공학박사 학위논문

Towards Fast and Accurate Information Transmission in Deep Spiking Neural Networks

딥 스파이킹 뉴럴 네트워크의
빠르고 정확한 정보 전달

2021년 2월

서울대학교 대학원
전기·컴퓨터공학부
김 세 준

Towards Fast and Accurate Information Transmission in Deep Spiking Neural Networks

지도교수 윤 성 로

이 논문을 공학박사 학위논문으로 제출함

2021년 2월

서울대학교 대학원

전기·컴퓨터공학부

김 세 준

김세준의 박사 학위논문을 인준함

2021년 2월

위 원 장	김 태 환	(인) [illegible]
부위원장	윤 성 로	(인) [illegible]
위 원	정 의 영	(인) [illegible]
위 원	유 승 주	(인) [illegible]
위 원	김 재 하	(인) [illegible]

Abstract

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. Nonetheless, solving such advanced machine learning problems in real world applications requires a more sophisticated model with a vast number of parameters and training data, which leads to substantial amounts of computational overhead and power consumption. Given these circumstances, spiking neural networks (SNNs) have attracted growing interest as the third generation of neural networks due to their event-driven and low-powered nature. SNNs were introduced to mimic how information is encoded and processed in the human brain by employing spiking neurons as computation units. SNNs utilize temporal aspects in information transmission as in biological neural systems, thus providing sparse yet powerful computing ability.

SNNs have been successfully applied in several applications, but these applications only include relatively simple tasks such as image classification, and are limited to shallow neural networks and datasets. One of the primary reasons for the limited application scope is the lack of scalable training algorithms attained from non-differential spiking neurons. In this dissertation, we investigate deep SNNs in a much more challenging regression problem (i.e., object detection), and propose a first object detection model in deep SNNs which is able to achieve comparable results to those of DNNs in non-trivial datasets. Furthermore, we introduce novel approaches to improve performance of the object detection model in terms of accuracy, latency and energy efficiency. This dissertation contains mainly two approaches: (a) object detection model in deep SNNs, and (b) improving performance of object detection model in deep SNNs. Consequently, the two approaches enable fast and accurate object detection in deep SNNs.

The first approach is an object detection model in deep SNNs. We present a spiked-based object detection model, called Spiking-YOLO. To the best of our knowledge, Spiking-YOLO is the first spiked-based object detection model that is able to achieve comparable results to those of DNNs on a non-trivial dataset, namely PASCAL VOC and MS COCO. In doing so, we introduce two novel methods: a channel-wise weight normalization and a signed neuron with imbalanced threshold, both of which provide fast and accurate information transmission in deep SNNs. Our experiments show that Spiking-YOLO achieves remarkable results that are comparable (up to 98%) to those of Tiny YOLO (DNNs) on 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 DNN-to-SNN conversion methods.

The second approach aims to provide a more effective form of computational capabilities in SNNs. Even though, SNNs enable sparse yet efficient information transmission through spike trains, leading to exceptional computational and energy efficiency, the critical challenges in SNNs to date are two-fold: (a) latency: the number of time steps required to achieve competitive results and (b) synaptic operations: the total number of spikes generated during inference. Without addressing these challenges properly, the potential impact of SNNs may be diminished in terms of energy and power efficiency. We present a threshold voltage balancing method for object detection in SNNs, which utilizes Bayesian optimization to find optimal threshold voltages in SNNs. We specifically design Bayesian optimization to consider important characteristics of SNNs, such as latency and number of synaptic operations. Furthermore, we introduce two-phase threshold voltages to provide faster and more accurate object detection, while providing high energy efficiency. According to experimental results, the proposed methods achieve the state-of-the-art object detection accuracy in SNNs, and converge 2x and 1.85x faster than conventional methods on PASCAL VOC and MS COCO, respectively. Moreover, the total number of synaptic operations is reduced by 40.33% and 45.31% on PASCAL VOC and MS COCO, respectively.

keywords: Neural Network, Deep Learning, Spiking Neural Network, Energy Efficiency, Object Detection, Bayesian Optimization

student number: 2013-23104

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past decade, deep neural networks (DNNs) have been becoming one of the most popular choice for solving machine learning problems. Given its great success in a variety of applications such as image classification [51, 34], speech recognition [4, 14], and natural language processing (NLP) [17, 15], DNNs are now being further applied in more intriguing tasks such as genome sequencing [3, 101], and self-driving [62, 48], making our daily lives better in all aspects. One of the primary reasons behind the recent success of 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 [30, 36], compression [33, 50], quantization [28, 70], and knowledge distillation [37, 56], some of which have shown promising results. Despite these efforts, the demand for computing and power resources will most likely to increase as deeper and more complicated neural networks achieve higher accuracy [86]. For instance, [11] recently proposed a GPT-3 language model which has 175 billion parameters and requires approximately 3.11×10^{23} floating point operations (FLOPs) to train the model. Nonetheless, GPT-3 has already shown remarkable performance in various NLP tasks such as generating news articles, language translation, and answering standardized test questions.

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 [60]. Unlike conventional neural networks such as DNNs, 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 [61], thus providing sparse yet powerful computing ability [66, 6]. 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 [63, 75].

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 [52, 94]. 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. The direct training method of SNNs consists of unsupervised learning with spike-timing-dependent plasticity (STDP) [18] and supervised learning with gradient descent and error back-propagation

[52]. 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 can approximate the non-differentiable part (e.g., integrate-and-fire) of SNNs [44, 52] to improve the learning performance. Furthermore, [98] proposed temporal spike sequence learning back-propagation to train SNNs with fewer steps while improving classification accuracy. Despite these efforts, most previous works have been limited to the image classification task and MNIST and CIFAR dataset on relatively shallow neural networks.

DNN-to-SNN conversion methods, as an alternative approach, have been studied widely in recent years [13, 19, 83]. These methods are also known as in-direct training method of SNNs and are based on the idea of importing pre-trained parameters (e.g., weights and biases) from a DNN to an SNN. In the DNN-to-SNN conversion method, DNNs are converted into SNNs that can be directly mapped to spike-based neuromorphic hardware with minimum performance loss [13]. DNN-to-SNN conversion methods have a great advantage over the direct training methods in a way that it is not necessary to train sophisticated deeper neural networks in SNNs, but simply use trained parameters from DNNs in SNNs. DNN-to-SNN conversion methods aim to bridge a large performance gap between DNNs and SNNs and have successfully achieved comparable results in deep SNNs to those of original DNNs (e.g., VGG and ResNet); however, the results from MNIST and CIFAR datasets were competitive, while those of complex dataset, such as ImageNet dataset, were unsatisfactory when compared with DNN's accuracy.

Object detection is a core task in computer vision that is adopted in a wide range of applications such as autonomous driving [92, 54], surveillance systems [67], robots [43], and drone navigation [12]. It is often considered as a more challenging task than image classification, as it involves recognizing multiple and possibly overlapping objects, calculating precise coordinates of bounding boxes, and predicting associated class probabilities. Since localizing objects in object detection is referred to as a regression problem, high-numerical precision is required to achieve high object detection accuracy, which leads to large computational overhead. This limits their ability to perform real-time detection. In recent years, various architectures and techniques have been proposed to enable real-time and efficient object detection, namely one-stage detectors [79, 8, 59], their variants [91, 25, 39, 24], and quantized detectors [55, 20, 89]. Nevertheless, these efforts still fall short of the demand of real-world applications on resource-constrained edge devices.

In this dissertation, we first 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 weight 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 weight 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 [80]. 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.

Chapter 3 is based on the following paper:

- Seijoon Kim, Seongsik Park, Byunggook Na, Sungroh Yoon, "Spiking-YOLO: Spiking Neural Network for Energy-Efficient Object Detection," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Our contributions in Chapter 3 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 weight normalization** We developed a fine-grained weight 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.

The proposed Spiking-YOLO improves efficiency of object detection [49] when compared to object detection models in DNNs. However, one of its drawbacks is that it requires a significant amount of time steps (latency) and spikes (synaptic operations) to provide high-numerical precision and high object detection accuracy, which can directly translate into higher energy and power consumption. For instance, SNNs executing object detection would require over 2x the latency and 100x the number of spikes, when compared to image classification [47, 83, 49]. Without addressing these problems properly, the potential impact of SNNs can be diminished, particularly in terms of energy efficiency. Consequently, the significant benefits of SNNs over DNNs may no longer be the same. For instance, in image classification, SNNs (e.g., VGG and ResNet architecture) yield latency of between 2,000 and 2,500 time steps and generate up to 86.5M spikes on CIFAR-100 [47, 83]. As for a more complex object detection, SNNs (e.g., Tiny YOLO) require up to 5,000 time steps and produce 38.2B spikes on PASCAL VOC [49]. That is, SNNs executing object detection would require over 2x the latency and 100x the number of spikes, when compared to image classification.

In recent years, various approaches have been proposed to improve performance of SNNs in terms of both accuracy and efficiency, but these have been limited to image classification. These approaches include weight and threshold voltage balancing methods [19, 73, 83] and neural coding schemes [71, 97, 72]. In most of the existing approaches, the firing rate of neurons is regulated by a single uniform value in each channel (e.g., maximum activation value) while having the same threshold voltage. In neuroscience literature, neurons in different regions of brain represent distinct dynamics and process information differently than other regions [10, 87, 100]. The threshold voltage of neurons is also known to have a broad range rather than a single value [5]. A few studies have used various threshold voltages, but these values were determined heuristically, leaving much room for optimization [32, 31].

Inspired from these observations, we propose a threshold voltage balancing method that improves object detection performance in SNNs in terms of three important aspects: accuracy, latency, and number of synaptic operations. The proposed threshold voltage balancing method can automatically scale the threshold voltages in each hidden layer to optimal values by employing Bayesian optimization. We specifically design Bayesian optimization such that it considers distinctive characteristics of SNNs, namely, latency and number of synaptic operations. We thoroughly investigated optimal evaluation procedure and hyper-parameters of Bayesian optimization specifically for SNNs. Moreover, proxy evaluation is introduced to accelerate optimization process with a consideration of accuracy-latency trade-off in SNNs.

Furthermore, we introduce the concept of two-phase threshold voltages. The threshold voltages in each phase have two different objectives; phase-1 for fast object detection and phase-2 for accurate object detection. By utilizing phase-1 threshold voltages for the early part of time steps in SNNs, then swapping in phase-2 threshold voltages later, SNNs achieve not only fast convergence speed but also state-of-the-art object detection accuracy with significantly less number of synaptic operations.

Chapter 4 is based on the following paper:

- Seijoon Kim, Seongsik Park, Byunggook Na, Jongwan Kim, Sungroh Yoon, ”Towards Fast and Accurate Object Detection in Bio-Inspired Spiking Neural Networks Through Bayesian Optimization,” in *IEEE Access*, 2021.

The key contributions of Chapter 4 can be summarized as follows:

- **Threshold voltage balancing through Bayesian optimization** We present a threshold voltage balancing method using Bayesian optimization that finds optimal threshold voltages to improve performance of SNNs in terms of accuracy, latency and number of synaptic operations.

- **Bayesian optimization specifically designed for SNNs** We design Bayesian optimization to consider two important characteristics of SNNs in addition to object detection accuracy: latency, and number of synaptic operations. Moreover, we employ proxy evaluation of SNNs to reduce large computational overhead of the optimization process.

- **Two-phase threshold voltages for faster and more accurate object detection in SNNs** We introduce the concept of two-phase threshold voltages that can provide low latency while achieving state-of-the-art object detection accuracy. By substantially reducing latency and number of synaptic operations, the proposed methods can provide highly energy-efficient object detection in SNNs.

In summary, this dissertation underlines the importance of energy efficiency of DNNs, which are increasingly becoming a critical factor as we try to deploy DNNs in resource-constrained environments (e.g., embedded systems, edge devices) for various applications. As demonstrated in Chapter 3, we propose the first object detection model in SNNs, namely Spiking-YOLO, which can significantly improve energy efficiency compared to DNNs while achieving detection accuracy close to DNNs (up to 98%). In Spiking-YOLO, we propose two approaches: channel-wise weight normalization and signed neuron with imbalanced threshold voltage. Furthermore, in Chapter 4, we propose a new threshold voltage balancing method and introduced two-phase threshold voltages which can improve performance of Spiking-YOLO in terms of detection accuracy, latency, and total number of synaptic operations.

All the proposed methods in Chapter 3 and 4 ultimately enable fast and accurate object detection while offering extremely high energy efficiency. Other than signed neuron with imbalanced threshold voltage, the proposed methods in Chapter 3 and 4 are interchangeable and compliment with others. To elaborate on this point, signed neuron with imbalanced threshold precisely implements a leakage term in leaky-ReLU for SNNs, and object detection model struggles to detect objects without the proposed signed neuron with imbalanced threshold. Thus, signed neuron with imbalanced threshold are used by default in the object detection model. On the other hand, channel-wise weight normalization and threshold voltage balancing through

Bayesian optimization are compliment to each other and using the both provides more sufficient activation of spiking neurons in deep SNNs. Furthermore, two-phase threshold voltages can be looked as a results from various design choices of the proposed threshold voltage balancing through Bayesian optimization, which are obtained by alternating a number of time steps during evaluation in Bayesian optimization.

The rest of the dissertation is organized as follows: Chapter 2 illustrates background in details. Chapter 3-4 present the proposed methods that enable fast and accurate information transmission in deep SNNs while providing high energy efficiency. We conclude this dissertation with conclusion and future work in Chapter 5.

Chapter 2

Background

2.1 Object detection

Object detection is one of the most fundamental computer vision task that is applied in a variety of applications such as autonomous driving [92, 54], surveillance systems [67], robots [43], and drone navigation [12]. It aims to provide semantic understanding of an image or video by recognizing multiple and possibly overlapping objects (if existed), classifying associated class that each object belongs to, and predicting location of each object with precise coordinates. 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. Figure 2.1 compares object detection and image classification.

With the recent success of DNNs, object detection has made a significant improvement in terms of detection accuracy over the last decade. However, there are still difficulties encountered in general, namely overlapped objects, objects in extremely small size, and quality of an input image (or video) (e.g., occlusion, blur, and

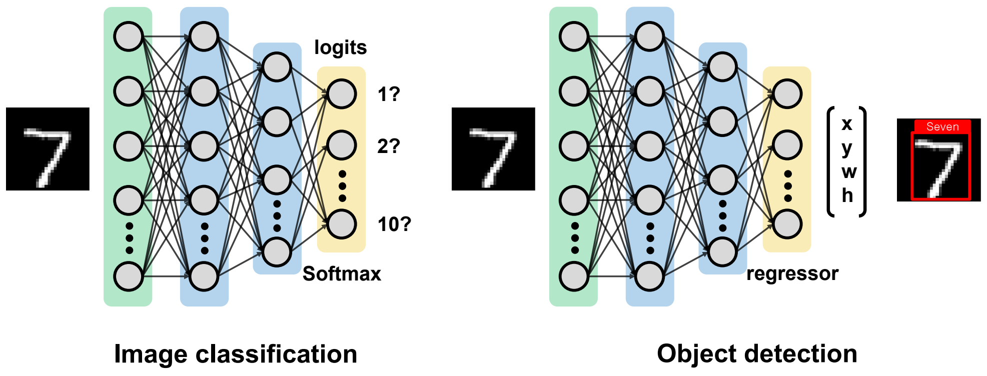


Figure 2.1: Image classification vs. object detection

different angle). These have led to much attention to neural network in object detection to overcome such difficulties and various algorithms and techniques have been proposed in recent years [26, 81, 59, 79, 79]. Nonetheless, the main challenges in object detection to this date are detection speeds (FPS: frame per second) and computational overhead (FLOPS: floating point operations per second) while providing high object detection accuracy.

These main challenges originated from how traditional object detection framework was designed. As mentioned previously, object detection aims to recognize objects, classifies their associated class (classification), and draws precise bounding boxes around them (localization). Naturally, the traditional object detection framework was designed in two stages: (a) first stage searches for possible regions that would have objects in them (i.e., region proposals) using algorithms such as selective search, (b) second stage classifies each region proposal. This two-stage object detection framework limits their ability to perform real-time detection. To provide real-time object detection, one-stage object detection frameworks have been proposed. These frameworks perform classification and localization in unified network to significantly reduce computation overhead and execution time. The details of two-stage and one-stage object detection framework is shown in Figure 2.2 and is explained in

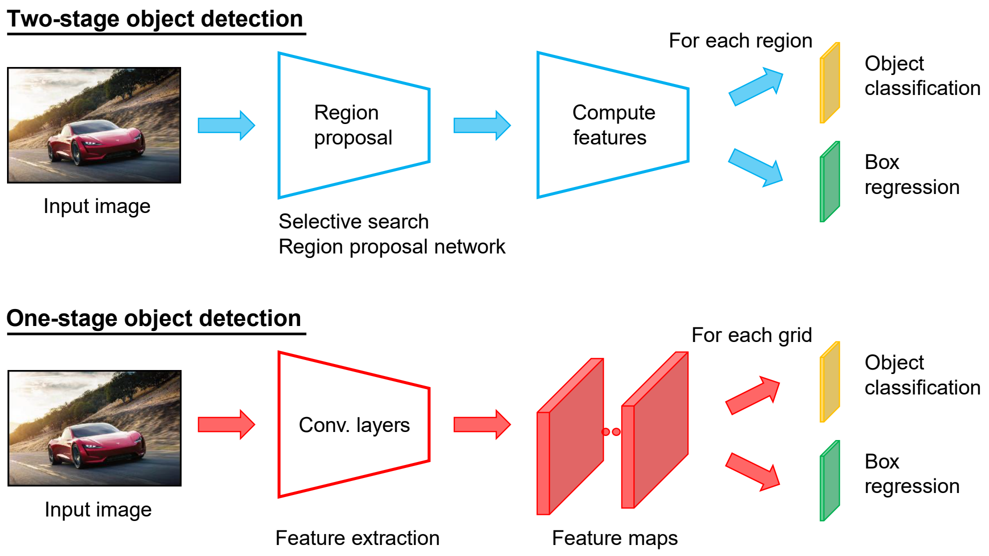
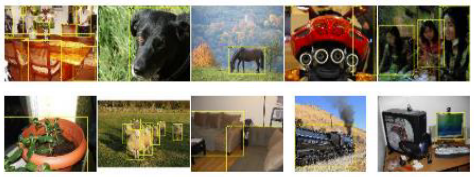


Figure 2.2: Two-stage object detection vs. one-stage object detection

details as follows.

Region-based CNN (R-CNN) [27], two-stage object detection framework, is considered to be one of the most significant advances in object detection. R-CNN extract a set of vector from each region proposal through selective search [88]. Then each proposal is scaled and fed into a CNN model trained on ImageNet to extract features. Linear SVM classify the object in one image at the last layer of the CNN. Then, a linear regressor predicts the bounding box coordinates of the classified object more precisely. Although R-CNN has made great progress, its drawbacks are redundant overlapped proposals without sharing computation. This leads to an extremely slow detection speed. To improve detection performance and speed, various extended versions of R-CNN have been proposed, namely fast R-CNN [26], faster R-CNN [81], and Mask R-CNN [35]. 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 object detection framework has been pro-


(a) PASCAL VOC

(b) MS COCO

Figure 2.3: Example of object detection datasets

posed, where the bounding box information is extracted and objects are classified in a unified network. In one-stage object detection framework, Single-shot multi-box detector (SSD) [59] and You only look once (YOLO) [79] achieve the state-of-the-art performance. YOLO divides the image into grids and predicts bounding boxes and probabilities for each region simultaneously. The author has made a series of improvements on basis of YOLO and has proposed its YOLO v2 [78], YOLO v3 [79], which further improve the detection accuracy while keeps a detection speed. Particularly, YOLO has superior inference speed (FPS) without a significant loss of accuracy, which is a critical factor in real-time object detection. In recent years, variants of these one-stage object detection framework have been proposed such as architecture variants [91, 25, 39, 24], and quantized detectors [55, 20, 89]. Nevertheless, these efforts still fall short of the demand of real-world applications on resource-constrained edge devices. Recent years have seen considerable interest in SNNs because of their exceptional energy efficiency.

Popular datasets for object detection are PASCAL VOC [22] and MS COCO [57]. PASCAL VOC is based on PASCAL Visual Object Classes Challenges that took place in 2005 to 2012. PASCAL VOC 2012 is consisted of total of 20 classes including person, dogs, horses, chairs, sofas and so on. The train and validation data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations.

Most of the researchers to date uses PASCAL VOC (2007 + 2012) when validating performance of their proposed model. Another popular dataset for object detection is MS COCO which is known to be more challenging dataset than PASCAL VOC. It includes large-scale object detection, segmentation, and captioning dataset. The train and validation data has over 300k images (200k are labeled) with 1.5 million instances and 80 object classes. Moreover, MS COCO has sample boxes of 7.4 per image when compared to 2.4 in PASCAL VOC (2007 + 2012). Since MS COCO is considered as more challenging object detection dataset and detection accuracy on PASCAL VOC is somewhat saturated, more researchers in the recent years use MS COCO dataset to report detection performance in their research paper.

In image classification, accuracy is often used as performance metric of the model. Since image classification only involves predicting a class of an input image, accuracy can be easily calculated by comparing ground truth and class prediction of the model. On the other hand, object detection involves multiple objects with their associated classes and bounding boxes. That is, performance metric for object detection has to consider both classification and bounding boxes regression. Thus, object detection uses mean average precision (mAP) as the performance metric. In order to calculate mAP, we first need to calculate recall and precision which can be defined as

$$Precision = \frac{\text{True positive}}{\text{True positive} + \text{False positive}}, \tag{2.1}$$

$$Recall = \frac{\text{True positive}}{\text{True positive} + \text{False negative}}. \tag{2.2}$$

In object detection, precision measures false positive rate and can be defined as fraction of correctly predicted objects among all predicted objects made by the model. Recall, also refers to sensitivity, measures false negative rate and can be de-

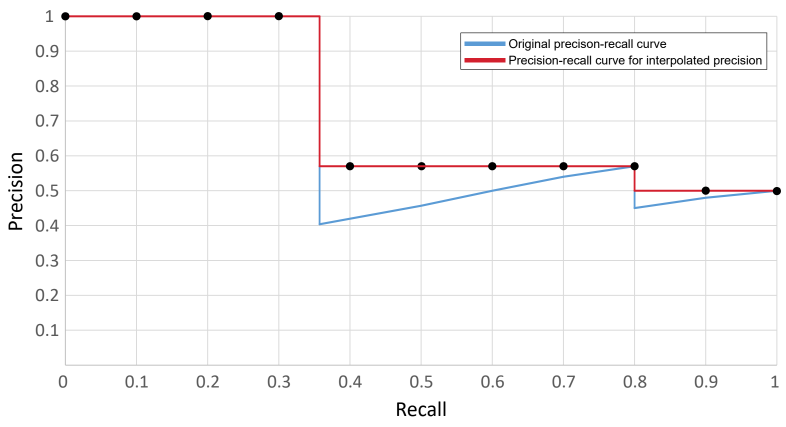


Figure 2.4: Calculation of mean average precision (mAP)

fined as fraction of correctly predicted objects among all objects in an image. Then precision-recall curve is computed in order to calculate average precision (AP). Using the precision-recall curve, AP computed as average of maximum precision values on 11 sample points in recall as shown in Figure 2.4. Please note that precision value at each sample point is obtained with the maximum precision value to the right of the sample point. AP is calculated on each class then mAP calculates average of AP for the entire classes. Also note that intersection over union (IoU) is a measure of how two bounding boxes are overlapped which can be calculated as

$$IoU = \frac{\text{Area of overlap}}{\text{Area of union}}. \tag{2.3}$$

IoU over 50% between the ground truth and predicted bounding box is considered as correct prediction (correct = True). In MS COCO, mAP is calculated on multiple IoU values rather than a single IoU at 50% (PASCAL VOC). For instance, [79] reports mAP in three different IoU values (i.e., mAP, mAP_{50}, mAP_{75}).

