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

Preprint · November 2019

DOI: 10.48550/arXiv.1903.06530

4 authors, including:

Seijoon Kim
Seoul National University
17 PUBLICATIONS **643** CITATIONS
SEE PROFILE

Seongsik Park
Korea Institute of Science and Technology
44 PUBLICATIONS **705** CITATIONS
SEE PROFILE

Byunggook Na
21 PUBLICATIONS **598** CITATIONS
SEE PROFILE

Spiking-YOLO: Spiking Neural Network for Real-time Object Detection

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

Abstract

Over the past decade, deep neural networks (DNNs) have become a *de-facto* standard for solving machine learning problems. As we try to solve more advanced problems, growing demand for computing and power resources are inevitable, nearly impossible to employ DNNs on embedded systems, where available resources are limited. Given these circumstances, spiking neural networks (SNNs) are attracting widespread interest as the third generation of neural network, due to event-driven and low-powered nature. However, SNNs come at the cost of significant performance degradation largely due to complex dynamics of SNN neurons and non-differential spike operation. Thus, its application has been limited to relatively simple tasks such as image classification. In this paper, we investigate the performance degradation of SNNs in the much more challenging task of object detection. From our in-depth analysis, we introduce two novel methods to overcome a significant performance gap: channel-wise normalization and signed neuron with imbalanced threshold. Consequently, we present a spiked-based real-time object detection model, called Spiking-YOLO that provides near-lossless information transmission in a shorter period of time for deep SNN. Our experiments show that the Spiking-YOLO is able to achieve comparable results up to 97% of the original YOLO on a non-trivial dataset, PASCAL VOC.

1 Introduction

One of the primary reasons behind the recent success of deep neural networks (DNNs) lies in the development of high-performance parallel computing systems and the availability of enormous amounts of data for training a complex model. However, solving advanced machine learning problems in real world applications would certainly require a more sophisticated model with a vast number of parameters and training data, which would result in substantial amounts of computational overhead and power consumption. This leads to the question of how large-scale DNNs could be employed on embedded systems, such as mobile devices, where available power and computation resources are limited.

In recent years, many have attempted to design an energy and computation-efficient DNNs, some of which have shown promising results. Pruning [He et al., 2017b, Guo et al., 2016] and compression [Kim et al., 2015, Hinton et al., 2015, Zagoruyko and Komodakis, 2017] techniques aim to reduce computational overheads by eliminating redundancy, keeping only important parts of the network, while preserving accuracy. [Han et al., 2016, Park et al., 2018] adopted a quantization technique to reduce the number of bits required to represent a model, which led to a decrease in the amount of storage and memory access needed.

Despite these efforts, employing DNNs in a resource-constrained environment remains to be a great challenge due to the nature of how DNNs were designed in the first place. DNNs negate the fact that an actual biological neuron in the human brain processes information based on discrete signals known as a spike train (a group of spikes), rather than a continuous value. Although the recent success of DNNs cannot be overlooked, a DNN is not biologically plausible, and overall efficiency and performance do not even come close to those of the human brain.

Spiking neural networks (SNNs), which are bio-inspired artificial neural networks, were introduced to mimic how information is processed in the human brain. Unlike conventional neural networks (e.g., CNNs, RNNs), SNNs transmit information via timing (temporal) and a spike train (discrete), rather than a real (continuous) value [Kasabov, 2014]. In a SNN, a neuron integrates spikes to membrane potential then, reaching a certain threshold, leads to firing of the neuron. This enables event-driven computation and therefore offers exceptional power-efficiency. Driven by these characteristics, SNNs hold stronger resemblance to an actual human brain than the the conventional neural networks, and more plausible in neuromorphic architecture [Merolla et al., 2014, Poon and Zhou, 2011].

SNNs have been successfully applied to various applications, yet relatively simple tasks such as image classification, moreover, only limited to shallow DNNs. The primary reason for its limited application scope is due to the lack of scalable training algorithm. Recent works proposed a DNN-to-SNN conversion method [Cao et al., 2015, Diehl et al., 2015] to

*Corresponding author

convert SNN successfully and achieve excellent performance. The DNN-to-SNN conversion methods are based on an idea of importing pre-trained parameters (e.g., weights and biases) from a DNN to an SNN. The DNN-to-SNN conversion methods achieved comparable performance in a deep SNN (e.g., VGG and ResNet) compared to that of an original DNN, although they have focused solely on classification tasks.

