# T2FSNN: Deep Spiking Neural Networks with Time-to-first-spike Coding

Seongsik Park, Seijoon Kim, Byunggook Na, Sungroh Yoon*
Department of Electrical and Computer Engineering, ASRI, INMC, and Institute of Engineering Research
Seoul National University, Seoul 08826, South Korea

***Abstract*—Spiking neural networks (SNNs) have gained considerable interest due to their energy-efficient characteristics, yet lack of a scalable training algorithm has restricted their applicability in practical machine learning problems. The deep neural network-to-SNN conversion approach has been widely studied to broaden the applicability of SNNs. Most previous studies, however, have not fully utilized spatio-temporal aspects of SNNs, which has led to inefficiency in terms of number of spikes and inference latency. In this paper, we present T2FSNN, which introduces the concept of time-to-first-spike coding into deep SNNs using the kernel-based dynamic threshold and dendrite to overcome the aforementioned drawback. In addition, we propose gradient-based optimization and early firing methods to further increase the efficiency of the T2FSNN. According to our results, the proposed methods can reduce inference latency and number of spikes to 22% and less than 1%, compared to those of burst coding, which is the state-of-the-art result on the CIFAR-100.**

***Index Terms*—Biological neural networks, Neuromorphics, Supervised learning, Image classification**

## I. Introduction

Recent advances in deep neural networks (DNNs) have led to state-of-the-art results in a variety of applications. These DNNs, however, demand a substantial amount of computation and power consumption, which restricts the employment of deep learning to edge devices. To overcome these challenges, many studies have focused on reducing the model size and the amount of computation through quantization [1] and pruning [2]. Despite these attempts, the complexity of the DNN models has increased rapidly, making it difficult to deploy the DNNs efficiently in resource-constrained environments, such as mobile devices [3].

Spiking neural networks (SNNs) have emerged as the next generation of neural networks for their superior energy efficiency caused by the features of integrate-and-fire and event-based operations [4]. Although SNNs have the potential for improving energy efficiency of artificial neural networks, deep SNNs have not been widely used in many applications due to lack of scalable training algorithms. Deep SNNs have been struggling to achieve competitive results compared to DNNs. Although many studies about direct training with approximate stochastic gradient descent (SGD) have been published recently [5], [6], they have shown unsatisfactory results.

DNN-to-SNN conversion methods have been proposed in recent years [7], [8], [9], [10], [11], [12], [13] to address the issue of training deep SNNs. Using the conversion methods, deep SNNs can exploit the DNNs' training performance with the pretrained synaptic weights of DNNs, which results in the competitive performance of SNNs. There are several factors affecting the efficiency of deep SNNs, including neuron types and neural coding schemes, during the conversion. Among these factors, many studies have focused on neural coding scheme, which is critical in transmitting accurate information with low energy consumption and latency [10], [11], [12], [13].

To date, various neural coding schemes have been applied to SNNs, such as rate [14] and temporal coding [10], [15], [16], [17]. Rate coding, which is a well-known and commonly used neural coding, utilizes firing rate to represent information [14]. Many DNN-to-SNN conversion methods have adopted rate coding for its simple implementation and robustness [7], [8], [9]. However rate coding generates a large number of spikes and has a slow information transmission. In the wake of rate coding, many temporal coding schemes, including phase [16] and burst [10], were introduced to deep SNNs to utilize the temporal information in the spike trains as in biological neural systems. Deep SNNs with phase coding [11] and burst coding [10] have improved the efficiency of inference with comparable results to those of DNNs. However, they still fall short of the desired performance in terms of latency and number of spikes.

