient information transmission with burst spikes in deep spiking neural networks. In *DAC*.

Pfeiffer, M., and Pfeil, T. 2018. Deep learning with spiking neurons: opportunities and challenges. *Frontiers in neuroscience* 12.

Poon, C.-S., and Zhou, K. 2011. Neuromorphic silicon neurons and large-scale neural networks: challenges and opportunities. *Frontiers in neuroscience* 5:108.

Redmon, J., and Farhadi, A. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *CVPR*.

Redmon, J. 2016. https://pjreddie.com/darknet/yolov2/.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*.

Rueckauer, B.; Lungu, I.-A.; Hu, Y.; Pfeiffer, M.; and Liu, S.-C. 2017. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience* 11:682.

Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; and Roy, K. 2019. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience* 13:95.

Tan, M., and Le, Q. V. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*.

Wilson, H. R., and Cowan, J. D. 1972. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical journal* 12(1):1–24.

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; Xie, Y.; and Shi, L. 2019. Direct training for spiking neural networks: Faster, larger, better. In *AAAI*.

Xu, B.; Wang, N.; Chen, T.; and Li, M. 2015. Empirical evaluation of rectified activations in convolutional network. In *CVPR*.