In this paper, we explore a deep SNN on a more advanced machine learning problem, namely, object detection. The object detection is considered a much more challenging task; both calculating precise coordinates for bounding boxes recognizing overlapping multiple objects. Thus, it requires the prediction of an accurate output value, rather than picking one class with the highest probability (i.e., argmax function), as done in the image classification. There are several concerns when applied to SNNs: inefficiency of the conventional normalization technique, and implementation of leaky-ReLU in SNNs.

To overcome these issues, we introduce two novel methods, called channel-wise normalization and signed neuron with imbalanced threshold. Consequently, we present a spike-based real-time object detection model, called Spiking-YOLO. As a first step towards object detection in the deep SNN, we implemented Spiking-YOLO based on Tiny YOLO [Redmon and Farhadi, 2017]. Tiny YOLO is known to have exceptional inference speed and is well-suited for a real-time object detection. To the best of our knowledge, this is the first deep SNN for a real-time object detection that achieves comparable results to those of DNNs on a non-trivial dataset, PASCAL VOC [Everingham et al., 2015]. Our contributions can be summarized as follows:

- **The first deep SNN for object detection** We present Spiking-YOLO that enables fast and accurate information transmission in the deep SNN. This is the first work that applies the deep SNN successfully to the object detection task.
- **Channel-wise normalization** We developed a fine-grained normalization technique for the deep SNN called channel-wise normalization. The proposed method enables a higher firing rate in multiple neurons, which led to fast and accurate information transmission.
- **Signed neuron featuring imbalanced threshold** We proposed a novel method featuring the signed neuron with imbalanced threshold which allows implementation of leaky-ReLU in SNNs. This creates opportunities for the deep SNN in various models and applications.

2 Related work

2.1 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 the input z into V_{mem} as

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

where $\Theta^l_j(t)$ is a spike and $z^l_j(t)$ is the input of ith neuron the in lth layer. $z^l_j(t)$ can be described as follows:

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

where w is a weight and b is a bias. A spike Θ is generated in the neuron when the integrated value V_{mem} exceeds a certain threshold V_{th} as

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

where $U(x)$ is a unit step function. This event-driven nature of SNNs has shown the possibility of energy-efficient operation, but it is difficult to train SNN, which has become the major obstacle in applying SNNs to various applications.

The training methods of SNNs can be divided into two types: direct and indirect. The direct training method consists of unsupervised learning with spike-timing-dependent plasticity (STDP) [Diehl and Cook, 2015], and supervised learning with gradient descent and error back-propagation [Jin et al., 2018, Wu et al., 2018]. Although STDP is biologically plausible, the learning performance is significantly lower than that of supervised learning, and there have been very few studies successfully applied STDP to the deep SNN.

Recent work proposed a new supervised learning algorithm with a function approximates the non-differential portion (integrate-and-fire) of SNNs. These works successfully implemented gradient descent and error back-propagation through a function approximation, however, they were limited to shallow SNNs or applied in relatively simple tasks such as image classification.

In recent years there has been growing interest in conversion of trained DNNs to SNNs. [Cao et al., 2015] proposed DNN-to-SNN mapping methods that neglected bias and max-pooling as an early stage work. In a subsequent work, [Diehl et al., 2015] proposed a data-based normalization in order to improve performance in the deep SNN.

[Rueckauer et al., 2017] demonstrated an implementation method of batch normalization and spike max pooling [Rueckauer et al., 2017]. [Kim et al., 2018, Park et al., 2019] proposed phases and burst coding, respectively, in order to transmit information precisely and effectively. These works have limited application of their methods such as image classification task. To the best of the author's knowledge, there has been few study which applied SNNs to regression problems such as object detection.

2.2 Object detection

The object detection locates multiple objects in a given image or video by drawing bounding boxes and classifying their classes. Hence, the object detection model consists of not only a classifier that classifies multiple objects, but also a regressor that predicts the precise coordinate (row and column), and size (width and height), of the bounding boxes. Since predicting precise coordinate of bounding boxes are crucial in object detection, it is considered as a much more challenging task than image classification where an argmax function is used to pick one class with the highest probability.