To fully utilize temporal coding, time-to-first-spike (TTFS) coding has been applied to deep SNNs recently [12], [13]. The study [13], where the important information was delivered first, successfully introduced TTFS coding into deep SNNs, leading to a significant reduction in number of spikes. However, this research did not show satisfactory results. The TDSNN [12] with reverse coding, which is a kind of TTFS coding, was proposed to improve the accuracy of deep SNNs. The deep SNNs with reverse coding achieved competitive results with DNNs. However, the TDSNN does not report the number of spikes and latency, which are critical measures to evaluate the efficiency of deep SNNs. Furthermore, the additional spikes from auxiliary neurons and the reverse coding prohibited the deep SNNs from improving inference efficiency.

In this paper, we propose a novel deep SNN model, called T2FSNN, for efficient implementation of TTFS coding in deep SNNs. The T2FSNN exploits a kernel-based dynamic threshold and dendrite with TTFS coding, where the earlier spikes represent more critical information. In addition, we propose a gradient descent algorithm to optimize the kernels in the dynamic threshold and dendrite to further improve infor-

*corresponding author: Sungroh Yoon (sryoon@snu.ac.kr)

978-1-7281-1085-1/20/$31.00 ©2020 IEEE

mation transmission efficiency. Besides, we introduce an early firing method to further reduce inference latency. To validate the proposed approach, we conducted extensive experiments on the MNIST, CIFAR-10, and CIFAR-100, measuring the accuracy, number of spikes, and latency. According to the results, we were able to increase the accuracy as well as reduce latency and number of spikes, by applying the proposed methods to the T2FSNN. Our contributions can be summarized as follows:

- **T2FSNN:** We propose T2FSNN, which is an SNN model with a dynamic threshold and dendrite for TTFS coding to reduce the number of spikes and latency of inference.
- **Gradient-based optimization:** We propose a gradient-based optimization approach for improving the efficiency of the T2FSNN.
- **Early firing:** We propose an early firing method to further reduce the inference latency of T2FSNN.

## II. Background and Related Work

### A. Spiking Neural Networks

The main difference between SNNs and DNNs lies in how information is transmitted between neurons. SNNs transmit information through spike trains which contain binary spikes (discrete) rather real values (continuous). The spike train $S_i^l(t)$ of $i$th neuron in $l$th layer can be represented as

$$S_i^l(t) = \sum\nolimits_{t_i^{l,(f)} \in F_i^l} \delta(t - t_i^{l,(f)}), \tag{1}$$

where $\delta(t)$ is the Dirac delta function, $f$ is an index of spike, and $F_i^l$ is a set of spike times satisfying firing condition which is stated as:

$$t_i^{l,(f)} : u_i^l(t_i^{l,(f)}) \geq \theta_i^l(t_i^{l,(f)}), \tag{2}$$

where $u_i^l(t)$ is a membrane potential and $\theta_i^l(t)$ is a threshold at time $t$. Due to this information transmission based on the discrete spikes, SNNs have a feature of event-driven operation, which leads to improving computational energy efficiency. Please note that we will use $t_i^l$ instead of $t_i^{l,(f)}$ in the rest of this paper for simplicity because each neuron generates only one spike in TTFS coding.

In an integrate-and-fire (IF) neuron, which is one of the widely used neuron types in SNNs, a synaptic input is integrated into a membrane potential as follows:

$$u_j^l(t) = u_j^l(t-1) + z_j^l(t), \tag{3}$$

where $z_j^l$ is a sum of postsynaptic potential (PSP), which can be described as:

$$z_j^l(t) = \sum\nolimits_i w_{ij}^l d_j^l(t) S_i^{l\text{-}1}(t) + b_j^l, \tag{4}$$

where $w_{ij}^l$ is a synaptic weight, $d_j^l$ is a dendrite, and $b_j^l$ is a bias. Only the neurons satisfying the firing condition (Eq. 2) generate spikes, which increases the sparsity of the spikes.