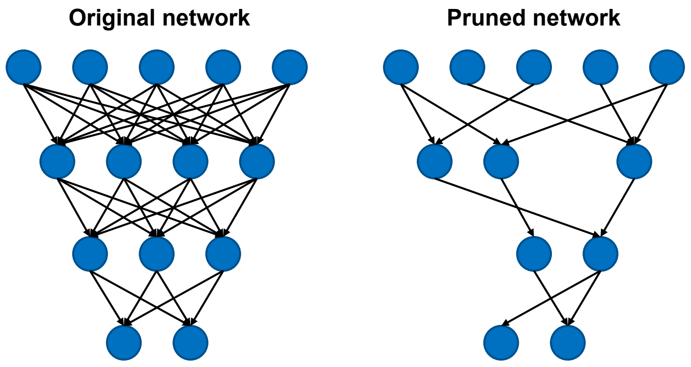


Figure 2.5: Example of pruning neural networks

2.2 Spiking Neural Networks

Recently, many have attempted to design energy- and computation-efficient DNNs via pruning [30, 36, 65], compression [33, 50], and quantization [28, 70, 21], some of which have shown promising results. Pruning [36] and compression [33] techniques aim to reduce computational overheads by eliminating redundancy, keeping only important parts of network, while preserving accuracy. [70] adopted a quantization technique to reduce number of bits required to represent a model, which led to a decrease in amount of storage and memory access needed. Recently, concept of knowledge transfer has been proposed to enhance compression ratio of a network by transferring the knowledge (e.g., attention map [96], softened distribution [37]), from a teacher (larger) network to a student (smaller) network.

Despite these efforts, employing DNNs in a resource-constrained environment remains a great challenge due to the nature of how DNNs are designed in the first place. DNN processes information based on continuous value through number of hidden layers. Through heavy matrix multiplications and summations, DNN predicts a final answer based on extracted features and any patterns learned from training data. 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

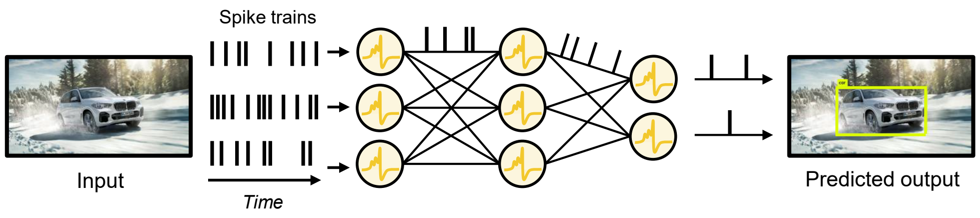


Figure 2.6: Overview of Spiking Neural Networks

than a continuous value. Although the recent success of DNNs cannot be overlooked, DNNs are not biologically plausible, and overall, their efficiency and performance do not even come close to those of the human brain.

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

$$V^l_{\text{mem},j}(t) = V^l_{\text{mem},j}(t-1) + z^l_j(t) - V_{\text{th}}\Theta^l_j(t), \tag{2.4}$$

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

$$z^l_j(t) = \sum_i w^l_{i,j}\Theta^{l\text{-}1}_i(t) + b^l_j, \tag{2.5}$$

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

$$\Theta^l_i(t) = U(V^l_{\text{mem},i}(t) - V_{\text{th}}), \tag{2.6}$$

where $U(x)$ is a unit step function. Due to the event-driven nature, SNNs offer

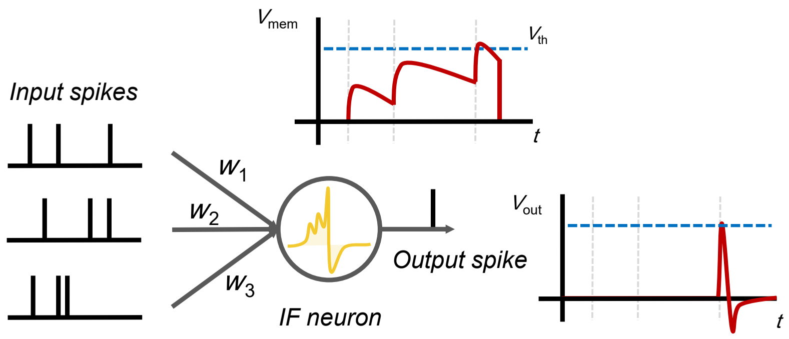


Figure 2.7: Dynamics of integrate-and-fire neurons

energy-efficient operations [73]. However, they are difficult to train which has been one of the major obstacles when deploying SNNs in various applications [94].

The training method of SNNs consists of unsupervised learning with spike-timing-dependent plasticity (STDP) [18] and supervised learning with gradient descent and error back-propagation [52]. 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 [44, 52] 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.

2.3 DNN-to-SNN conversion

As an alternative approach, DNN-to-SNN conversion methods have been recently proposed [83, 19, 13]. These methods are based on the idea of directly importing pre-trained parameters such as synaptic weights from a DNN to an SNN while employing spiking neurons (e.g., integrate-and-fire neurons) as shown in Figure 2.8. [13] proposed a DNN-to-SNN conversion method that removed biases, and used spatial linear sub-sampling instead of max-pooling operation. In subsequent work, [19] proposed

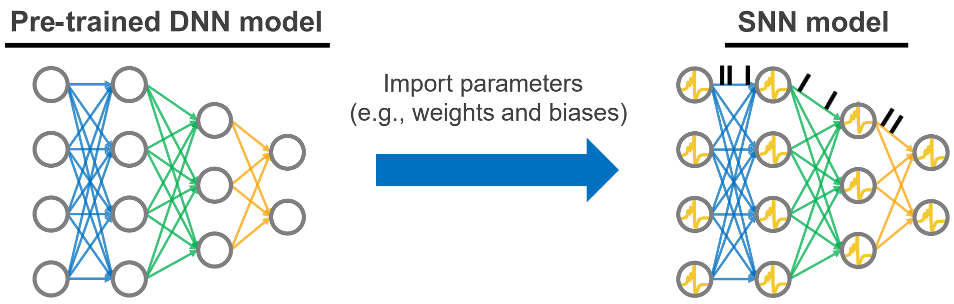


Figure 2.8: Overview of DNN-to-SNN conversion method

data-based normalization (also known as layer-wise weight normalization) to prevent insufficient activation of spiking neurons (i.e., over- and under- activation which is illustrated in Figure 2.9) by normalizing weights in a specific layer using the maximum activation of the corresponding layer. This led to sufficient and balanced activation of neurons, and achieved impressive results on MNIST dataset when compared to existing works. The layer-wise weight normalization can be calculated by

$$w^l \rightarrow w^l \frac{\lambda^{l-1}}{\lambda^l} \text{ and } b^l \rightarrow \frac{b^l}{\lambda^l}, \tag{2.7}$$

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.

[82] proposed robust normalization and demonstrated that biases can be implemented in SNNs with a constant input current. They also proposed an conversion method of batch normalization and implementation of spike max-pooling in SNNs. In [83], the authors aimed to expand the conversion method to deep SNNs (e.g., VGG and residual architectures) and proposed a spike-norm algorithm which balances the threshold voltage of each layer during the conversion process rather than before, which is commonly done in the most previous works.

More recent work [49] applied object detection in deep SNNs for the first time.

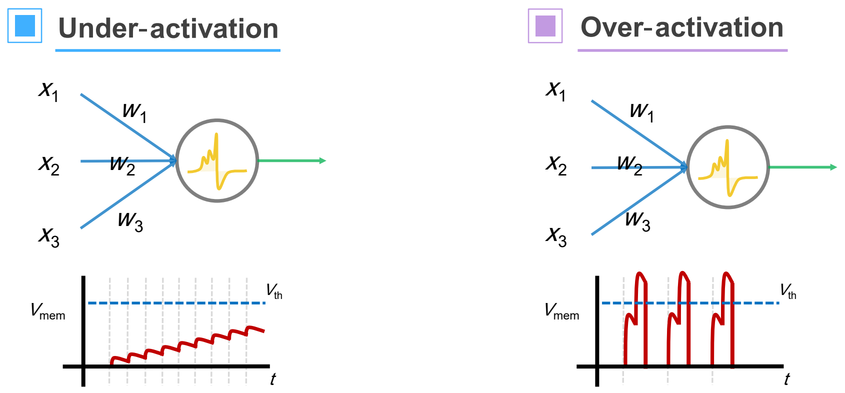


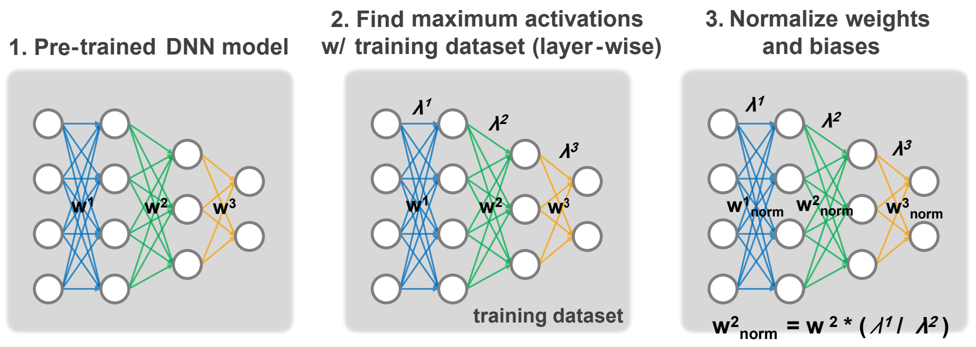
Figure 2.9: Example of under- and over-activation

In doing so, they proposed signed neuron with imbalanced threshold which can efficiently implement leaky-ReLU in SNNs, and developed a more fine-grained weight normalization method, called channel-wise weight normalization to provide fast and accurate information transmission in deep SNNs. The channel-wise weight normalization can be expressed as

$$\tilde{w}_{i,j}^{l} = w_{i,j}^{l}\frac{\lambda_i^{l-1}}{\lambda_j^{l}} \quad \text{and} \quad \tilde{b}_j^{l} = \frac{b_j^{l}}{\lambda_j^{l}}, \tag{2.8}$$

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_j^l in each channel. In the following layer, the normalized activations must be multiplied by λ_i^{l-1} to obtain the original activation prior to the normalization. Note that the threshold voltage is set to $1V$ for all neurons. The weight normalization is equivalent to threshold voltage balancing, carrying out the same effect in terms of providing sufficient and balanced activation of the spiking neurons. In case of the threshold voltage balancing, the threshold voltages will be scaled instead of the weights.

Nonetheless, the conventional conversion methods can be considered as a conservative approach. They strictly regulate firing rate of all neurons between 0 and 1



1. Pre-trained DNN model
2. Find maximum activations w/ training dataset (layer-wise)
3. Normalize weights and biases
w1
w2
w3
λ1
λ2
λ3
training dataset
w1 norm
w2 norm
w3 norm
w2 norm = w 2 * (λ1 / λ2)

Figure 2.10: Step-by-step process of layer-wise weight normalization

by normalizing the weights with a single value (e.g., maximum activation, λ_j^l) and retain a fixed threshold voltage of 1V for all neurons. To improve performance in terms of both latency and accuracy, a several approaches have explored in using different threshold voltages other than 1V, yet the threshold voltages were determined in a very heuristic manner. In their work, the threshold voltages is simply multiplied by a scaling factor (e.g., 0.8, 0.6, 0.4 and 0.2) and seeing if the overall performance improves [31]. Moreover, a single threshold voltage is applied to the entire network. [31] claims that the scaling factor of 0.8 achieves the optimal accuracy-latency trade-off but fails to provide an adequate proof nor address an important perspective in terms of a number of synaptic operations (spikes) which is more likely to increase due to lower threshold voltage.

2.4 Hyper-parameter optimization

Despite the much success of deep neural networks, design space exploration of the neural network has been gaining significant interests with much of the focuses centered on designing network architectures and tuning hyper-parameters. These design choices, typically, are complex and high dimensional, involving a large search space which makes difficult for humans to efficiently navigate all possible choices to pro-

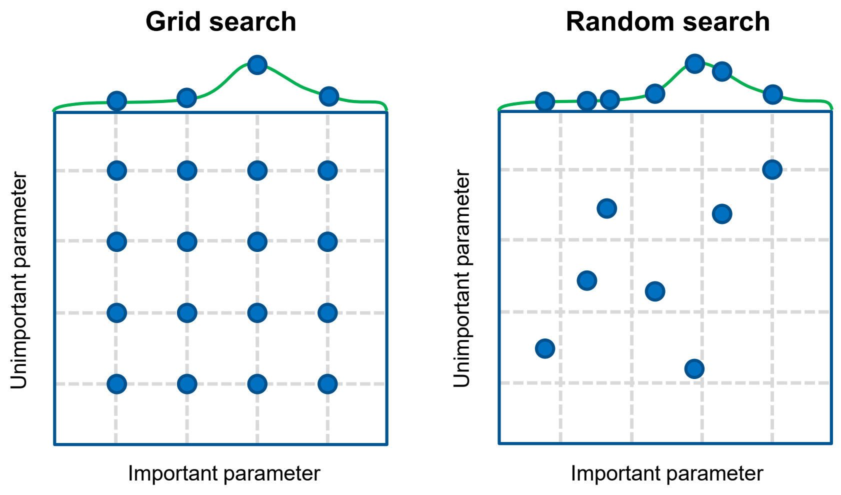


Figure 2.11: Overview of grid and random search

duce the best performance of the model. In the same vein, AutoML is a vital research area in that it automatically finds the optimal neural network design or hyper-parameters, taking the human out of the equation. In general, AutoML can be divided into three major categories; (a) automated feature learning, (b) architecture search and (c) hyper-parameter optimization. In this dissertation, we specifically focus on the hyper-parameter optimization.

The grid search is well-known to be a standard approach to hyper-parameter optimization over the years. However, it became highly ineffective as the search space dimension continues to increases for more sophisticated and bigger neural networks. To overcome such challenges, other approaches have been proposed such as gradient-based optimization [7], random search [1], and Bayesian optimization [85, 41, 2]. [7] optimizes hyper-parameters based on computed gradient of the model selection criterion with respect to the hyper-parameters have been chosen to be optimized. [1] empirically demonstrated that the random search outperforms the grid search while consuming less computational time in several cases. This is because the hyper-parameters have different degrees of importance to the model in that the grid search

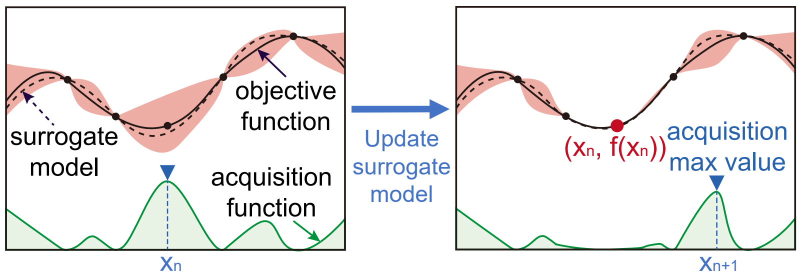


Figure 2.12: Overview of Bayesian optimization

may suffer from lack of converge in possibly important dimensions owing to pre-defined grid size. The random search can eliminate unnecessary search space and enables stochastic search for values located between the two grid points.

As illustrated in Figure 2.12, Bayesian optimization aims to efficiently find optimal input value x of objective function $f(x)$, so that the obtained value x maximizes the output value of $f(x)$ while using as few input candidates as possible. The input value x can be multiple hyper-parameters. Bayesian optimization first constructs a probabilistic model called surrogate model, M, based on input-output evaluation pair $(x_1, f(x_1), \ldots, (x_t, f(x_t))$ obtained from previous iteration. Then an acquisition function, u, suggests the most promising input value $x_{(t+1)}$ for the next iteration based on the probabilistic estimation results obtained thus far. By fitting a surrogate model $\mathcal{M}$ to the samples of an unknown objective function f, the Bayesian optimization procedure iteratively selects the new sample X_{t+1} as follows:

$$\mathbf{X}_{t+1} = \text{argmax}_{\mathbf{X}}\, \mathcal{A}(\mathbf{X}|\mathcal{D}_t), \tag{2.9}$$

where $\mathcal{A}$ is the acquisition function and D_t are the samples drawn from f. Various extended version of Bayesian optimizations have been proposed, namely Spearmint [85], SMAC [41], and TPE [2]. [85] uses a typical Gaussian process model as the surrogate model. [41] constructs a response surface model $P(y|x)$ based on random forests to optimize expected improvement criterion[xx] and also supports categorical parameters and multiple instances. [2] utilizes Adaptive Parzen Estimator to produce a variety of densities over the configuration space and optimizes the expected improvement in tree-structured configuration spaces. In this work, we used Spearmint implementation of Bayesian optimization which showed the best results among the other Bayesian optimization algorithms according to our preliminary experiments.

Chapter 3

Object detection model in deep SNNs

3.1 Introduction

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 [52, 94]. 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 [13, 19, 83]. 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 chapter, 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 weight 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 [80]. 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 weight 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 with 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.

3.2 Channel-wise weight normalization

3.2.1 Conventional weight 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. Layer-wise weight normalization [19] (abbreviated to layer-norm) is one of the most well-known weight 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

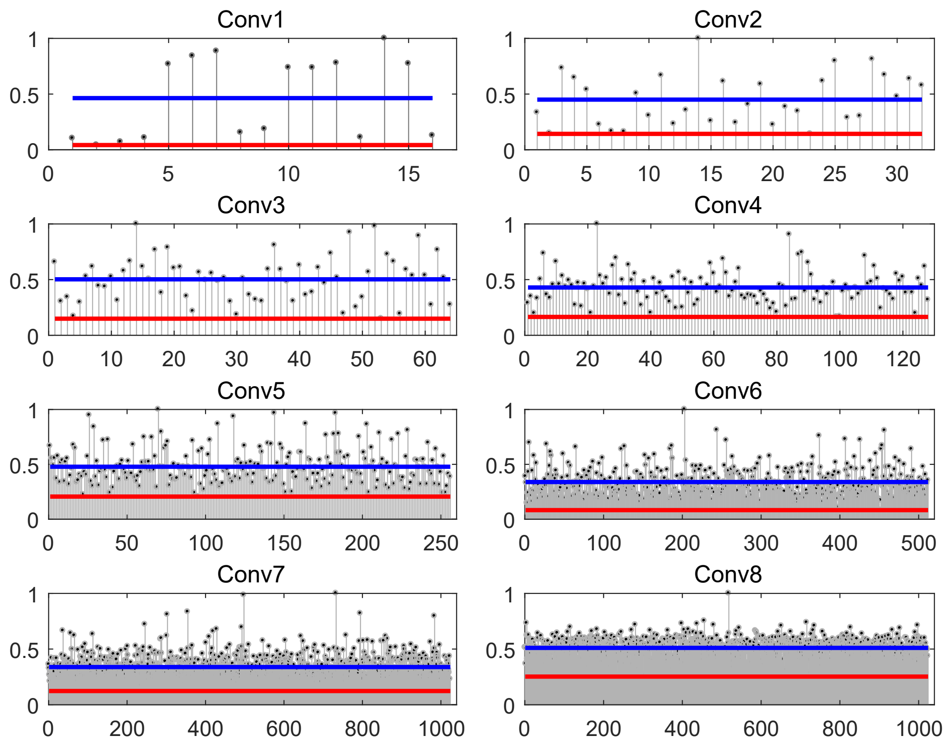


Figure 3.1: Normalized maximum activation via layer-wise weight normalization in each channel for eight convolutional layers in Tiny YOLO. Blue and red lines indicate the average and minimum of the normalized activations, respectively.

normalizing the output activation. Layer-norm can be calculated by

$$\tilde{w}^l = w^l \frac{\lambda^{l-1}}{\lambda^l} \quad \text{and} \quad \tilde{b}^l = \frac{b^l}{\lambda^l}, \tag{3.1}$$

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, [82] 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 weight normalization methods, the model suffers from significant performance degradation.

3.2.2 Analysis of limitations in layer-wise weight normalization

Figure 3.1 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 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 3.1, 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.

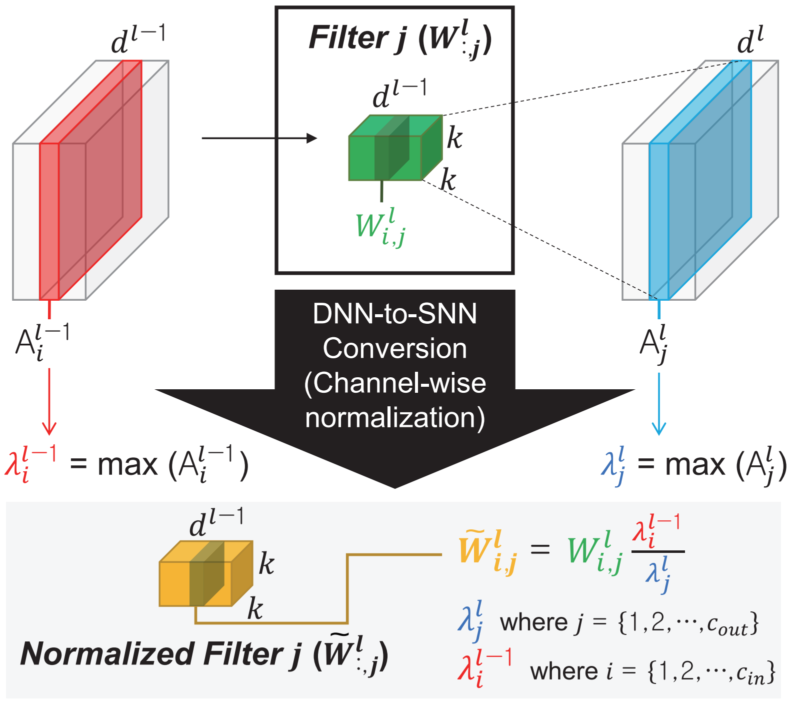

d^{l-1}
Filter j ($W^l_{:,j}$)
d^{l-1}
k
k
$W^l_{i,j}$
d^l
A^{l-1}_i
DNN-to-SNN Conversion (Channel-wise normalization)
A^l_j
λ^{l-1}_i = max (A^{l-1}_i)
λ^l_j = max (A^l_j)
d^{l-1}
k
k
$\tilde{W}^l_{i,j} = W^l_{i,j}\frac{\lambda^{l-1}_i}{\lambda^l_j}$
λ^l_j where $j = \{1,2,\cdots,c_{out}\}$
λ^{l-1}_i where $i = \{1,2,\cdots,c_{in}\}$
Normalized Filter j ($\tilde{W}^l_{:,j}$)

Figure 3.2: Proposed channel-wise weight normalization; A^l_j is jth activation matrix (i.e., feature map) in layer l.