Typical deep learning frameworks for object detection are divided into two categories [Liu et al., 2018]: two-stage and

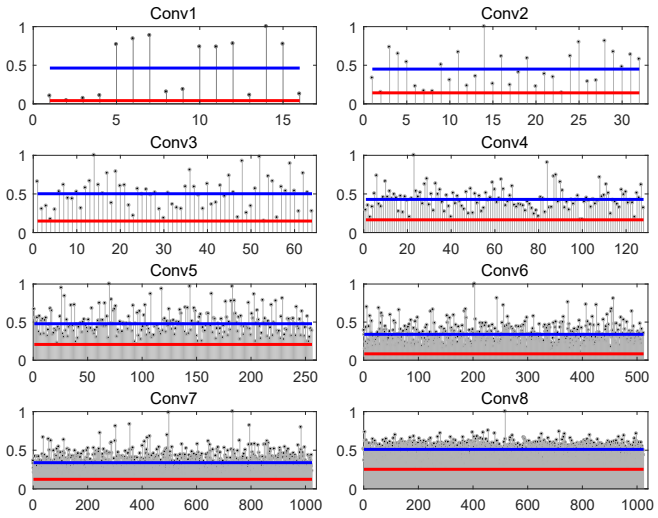

Figure 1: Distribution of normalized activation via layer-wise maximum activation in each channel for eight convolutional layers of Tiny YOLO. Blue and red line indicate the average and minimum of the normalized activations, respectively.

one-stage detection framework. In the two-stage detection frameworks, an additional component first obtains region proposals where the objects are likely to exist. Then, for each region proposal, the classifier determines which object is present, and the coordinate of the bounding box is produced by the regressor.

A major advance in object detection model, Region-based CNN (R-CNN) [Girshick et al., 2014] was proposed to perform the two-stage detection algorithm. Then, to accelerate detection speed and improve detection performance, several works have been proposed such as fast R-CNN [Girshick, 2015], faster R-CNN [Ren et al., 2015], and Mask R-CNN [He et al., 2017a] as an extended version of R-CNN.

However, the two-stage detection frameworks suffers from slow inference speed due to selective search which is an algorithm that creates region proposals has known to be the bottleneck of overall process. Moreover, processing through all possible region proposals is computationally expensive. Thus, it is not suitable for a real-time object detection task.

The one-stage detection framework presents an alternative approach to extract region proposals. The one-stage detection extracts the bounding box information and classifies the objects in one unified network. Single-shot multi-box detector (SSD) [Liu et al., 2016], and You look only once (YOLO) [Redmon et al., 2016] are known to be the state-of-the-art in one stage detection framework. SSD has been the most widely used model in object detection because of its superior performance in terms of accuracy [Liu et al., 2018].

YOLO, however, outperforms SSD in inference speed, or FPS (frames/s) by a large margin with slightly lower accuracy [Redmon and Farhadi, 2017, Redmon and Farhadi, 2018]. YOLO has simpler architecture and therefore requires less computational overheads. The inference speed is a critical factor in a real-time object detection and towards to employing object detection in an embedded system. Thus, we selected YOLO as our object detection model.

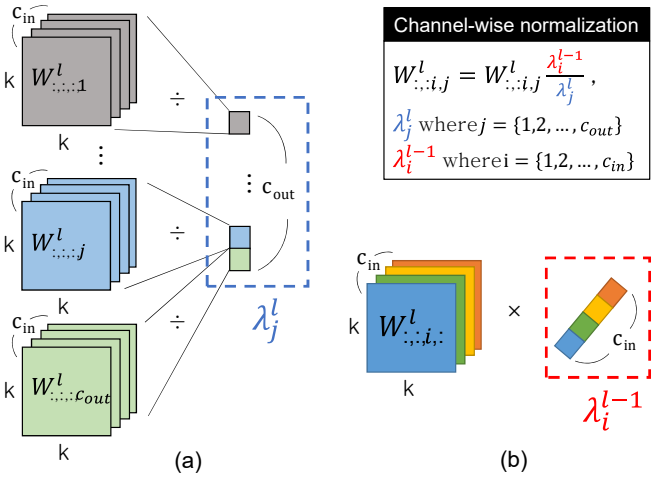

Figure 2: Proposed channel-wise normalization: (a) weights (denoted by W^l) are normalized by the maximum activation (λ^l) of each channel to obtain normalized activations, (b) the normalized activations are multiplied by λ^{l-1} to obtain original activations