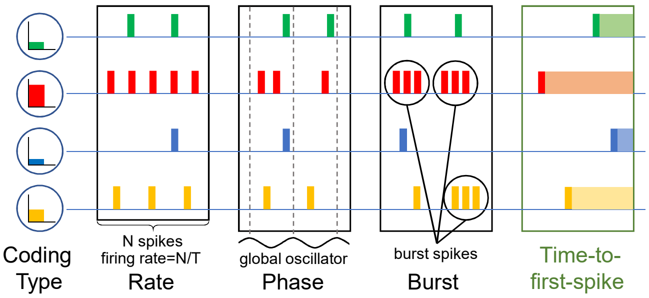


Fig. 1: Various neural coding methods

### B. Neural Coding

Neural coding is a method of representing information with spike trains, including encoding and decoding procedures. There have been four neural coding schemes in deep SNNs: rate [7], [8], [9], phase [11], burst [10], and TTFS [12], [13], as depicted in Fig. 1. Rate coding has the advantages of simple implementation and robustness to errors by using firing rate, $\frac{N}{T}$, where $N$ is the total number of spikes in a given time window $T$ [7], [8], [9], [14]. However, rate coding cannot utilize temporal information in spike trains and generates a large number of spikes, leading to high energy consumption and long inference latency.

Phase coding encodes temporal information into spike patterns based on a global oscillator [16], and it can significantly reduce the number of spikes in deep SNNs [11]. However, the efficiency cannot be guaranteed if the input changes dynamically and is unpredictable as hidden layers in deep SNNs [10]. Burst coding attempts to overcome this challenge by introducing burst spikes utilizing inter-spike interval [10]. Burst spikes can carry more information quickly and accurately by inducing PSP dramatically. Burst coding was able to significantly reduce the number of spikes and improve overall performance; however, it is still short of desired performance in terms of latency and efficiency.

TTFS coding has been adopted in deep SNNs to overcome such shortcomings [12], [13]. As illustrated in Fig. 2, the neurons with TTFS coding generate only one spike during inference and transmit the information using the timing of the spike. For instance, a red presynaptic neuron will be the first neuron that generates a spike in the fire phase to encode the largest amount of information (i.e., membrane potential). The postsynaptic neuron then integrates the PSP induced by the spike (red bar in $u^{l+1}$). Note that once a neuron generates a spike, it no longer generates additional spikes by applying a sufficiently long refractory period as described in [12].

The TDSNN [12] was able to reduce number of spikes substantially with the reverse coding, which is a kind of TTFS coding. Reverse coding, however, delivers larger values later, making it difficult to improve latency in deep SNNs. In addition, the auxiliary neurons, used to implement reverse coding, generate a large number of spikes, which deteriorates the improvement by TTFS coding. Thus, a new approach to thoroughly utilize the features of TTFS coding is needed for improving the inference of deep SNNs.

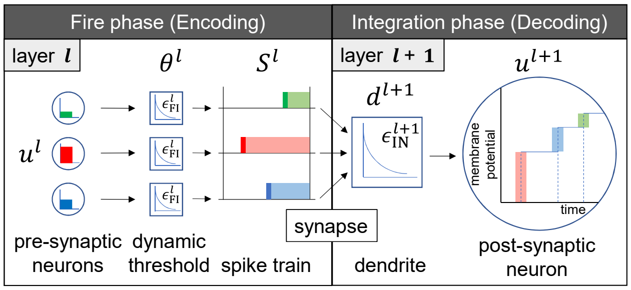


Fig. 2: Overview of T2FSNN: The height of each bars in the presynaptic neurons indicates the amount of integrated membrane potential (information being transmitted).

## III. Proposed Methods

In this paper, we propose an SNN model, called T2FSNN, that efficiently implements TTFS coding in deep SNNs. We adopted the DNN-to-SNN conversion method as described in [8], [9], [10], [11]. The T2FSNN exploits the concept of kernel-based dynamic threshold and dendrite for the encoding and decoding procedure, respectively. In addition, we propose a gradient-based optimization method of the kernels and early firing method for further increasing accuracy and reducing inference latency with the feature of TTFS coding.