3.2.3 Proposed weight normalization method

We propose a more fine-grained weight normalization method, called channel-wise weight 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

$$\tilde{w}^l_{i,j} = w^l_{i,j}\frac{\lambda^{l-1}_i}{\lambda^l_j} \quad \text{and} \quad \tilde{b}^l_j = \frac{b^l_j}{\lambda^l_j}, \tag{3.2}$$

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_i^{l-1} to obtain the original activation prior to the normalization. The detailed method is depicted in Algorithm 1 and Figure 3.2.

Algorithm 1: Channel-wise normalization

```
   // Calculate maximum activation ($\lambda$) for each channel from training dataset
1  for l in layers do
2      for j in output channels do
3          $\lambda_j^l = \max(A_j^l)$                    // $A$ = activation matrix
   // Apply channel-norm on inference (test dataset)
4  for l in layers do
5      for j in output channels do
6          $\tilde{b}_j^l = b_j^l \,/\, \lambda_j^l$                    // $b$ = bias
7          for i in input channels do
8              if l = first layer then
9                  $\tilde{w}_{i,j}^l = w_{i,j}^l \,/\, \lambda_j^l$              // $w$ = weight
10             else
11                 $\tilde{w}_{i,j}^l = w_{i,j}^l \,/\, \lambda_j^l \, \lambda_i^{l-1}$
```

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.

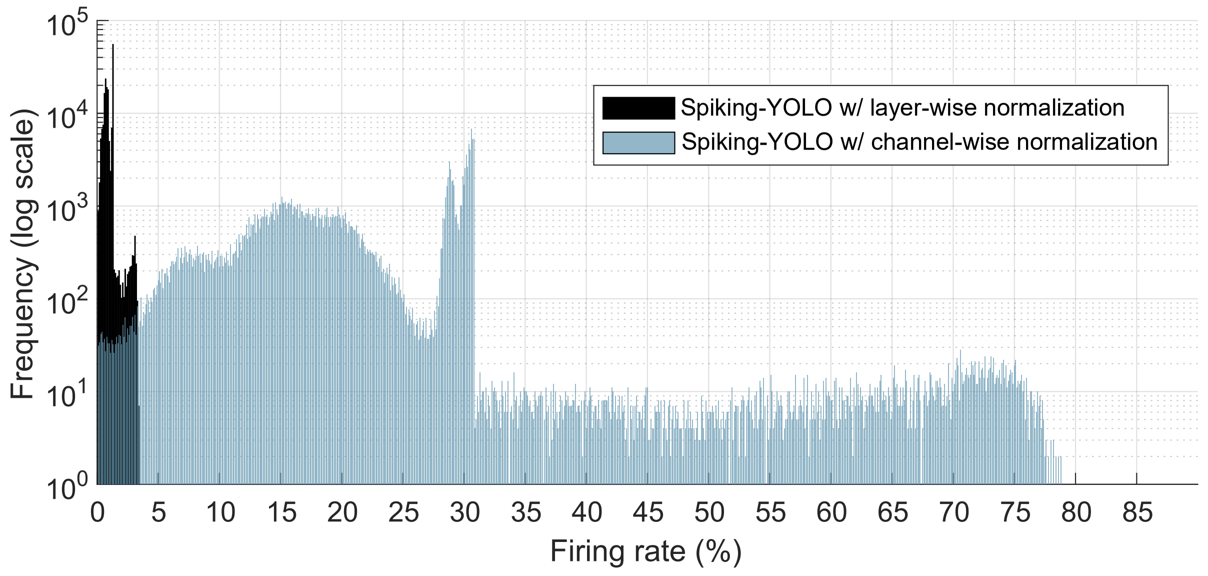


Figure 3.3: Firing rate distribution for layer-norm and channel-norm on channel 2 of Conv1 layer (single image)

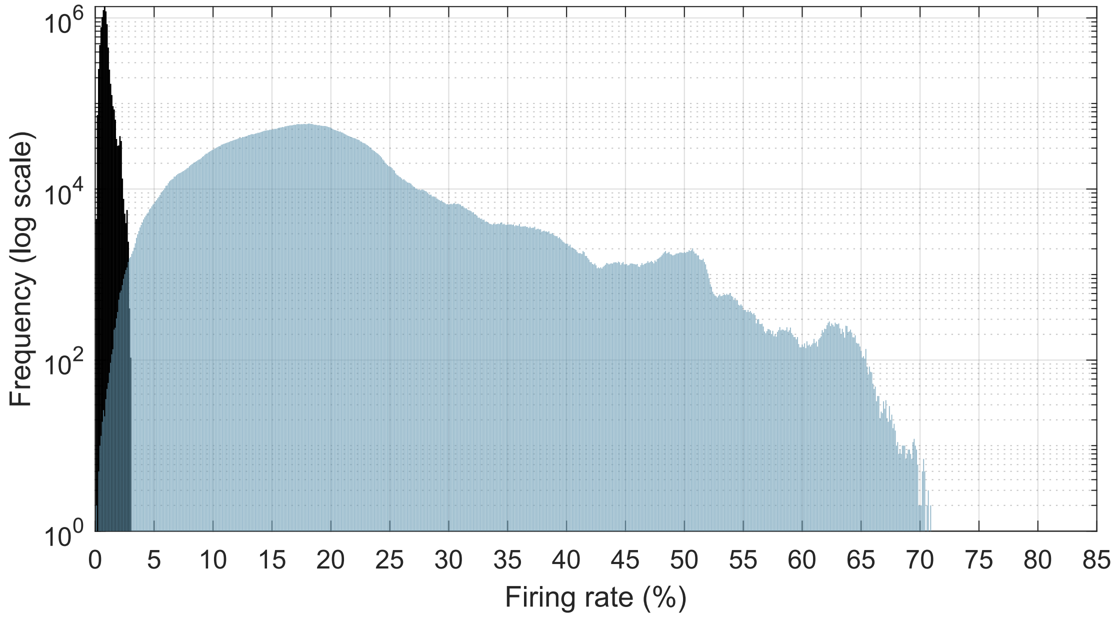


Figure 3.4: Average firing rate distribution for layer-norm and channel-norm on channel 2 of Conv1 layer (5,000 test images)

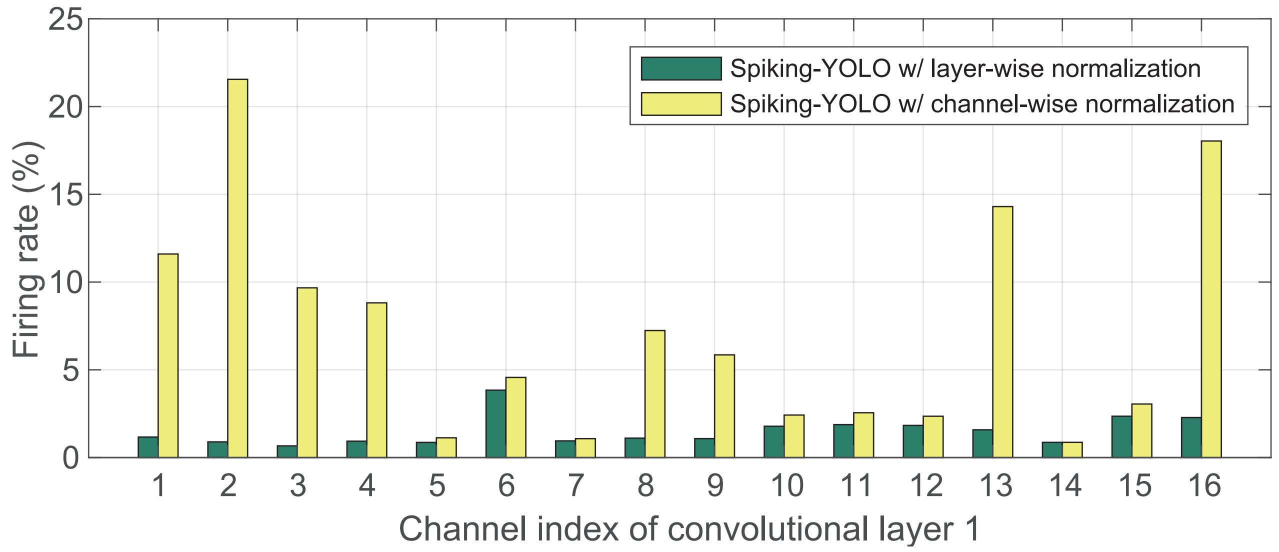


Figure 3.5: Firing rate of 16 channels in Conv1 layer for layer-norm and channel-norm of Tiny YOLO (single image)

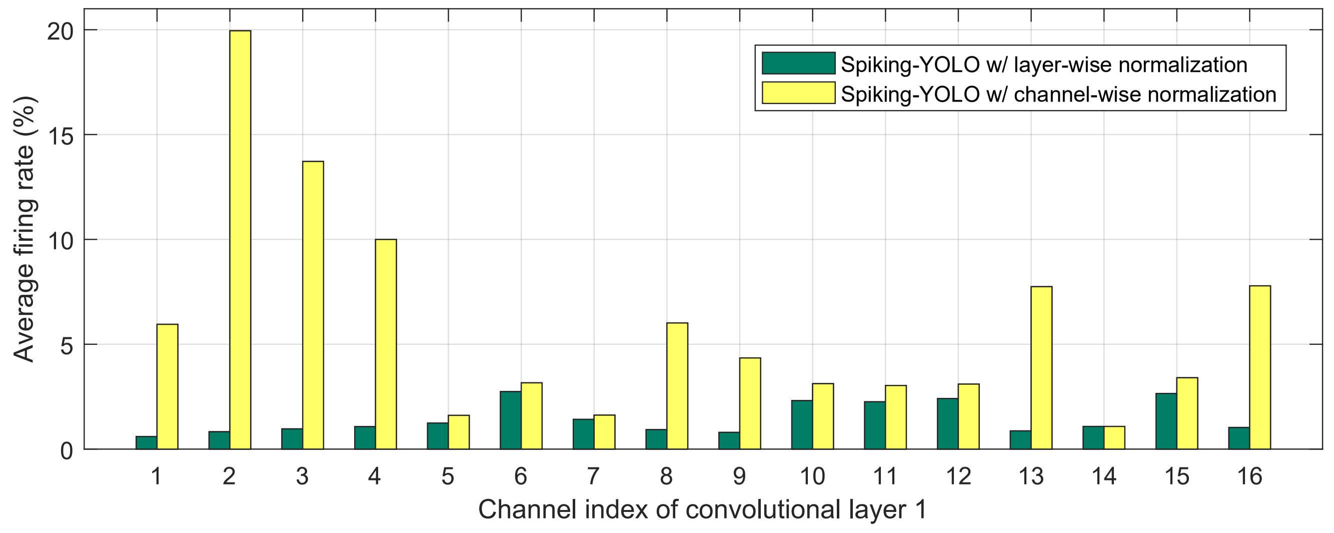


Figure 3.6: Average firing rate of 16 channels in Conv1 layer for layer-norm and channel-norm of Tiny YOLO (5,000 test images)

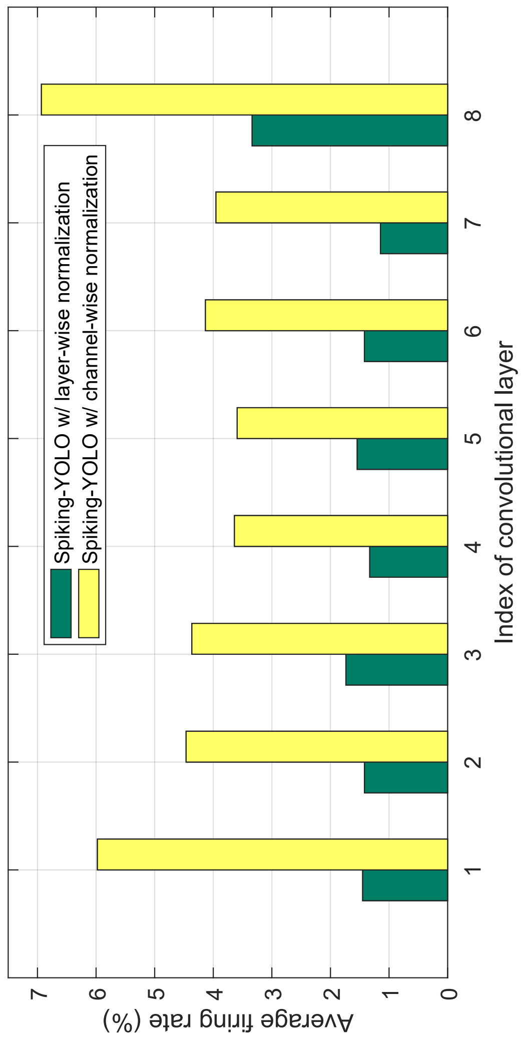
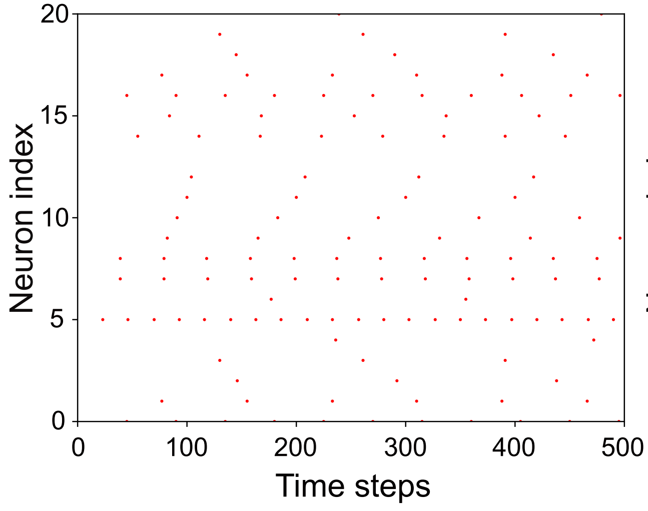


Figure 3.7: Average firing rate of 8 layers for layer-norm and channel-norm of Tiny YOLO (5,000 test images)



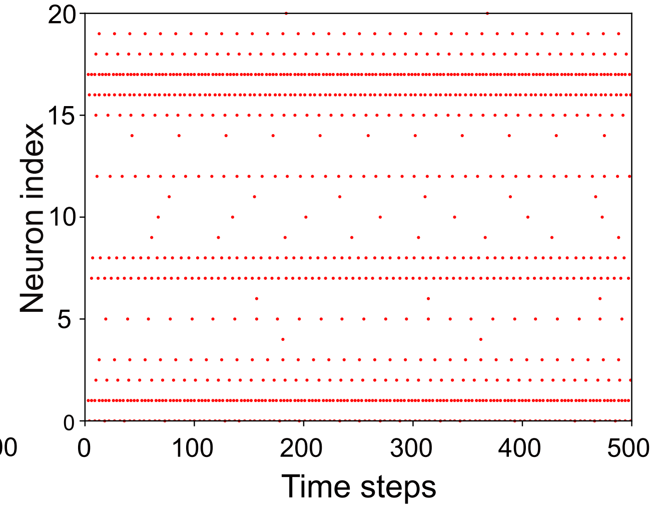


Figure 3.8: Raster plot of 20 sampled neurons' spike activity; layer-norm (left) vs. channel-norm (right)

3.2.4 Analysis of the improved firing rate

In Figure 3.3, 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. Figure 3.3 is measured on a single image and Figure 3.4 is average firing rate distribution on 5,000 test images. In addition, Figure 3.5 presents the firing rate of 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. Figure 3.7 also presents firing rate of each channel in the convolutional layer 1 for 5,000 test images while Figure 3.5 is measured on a single image. Moreover, Figure 3.6 is average firing rate of 8 layers for layer-norm and channel-norm of Tiny YOLO for 5,000 test images. In all layers, channel-norm showed improved firing rate. Lastly, Figure 3.8 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 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

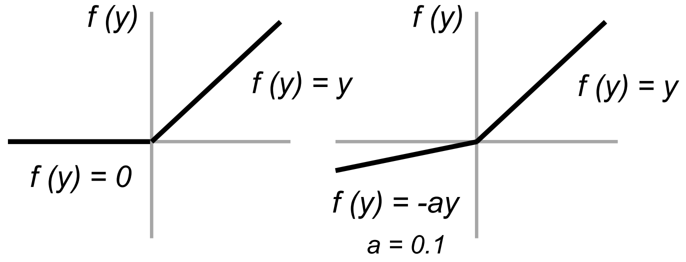


Figure 3.9: ReLU vs. Leaky-ReLU

viable solution for solving more advanced machine learning problems in deep SNNs.

3.3 Signed neuron with imbalanced threshold

3.3.1 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 $f(x) = \alpha x$ [95].

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. The details of distribution of activation on each convolutional layer in Tiny YOLO is depicted in Figure 3.10. To extend the activation function bound to the negative region in SNNs, [82] added a second V_{th} term (-1). Their method successfully converted BinaryNet [40] to SNNs, where the activations were constrained to $+1$ or -1 on CIFAR-10.

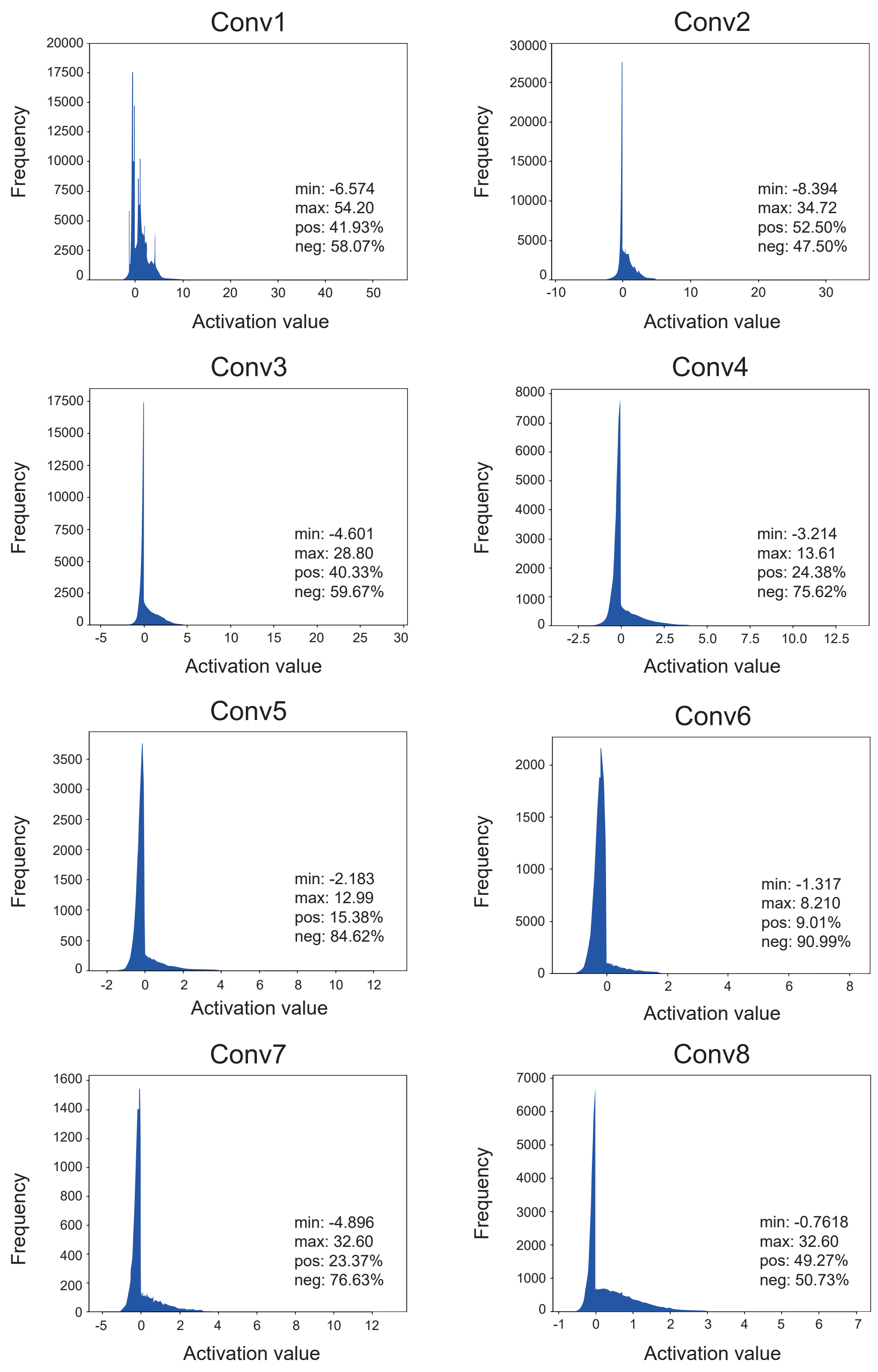


Figure 3.10: Distribution of activation on each convolutional layer

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) as shown in Figure 3.12. 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. Figure 3.10 represents distribution of activation values in each convolutional layer prior to normalization. The ratio of positive and negative activation values for each convolutional layer are also shown in Figure 3.10.

3.3.2 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, $V_{\mathrm{th,neg}}$. The second threshold voltage $V_{\mathrm{th,neg}}$ is equal to the V_{th} divided by the negative of the slope, $-\alpha$, and $V_{\mathrm{th,pos}}$ 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

$$\mathrm{fire}(V_{\mathrm{mem}}) = \begin{cases} 1 & \text{if } V_{\mathrm{mem}} \geq V_{\mathrm{th,pos}}(V_{\mathrm{th}}) \\ -1 & \text{if } V_{\mathrm{mem}} \leq V_{\mathrm{th,neg}}(-\frac{1}{\alpha}V_{\mathrm{th}}) \\ 0 & \text{otherwise, no firing.} \end{cases} \tag{3.3}$$

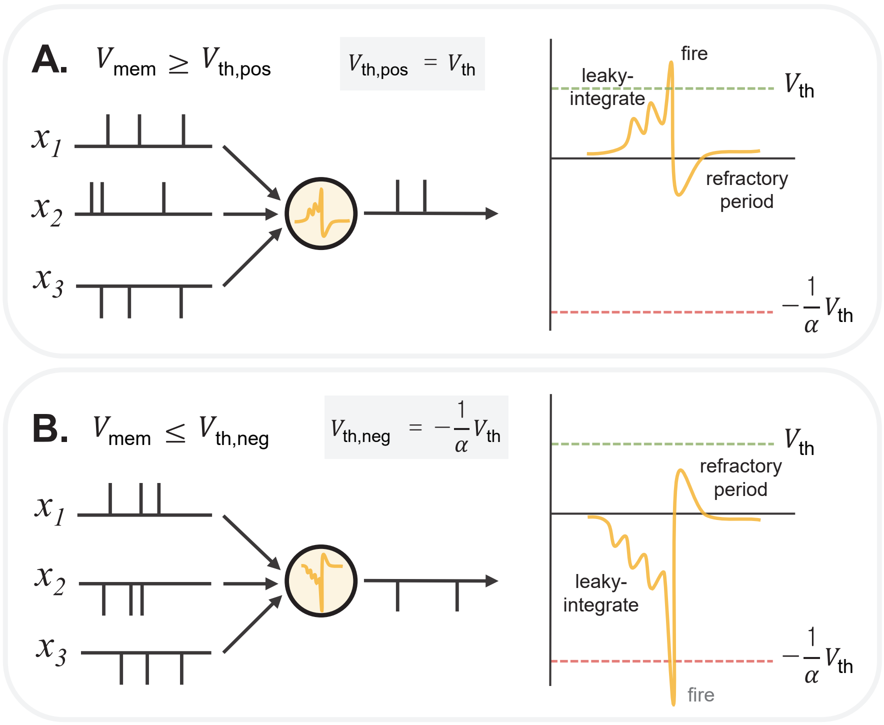

A. $V_{mem} \geq V_{th,pos}$
$V_{th,pos} = V_{th}$
x_1
x_2
x_3
fire
leaky-integrate
refractory period
V_{th}
$-\frac{1}{\alpha}V_{th}$
B. $V_{mem} \leq V_{th,neg}$
$V_{th,neg} = -\frac{1}{\alpha}V_{th}$
x_1
x_2
x_3
refractory period
leaky-integrate
fire
V_{th}
$-\frac{1}{\alpha}V_{th}$

Figure 3.11: Overview of proposed signed neuron featuring imbalanced threshold; two possible cases for a spiking neuron

As shown in Figure 3.11, 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 [16, 90]. 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.