3 Methods

3.1 Overview

In the object detection task, recognizing multiple objects and drawing bounding boxes around them (a regression problem), pose a great challenge, requiring high numerical precision in predicting output value. When applied in the SNN domain using conventional DNN-to-SNN conversion methods, it suffers from severe performance degradation. Our in-depth analysis presents possible explanations for the performance degradation: a) extremely low firing rate from layer-wise normalization, and b) absence of representation in negative activation for leaky-ReLU function. To overcome such obstacles, we propose two novel methods called channel-wise normalization and signed neuron with imbalanced threshold.

3.2 Channel-wise data-based normalization

In an SNN, it is important to ensure that a neuron generates spike trains according to the magnitude of the input value of a neuron. Both weights and threshold voltage, V_{th}, are responsible for sufficient and balanced activation of the neuron, otherwise under or over-activation occurs, which causes lost information, and poor performance [Diehl et al., 2015]. For instance, if the V_{th} is too large, then it will take a long period of time before the V_{mem} reaches the V_{th}, resulting in a low firing rate (i.e., under-activation). Vice versa, if the V_{th} is too small, then the V_{mem} is most likely to exceed the V_{th} and generate spikes regardless of the input value of the neuron (i.e., over-activation).

To prevent the neuron from under or over-activation, various data-based normalization techniques [Diehl et al., 2015, Rueckauer et al., 2016] have been proposed. The layer-wise normalization [Diehl et al., 2015] (abbreviated to layer-norm) is one of the most well-known techniques that normalizes weights in a specific layer by the maximum activation of the corresponding layer, calculated from running the training dataset (PASCAL VOC 2007+2012) in a DNN. This is based on an assumption that the distribution of the training and the

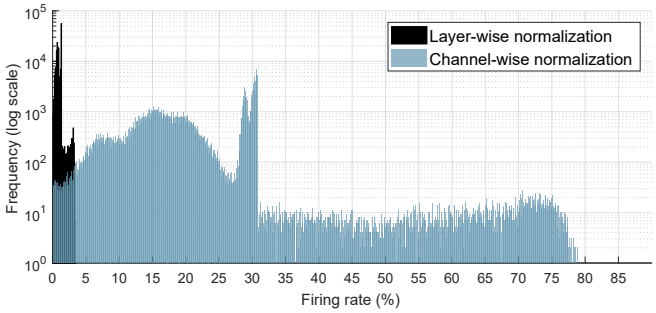

Figure 3: Spike count distribution for layer-wise and channel-wise normalization on channel 2 in conv. layer 1 in Tiny YOLO

test dataset are similar. The layer-norm can be calculated by

$$W^l \rightarrow W^l \frac{\lambda^{l-1}}{\lambda^l} \text{ and } b^l \rightarrow \frac{b^l}{\lambda^l}, \tag{4}$$

where w, b and λ are weights, bias and maximum activation in a layer l, respectively. Note that normalizing the weights by the maximum activation will have the same effect as normalizing the output activation.

As an extended version of the layer-norm, [Rueckauer et al., 2016] introduced an approach that normalizes activation by the 99.9th percentile of maximum activation which provides robustness to outliers ensuring sufficient firing of the neuron. According to our analysis, however, the conventional data-based normalization methods suffer from significant performance degradation when applied to object detection, as a result of under-activation.

Figure 1 represents the distribution of the normalized activation of all channels for eight convolutional layers in Tiny YOLO. Blue and red line indicate the average and minimum value of the normalized activation in each channel. As highlighted in Figure 1, the deviation of the maximum activation in a specific layer is relatively large. For example in Conv1 layer, the normalized maximum activation among 16 channels is close to 1, while the minimum activation is 0.041. Clearly, normalizing all weights by the maximum activation of the corresponding layer will result in exceptionally small activation (i.e., under-activation) in some channels, which had relatively small maximum activation prior to normalization.