### A. T2FSNN Model

The T2FSNN consists of IF neurons, synapses, and dendrites as depicted in Fig. 2. In the T2FSNN, each layer has two phases: one is an integration phase (decoding) and the other is a fire phase (encoding) in a time window $T$. In the integration phase, the neurons in the layer $l+1$ decode the spike trains generated by the presynaptic neurons in the layer $l$, then integrate the decoded information into their membrane potential $u^{l+1}$. After the integration phase, the integrated membrane potential of a neuron is encoded into a spike.

The TDSNN [12] employs TTFS coding which leads to a substantial reduction in number of spikes. Nonetheless, performance improvement of TDSNN deteriorated due to the overhead of the auxiliary neurons with a very high firing rate. In T2FSNN, to efficiently implement TTFS coding without any auxiliary neurons, we designed the encoding and decoding process based on the kernels in the threshold and dendrite, respectively. The kernels decrease monotonically as

$$\epsilon^l(t-t_{\text{ref}}^l) = \exp(-(t-t_{\text{ref}}^l-t_d^l)/\tau^l), \tag{5}$$

where $t_{\text{ref}}^l$ is a reference time that is defined as the start time of the fire phase, $t_{\text{d}}^l$ is a time delay, and $\tau^l$ is a time constant of layer $l$ ($t_{\text{d}}^l$ and $\tau^l$ are trainable parameters of each layer).

The encoding is a process of converting integrated information in the membrane potential $u^l$ into a spike time $t^l$ at the fire phase of each layer. We implemented TTFS encoding using a dynamic threshold $\theta^l(t)$ described as

$$\theta^l(t) = \theta_0 \epsilon_{\text{FI}}^l(t-t_{\text{ref}}^l), \tag{6}$$

where $\theta_0$ is a threshold constant and $\epsilon_{\text{FI}}^l$ is a fire kernel. With Eqs. 2 and 6, we were able to obtain the encoding function, which outputs a spike time $t^l$ as follows:

$$t^l = \lceil -\tau^l \ln(u_i^l(t_{\text{ref}}^l-1)/\theta_0) + t_{\text{d}}^l \rceil, \tag{7}$$

where $u_i^l(t_{\text{ref}}^l-1)$ is the integrated membrane potential of the neuron. We set the $\theta_0$ to one in this paper, because the range of integrated membrane potentials (activation values in DNNs) was limited [0, 1] by the data-based normalization [8], [10]. Because of the characteristic of the dynamic threshold, the larger amount of information integrated, the earlier the membrane potential exceeds the threshold, which means the neuron generates a spike earlier.

The decoding is a process of restoring the information encoded in a spike time at the integration phase. The decoded information $z_j^l(t)$ is accumulated in the form of the sum of PSP as follows:

$$z_j^l(t) = \sum\nolimits_i w_{ij}^l \epsilon_{\text{IN}}^l(t^{l-1}-t_{\text{ref}}^{l-1}) + b_j^l, \tag{8}$$

where $\epsilon_{\text{IN}}^l$ is an integration kernel. To make the decoding more accurate and effective, we set the time constant $\tau^l$ and time delay $t_{\text{d}}^l$ in the integration kernel $\epsilon_{\text{IN}}^l(t)$ to be equal to those in the fire kernel of the previous layer $\epsilon_{\text{FI}}^{l-1}(t)$.

### B. Gradient-Based Optimization Method

The key factors of T2FSNN are the integration and fire kernel in the dendrite and dynamic threshold, respectively. The kernels have a significant effect on the efficiency of inference in T2FSNN. The transmission error caused by the kernels mainly consists of two factors: precision error and small value encoding error. The precision error occurs because the precision of the information to be delivered is different from the precision that the kernels can represent. We can define the precision error as $|x-\hat{x}|$, where $x$ and $\hat{x}$ are the information before encoding and restored after decoding, respectively. With Eqs. 2, 5, and 7, we can obtain the precision error as $\hat{x}(\exp(1/\tau)-1)$. The precision error is inversely proportional to $\tau$, which means that the error can be reduced by increasing $\tau$.