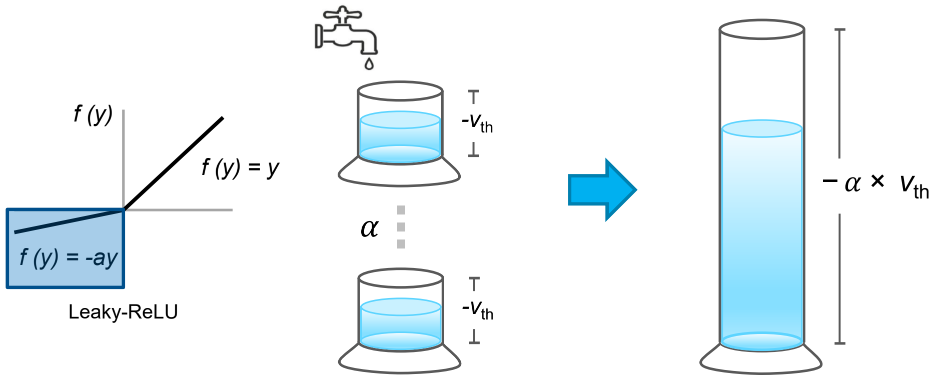


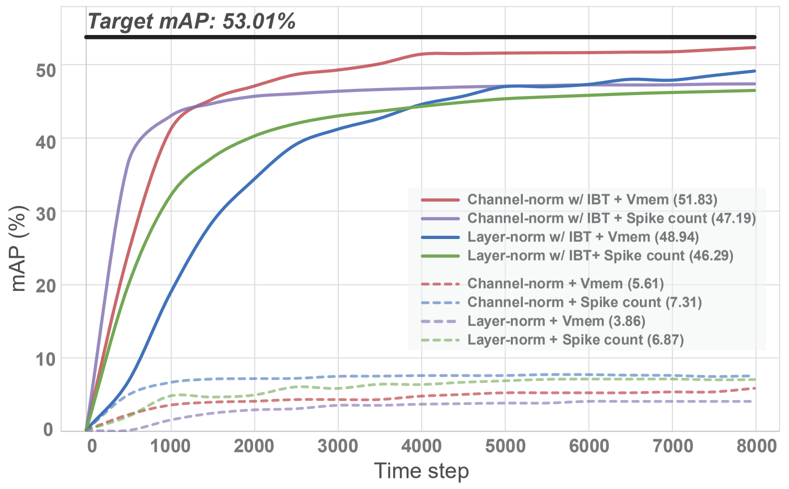
Figure 3.12: Overview of proposed signed neuron featuring imbalanced threshold

3.4 Evaluation

As the first step towards object detection in deep SNNs, we used a real-time object detection model, Tiny YOLO. Note that 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, which is a simpler but efficient version of YOLO. We implemented max-pooling and batch-normalization in SNNs according to [82]. 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.

3.4.1 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 3.14, 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 out-



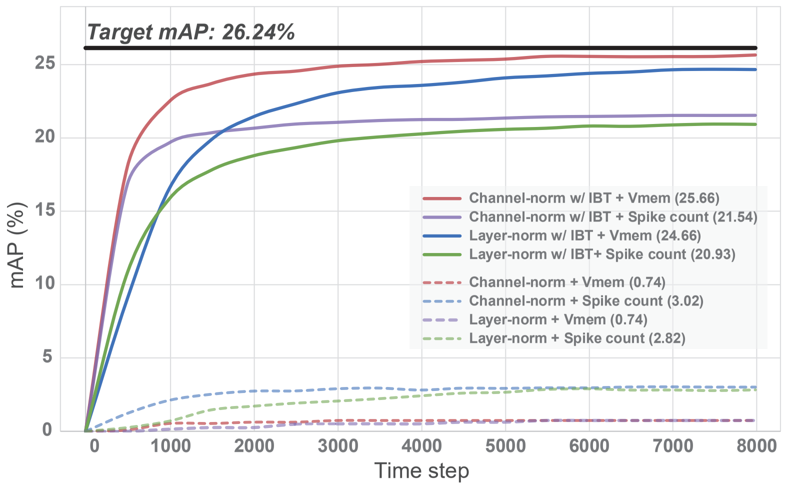


Figure 3.13: Experimental results of Spiking-YOLO on PASCAL VOC (left) and MS COCO (right) for various configurations (weight normalization methods + signed neuron w/ IBT + decoding scheme); maximum mAP is in parentheses.

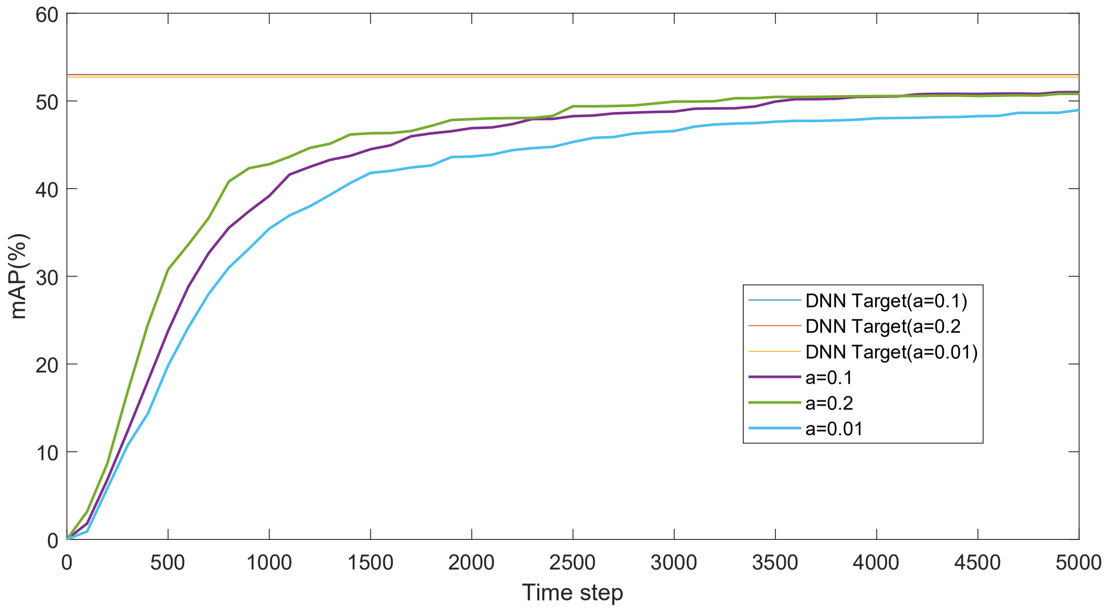


Figure 3.14: Object detection accuracy as time step increases for various slope (alpha) values

performs 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 3.1 for more detailed results.

Table 3.1: 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 3.14 verifies this assertion. The V_{mem}-based output decoding scheme outperforms the spike-count-based scheme and converges faster in channel-norm.

Figure 3.15 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 2,000 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; 5,000 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 3.15. The proposed channel-norm shows a clear advantage in detecting multiple and microscopic objects accurately in a shorter period of time. Please refer to Figures 3.16 - 3.23 for more object detection results.

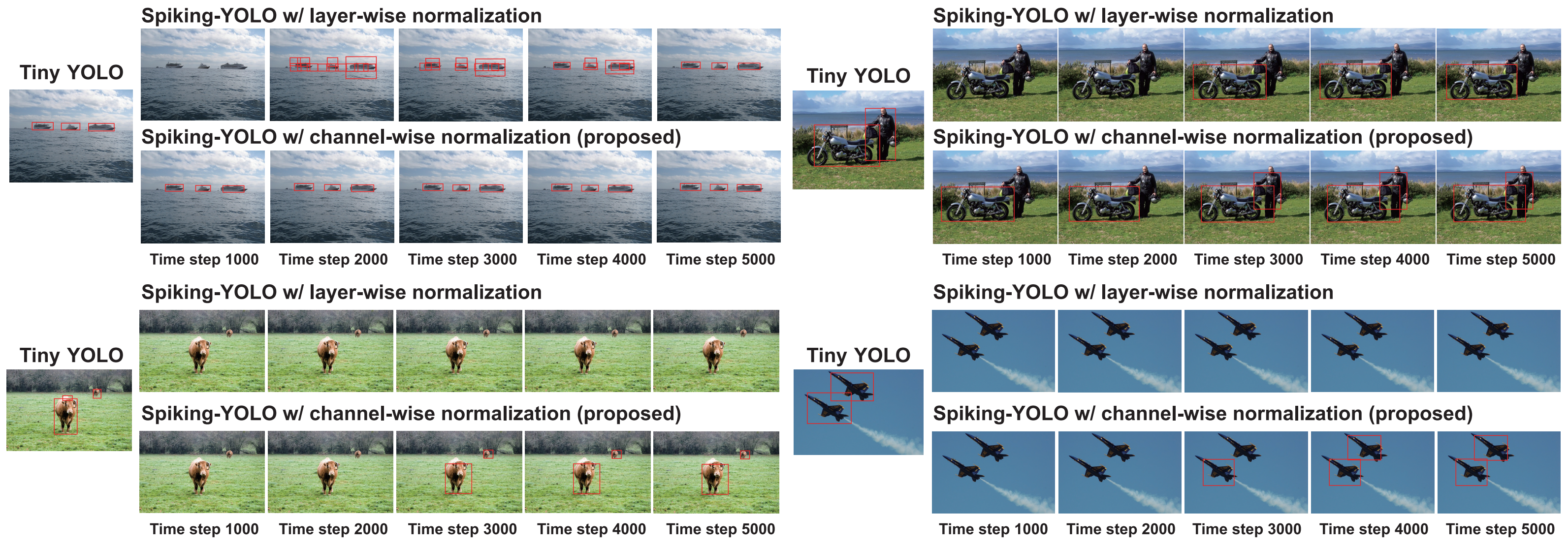

Tiny YOLO
Spiking-YOLO w/ layer-wise normalization
Spiking-YOLO w/ channel-wise normalization (proposed)
Time step 1000
Time step 2000
Time step 3000
Time step 4000
Time step 5000

Figure 3.15: Object detection results (Tiny YOLO vs. Spiking-YOLO with layer-norm vs. Spiking-YOLO with channel-norm)

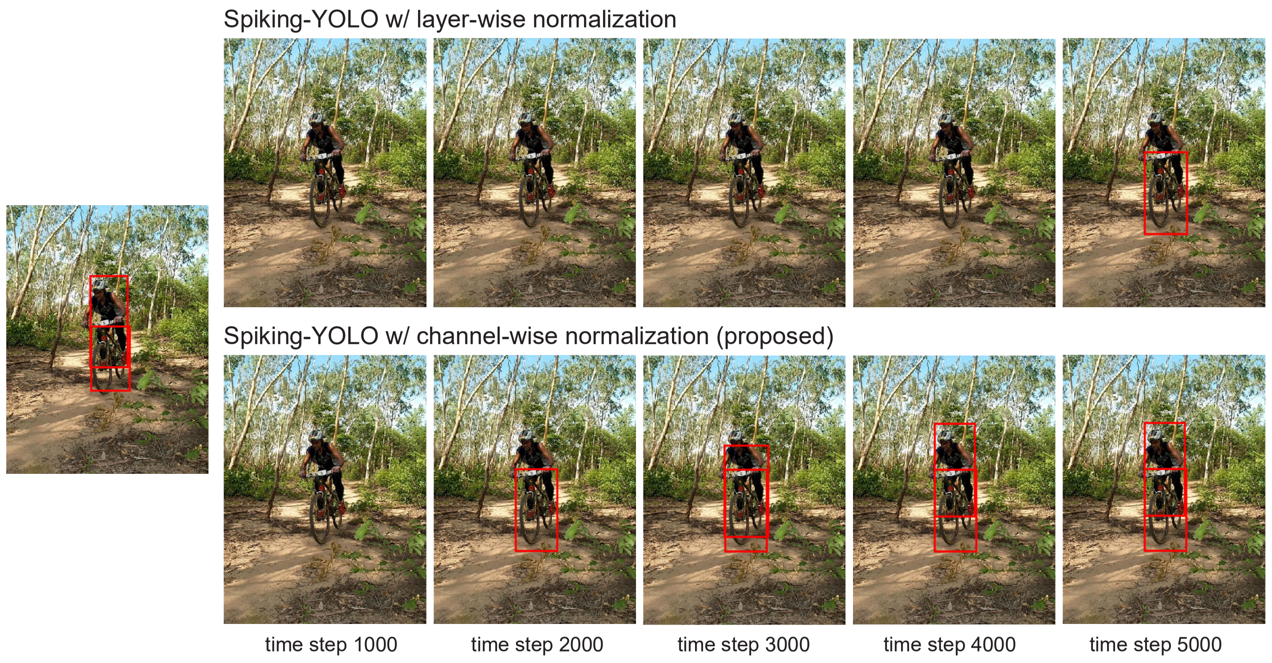

Spiking-YOLO w/ layer-wise normalization
Spiking-YOLO w/ channel-wise normalization (proposed)
time step 1000
time step 2000
time step 3000
time step 4000
time step 5000

Figure 3.16: More object detection results (Tiny YOLO vs. Spiking-YOLO with layer-norm vs. Spiking-YOLO with channel-norm)

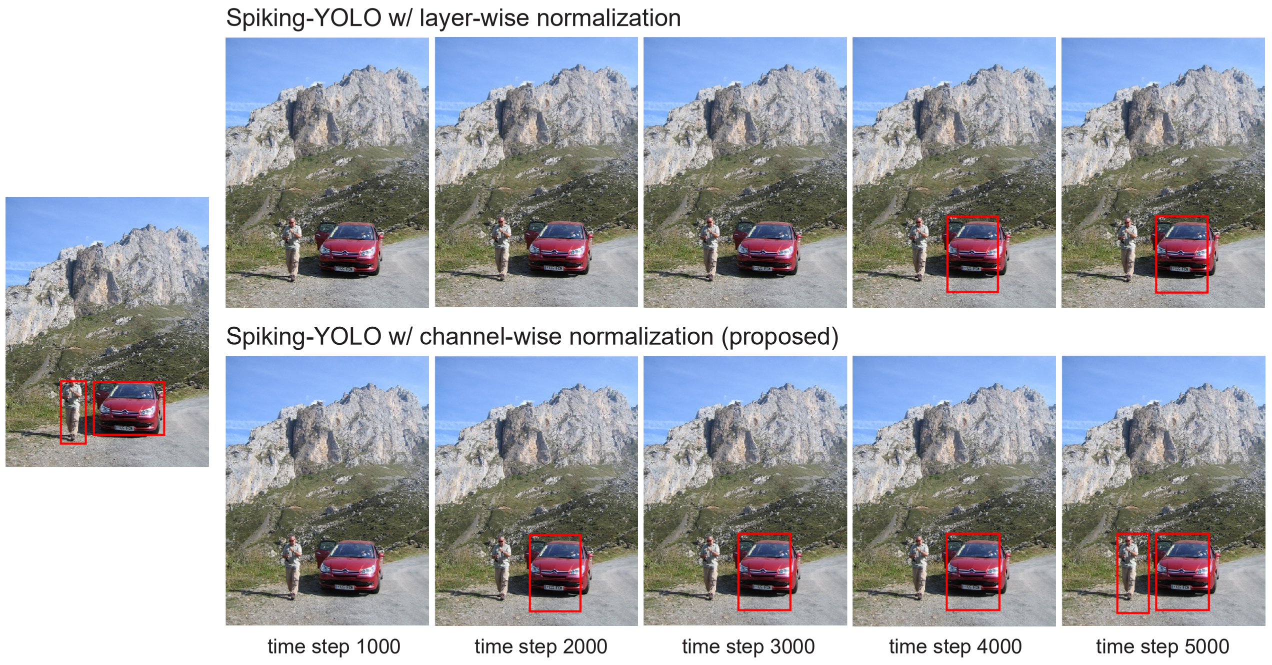
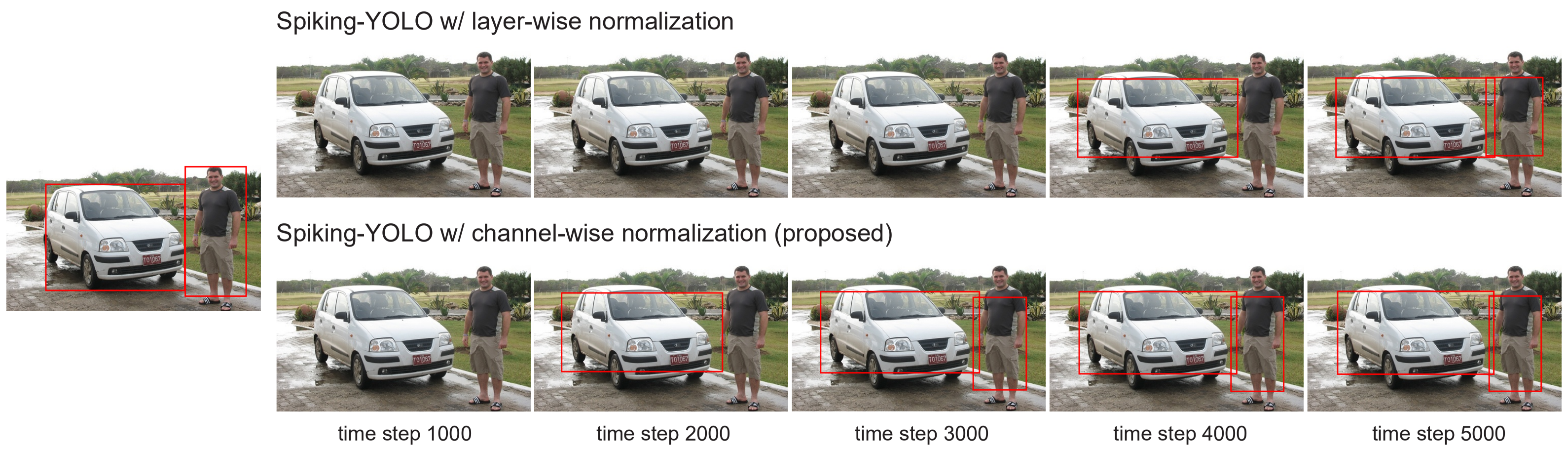

Spiking-YOLO w/ layer-wise normalization
Spiking-YOLO w/ channel-wise normalization (proposed)
time step 1000
time step 2000
time step 3000
time step 4000
time step 5000

Figure 3.17: More object detection results (Tiny YOLO vs. Spiking-YOLO with layer-norm vs. Spiking-YOLO with channel-norm)



Spiking-YOLO w/ layer-wise normalization
Spiking-YOLO w/ channel-wise normalization (proposed)
time step 1000
time step 2000
time step 3000
time step 4000
time step 5000

Figure 3.18: More object detection results (Tiny YOLO vs. Spiking-YOLO with layer-norm vs. Spiking-YOLO with channel-norm)

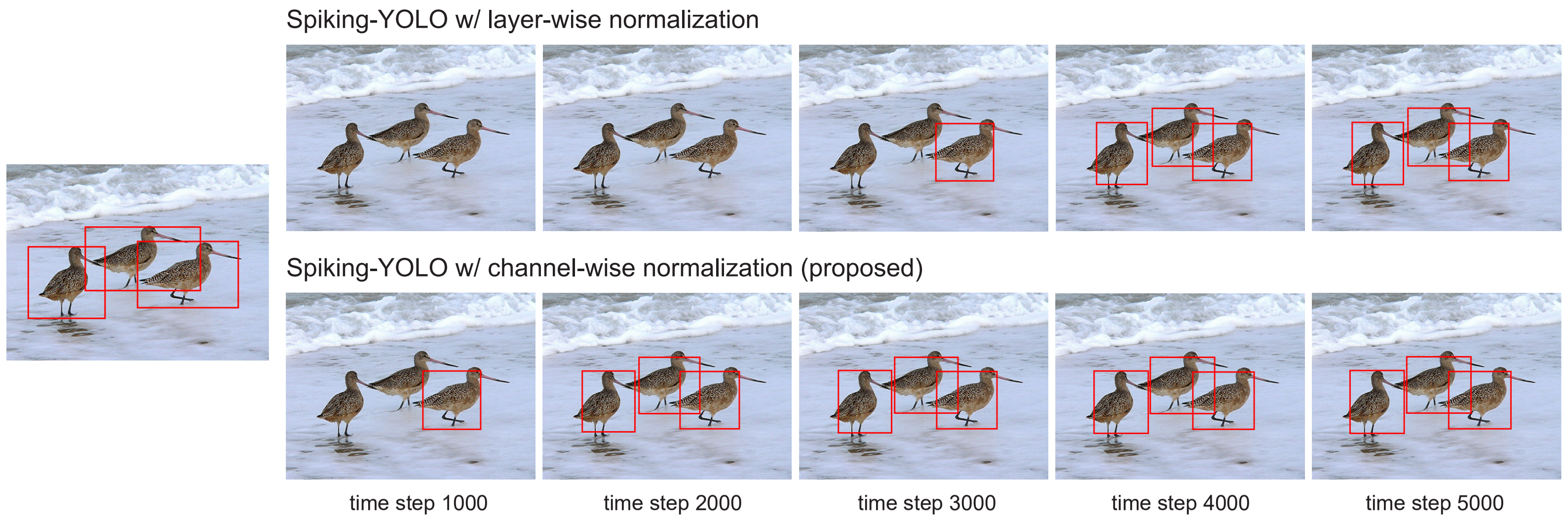
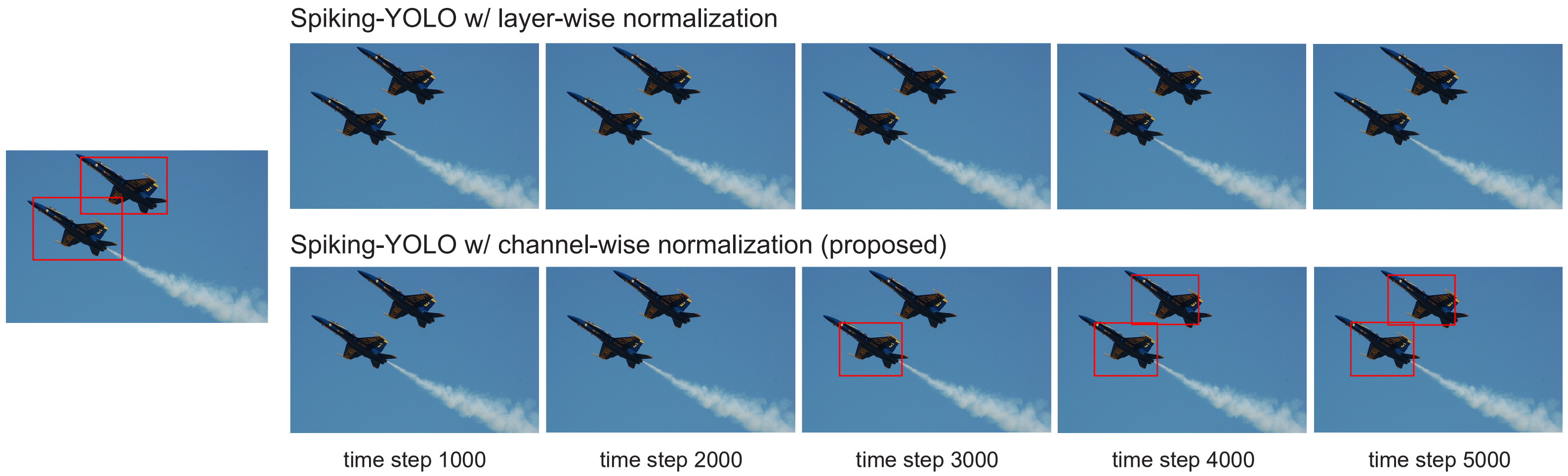