This can be extremely problematic in solving regression problems in a deep SNN. For instance, in order to transmit 0.7, a total of 7 spikes and 10 time steps are required. Applying the same logic, transmitting 0.007 would require 7 spikes and 1000 steps without any losses of information. Hence, to send either extremely small (e.g., 0.007) or precise activation (e.g., 0.9007 vs. 0.9000), a large number of time steps (high resolution) is inevitable. Even though a large number of required time steps are given, a small but possibly important activation can disappear as the number of hidden layers grows in the deep SNN.

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

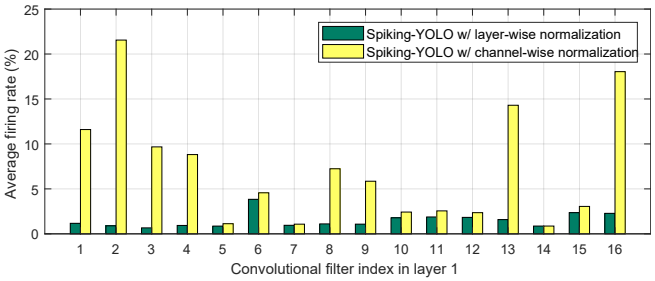

Figure 4: Average firing rate of 16 channels in conv. layer 1 for channel-wise and layer-wise normalization on Tiny YOLO

$$W^l_{:,:,i,j} \rightarrow W^l_{:,:,i,j} \frac{\lambda_i^{l-1}}{\lambda_j^l} \text{ and } b_j^l \rightarrow \frac{b_j^l}{\lambda_j^l}, \tag{5}$$

where i and j are the number of channels in the previous and current layer, respectively. Remember that activations (also an input to a following layer) are normalized to ensure sufficient activation by a factor of λ_j^l. In the following layer, normalized activation needs to be multiplied by λ_i^{l-1} to obtain original activation prior to normalization. The detailed method is depicted in Figure 2. Figure 2A describes how weights are normalized by the maximum activation of each channel, λ_j^l, obtained from training dataset. Figure 2B illustrates how normalized activations are re-amplified in the following by λ_i^{l-1} to obtain original activation.

By normalizing activation channel-wise, extremely small activations (under-activation) are eliminated. In other words, more neurons are properly normalized to obtain a higher firing rate, and eventually to lead to accurate results. The firing rate can be defined as

$$\text{firing rate } = \frac{N}{T}, \tag{6}$$

where N is the total number of spikes in a given time of period, T. Not only has the channel-norm provided sufficient activation of a neuron, but also led us to an important implication in efficient and accurate information transmission in the deep SNN.

Both figures 3 and 4 validate the point that channel-norm generates more spikes in a given fixed time step. Figure 3 demonstrates the spike count distribution in channel 2 of layer 1. It is obvious that channel-norm induces more neuron to generate a high number of spikes. Numerous neurons generated up to 80% firing rate. This is magnificent improvement in the firing rate produced by layer-norm. In the layer-norm, however, most of the neurons generated approximately in the rage of 0 and 3.5%. Figure 4 presents the average firing rate in each channel in layer 1. Evidently, the channel-norm produces much larger average spike rate in majority of the channels. Especially in channel 2, the channel-wise firing rate is 20 times more than the layer-wise firing rate.

The evidence from our in-depth analysis verifies that fine-grained channel-norm increases the firing rate of the neuron with extremely small activations. In other words, extremely

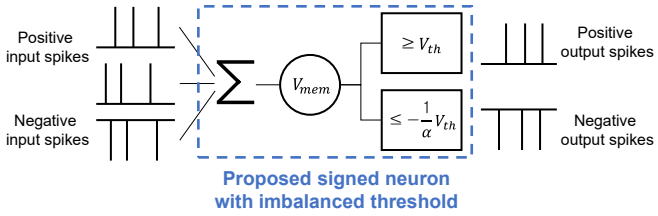

Figure 5: Implementation method of our signed neuron featuring imbalanced threshold

small activations are normalized properly that will transmit information accurately in a shorter period of time. These small activations may not mean much, and may have very little impact on the final output of the network in a simple application such as image classification. However, the small activations can be critical in regression problems to obtain accurate results. Thus, applying channel-norm provides a fast and accurate deep SNN and can be a viable solution to solving more advanced machine learning problems in the deep SNN.