However, because the encoding is based on the threshold operation, the information smaller than the smallest value $(\exp(-(T-t_{\text{d}})/\tau))$ that the kernel can express in a given time window $T$ cannot be transmitted. To prevent this loss in transmission of small values, $\tau$ is sufficiently small to represent the values. Thus, there is a trade-off between the precision and latency of information transmission between neurons, depending on the time constant $\tau$ in the kernel.

To address such a trade-off properly, we propose loss functions, including precision loss and representation loss, considering the accuracy and latency of the inference. In addition, we propose a gradient-based optimization, which is based on supervised learning and minimizes the loss functions in a layer-wise manner. We set ground truth of the supervised learning to $\bar{z}$, which is the value from DNN used in the DNN-to-SNN conversion. The precision loss is defined as

$$L_{\text{pre}}^l = \frac{1}{|F^l|} \sum\nolimits_{f\in F^l} \frac{1}{2}(\bar{z}_f^l - \hat{z}_f^l)^2, \tag{9}$$

where $F^l$ is a set of spike time of all neurons in layer $l$, $\bar{z}_f^l$, and $\hat{z}_f^l$ are the ground truth in DNN, and the decoded value in T2FSNN corresponds to the spike time $f$, respectively.

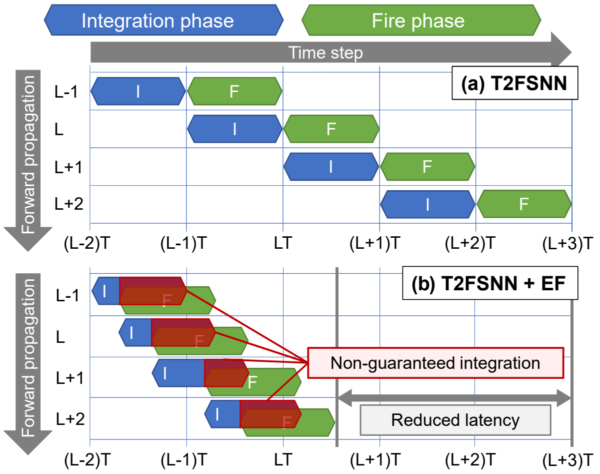


Fig. 3: Pipeline of the integration and fire phase

The representation loss consists of two terms; $L^l_{\text{min}}$ and $L^l_{\text{max}}$. These two loss terms consider the minimum and maximum representation values of each layer's kernel, so that the kernel can learn the distribution of ground truth $\bar{z}^l$. $L^l_{\text{min}}$ is defined as

$$L^l_{\text{min}} = \frac{1}{2}(\bar{z}^l_{\text{min}} - \hat{z}^l_{\text{min}})^2, \tag{10}$$

where $\bar{z}^l_{\text{min}}$ is the minimum value of $\bar{z}^l$, and $\hat{z}^l_{\text{min}}$ is the minimum value that the kernel can represent in a given time window $T$, which is $\exp(-(T - t^l_{\text{d}})/\tau^l)$. $L^l_{\text{max}}$ is defined as

$$L^l_{\text{max}} = \frac{1}{2}(\bar{z}^l_{\text{max}} - \hat{z}^l_{\text{max}})^2, \tag{11}$$

where $\bar{z}^l_{\text{max}}$ is the maximum value of $\bar{z}^l$, and $\hat{z}^l_{\text{max}}$ is the maximum value that the kernel can represent, which is $\exp(t^l_{\text{d}}/\tau^l)$.