Spiking-YOLO w/ layer-wise normalization
Spiking-YOLO w/ channel-wise normalization (proposed)
time step 1000
time step 2000
time step 3000
time step 4000
time step 5000

Figure 3.19: More object detection results (Tiny YOLO vs. Spiking-YOLO with layer-norm vs. Spiking-YOLO with channel-norm)



Spiking-YOLO w/ layer-wise normalization
Spiking-YOLO w/ channel-wise normalization (proposed)
time step 1000
time step 2000
time step 3000
time step 4000
time step 5000

Figure 3.20: More object detection results (Tiny YOLO vs. Spiking-YOLO with layer-norm vs. Spiking-YOLO with channel-norm)

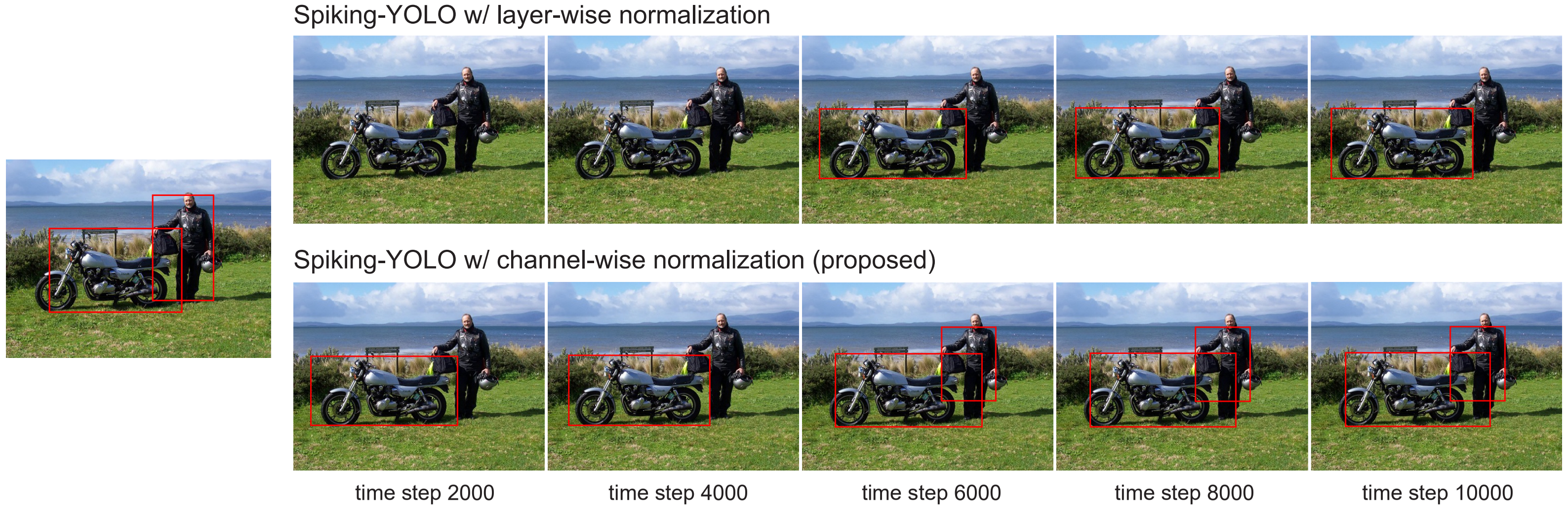

Spiking-YOLO w/ layer-wise normalization
Spiking-YOLO w/ channel-wise normalization (proposed)
time step 2000
time step 4000
time step 6000
time step 8000
time step 10000

Figure 3.21: More object detection results (Tiny YOLO vs. Spiking-YOLO with layer-norm vs. Spiking-YOLO with channel-norm)

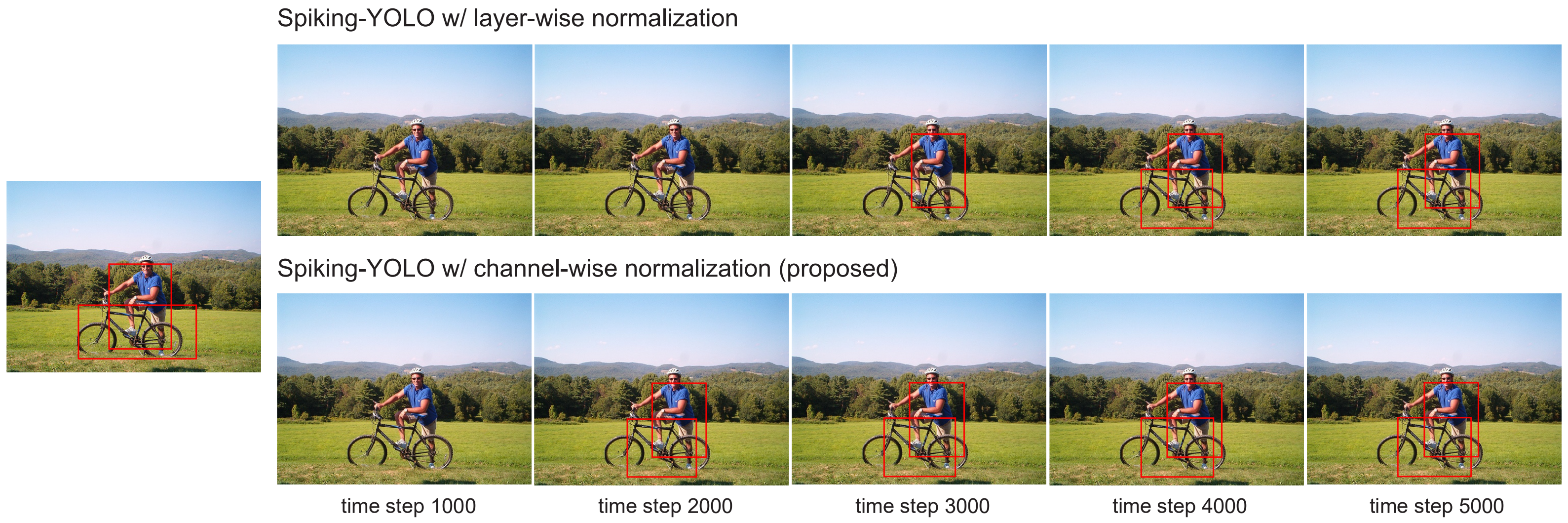

Spiking-YOLO w/ layer-wise normalization
Spiking-YOLO w/ channel-wise normalization (proposed)
time step 1000
time step 2000
time step 3000
time step 4000
time step 5000

Figure 3.22: More object detection results (Tiny YOLO vs. Spiking-YOLO with layer-norm vs. Spiking-YOLO with channel-norm)

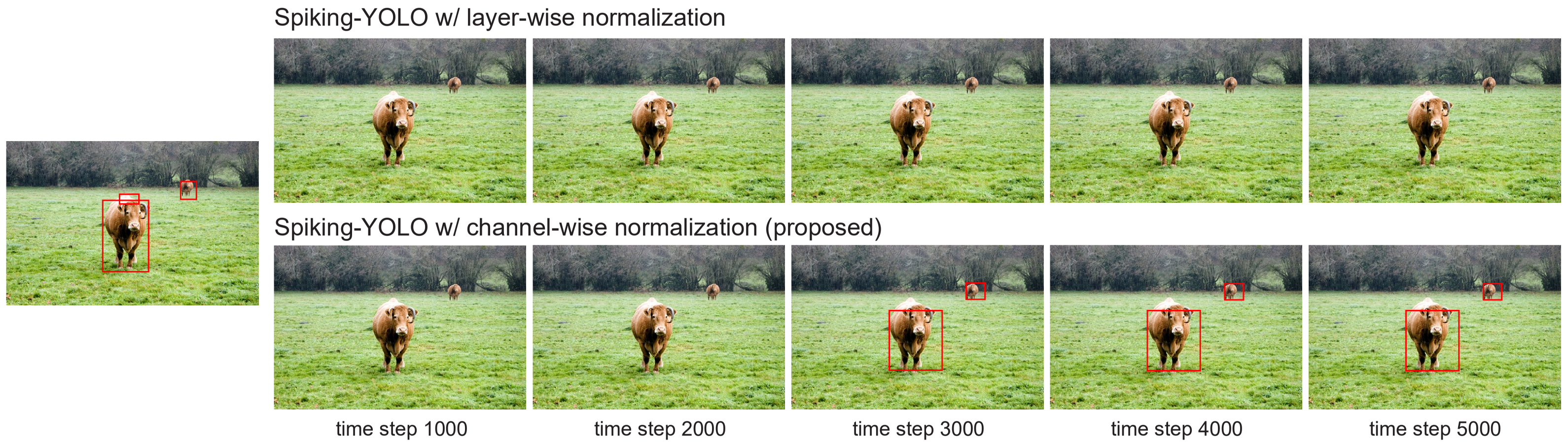

Spiking-YOLO w/ layer-wise normalization
Spiking-YOLO w/ channel-wise normalization (proposed)
time step 1000
time step 2000
time step 3000
time step 4000
time step 5000

Figure 3.23: More object detection results (Tiny YOLO vs. Spiking-YOLO with layer-norm vs. Spiking-YOLO with channel-norm)

3.4.2 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 and 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 [38], 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 integer (INT) MAC operation consumes 3.2 pJ (0.1 + 3.1 pJ) and 0.1 pJ for an AC operation.

Table 3.2: Energy table in 45nm CMOS process [Horowitz, 2014]

Operation	Energy (J)
32 bit int ADD	0.1
32 bit float ADD	0.9
32 bit int MULT	3.1
32 bit float MULT	3.7

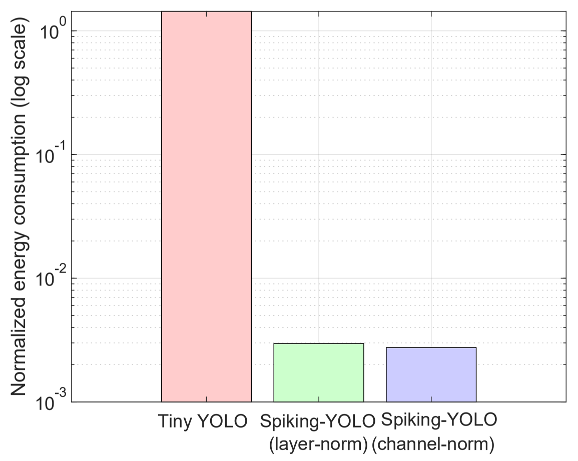
Based on these measures, we calculated the energy consumption of Tiny YOLO and Spiking-YOLO by multiplying FLOPs (floating-point operations) and the energy consumption of MAC and AC operations calculated, as shown below. FLOPs of Tiny YOLO are reported on [77], and that for Spiking-YOLO are calculated during our simulation. Figure 3.24 shows that regardless of the weight normalization methods, Spiking-YOLO demonstrates exceptional energy efficiency, over 2,000 times better than Tiny-YOLO for 32-bit FL and INT operations.

Secondly, SNNs on neuromorphic chips offer excellent energy efficiency, which is an important and desirable aspect of neural networks [73]. 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 [69], and GFLOPS/W for TrueNorth is reported on [63]. We define one time step as equal to 1ms (1 kHz synchronization signal in [63]).

Based on our calculations shown in Table 3.3, 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 3.3: Energy comparison of Tiny YOLO (GPUs) 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**



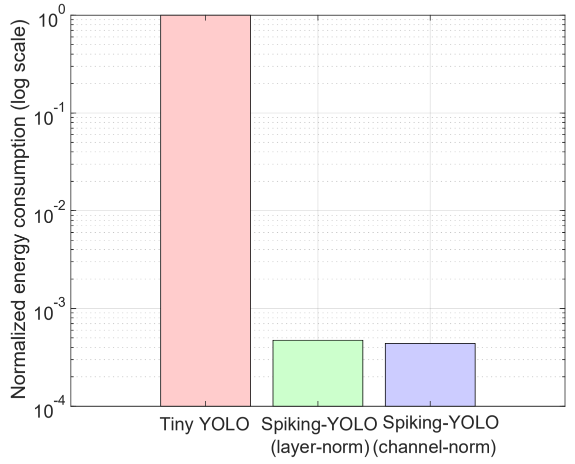


Figure 3.24: Energy comparison of Tiny YOLO and Spiking-YOLO for MAC and AC operations; 32-bit FL (left) and 32-bit INT (right)

Chapter 4

Improving performance and efficiency of deep SNNs

4.1 Introduction

An object detection model in SNNs, called Spiking-YOLO, has been recently proposed to substantially improve efficiency of object detection [49]. However, one of its drawbacks is that it requires a significant amount of time steps (latency) and spikes (synaptic operations) to provide high-numerical precision and high object detection accuracy, which can directly translate into higher energy and power consumption. For instance, SNNs executing object detection would require over 2x the latency and 100x the number of spikes, when compared to image classification [47, 83, 49]. Without addressing these problems properly, the potential impact of SNNs can be diminished, particularly in terms of energy efficiency. Consequently, the significant benefits of SNNs over DNNs may no longer be the same. For instance, in image classification, SNNs (e.g., VGG and ResNet architecture) yield latency of between 2,000 and 2,500 time steps and generate up to 86.5M spikes on CIFAR-100 [47, 83]. As for a more complex object detection, SNNs (e.g., Tiny YOLO) require up to 5,000 time steps and produce 38.2B spikes on PASCAL VOC [49]. That is, SNNs executing object detection would require over 2x the latency and 100x the number of spikes, when compared to image classification.

In recent years, various approaches have been proposed to improve performance of SNNs in terms of both accuracy and efficiency, but these have been limited to image classification. These approaches include weight and threshold balancing methods [19, 73, 83] and neural coding schemes [71, 97, 72]. In most of the existing approaches, the firing rate of neurons is regulated by a single uniform value in each channel (e.g., maximum activation value) while having the same threshold voltage. In neuroscience literature, neurons in different regions of brain represent distinct dynamics and process information differently than other regions [10, 87, 100]. The threshold voltage of neurons is also known to have a broad range rather than a single value [5]. A few studies have used various threshold voltages, but these values were determined heuristically, leaving much room for optimization [32, 31].

Inspired from these observations, we propose a threshold voltage balancing method that improves object detection performance in SNNs in terms of three important aspects: accuracy, latency, and number of synaptic operations. The proposed threshold voltage balancing method can automatically scale the threshold voltages in each hidden layer to optimal values by employing Bayesian optimization. We specifically design Bayesian optimization such that it considers distinctive characteristics of SNNs, namely, latency and number of synaptic operations. Furthermore, we introduce the concept of two-phase threshold voltages. The threshold voltages in each phase have two different objectives; phase-1 for fast object detection and phase-2 for accurate object detection. By utilizing phase-1 threshold voltages for the early part of time steps in SNNs, then swapping in phase-2 threshold voltages later, SNNs achieve not only fast convergence speed but also state-of-the-art object detection accuracy with significantly less number of synaptic operations. The key contributions of this work can be summarized as follows:

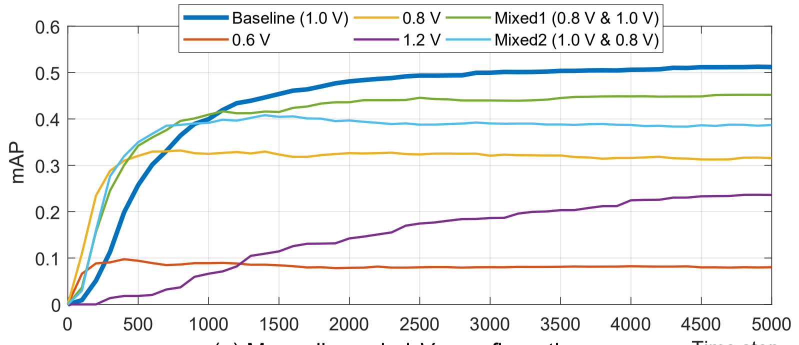
- **Threshold voltage balancing through Bayesian optimization** We present a threshold balancing method using Bayesian optimization that finds optimal threshold voltages to improve performance of SNNs in terms of accuracy, latency and number of synaptic operations.
- **Bayesian optimization specifically designed for SNNs** We design Bayesian optimization to consider two important characteristics of SNNs in addition to object detection accuracy: latency, and number of synaptic operations. Moreover, we employ proxy evaluation of SNNs to reduce large computational overhead of the optimization process.
- **Two-phase threshold voltages for faster and more accurate object detection in SNNs** We introduce the concept of two-phase threshold voltages that can provide low latency while achieving state-of-the-art object detection accuracy. By substantially reducing latency and number of synaptic operations, the proposed methods can provide highly energy-efficient object detection in SNNs.

4.2 Threshold voltage balancing through Bayesian optimization

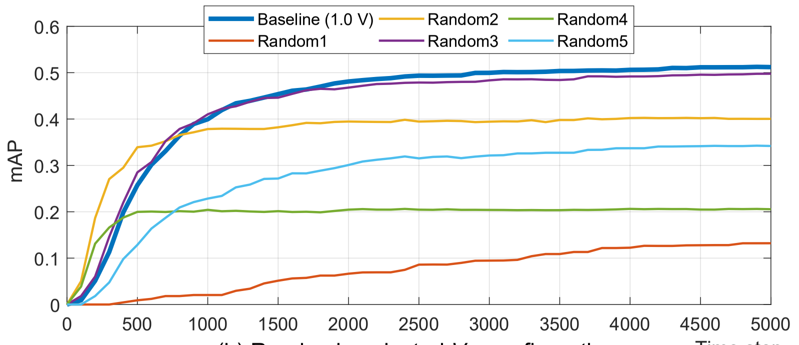
4.2.1 Motivation

The conventional conversion methods can be considered as a conservative approach and a sub-optimal solution. They strictly regulate firing rate between 0 and 1 by normalizing the weights with a single value, and retain a fixed threshold voltage for all spiking neurons. Moreover, Object detection in SNNs requires a significant amount of time steps (latency) and spikes (synaptic operations). For instance, in image classification, SNNs (e.g., VGG and ResNet architecture) yield latency of between 2,000 and 2,500 time steps and generate up to 86.5M spikes on CIFAR-100 [47, 83]. As for a more complex object detection, SNNs (e.g., Tiny YOLO) require up to 5,000 time steps and produce 38.2B spikes on PASCAL VOC [49]. That is, SNNs executing object detection would require over 2x the latency and 100x the number of spikes, when compared to image classification. Without addressing these issues properly, significant benefits of SNNs over DNNs may no longer be the same and could pose challenges in wide-spread use of SNNs for variety applications of DNNs, particularly in object detection.

Despite channel-wise weight normalization [49] is applied to eliminate extremely small activations, some neurons still remain to have very small activation values, thereby producing extremely low firing rate. This becomes a more critical issue in two circumstances; a more challenging task, namely object detection which requires high-numerical precision, and deeper SNNs which require more time steps for information to reach the output layer without significant information loss. Most of the previous works have proposed to improve the efficiency of SNNs in image classification, and these can be considered sub-optimal solutions. This is because they strictly regulate the firing rate of neurons between 0 and 1 by normalizing the weights with a single value in each channel (e.g., channels in convolutional layer 1 have 173,056 spiking neurons) and retain the same threshold voltage, leaving much room for optimization. Several approaches have explored various threshold voltages to improve the efficiency, but the threshold voltages were determined in a heuristic manner [32, 31]. The threshold voltages were simply multiplied by a scaling factor of between 0.2 and



(a) Manually-scaled V_{th} configurations



(b) Randomly-selected V_{th} configurations

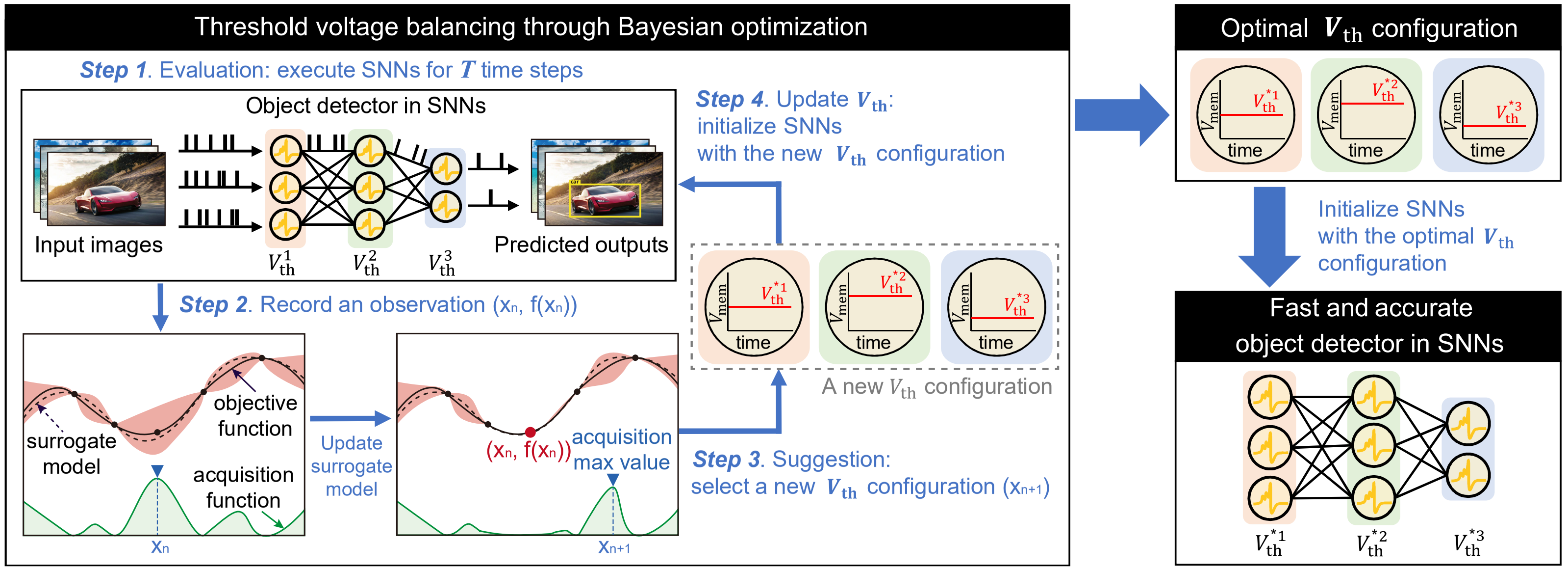
Figure 4.1: Comparison of object detection accuracy (mAP %) as the time step increases for various threshold voltage configurations

0.8 and checked in terms of which one performs the best accuracy-latency trade-off. More importantly, they neglected the total number of spikes, which is more likely to increase with scaled threshold voltages (i.e., scaling factor of less than 1.0).