3.3 Signed neuron featuring imbalanced threshold

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, typically set to 0.01; $f(x) = x$ when $x \geq 0$, otherwise $f(x) = \alpha x$ [Xu et al., 2015].

Most of the previous studies focused on converting IF (integrate-and-fire) neurons to ReLU function, and completely neglected the leakage term. That is, once a neuron, which might contain useful information, becomes negative, then it will be zero, and information contained will be useless over an entire course of the network. To extend the activation function bound to the negative region in SNNs, [Rueckauer et al., 2017] added second V_{th} term of -1 to generate spikes of size -1. Their method successfully converted BinaryNet [Hubara et al., 2016] to SNNs where in BinaryNet activations are constrained to $+1$ or -1, in SNNs on the CIFAR-10 dataset.

In a similar manner, we applied their method to object detection in the SNN domain. Despite the additional negative V_{th}, the results were very poor. Our analysis shows that over-activation occurs frequently as a result of neglecting the leakage term, the slope of α in leaky-ReLU. Note that Tiny YOLO account for over 40% in all acitvations. To take the leakage term into consideration, a different approach should be taken.

In this paper, we introduce a signed neuron featuring imbalanced threshold (henceforth abbreviated as IBT) that can not only interpret both the positive and negative activations, but also compensate for the leakage term in the negative region in leaky-ReLU. As shown in Figure 5, we add another V_{th} that is responsible for negative activation. Then V_{th} is divided by the slope, $-\alpha$. This would replicate the leakage term, slope, α in the negative region of leaky-ReLU, in the SNN domain. For example, if the $\alpha = 0.1$ then the threshold voltage responsible for positive activation $V_{\text{th,pos}}$ is equal to 1, and for negative activation $V_{\text{th,neg}}$ is equal to -10 . In other

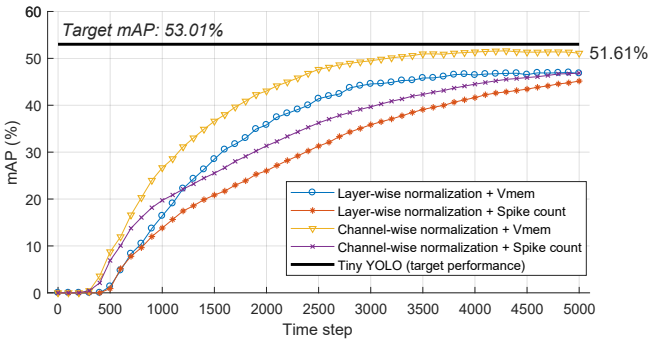

Figure 6: Experimental results for Spiking-YOLO on various configurations (normalization technique + decoding scheme)

words, the V_{mem} needs to be integrated ten times more when compared to $V_{\text{th,pos}}$, in order to reach the $V_{\text{th,neg}}$ and generate negative spikes. The basic dynamics of the signed neuron with IBT are represented by the V_{th}

$$fire(V_{\text{mem}}) = \begin{cases} 1 & \text{if } V_{\text{mem}} \geq V_{\text{th}} \\ -1 & \text{if } V_{\text{mem}} < -\frac{1}{\alpha}V_{\text{th}} \\ 0 & \text{otherwise, no firing.} \end{cases} \quad (7)$$

Typically, leaky-ReLU is implemented in a way that α is multiplied in all negative activations. Nonetheless, it is not biologically plausible because a spike is a discrete signal, not a continuous value. Additional hardware operation and resources are also required to multiply a neuron's activation by α from a hardware implementation standpoint. The proposed IBT, however, allows for discrete characteristics of a spike by implementing a leaky-ReLU term by having an additional threshold, $V_{\text{th,neg}}$ / α. In addition, our implementation features one signed neuron, though excitatory and inhibitory neurons can be implemented, which is more biologically plausible. Through the use of the proposed signed neuron with IBT, leaky-ReLU can be employed in SNNs creating more possibilities in converting various DNN models to SNNs on wide range of applications.