To minimize the proposed loss functions, we use a gradient-based optimization method. The derivative of the precision loss with respect to $\tau^l$ is obtained as

$$\frac{\partial L^l_{\text{prec}}}{\partial \tau^l} = \frac{\partial L^l_{\text{prec}}}{\partial \hat{z}^l}\frac{\partial \hat{z}^l}{\partial \tau^l} = -\frac{1}{|F^l|}\sum\nolimits_{f\in F^l}\frac{t_f - t^l_{\text{d}}}{(\tau^l)^2}(\bar{z}^l_f - \hat{z}^l_f)\hat{z}^l_f \tag{12}$$

and the derivative of minimum representation loss with respect to $\tau^l$ is stated as

$$\frac{\partial L^l_{\text{min}}}{\partial \tau^l} = \frac{\partial L^l_{\text{min}}}{\partial \hat{z}^l_{\text{min}}}\frac{\partial \hat{z}^l_{\text{min}}}{\partial \tau^l} = -\frac{T - t^l_{\text{d}}}{(\tau^l)^2}(z^l_{\text{min}} - \hat{z}^l_{\text{min}})\hat{z}^l_{\text{min}}. \tag{13}$$

Because the maximum representation is most affected by $t_{\text{d}}$, maximum representation loss is differentiated with respect to $t_{\text{d}}$ as follows:

$$\frac{\partial L^l_{\text{max}}}{\partial t^l_{\text{d}}} = \frac{\partial L^l_{\text{max}}}{\partial \hat{z}^l_{\text{max}}}\frac{\partial \hat{z}^l_{\text{max}}}{\partial t^l_{\text{d}}} = -\frac{1}{\tau^l}(z^l_{\text{max}} - \hat{z}^l_{\text{max}})\hat{z}^l_{\text{max}}. \tag{14}$$

From the equations above, we can calculate the gradient of $\tau^l$ and $t^l_{\text{d}}$, and optimize the kernel function with those gradients.

### C. Early Firing Method

The integration and fire phase of T2FSNN are operated with dependencies in order, as shown in Fig. 3. In each layer of this baseline pipeline (Fig. 3-(a), T2FSNN), the integration and fire phases in a layer are executed sequentially. The integration phase of layer $l$ is executed simultaneously with the fire phase of the previous layer $l-1$ from $(L-1)T$ to $LT$ time step. After the integration is finished, the fire phase begins at $LT$ time step, generating spikes based on the integrated information. This baseline pipeline can guarantee the integration of all information from the previous layer in a given time window $T$ before encoding, which leads to accurate information transmission to the subsequent layer. However, the dependency between integration and fire phase considerably increases inference latency in deep SNNs with many layers.

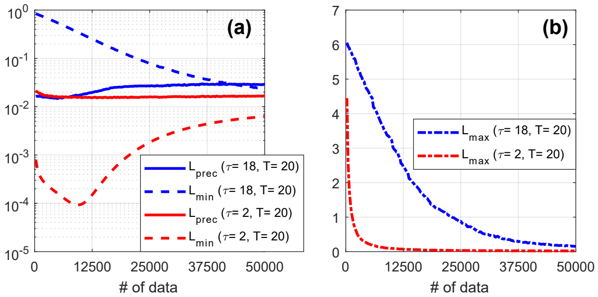


Fig. 4: Loss graphs of the proposed gradient-based optimization method: (a) $L_{\text{prec}}$ and $L_{\text{min}}$, (b) $L_{\text{max}}$

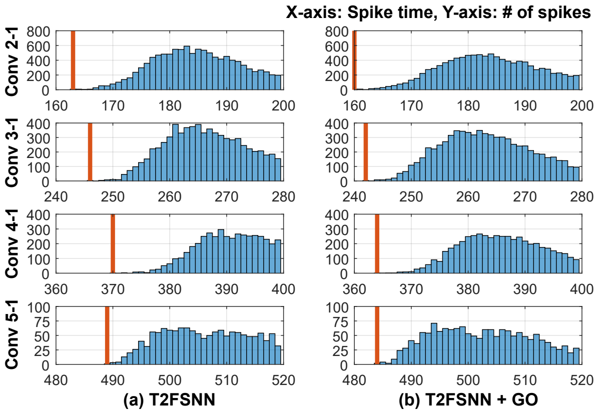
To alleviate this issue, we propose a technique called early firing, which is a method of starting the firing phase before the integration is complete at each layer. By using this method, we can overlap the integration and fire phase, which results in reducing inference latency as shown in Fig. 3-(b) (T2FSNN+EF). However, this approach causes non-guaranteed integration, where the information may not contribute to generating a spike. The information accumulated in the neurons that have already fired during the non-guaranteed integration cannot affect the spike generation, because each neuron generates at most one spike in the T2FSNN. Thus, the starting time of the early firing should be set to ensure the guaranteed integration of critical information to reduce inference latency without significant loss of accuracy.