To validate their claims in object detection, we applied the scaled threshold voltages suggested by [32, 31] and also explored various threshold voltages in each hidden layer. Figure 4.1 shows the object detection accuracy of SNNs (consisting of eight total convolutional layers) as the time step increases on various threshold voltages applied in each hidden layer. The threshold voltages are either manually scaled (Figure 4.1 (a)) or randomly selected (Figure 4.1 (b)). For instance, Mixed2 (1.0 V and 0.8 V) in Figure 4.1 (a) indicates that the threshold voltages of the first four layers and the last four layers in the hidden layers are set to 1.0 V and 0.8 V, respectively. Note that the baseline threshold voltages are set to 1.0 V.

As shown in Figure 4.1, SNNs show significant performance discrepancy over various threshold voltage configurations. Several threshold voltage configurations (0.8 V, Mixed1, Mixed2, Random2 and Random3) achieve better object detection accuracy at an earlier time step than the baseline. Particularly, Mixed2 shows higher object detection accuracy than the baseline until approximately 1,000 time steps, then the object detection accuracy saturates and achieves object detection accuracy below the baseline at 5,000 time steps. Moreover, the accuracy curve of Random3 is similar to that of the baseline, and Random3 actually performs slightly better than the baseline until 1,000 time steps.

From these observations, the threshold voltage scaling method in previous works on image classification fails to achieve optimal accuracy-latency trade-off when applied in object detection. The results shown in Figure 4.1 are encouraging in terms of convergence speed, but the overall object detection accuracy is not satisfactory. Consequently, there is a need for a new threshold voltage balancing method that



Threshold voltage balancing through Bayesian optimization
Step 1. Evaluation: execute SNNs for T time steps
Object detector in SNNs
Input images
V_{th}^1
V_{th}^2
V_{th}^3
Predicted outputs
Step 2. Record an observation (x_n, $f(x_n)$)
surrogate model
objective function
acquisition function
x_n
Update surrogate model
(x_n, $f(x_n)$)
acquisition max value
x_{n+1}
Step 3. Suggestion: select a new V_{th} configuration (x_{n+1})
A new V_{th} configuration
Step 4. Update V_{th}: initialize SNNs with the new V_{th} configuration
V_{mem}
time
V_{th}^{*1}
V_{th}^{*2}
V_{th}^{*3}
Optimal V_{th} configuration
Initialize SNNs with the optimal V_{th} configuration
Fast and accurate object detector in SNNs

Figure 4.2: Overall process of threshold voltage balancing through Bayesian optimization

can improve object detection accuracy and latency. We therefore introduce a voltage threshold balancing method that scales the threshold voltages to optimal values through Bayesian optimization to improve performance in terms of three important aspects in SNNs: accuracy, latency (time step), and number of synaptic operations (spikes).

4.2.2 Overall process and setup

The overall process and setup of the proposed method are explained as follows. First, we describe several terms used in the proposed threshold voltage balancing through Bayesian optimization. An input X is a threshold voltage configuration and an objective function $f(X)$ that we are trying to maximize is the object detection model in SNNs (Spiking-YOLO [49]) whose output is object detection accuracy (mAP). Note that the threshold voltage configuration consists of a number of threshold voltages which is assigned to each hidden layer. In Bayesian optimization, Gaussian process (GP) with Matern 5/2 kernel is used for a surrogate model $\mathcal{M}$. As for an acquisition function $\mathcal{A}$, expected improvement (EI) is used in this work. Both GP and EI are commonly used in Bayesian optimization and are known to be efficient in finding optima with a minimum number of evaluations [85].

Other hyper-parameters include a number of initial points (N_{init}) and evaluations (I_{eval}), range of input values ($[V_{\text{init}_{\text{min}}}, V_{\text{init}_{\text{max}}}]$), and xi parameter which controls exploitation-exploration trade-off. We empirically found hyper-parameters specifically for the object detection model in SNNs and these hyper-parameters are summarized in Table 4.1. Please note that the overall process is performed on training dataset based on an assumption that the distributions of the training and test datasets are similar. The overall process of the proposed threshold balancing method is presented in Figure 4.2. The details of each step are explained as follows:

Table 4.1: Hyper-parameters in Bayesian optimization

Hyper-parameter	Values
Input (X)	Threshold voltage configuration
Objective function $(f(X))$	Object detection model in SNNs
Surrogate model ($\mathcal{M}$)	Gaussian process w/ Matern 5/2 kernel
Acquisition function ($\mathcal{A}$)	Expected improvements
Range of input	[0.5 V, 1.5 V]
# of initial points (N_{init})	20
# of evaluations (I_{eval})	40
xi^{a}	0.1
α^{b}	0.01

a] exploitation-exploration trade-off parameter
b] noise-level parameter in kernel

- ***Step 0:*** This is an initialization step that is not illustrated in Figure 4.2. Prior to the start of Bayesian optimization, ***Step 0*** is processed once to build initial probabilistic estimation of SNNs according to the number of initial points N_{init}.

- ***Step 1:*** The evaluation step begins by setting the threshold voltages in SNNs with the threshold voltage configuration X_i recommended by the acquisition function $\mathcal{A}$ in previous iteration, where i is the index of iterative evaluation. Each threshold voltage in the suggested threshold voltage configuration is in the range of inputs values $[V_{\text{init}_{\text{min}}}, V_{\text{init}_{\text{max}}}]$. SNNs are now executed for T time steps and produce object detection accuracy (mAP).

- ***Step 2:*** An input-output evaluation pair $(X_i,\ f(X_i))$ will be fed to the surrogate model $\mathcal{M}$ to update probabilistic estimation of SNNs. As the iteration of

Bayesian optimization progresses, the obtained input-output evaluation pairs [(X_1, $f(X_1)$), ..., (X_i, $f(X_i)$)] will be recorded separately for collecting history of the object detection accuracy depending on the threshold voltage configuration. Additional information such as the total number of spikes will be recorded as well.

- ***Step 3:*** Based on the input-output evaluation pair (X_i, $f(X_i)$), the surrogate model $\mathcal{M}$ continues to update the probabilistic estimation of SNNs. Then, the acquisition function $\mathcal{A}$ determines the next sample point (a threshold voltage configuration) $X_{\text{i+1}}$ which maximizes EI over the current best.

- ***Step 4:*** The threshold voltages in SNNs are set to the threshold voltage configuration $X_{\text{i+1}}$ suggested by the acquisition function $\mathcal{A}$ from ***Step 3***. Now, the next evaluation of SNNs starts again. ***Step 1*** through ***Step 4*** is equivalent to one iteration in Bayesian optimization and will be repeated for I_{eval} times.

4.2.3 Design of Bayesian optimization for SNNs

As mentioned in Chapter 2, SNNs operate fundamentally different from DNNs in that they encode and transmit information via spikes in time domain. When using SNNs as objective functions in Bayesian optimization, their characteristics need to be carefully considered. We describe a detailed design of Bayesian optimization proposed in this work, which reflects important characteristics of SNNs.

Threshold voltage search level

In neuroscience literature, it is well-known that neurons in different brain regions process information differently depending on their functionalities [64]. In a similar vein, we applied Bayesian optimization to find different threshold voltages in each

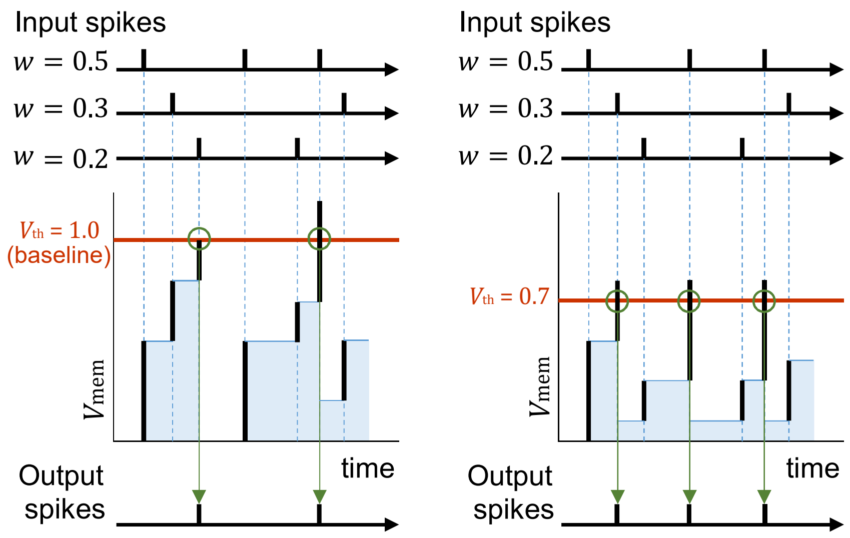


Figure 4.3: Dynamics of IF neuron depending on different threshold voltages

hidden layer, which can be formulated as

$$\max_{V_{\text{th}} \in \mathbb{A}} f(V_{\text{th}}^1, V_{\text{th}}^h, \ldots, V_{\text{th}}^H), \tag{4.1}$$

where V_{th}^H is a threshold voltage for each hidden layer with H being the total number of hidden layers.

Considering the number of synaptic operations

Figure 4.3 illustrates how the total number of spikes is affected by a different threshold voltage value. Note that the number of spikes is equivalent to the number of synaptic operations in SNNs. Compared with the baseline threshold voltage set to 1.0 V, a spiking neuron whose threshold voltage is 0.7 V, will be more likely to generate a spike since a lower amount of integrated membrane potential is required to reach the threshold voltage of 0.7 V. Contrarily, spiking neurons with a threshold voltage greater than the baseline will less likely generate a spike since more membrane potential needs to be integrated before generating a spike. The total number of spikes generated has a significant impact on the power consumption of SNNs. Thus, it is important to reduce the total number of spikes while maintaining object detection accuracy.

In this study, we implement Bayesian optimization so that the total number of spikes is also considered along with object detection accuracy when selecting a final optimal threshold voltage configuration. For instance, in ***Step 2*** of the overall process, the history of the input-output evaluation pairs [(X_1, $f(X_1)$), ..., (X_i, $f(X_i)$)] is recorded as well as the total number of spikes. When selecting the final threshold voltage configuration, one may select the threshold voltage configuration with the less number of spikes if the object detection accuracy among various threshold voltage configurations is tied or their difference is within an affordable range.

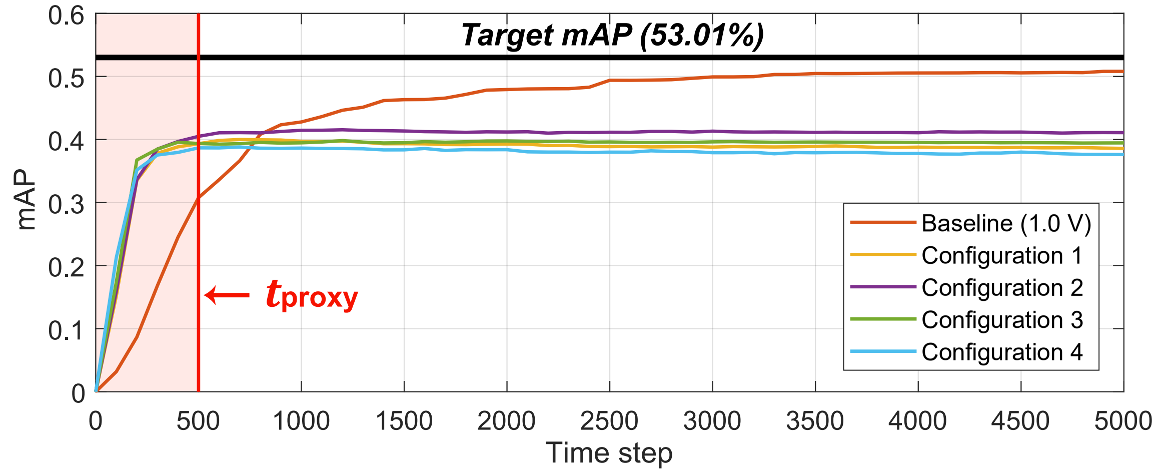


Figure 4.4: Object detection accuracy (mAP %) curve of phase-1 threshold voltages ($V_{\text{th}}^{\text{fast}}$) as time step increases

Proxy evaluation of objective function

When Bayesian optimization is applied in SNNs, two factors mainly affect computational overhead of the overall process: (a) the number of iterations in Bayesian optimization and (b) the number of time steps in SNNs. Particularly for number of time steps in SNNs, a sufficient amount of time steps T (e.g., 5,000 time steps for object detection) are required to precisely assess the quality of threshold voltages suggested during optimization process. This would prohibitively yield large computational overhead which becomes extremely time-consuming and thus difficult to apply in practice.

In this study, we propose a proxy evaluation of SNNs to reduce computational overhead and execution time of the optimization process. In the proxy evaluation, we execute SNNs for only t_{proxy} time steps which are set to 500 (10% of originally targeted time step T of 5,000). This also improves the convergence speed of SNNs which will be discussed in the following section. Also note that Bayesian optimization is not considered scalable with respect to network size. Thus, Spiking-YOLO is selected as the object detection model in SNNs, which uses a real-time object detection model called Tiny YOLO. Tiny YOLO is a simpler but efficient version of YOLO with much less computational overheads.

Since the proxy evaluation only executes for t_{proxy} time steps, the threshold voltages obtained via Bayesian optimization may not perform well in the targeted time step T, as shown in Figure 4.4. To validate that t_{proxy} time step reflects well on the targeted time step T, we report a rank correlation coefficient on the object detection accuracy of 10 threshold voltage configurations obtained via Bayesian optimization in PASCAL VOC [22] and MS COCO [57]. Two commonly-used rank correlation coefficients, namely Spearman rho [74] and Kendall tau [46], are reported in this study. The rank correlation coefficient is in the range of [-1, 1] and a larger value

Table 4.2: Rank correlation coefficients of proxy evaluation of SNNs in Bayesian optimization

	PASCAL VOC	MS COCO
Spearman rho	0.770	0.855
Kendall tau	0.664	0.689

indicates a stronger correlation. As shown in Table 4.2, the proxy evaluation of 500 time steps shows a fairly high rank correlation coefficient to 5,000 time steps (0.770 and 0.855 for PASCAL VOC and MS COCO, respectively). The results confirm that using the proxy evaluation (e.g., 500 time steps) are consistent with those via the full evaluation (e.g., 5,000 time steps).

4.3 Fast and accurate object detection with two-phase threshold voltages

4.3.1 Motivation

Generally, SNNs are executed for T time steps and can produce results on each time step, where time step can be defined as a unit of time that a single spike can be processed. Hence, a trade-off exists between the number of time steps and the accuracy of the network; the more time steps the model is executed for, the more likely it is for the model to achieve higher accuracy. The number of time steps (equivalent to latency), however, has a direct impact on the energy consumption of SNNs. Hence, reducing the number of time steps while achieving competitive results is an important and desirable facet of SNNs, particularly in latency-critical applications such as autonomous vehicles and data centers.

In this work, we propose the concept of two-phase threshold voltages to provide

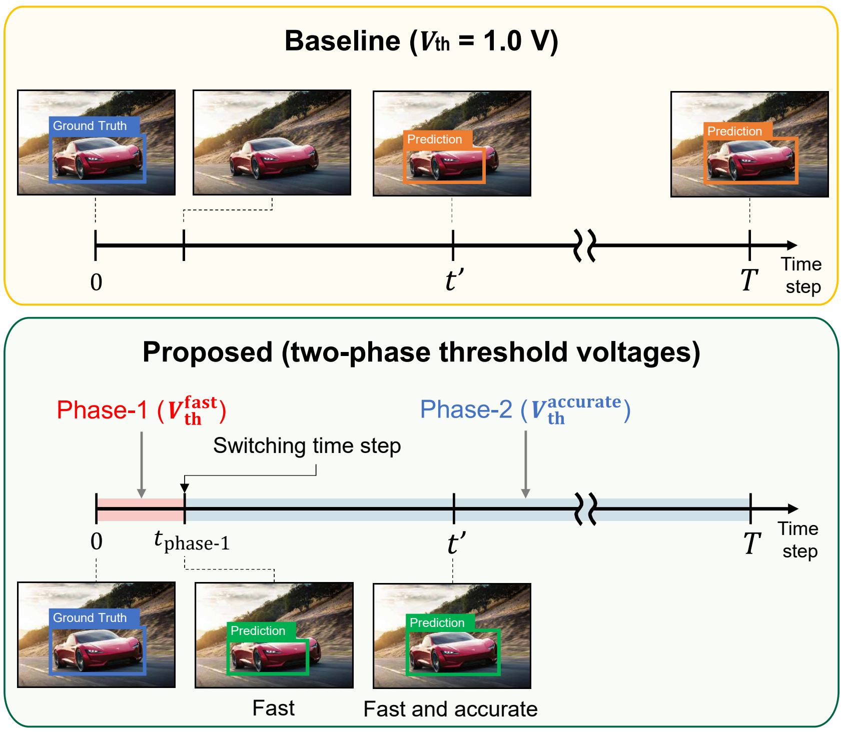


Figure 4.5: Two-phase threshold voltages for fast and accurate object detection in SNNs

fast and accurate object detection in SNNs. The threshold voltages in each phase aim to achieve two distinct goals: phase-1 threshold voltages for fast object detection and phase-2 threshold voltages for accurate object detection. Phase-1 threshold voltages primarily focus on transmitting information fast and early, reducing latency. This may, however, result in rough estimate detection. Phase-2 threshold voltages, on the other hand, concentrate on achieving accurate object detection. This may result in high latency. Combining the two, we can achieve faster and more accurate object detection in SNNs as opposed to conventional methods. The overview of two-phase threshold voltages is presented in Figure 4.5.

4.3.2 Phase-1 threshold voltages: fast object detection

To obtain phase-1 threshold voltages ($V_{\text{th}}^{\text{fast}}$) that can provide fast object detection, we simply execute SNNs for only $t_{\text{phase-1}}$ time steps (e.g., 500 time steps) during Bayesian optimization, which is typically less than the targeted time step T (e.g., 5,000 time steps). Bayesian optimization indeed seeks optimal threshold voltages in each hidden layer that can maximize the object detection accuracy of SNNs at time step $t_{\text{phase-1}}$. Phase-1 threshold voltages primarily focus on transmitting information fast and early. This also has a similar effect to the warm-up phase [29] designed to improve convergence speed. Furthermore, executing SNNs for $t_{\text{phase-1}}$ time step is well aligned with the proxy evaluation (i.e., $t_{\text{proxy}} = t_{\text{phase-1}}$).

Figure 4.4 presents the object detection accuracy of SNNs as the time step increases with the optimal threshold voltages obtained at 500 time steps. Note that target mAP indicates object detection accuracy of Tiny YOLO on PASCAL VOC. SNNs achieve approximately 40% mAP in only 500 time steps while the baseline requires about 800 time steps to achieve a similar object detection accuracy. That is, SNNs with phase-1 threshold voltages show faster convergence speed (1.6x) compared with the baseline. The accuracy curve, however, saturates after 500 time steps and then fails to reach the baseline's best object detection accuracy of 50.81% at 5,000 time steps. To overcome this issue, we introduce phase-2 threshold voltages that provide accurate object detection.

4.3.3 Phase-2 threshold voltages: accurate detection

As expected, SNNs using phase-1 threshold voltages show great performance at $t_{\text{phase-1}}$ time steps, where we specifically design Bayesian optimization to evaluate SNNs for $t_{\text{phase-1}}$ time steps. Phase-1 threshold voltages might be optimal at $t_{\text{phase-1}}$ time steps, yet in the end, they may not be the optimal threshold voltages to achieve state-of-the-art object detection accuracy at the targeted time step T (e.g., 5,000 time steps) as highlighted in Figure 4.4. To overcome such limitation, we introduce phase-2 threshold voltages obtained via Bayesian optimization evaluating SNNs at T time steps. That is, Bayesian optimization is designed to find threshold voltages that can maximize object detection accuracy at the targeted time step T. Hence, SNNs using phase-2 threshold voltages can provide an accurate object detection model.

Furthermore, we propose two-phase threshold voltages, which combine phase-1 and phase-2 threshold voltages. The two-phase threshold voltages can provide a fast and accurate object detection model in SNNs. For fast object detection, SNNs are first set to phase-1 threshold voltages until time step $t_{\text{phase-1}}$, then phase-2 threshold voltages are swapped in to provide accurate object detection. As illustrated in Figure 4.5, the proposed two-phase threshold voltages enable fast object detection at $t_{\text{phase-1}}$ time steps then achieve accurate object detection in time step t', which is much smaller than the targeted time step T. The baseline, on the other hand, starts to draw bounding boxes much later than time step $t_{\text{phase-1}}$ and provides accurate object detection at time step T.