4 Evaluation

4.1 Experimental setup

As the first step towards object detection in deep SNN, we used a real-time object detection model called Tiny YOLO which is an efficient version of YOLO. We implemented max-pooling and batch-normalization layers in SNNs according to [Rueckauer et al., 2017]. Tiny YOLO is trained on PASCAL VOC 2007 and 2012, and tested on the PASCAL VOC 2007, a non-trivial dataset. Our simulation is based on the Tensorflow Eager framework and we conducted all experiments on NVIDIA Tesla V100 GPUs.

4.2 Results

To verify and analyze the roles of proposed methods, we experimented on the presence or absence of applying channel-norm and signed neuron with IBT. As depicted in Figure 6, when both channel-norm and signed neuron with IBT are applied, Spiking-YOLO achieves the remarkable performance of 51.61% mAP (Note that target mAP of Tiny YOLO is

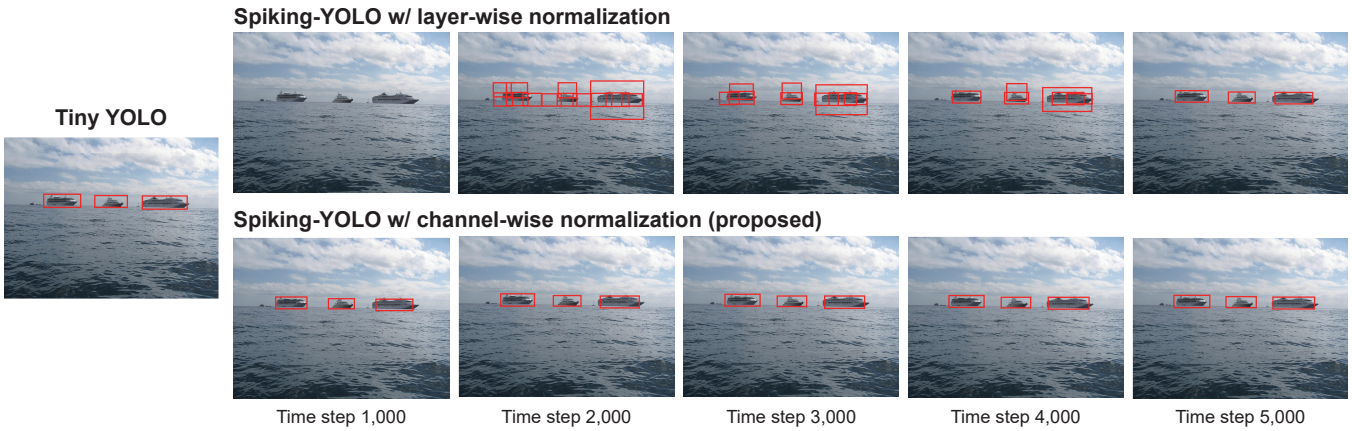

Figure 7: Object detection results (Tiny YOLO vs. Spiking-YOLO with layer-norm vs. Spiking-YOLO with channel-norm as time step increases)

53.01%). In contrast to layer-norm (46.98% mAP), channel-norm outperforms by a large margin and converges faster at the same time. These results verify that channel-norm enhances the firing rate and enables faster yet more accurate, information transmission than layer-norm. Furthermore, it should be noted that without both of our proposed methods, Spiking-YOLO failed miserably in detecting any object, reporting 0% mAP. When only the signed neuron with IBT is applied, it still struggles to detect objects, reporting approximately 7.3% mAP at best (both results are omitted from the graph of Figure 6). This is a great indication that the signed neuron with IBT compensates for leakage term in leaky-ReLU and plays a critical role in solving such high numerical precision problems in a deep SNN. Thus, the rest of the experiments (shown in Figure 6) were conducted by applying the signed neuron with IBT method as a default.

Figure 8: Object detection results for small objects (Tiny YOLO vs. Spiking-YOLO w/ layer-norm vs. Spiking-YOLO w/ channel-norm @ 5000 time steps)

For further analysis, we provide additional experiments on two different output decoding schemes: one based on accumulated $V_{\rm th}$, and another based on spike count. The accumulated spike count can be simply calculated by dividing $V_{\rm mem}$ by $V_{\rm th}$. The quotient from the $V_{\rm mem}$ / $V_{\rm th}$ is the spike count, and the remainder are rounded off, which will eventually be an error and lost information. Therefore, the $V_{\rm th}$ based output decoding scheme will be more precise at interpreting spike trains and Figure 6 verifies that assertion. The $V_{\rm th}$ based output decoding scheme outperforms spike based and also converges faster in both channel and layer-norm.