## IV. Experimental Results

We empirically set the $\tau$, $t_{\text{d}}$, and $T$ at the initial stage. Based on the initial configurations of T2FSNN, we applied the proposed gradient-based optimization and early firing. We used a train dataset and mini-batch SGD for the optimization. For the early firing, we set the starting time of the early firing to half of the time window $T$ based on the experiments. We assessed the T2FSNN, including the proposed methods, and compared the other neural coding methods on various datasets.

### A. Evaluation of the Proposed Methods

To evaluate the proposed optimization method, we measured the three loss terms ($L_{\text{prec}}$, $L_{\text{min}}$, and $L_{\text{max}}$). We set two different initial conditions to validate the trade-off between precision and latency of information transmission depending on the $\tau$: 1) a small time constant ($\tau$=2), and 2) a large time constant ($\tau$=18) on a given time window ($T$=20). When the time constant is a small value ($\tau$=2), the kernel $\epsilon$ can represent small values sufficiently, but the precision of transmission is low. Thus, as the training progresses, the time constant $\tau$ increased and the precision loss $L_{\text{prec}}$ decreased as shown in Fig. 4-(a) (red solid line). In contrast, if the time constant



(a) T2FSNN (b) T2FSNN + GO

Fig. 5: Distributions of spike time: Vertical orange bars represent the first spike time of each layer (VGG-16, CIFAR-10).

TABLE I: Ablation study

| Methods | Latency | CIFAR-10 | | CIFAR-100 | |
|---|---|---|---|---|---|
| | | Accuracy | Spikes | Accuracy | Spikes |
| T2FSNN | 1280 | 91.36 | 6.898E+4 | 66.04 | 8.626E+4 |
| T2FSNN+GO[a] | 1280 | 91.37 | 6.887E+4 | 66.97 | 8.464E+4 |
| T2FSNN+EF[a] | 680 | 91.37 | 6.893E+4 | 68.09 | 8.603E+4 |
| **T2FSNN+GO+EF**[a] | **680** | **91.43** | **6.881E+4** | **68.79** | **8.444E+4** |

[a] GO: Gradient-based Optimization, EF: Early Firing

is a large value ($\tau$=18), the proposed method will train $\tau$ to minimize the minimum representation loss $L_{\text{min}}$ as seen in the blue dashed line on Fig. 4-(a), which results in decreasing $\tau$.

The maximum representation loss is depicted in Fig. 4-(b). In the case of a small time constant ($\tau$=2), the maximum representation loss $L_{\text{max}}$ decreases rapidly because the first spike of each layer occurs at an earlier time in each layer's fire phase. Through this experiment, we can verify that $L_{\text{prec}}$ and $L_{\text{min}}$ are trained competitively in a given time window $T$, and $L_{\text{min}}$ has a greater impact than $L_{\text{prec}}$. We can also validate the effect of the gradient-based optimization by the distribution of spike time in each layer as depicted in Fig. 5. Compared to the T2FSNN (Fig. 5-(a)), the optimized T2FSNN (T2FSNN+GO, Fig. 5-(b)) can shorten the first spike time of each layer (vertical orange bar), and reduce the number of spikes. This indicates that the T2FSNN+GO enables the efficient inference that is fast and requires less computation.