To validate the performance improvement of the proposed two-phase threshold voltages in another aspect, we calculate the mean square error (MSE) between the output values of SNNs and those of DNNs. In the DNN-to-SNN conversion method, DNNs are converted into SNNs that can be directly mapped to spike-based neuromorphic hardware with minimum performance loss [13]. Thus, the original target output values would be those of DNNs, not ground truth. As compared in Figure 4.6, the MSE of SNNs with phase-1 threshold voltages rapidly decreases and is smaller than that of the baseline at 500 time steps. In other words, phase-1 threshold voltages enable fast and relatively accurate information transmission up to 500 time steps. After 500 time steps, however, the MSE starts to level off and actually becomes higher

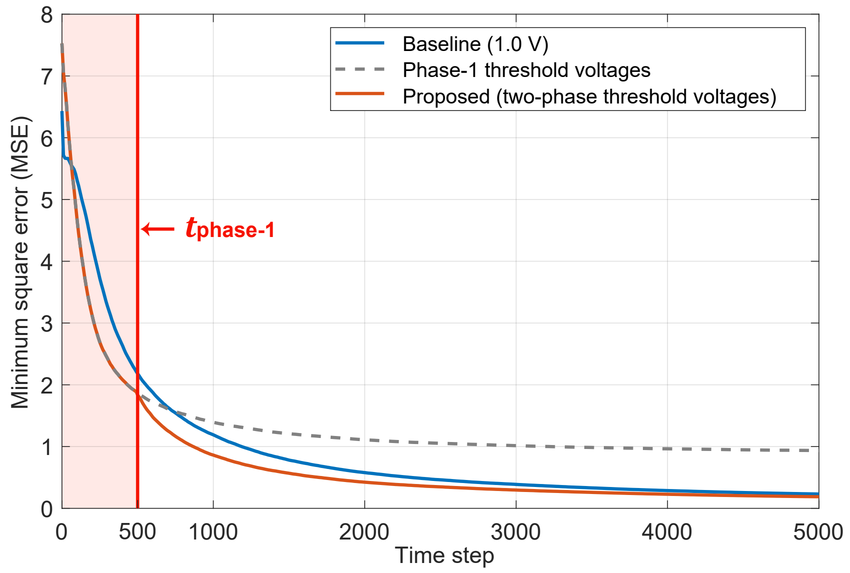


Figure 4.6: MSE comparison of baseline vs. phase-1 voltages vs. proposed (two-phase threshold voltages) as time step increases

than that of the baseline. When the two-phase threshold voltages are applied, the MSE continues to decrease after 500 time steps and follows the similar trend as the baseline. In the end, they achieve a lower MSE than that of the baseline at 5,000 time steps. According to Figure 4.6, the two-phase threshold voltages enable fast and accurate object detection in SNNs when compared with the baseline.

Finally, optimal threshold voltages obtained in each convolutional layer for phase-1 and phase-2 voltages are presented in Table 4.3. There is a tendency that certain layers have a very low threshold voltage value. Particularly, convolutaional layer 2, 5, and 6 have low threshold voltage values regardless of the phase as shown in Table 4.1, which will most likely to generate more spikes. Please note that input range is set to [0.5V, 1.5V]. In fact, in convolutaional layer 5 and 6 have the minimum threshold voltage of 0.5 V. Moreover, average of all threshold voltages in each convolutional layer for Phase-1 threshold voltages is smaller that that of Phase-2 threshold voltages (0.98994 V vs. 1.03458 V). In general, more spikes would indicate more information being transmitted through deep layers. This leads to faster convergence speed at earlier time steps.

Table 4.3: Optimal threshold voltages obtained in each convolutional layer for two-phase voltages

	Conv1	Conv2	Conv3	Conv4	Conv5	Conv6	Conv7	Conv8
Phase-1 V_{th}	1.15405	0.76995	1.100709	1.28183	0.5	0.5	1.39703	1.30961
Phase-2 V_{th}	1.12819	0.88199	0.99469	1.07970	0.83360	0.72381	1.13461	1.5

4.4 Evaluation

4.4.1 Experimental setup

Object detection in SNNs used in this work is based on Spiking-YOLO [49]. In regards to selecting object detection model, the scalability of network size in Bayesian optimization is considered as well. That is, more complex and deeper network can lead to large computational overhead and execution time of Bayesian optimization. Thus we selected Spiking-YOLO which utilizes Tiny YOLO as object detection model in DNNs, which is a simpler but efficient version of YOLO. The implementation of Bayesian optimization is based on [68]. Moreover, we used PASCAL VOC [22] and MS COCO [57] as object detection datasets. Our implementation is based on TensorFlow Eager, and all experiments are conducted on NVIDIA Tesla V100 GPUs.

4.4.2 Experimental results

Object detection accuracy

To present the performance improvement of the proposed methods, we first compared the object detection accuracy of the proposed method (threshold voltage bal-

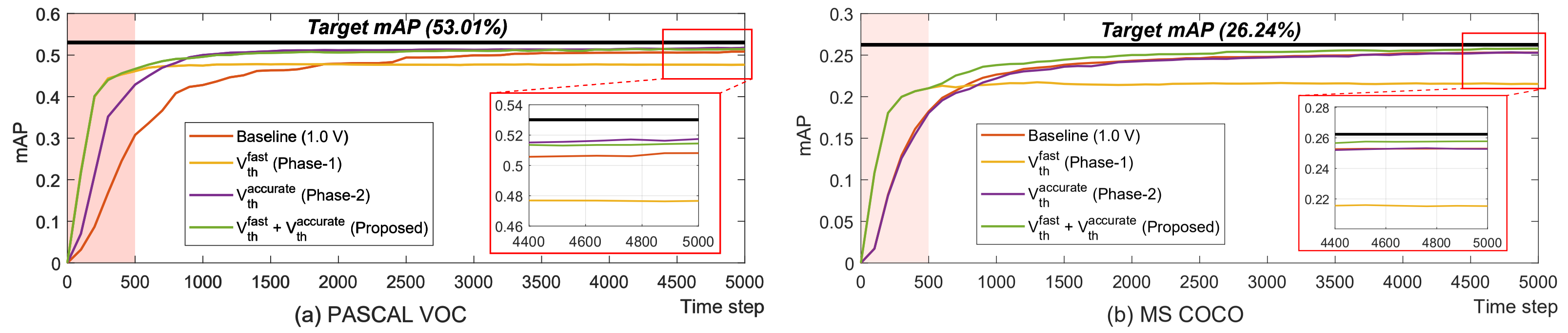


Figure 4.7: Object detection accuracy (mAP %) as time step increases on (a) PASCAL VOC and (b) MS COCO for various threshold voltage configurations

ancing through Bayesian optimization + two-phase threshold voltages) and the baseline (threshold voltage equal to 1.0 V) as the time step increases. The target mAP of Tiny YOLO is 53.01% (PASCAL VOC) and 26.24% (MS COCO). As demonstrated in Figure 4.7 (a), in PASCAL VOC, the proposed method achieved an object detection accuracy of 51.45% at 5,000 time steps. More importantly, to reach 95% of the DNN's target accuracy (53.01%), only 1,300 time steps are required compared with 3,400 time steps at the baseline (2.6x faster). Moreover, the proposed method only took approximately 2,500 time steps to reach the baseline's highest accuracy at 5,000 times steps (50.81%), which is only half the time steps.

When phase-1 threshold voltages ($V_{\text{th}}^{\text{fast}}$) were applied in SNNs for fast object detection, SNNs converged quickly and achieved an object detection accuracy of 46.66% at 500 time steps when compared with 30.78% at the baseline. After 500 time steps, object detection accuracy starts to saturate and achieves only 47.66% at 5,000 time steps. Phase-2 threshold voltages ($V_{\text{th}}^{\text{accurate}}$), on the other hand, achieved the state-of-the-art object detection accuracy of 51.74% at 5,000 time steps. Convergence speed was also relatively fast but not as fast as phase-1 threshold voltages. These results are consistent with the purpose of phase-2 threshold voltages, which is to provide accurate object detection without considering the convergence speed.

As regards to object detection accuracy in MS COCO, the proposed method showed a similar trend as illustrated in Figure 4.7 (b). The proposed method achieved 25.78% at 5,000 time steps while taking only 1,900 time steps to reach 95% of the DNN's target accuracy of 26.24%. That is 1.8x faster than the baseline at 3,400 time steps. Moreover, the proposed method reached the baseline's best object detection accuracy (25.30%) in only 2,700 time steps. Evidently, the proposed method outperforms the baseline in terms of the object detection accuracy and convergence speed for PASCAL VOC and MS COCO. More detailed results are illustrated in Table 4.4.

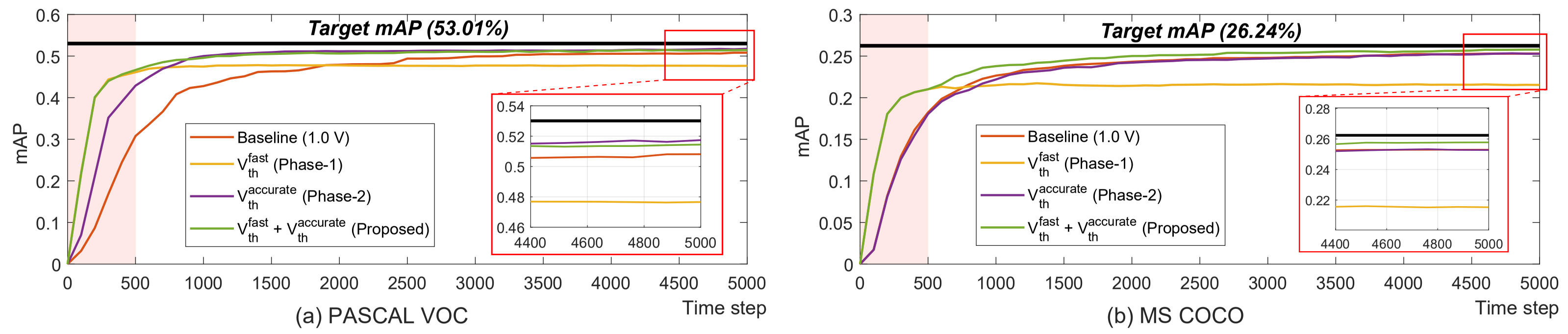


Figure 4.8: Object detection accuracy (mAP %) as a number of spikes increases on (a) PASCAL VOC and (b) MS COCO for various threshold voltage configurations

Table 4.4: Comparison of object detection accuracy (mAP %) in SNNs using various methods

Method	PASCAL VOC[a]		MS COCO[b]	
	500 ts[c]	5,000 ts	500 ts	5,000 ts
Baseline [49]	30.78	50.81	18.23	25.30
Scaled V_{th} (x0.6) [31]	8.36	7.91	5.25	5.29
Scaled V_{th} (x0.8) [32]	29.86	30.93	16.97	16.95
$V_{\text{th}}^{\text{fast}}$	**46.66**	47.66	**21.05**	21.53
$V_{\text{th}}^{\text{accurate}}$	42.88	**51.74**	18.04	25.27
$V_{\text{th}}^{\text{fast}}$ + $V_{\text{th}}^{\text{acc.}}$ **(Proposed)**	**46.66**	51.44	**21.05**	**25.78**

a] Target mAP - 53.01 (%), b] Target mAP - 26.24 (%), c] time step

Total number of synaptic operations

As mentioned previously, the total number of spikes (synaptic operations) directly affects the power consumption of SNNs. Figure 4.8 presents the total number of spikes required to achieve a certain object detection accuracy in PASCAL VOC and MS COCO. As shown in Figure 4.8 (a), in PASCAL VOC, the proposed method generated up to 3.63×10^{10} spikes to achieve the baseline's best accuracy (50.81%) compared with the baseline's 9.00×10^{10} spikes. That is, the proposed method reduced the total number of spikes by 40.33%. Moreover, at 500 time steps, the proposed method only generated 7.51×10^{9} spikes to achieve an accuracy of 46.66% while the baseline generated 8.70×10^{9} spikes to achieve 30.78%. The proposed method clearly has an advantage over the baseline in terms of accuracy, latency, and total number of spikes.

Similar results can be observed for MS COCO as depicted in Figure 4.8 (b). The proposed method generated only 3.56×10^{10} spikes as opposed to 6.51×10^{10} spikes to achieve the best accuracy of the baseline (25.30%). That is, the total number of spikes are reduced by 45.31%. These results confirm that the proposed threshold voltage balancing method with two-phase threshold voltages finds the optimal threshold voltages to achieve fast and accurate object detection with significantly less spikes. With less number of synaptic operations and lower latency, the proposed method provides a highly energy- and power-efficient object detection in SNNs.

Chapter 5

Conclusion

DNNs have shown promising results in various applications yet, their computational overhead and energy efficiency have been a major concern in deploying DNNs on embedded systems such as mobile devices, where available power and computation resources are limited. More sophisticated and extremely large neural networks are required as we try to apply DNNs to more advanced problems existed in our daily lives. The vast amount of training data is also demanded as well for training such networks. In order to improve computational overhead and energy efficiency of DNNs, SNNs have gathered much attention as the third generation of neural networks. SNNs transmit information via spike trains which is consisted of a series of spikes, and use spiking neurons as computational units to enable sparse and event-driven computation. These characteristics lead to extremely high energy efficiency. Nonetheless, there is a lack of scalable training algorithms for SNNs due to their complex dynamics and non-differentialable operations (back-propagation not applicable because of spike-based operations). Therefore, their application have been mainly limited to image classification using indirect training method (e.g., DNN-to-SNN conversion method). In this dissertation, we investigate DNN-to-SNN conversion method in a more challenging task, namely object detection. We introduce a spike-based object detection model called, Spiking-YOLO, and propose novel methods to improve efficiency of the model while achieving the state-of-the-art detection accuracy. This final chapter summarizes our contributions and provides future research directions.

5.1 Dissertation summary

In Chapter 3, we present Spiking-YOLO, the first SNN model that successfully performs object detection and achieves comparable results (up to 98%) to those of the original DNNs on non-trivial datasets, PASCAL VOC and MS COCO. In doing so, we proposed two novel methods; channel-wise weight normalization and a signed neuron with imbalanced threshold. In the conventional methods, weights are normalized by maximum activation value in each layer to prevent under- or over-activation in neurons. This results in sufficient and balanced activation of neurons for image classification. However, when the conventional method is applied in object detection task which is known to be more challenging for its bounding box regression, significant performance degradation occurs due to large deviation among normalized activations. Note that this went unnoticed in image classification because it only selects class with the highest probability. But in object detection, high numerical precision is required in predicting output value of neural network for bounding box regression.

To improve performance of the object detection model in deep SNNs, we introduce channel-wise weight normalization in DNN-to-SNN conversion method. The channel-wise weight normalization is a more fine-grained normalization technique that normalizes the weights by the maximum activation in each channel rather than the layer. It eliminates under-activation in multiple neurons to transmit information fast and accurately. Additionally, we propose a signed neuron with imbalanced threshold voltage. This is an efficient implementation method of leaky-ReLU in SNNs and can be directly mapped in today's neuromorphic architecture without any additional hardware overheads. With the propose methods, Spiking-YOLO achieves detection accuracy that are comparable to those of DNNs (up to 98%) on non-trivial datasets, PASCAL VOC and MS COCO. Furthermore, Spiking-YOLO on a neuromorphic chip consumes roughly 280 times less than DNNs and converges 2.3 to 4 times faster than conventional conversion methods. Spiking-YOLO represents the first step towards solving more advanced machine learning problems in deep SNNs.

Spiking-YOLO achieves comparable results to those of DNNs, yet overall performance can be improved further in terms of latency and number of synaptic operations (spikes). Note that the latency and the number of synaptic operations directly impact energy and power consumption of the model, respectively. Many have attempted to improve performance of SNNs in terms of accuracy and efficiency, but their methods have been primarily limited to image classification. In Chapter 4, we present a new threshold voltage balancing method for object detection in SNNs to improve performance in terms of three important aspects of SNNs: accuracy, latency, and number of synaptic operations. The proposed methods find an optimal threshold voltage in each hidden layer via Bayesian optimization. We also design Bayesian optimization to consider important characteristics of SNNs and introduce proxy evaluation and consideration of total number of spikes when choosing the final optimal threshold voltages. Furthermore, we introduce two-phase threshold voltages that enable faster and more accurate object detection in SNNs while providing high energy efficiency when compared to the conventional methods. The proposed methods achieve the state-of-the-art detection accuracy while converging 2x and 1.85 faster than the conventional methods on PASCAL VOC and MS COCO, respectively. Total number of synaptic operations is also reduced by 40.33% and 45.31% on PASCAL VOC and MS COCO, respectively.

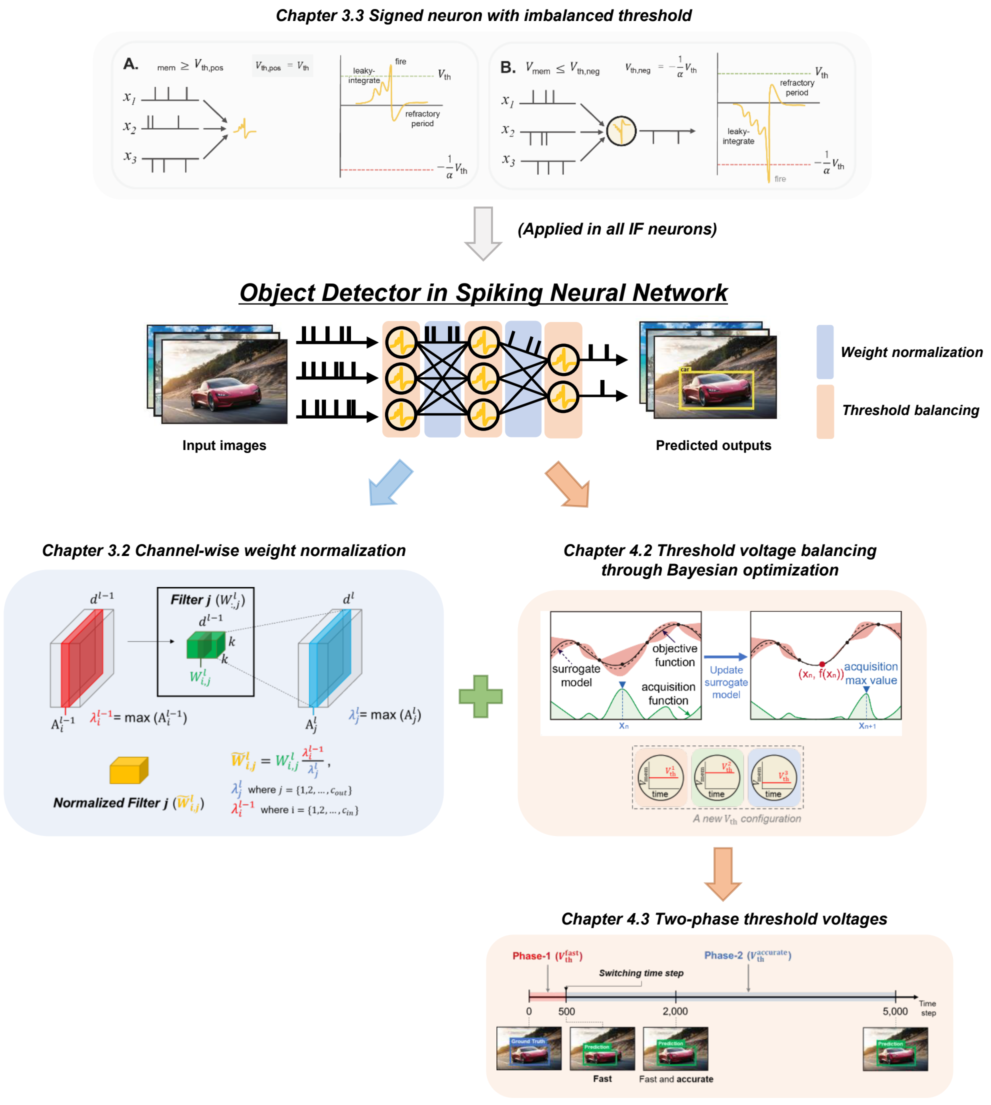
In this dissertation, we proposed several methods that are the first step toward enhancing object detection efficiency, providing fast and accurate object detection in deep SNNs. We believe that bio-inspired SNNs would be valuable in competing with the human brain's learning capability and efficiency in the future.

5.2 Discussion

5.2.1 Overview of the proposed methods and their usages

The overview of this dissertation is shown in Figure 5.1. To enable fast and accurate information transmission in deep SNNs, we propose various DNN-to-SNN conversion methods which can be divided into four: (a) channel-wise weight normalization, (b) signed neuron with imbalanced threshold voltage, (c) threshold voltage balancing through Bayesian optimization, and (d) two-threshold voltages. These methods can be applied at the same time and eventually are compliment to each other. Signed neuron with imbalanced threshold voltage, which allows negative activation of spiking neurons and precisely implements slope α, is required to successfully perform object detection in deep SNNs. As shown in Figure 3.14, Spiking-YOLO fails to detect any objects when signed neuron with imbalanced threshold voltage is not applied. This is because all the negative activations are neglected, which is over 40% of activation in Tiny YOLO. Thus, signed neuron with imbalanced threshold voltage is applied to all spiking neurons by default as shown in top portion of Figure 5.1.

As was explained in Chapter 2, either weight normalization or threshold voltage balancing is applied to provide sufficient activation of neurons in deep SNNs. In this dissertation, we propose to use both weight normalization and threshold voltage balancing as presented in bottom portion of Figure 5.1. We first propose channel-



Chapter 3.3 Signed neuron with imbalanced threshold
(Applied in all IF neurons)
Object Detector in Spiking Neural Network
Input images
Predicted outputs
Weight normalization
Threshold balancing
Chapter 3.2 Channel-wise weight normalization
Normalized Filter j
Chapter 4.2 Threshold voltage balancing through Bayesian optimization
objective function
surrogate model
acquisition function
Update surrogate model
acquisition max value
A new V_{th} configuration
Chapter 4.3 Two-phase threshold voltages
Phase-1 (V_{th}^{fast})
Phase-2 ($V_{th}^{accurate}$)
Switching time step
Time step
Fast
Fast and accurate

Figure 5.1: Final overview of all the proposed methods

wise weight normalization which enables more fine-grained normalization method to eliminate under-activation of neurons and improve firing rate. Second, we propose a new threshold voltage balancing method by using Bayesian optimization which automatically finds optimal threshold voltages in each hidden layer to maximize object detection accuracy while reducing latency and synaptic operations.

Furthermore, we propose two-phases threshold voltages to provide faster and more accurate information transmission in deep SNNs. The two-phase threshold voltages can be considered as an extension from threshold voltage balancing method. Each threshold voltage in two-phase threshold voltages is obtained from the threshold voltage balancing through Bayesian optimization by modifying an evaluation time step T during Bayesian optimization process. As shown in experimental results in 4, using both weight normalization and threshold voltage balancing methods certainly improve performance of object detection model in SNNs. However, one may use either weight normalization or threshold voltage balancing method alone. Also various two-phase threshold voltages may be applied by modifying evaluation time step T depending on application's needs (e.g., detection accuracy, latency, or a number of synaptic operations).

5.3 Challenges in SNNs

The most common training algorithm in DNNs, namely error back-propagation algorithm made a huge impact in success of DNNs for various applications. In the similar vein, the biggest challenge that SNNs community faces is development of scalable training algorithm specifically for SNNs; error back-propagation algorithm version of SNNs. As an alternative approach, DNN-to-SNN conversion methods have been shown promising results in object detection, bridging the performance gap between

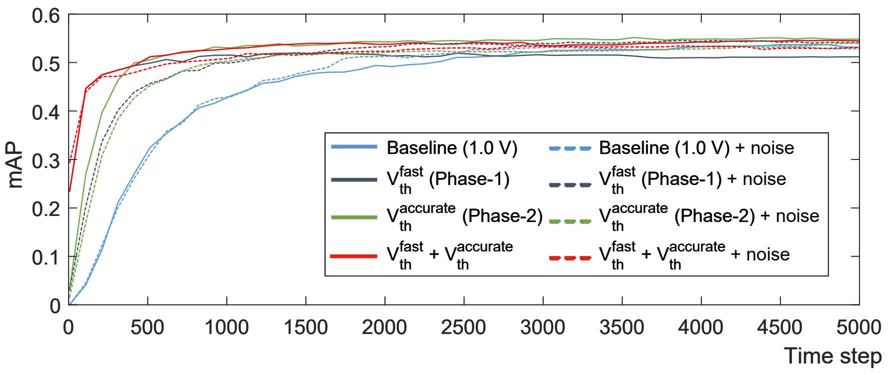


Figure 5.2: Object detection accuracy (mAP %) as a number of time increases for all proposed methods with and without noise

DNNs and SNNs. Moreover, converted SNNs only execute accumulation operations (i.e., simply adding input spikes) while matrix multiplication accounts for most of the operations in DNNs. As presented in Table 3.2 of Chapter 3, MAC operation consumes 4.6 pJ while AC operation consumes 0.1 pJ. Thus, DNN-to-SNN conversion methods show greater energy efficiency. Nevertheless, they are indirect training methods and use hyper-parameters from trained DNNs. SNNs are still in need of scalable training method that is specifically designed for SNNs or potential of SNNs such as powerful learning capability and energy efficiency can be diminish.