Figure 7 illustrates how well Spiking-YOLO detects objects as the time steps increases. Far left picture (Tiny YOLO) is the ground truth label that our Spiking-YOLO is trying to replicate. Remarkably, after only 1000 time steps, Spiking-YOLO with channel-norm was able to successfully detect all three objects. On the other hand, Spiking YOLO with layer-norm failed to detect any object. After 2000 time steps, it starts to draw bounding boxes of a few objects, but none of them are accurate. There are multiple bounding boxes over one object and sizes are just different. It improves detection performance as the time step increases though still unsatisfactorily, and 5000 time steps are required to reach the detection performance that of proposed channel-norm. Our channel-norm shows a clear advantage over detecting multiple objects in a shorter period of time.

Figure 8 shows another sample picture with drawn bounding boxes from Tiny YOLO (DNN), and Spiking-YOLO with layer-norm and channel-norm after 5000 time steps. Notice that Spiking-YOLO with channel-norm is accurately recognizing very small objects compared to layer-norm. Both of these observations clearly demonstrate that channel-norm increases firing rate of multiple neurons (especially with small activation) and provides fast and accurate information transmission in the deep SNN.

5 Conclusion

In this paper, we have presented Spiking-YOLO, a spiking neural network for real-time object detection. To the best of our knowledge, Spiking-YOLO is the first SNN that was applied in object detection and achieved comparable results to those of the original Tiny YOLO on a non-trivial dataset, PASCAL VOC. In doing so, we propose two novel methods, called channel-wise normalization and signed neuron featuring imbalanced threshold. Our fine-grained, channel-wise normalization method was able to increase the firing rate of extremely small activations to ensure fast yet accurate information transmission. The notion of the imbalanced threshold also enables implementation of a leaky-ReLU in the SNN domain. Note that our proposed method can be applied to any variant of an activation function such as PReLU, ELU and so on. We believe that this study is the first step towards solving more advanced machine learning problems in a deep SNN.

References

[Cao et al., 2015] 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.

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

[Diehl et al., 2015] 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*.

[Everingham et al., 2015] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136.

[Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *ICCV*.

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

[Guo et al., 2016] Guo, Y., Yao, A., and Chen, Y. (2016). Dynamic network surgery for efficient dnns. In *NIPS*.

[Han et al., 2016] 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 et al., 2017a] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017a). Mask r-cnn. In *ICCV*.

[He et al., 2017b] He, Y., Zhang, X., and Sun, J. (2017b). Channel pruning for accelerating very deep neural networks. In *ICCV*.

[Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[Hubara et al., 2016] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks. In *NIPS*.

[Jin et al., 2018] Jin, Y., Zhang, W., and Li, P. (2018). Hybrid macro/micro level backpropagation for training deep spiking neural networks. In *NIPS*.

[Kasabov, 2014] Kasabov, N. K. (2014). Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*, 52:62–76.

[Kim et al., 2018] Kim, J., Kim, H., Huh, S., Lee, J., and Choi, K. (2018). Deep neural networks with weighted spikes. *Neurocomputing*, 311:373–386.

[Kim et al., 2015] 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. *arXiv preprint arXiv:1511.06530*.

[Liu et al., 2018] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. (2018). Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*.

[Liu et al., 2016] 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*.

[Merolla et al., 2014] 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.

[Park et al., 2019] 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*.

[Park et al., 2018] Park, S., Kim, S., Lee, S., Bae, H., and Yoon, S. (2018). Quantized memory-augmented neural networks. In *AAAI*.

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

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *CVPR*.

[Redmon and Farhadi, 2017] Redmon, J. and Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In *CVPR*.

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

[Ren et al., 2015] 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 et al., 2016] Rueckauer, B., Lungu, I.-A., Hu, Y., and Pfeiffer, M. (2016). Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052*.

[Rueckauer et al., 2017] 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.

[Wu et al., 2018] Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. (2018). Direct training for spiking neural networks: Faster, larger, better. In *AAAI*.

[Xu et al., 2015] Xu, B., Wang, N., Chen, T., and Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

[Zagoruyko and Komodakis, 2017] Zagoruyko, S. and Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*.