As mentioned in Chapter 1 in SNNs are the preferred neural networks in neuromorphic architectures [63, 75] because they similarly process information (data) through spikes (signals). The current neuromorphic architectures, however, are typically based on analog-digital neuromorphic circuits. Particularly in [63], 1 million digital neurons and 256 million synapses are used in 5 billion transistors chip. Digital circuits are known to be more latest technology than analog circuits, which known to work better and are more popular technology in majority of applications. However, analog circuits have several advantages that can be work better in neural networks: (a) can store synaptic weights and process signals (i.e., multiplication), and (b) can reduce power consumption and increases circuit density dramatically, both of which can overcome bottlenecks existed in the current von Neumann architecture.

The proposed methods in this work, however, are simulated on GPUs and the characteristics of analog circuits have been neglected. In this section, we analyze performance of Spiking-YOLO and the proposed methods when applied in analog circuits. Analog circuits are operated based on analog signals which tend to have lower quality than digital. Consequently, we have added Gaussian noise (mean and standard deviation are set to 0 and 0.1, respectively) to threshold voltage V_{th} and input of jth neuron in the lth layer $z_j^l(t)$. Figure 5.2 shows the object detection accuracy curve as the time step increases for the proposed methods with and without noise.

Overall, adding noise to V_{th} and $z_j^l(t)$ leads to very small performance degradation in object detection accuracy at 5,000 time steps. However, Phase-1 and Phase-2 threshold voltages yield higher latency and converges much slower when noises are added. Two-phase threshold voltages (Phase-1 + Phase-2) have minimum effect when noise is added in terms of both accuracy and latency. Nevertheless, the noise applied in this experiment is just one of many variations in analog circuits. The noise level is also significantly low compared to noise level in actual analog circuits which becomes even higher in advanced scaled processes. Despite the proposed DNN-to-SNN conversion methods have shown promising results, direct training of SNNs is a vital future research direction for analog-based neuromorphic architectures.

5.4 Future Work

5.4.1 Extension to various applications and DNN models

The proposed methods have shown promising results in object detection task. First, the channel-wise weight normalization provides a more fine-grained normalization eliminating under-activation in multiple neurons to provide sufficient activation of neurons for fast and accurate information transmission. Second, the signed neuron with imbalanced threshold provides efficient implementation of leaky-ReLU in SNNs. The signed neuron with imbalanced threshold is a more general approach in that various DNNs these days often use leaky-relu as an activation function [42, 99, 9]. Building on these insights, the proposed methods can be applied in variety of application and DNN models. For instance image segmentation outputs a pixel-wise mask of the image in that each pixel is given a label (e.g., class). Image segmentation can be considered as more challenging task than object detection. Its applications are autonomous driving and medical imaging. The proposed methods can be extended to image segmentation and other computer vision task as well for future work.

Furthermore, SNNs transmit information through precise timing of spikes in that they utilize temporal aspects in information transmission among neurons. Naturally, SNNs open possibility to exploit time series applications such as automatic speech recognition [93].

5.4.2 Further improve efficiency of SNNs

As for another field of study in indirect training methods, various neural coding schemes have been proposed in recent years. The neural coding is a neural representation of information in spike trains. In SNNs, neurons transmit information via a series of spikes, known as a spike train, and neural coding determines how the information is encoded in spikes trains and is be decoded when information is received. One of the most well-known neural coding schemes is rate coding. Nevertheless other neural coding schemes such as temporal coding and burst coding have been extensively studied in recent years. The rate coding is based on firing rate of neurons and have advantages in simple implementation and its robustness. However, it suffers from large latency and total number of spikes. Temporal coding utilizes temporal aspects where timing of receiving spikes represents the amount of information being transmitted. Certainly, it has advantages in significantly less number of spikes being generated but a periodic oscillation function is typically required for global reference. Note that Spiking-YOLO is based on rate coding, and various neural coding schemes can be applied in order to improve efficiency of the model. Furthermore, [71] claims that hybrid neural coding schemes can improve efficiency further, which can be also considered as a future work.

As was mentioned previously, the channel-wise weight normalization eliminates under-activation of neurons and provides a more fine-grained normalization. Nevertheless, each channel in Convolutional layer 1 has 173,056 neurons. This indicates that the deviation of normalized activations is still large and there is a room for improvement. In fact, a more fine-grained normalization method can be studied extensively. For instance, we can select a certain group of neurons based on their importance in predicting output of the network (or subsequent layer), and normalize weights by the maximum activation value in that specific group. The importance of the neuron can be calculated by the activation value (e.g., maximum or L1-norm). This will eliminate under-activation in neurons further, and improve firing rate of those neurons, which lead to faster convergence of object detection in SNNs.

5.4.3 Optimization of deep SNNs

As illustrated in Chapter 4, we used Bayesian optimization as optimization algorithm when finding optimal values for threshold voltage in each hidden layer. Bayesian optimization has advantages in simple implementation and also finds optimal values relatively well when compared to random and grid search. However, there are a number of hyper-parameters in Bayesian optimization such as kernel, exploitation-exploration factor, and so on. These hyper-parameters need to be fine-tuned in order to improve the optimization process. Moreover, there are other optimization algorithms such as [23, 45, 84, 53], some of which has shown promising results. These can be applied and compared in optimization of threshold voltages. Furthermore, threshold search level can be extended to channel-wise or even further, to provide more fine-grained threshold voltage balancing.

Furthermore, there are other optimization problems existed in SNNs. In this dissertation, both weight normalization and threshold voltage balancing is used, but only one of the methods can be applied by using the optimization algorithm. In fact, the weight normalization can be optimized with the optimization algorithm. In this dissertation, we consider the search of optimal threshold voltage as hyper-parameter optimization. We can further optimize other parameters in SNNs. In fact, actually architecture of SNNs can be optimized with AutoML. AutoML is a field of finding the optimal neural network design automatically, taking the human out of the equation. Many previous works have successfully found optimal architectures in various applications that human has not be able to come up with before [58, 76].

Bibliography

[1] James and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. learning research*, 13(Feb):281–305, 2012.

[2] James S , Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proc. 24th Annu. Conf. Adv. Neural Inf. Process. Syst. (NeurIPS)*, pages 2546–2554, 2011.

[3] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

[4] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[5] Rony Azouz and Charles M Gray. Dynamic spike threshold reveals a mechanism for synaptic coincidence detection in cortical neurons in vivo. *Proc. National Acad. Sciences*, 97(14):8110–8115, 2000.

[6] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and

Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *NIPS*, 2018.

[7] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

[8] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[9] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 9157–9166, 2019.

[10] Korbinian Brodmann. *Vergleichende Lokalisationslehre der Grosshirnrinde in ihren Prinzipien dargestellt auf Grund des Zellenbaues*. Barth, 1909.

[11] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[12] Widodo Budiharto, Alexander AS Gunawan, Jarot S Suroso, Andry Chowanda, Aurello Patrik, and Gaudi Utama. Fast object detection for quadcopter drone using deep learning. In *Proc. 2018 3rd Int. Conf. on Comput. and Commun. Syst. (ICCCS)*, pages 192–195. IEEE, 2018.

[13] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015.

[14] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

[15] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[16] Nima Dehghani, Adrien Peyrache, Bartosz Telenczuk, Michel Le Van Quyen, Eric Halgren, Sydney S Cash, Nicholas G Hatsopoulos, and Alain Destexhe. Dynamic balance of excitation and inhibition in human and monkey neocortex. *Scientific reports*, 6:23176, 2016.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[19] 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 *IJCNN*, 2015.

[20] Caiwen Ding, Shuo Wang, Ning Liu, Kaidi Xu, Yanzhi Wang, and Yun Liang. Req-yolo: A resource-aware, efficient quantization framework for object detection on fpgas. In *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, pages 33–42, 2019.

[21] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 293–302, 2019.

[22] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comp. Vis.*, 88(2):303–338, 2010.

[23] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *Proc. Int. Conf. Mach. Learning (ICML)*, 2018.

[24] Hongxiang Fan, Shuanglong Liu, Martin Ferianc, Ho-Cheung Ng, Zhiqiang Que, Shen Liu, Xinyu Niu, and Wayne Luk. A real-time object detection accelerator with compressed ssdlite on fpga. In *Proc. 2018 Int. Conf. Field-Program. Tech. (FPT)*, pages 14–21. IEEE, 2018.

[25] Wei Fang, Lin Wang, and Peiming Ren. Tinier-yolo: A real-time object detection method for constrained environments. *IEEE Access*, 8:1935–1944, 2019.

[26] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.

[27] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[28] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. In *CVPR*, 2014.

[29] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz

Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[30] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *NIPS*, 2016.

[31] Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In *Proc. IEEE Eur. Conf. Comput. Vis. (ECCV)*, 2020.

[32] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 13558–13567, 2020.

[33] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[35] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[36] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017.

[37] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[38] Mark Horowitz. 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)*, pages 10–14. IEEE, 2014.

[39] Rachel Huang, Jonathan Pedoeem, and Cuixian Chen. Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers. In *Proc. 2018 IEEE Int. Conf. Big Data (Big Data)*, pages 2503–2510. IEEE, 2018.

[40] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NIPS*, 2016.

[41] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. Int. Conf. learning Intell. Opt.*, pages 507–523. Springer, 2011.

[42] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[43] Zhaoyin Jia, Ashutosh Saxena, and Tsuhan Chen. Robotic object detection: Learning to improve the classifiers using sparse graphs for path planning. In *Proc. 22nd Int. Joint Conf. Artif. Intell. (IJCAI)*, 2011.

[44] Yingyezhe Jin, Wenrui Zhang, and Peng Li. Hybrid macro/micro level back-propagation for training deep spiking neural networks. In *NIPS*, 2018.

[45] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration

in multi-armed bandits. In *International Conference on Machine Learning*, pages 1238–1246, 2013.

[46] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2): 81–93, 1938.

[47] Jaehyun Kim, Heesu Kim, Subin Huh, Jinho Lee, and Kiyoung Choi. Deep neural networks with weighted spikes. *Neurocomputing*, 311:373–386, 2018.

[48] Jinkyu Kim and John Canny. Interpretable learning for self-driving cars by visualizing causal attention. In *Proceedings of the IEEE international conference on computer vision*, pages 2942–2950, 2017.

[49] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: Spiking neural network for real-time object detection. In *Proc. 34th AAAI Conf. Artif. Intell. (AAAI)*, 2020.

[50] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. In *CVPR*, 2015.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[52] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

[53] Stefan Lessmann, Robert Stahlbock, and Sven F Crone. Optimizing hyperpa-

rameters of support vector machines by genetic algorithms. In *IC-AI*, pages 74–82, 2005.

[54] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 7644–7652, 2019.

[55] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 2810–2819, 2019.

[56] Tianhong Li, Jianguo Li, Zhuang Liu, and Changshui Zhang. Few sample knowledge distillation for efficient network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14639–14647, 2020.

[57] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. IEEE Eur. Conf. Comput. Vis. (ECCV)*, pages 740–755. Springer, 2014.

[58] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[59] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

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

[61] Zachary F Mainen and Terrence J Sejnowski. Reliability of spike timing in neocortical neurons. *Science*, 268(5216):1503–1506, 1995.

[62] Ana I Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5419–5427, 2018.

[63] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[64] Yasuhiro Mochizuki, Tomokatsu Onaga, Hideaki Shimazaki, Takeaki Shimokawa, Yasuhiro Tsubo, Rie Kimura, Akiko Saiki, Yutaka Sakai, Yoshikazu Isomura, Shigeyoshi Fujisawa, et al. Similarity in neuronal firing regimes across mammalian species. *Journal of Neuroscience*, 36(21):5736–5747, 2016.

[65] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019.

[66] Hesham Mostafa, Bruno U Pedroni, Sadique Sheik, and Gert Cauwenberghs. Fast classification using sparsely active spiking networks. In *2017 IEEE In-*

ternational Symposium on Circuits and Systems (ISCAS), pages 1–4. IEEE, 2017.

[67] Jacinto C Nascimento and Jorge S Marques. Performance evaluation of object detection algorithms for video surveillance. *IEEE Trans. Multimedia*, 8(4): 761–774, 2006.

[68] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–. URL `https://github.com/fmfn/BayesianOptimization`.

[69] Tesla NVIDIA. V100 gpu architecture, 2017.

[70] Seongsik Park, Seijoon Kim, Seil Lee, Ho Bae, and Sungroh Yoon. Quantized memory-augmented neural networks. In *AAAI*, 2018.

[71] Seongsik Park, Seijoon Kim, Hyeokjun Choe, and Sungroh Yoon. Fast and efficient information transmission with burst spikes in deep spiking neural networks. In *2019 56th ACM/IEEE Design, Auto. Conf. (DAC)*, pages 1–6. IEEE, 2019.

[72] Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2fsnn: Deep spiking neural networks with time-to-first-spike coding. In *Proc. Design Auto. Conf. (DAC)*, 2020.

[73] Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: opportunities and challenges. *Frontiers in neuroscience*, 12, 2018.

[74] W Pirie. Spearman rank correlation coefficient encyclopedia of statistical sciences, 2004.

[75] Chi-Sang Poon and Kuan Zhou. Neuromorphic silicon neurons and large-scale neural networks: challenges and opportunities. *Frontiers in neuroscience*, 5: 108, 2011.

[76] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[77] Joseph Redmon. Yolo: Real-time object detection. `https://pjreddie.com/darknet/yolov2/`, 2016.

[78] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, 2017.

[79] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[80] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[81] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[82] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11: 682, 2017.

[83] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy.

Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience*, 13:95, 2019.

[84] Yuhui Shi and Russell C Eberhart. Parameter selection in particle swarm optimization. In *International conference on evolutionary programming*, pages 591–600. Springer, 1998.

[85] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Proc. 25th Annu. Conf. Adv. Neural Inf. Process. Syst. (NeurIPS)*, pages 2951–2959, 2012.

[86] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proc. Int. Conf. Mach. Learning (ICML)*, 2019.

[87] Arthur W Toga, Paul M Thompson, Susumu Mori, Katrin Amunts, and Karl Zilles. Towards multimodal atlases of the human brain. *Nature Reviews Neuroscience*, 7(12):952–966, 2006.

[88] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. Segmentation as selective search for object recognition. In *2011 International Conference on Computer Vision*, pages 1879–1886. IEEE, 2011.

[89] Yi Wei, Xinyu Pan, Hongwei Qin, Wanli Ouyang, and Junjie Yan. Quantization mimic: Towards very tiny cnn for object detection. In *Proc. IEEE Eur. Conf. Comput. Vis. (ECCV)*, pages 267–283, 2018.

[90] Hugh R Wilson and Jack D Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical journal*, 12(1):1–24, 1972.

[91] Alexander Womg, Mohammad Javad Shafiee, Francis Li, and Brendan Chwyl. Tiny ssd: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection. In *Proc. 2018 15th Conf. Comp Robot Vis. (CRV)*, pages 95–101. IEEE, 2018.

[92] Bichen Wu, Forrest Iandola, Peter H Jin, and Kurt Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshop (CVPRW)*, pages 129–137, 2017.

[93] Jibin Wu, Emre Yılmaz, Malu Zhang, Haizhou Li, and Kay Chen Tan. Deep spiking neural networks for large vocabulary automatic speech recognition. *Frontiers in Neuroscience*, 14:199, 2020.

[94] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *AAAI*, 2019.

[95] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. In *CVPR*, 2015.

[96] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

[97] Lei Zhang, Shengyuan Zhou, Tian Zhi, Zidong Du, and Yunji Chen. Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding. In *Proc. 33rd AAAI Conf. Artif. Intell. (AAAI)*, volume 33, pages 1319–1326, 2019.

[98] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropa-

gation for deep spiking neural networks. In *Proc. 33th Annu. Conf. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020.

[99] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[100] Karl Zilles and Katrin Amunts. Centenary of brodmann's map—conception and fate. *Nature Reviews Neuroscience*, 11(2):139–145, 2010.

[101] James Zou, Mikael Huss, Abubakar Abid, Pejman Mohammadi, Ali Torkamani, and Amalio Telenti. A primer on deep learning in genomics. *Nature genetics*, 51(1):12–18, 2019.

초 록

오늘 날 딥러닝의 큰 성공은 고성능 병렬 컴퓨팅 시스템의 발전과 복잡한 모델을 학습하기 위해 필요한 많은 양의 데이터가 수집되어 접근이 가능해진 점이라고 할 수 있다. 하지만 실제 세상에 존재하는 더 어려운 문제들을 풀고자할 때는 더욱 더 섬세하고 복잡한 모델과 이 모델을 성공적으로 학습할 수 있는 방대한 양의 데이터를 필요한다. 하지만 이러한 점들은 모델 수행 시 연산 오버헤드와 전력 소모를 급격하게 증가시킬 수 밖에 없다. 이러한 문제점들을 극복하는 여러 방법들 중 하나로 스파이킹 뉴럴 네트워크가 최근 많은 주목을 받고 있다. 스파이킹 뉴럴 네트워크는 제 3세대 인공 신경망으로 불리며 이벤트 중심의 동작을 기반으로 하여 저전력이 가장 큰 장점이다. 스파이킹 뉴럴 네트워크는 실제 인간의 뇌에서 뉴런들 간 정보를 전달하는 방식을 모방하며 스파이킹 뉴런을 연산 단위로 사용하고 있다. 스파이킹 뉴럴 네트워크는 생물학적 신경계와 동일하게 시간적 정보를 활용할 수 있기 때문에 매우 뛰어난 연산 능력을 가지고 있다.

하지만 스파이킹 뉴럴 네트워크는 이미지 분류와 같은 비교적 쉬운 응용에만 주로 사용되고 있으며 얕은 인공 신경망과 간단한 데이터셋에서만 주로 수행되고 있다. 이러한 제약이 존재하는 가장 큰 요인 중 하나는 스파이크 뉴럴 네트워크에 적합한 학습 알고리즘이 아직 존재하지 않기 때문이다. 스파이크로 정보를 전달하고 연산을 수행하기 때문에 미분이 불가능하다. 따라서 딥 뉴럴 네트워크에서 주로 사용되는 역전파 알고리즘의 사용이 불가능하다. 본 논문에서 딥 스파이킹 뉴럴 네트워크를 이미지 분류보다 더 어려운 회귀 문제 (객체 인식)에 적용해 보고, 딥 뉴럴 네트워크의 성능에 버금가는 객체 인식 모델을 스파이킹 뉴럴 네트워에서 처음으로 제안한다. 더 나아가, 객체 인식 모델의 성능과 지연시간, 에너지 효율성을 향상시킬 수 있는 여러 방법들을 제안한다. 본 논문은 크게 두 가지 주제로 나누어 설명한다: (a) 딥 스파이킹 뉴럴 네트워크에서의 객체 인식 모델, (b) 딥 스파이킹 뉴럴

네트워크에서의 객체 인식 모델의 성능 및 효율성 향상. 제안하는 방법들을 통해 빠르고 정확한 객체 인식 모델을 딥 스파이킹 뉴럴 네트워크에서 수행할 수 있다.

첫 번째 방법은 딥 스파이킹 뉴럴 네트워크에서의 객체 인식 모델이다. 객체 인식 모델은 Spiking-YOLO로 부르고, 저자들이 아는 바에 의하면 PASCAL VOC, MS COCO와 같은 데이터 셋에서 딥 뉴럴 네트워크의 성능에 버금가는 결과를 보여준 첫 번째 스파이킹 뉴럴 네트워크를 기반으로 하는 객체 인식 모델이다. Spiking-YOLO에서는 크게 두 가지 방법을 제안한다. 첫번 째는 채널 별 가중치 정규화이고 두번째는 불균형 한계 전압을 가지는 양음수 뉴런이다. 두 가지 방법을 통해 빠르고 정확한 정보를 딥 스파이킹 뉴럴 네트워크에서 전달 가능하게 한다. 실험 결과, Spiking-YOLO는 PASCAL VOC와 MS COCO 데이터셋에서 딥 뉴럴 네트워크의 객체 인식률의 98%에 뛰어난 성능을 보였다. 또한 Spiking-YOLO가 뉴로모픽 칩에 구현되었음 가정하였을 때, Tiny YOLO보다 약 280의 에너지를 적게 소모하였고 기존의 DNN-to-SNN 전환 방법들 보다 2.3배에서 4배 더 빠르게 수렴하는 것을 확인할 수 있었다.

두 번째 방법은 스파이킹 뉴럴 네트워크에 조금 더 효율적인 연산 능력을 부여하는데 중점을 주고 있다. 비록 스파이킹 뉴럴 네트워크가 희박한 양의 스파이크로 정보를 효율적으로 전달하며 연산 오버헤드와 에너지 소모가 적지만, 두 가지 매우 중요한 문제들이 존재한다: (a) 지연속도: 좋은 성능을 내기 위해 필요한 타임스탭, (b) 시냅틱 연산수: 추론 시 생성된 총 스파이크의 수. 이러한 문제들을 적절히 해결하지 못한다면 스파이킹 뉴럴 네트워크의 큰 장점이라고 할 수 있는 에너지와 전력 효율성이 크게 저하될 수 있다. 이를 해결하기 위해 본 논문에서는 한계 전압 균형 방법론을 새로 제안한다. 제안하는 방법론은 베이시안 최적화 알고리즘을 사용하여 가장 최적의 한계전압 값을 찾는다. 또한 베이시안 최적화 알고리즘을 지연속도나 시냅틱 연산수 등의 스파이킹 뉴럴 네트워크의 특성을 고려할 수 있게 디자인한다. 더 나아가, 두 단계의 한계 전압을 제안하여 높은 에너지 효율을 가지며 더 빠르고 더 정확한 객체 인식 모델을 가능하게 한다. 실험 결과에 따르면

제안하는 방법들을 통해 state-of-the-art 객체 인식률을 달성하였고 기존의 방법들보다 PASCAL VOC에서는 2배, MS COCO에서는 1.85배 빠르게 수렴하는 것을 확인하였다. 또한 시냅틱 연산수도 PASCAL VOC에서는 40.33%, MS COCO에서는 45.31%를 줄일 수 있었다.

주요어: 인공신경망, 딥 러닝, 스파이킹 신경망, 에너지 효율, 객체 인식, 베이시안 최적화

학번: 2013-23104