We conducted an ablation study to demonstrate individual effects of the proposed methods as described in Table I. When the gradient-based optimization was applied to the T2FSNN (T2FSNN+GO), accuracy on CIFAR-10 and CIFAR-100 improved by approximately 0.01% and 0.93% compared to the T2FSNN, respectively. In addition, the number of spikes decreased by 0.2% and 1.9%. When the early firing method was applied to the T2FSNN (T2FSNN+EF), inference latency decreased by 46.9%, while accuracy increased by 2.05% on CIFAR-100. Remarkably, the T2FSNN+GO+EF achieved 46.9% reduction in inference latency while improving accuracy of 0.07% and 2.75% on CIFAR-10 and CIFAR-100, respectively. Furthermore, the number of spikes also decreased by 0.3% and 2.1%. It is interesting to note that the accuracy was improved with a lower number of spikes, despite that early firing causes non-guaranteed integration. This result can be interpreted as a generalization effect caused by the stochastic property of non-guaranteed integration that contributes to the generation of a spike depending on the state of the neuron.

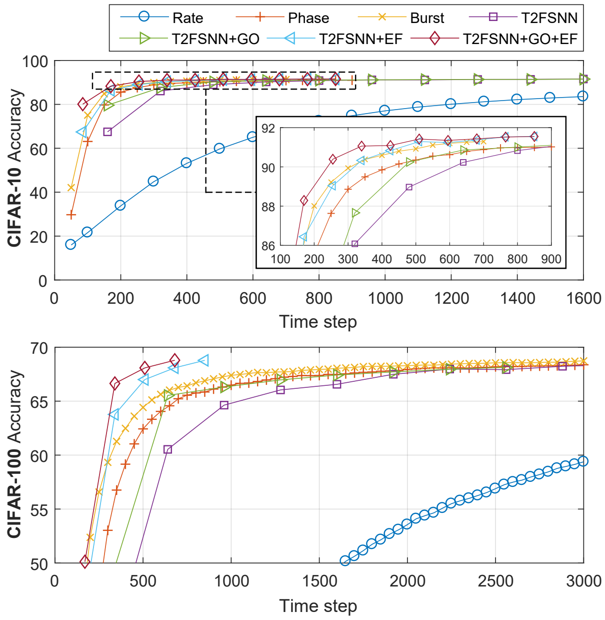


Fig. 6: Inference curve of various neural coding methods for VGG-16 on the CIFAR-10 (top) and CIFAR-100 (bottom)

## B. Comparison with Other Methods

We compared our proposed approach with the other neural coding methods, including rate [7], [8], phase [11], burst coding [10]. To assess inference speed, we measured the accuracy depending on the inference time as depicted in Fig. 6. In CIFAR-10, the order of fast-to-slow inference speed is burst, phase, T2FSNN, and rate coding. When we applied the proposed optimization or early firing to the T2FSNN (T2FSNN+GO, T2FSNN+EF), the inference speed of T2FSNN+GO and T2FSNN+EF was as fast as that of phase and burst coding, respectively. In the case of T2FSNN with the optimization and early firing, T2FSNN+GO+EF showed the fastest inference speed. We observed similar tendency in the experimental results on CIFAR-100. T2FSNN+EF showed faster inference speed than burst coding, and T2FSNN+GO+EF achieved the fastest inference speed.

Table II summarizes the accuracy, latency, number of spikes, and estimated energy consumption of various neural coding on MNIST, CIFAR-10, and CIFAR-100. Our method (T2FSNN+GO+EF) achieved the best accuracy with the lowest number of spikes in all datasets. Particularly, in CIFAR-100, we were able to reduce the number of spikes to less than 1% of that of burst coding, given the fact that the T2FSNN generates at most one spike per neuron. Inference latency also decreased to 22% compared to that of burst coding.

The energy consumption takes both latency and number of spikes into account, and thus it is an important metric for the efficiency of deep SNNs [10]. We estimated the

TABLE II: Comparison with other DNN-to-SNN conversion methods with various neural coding schemes

| Neural Coding | Accuracy (%) | Latency (time step) | Spikes ($10^6$) | Normalized Energy TN [18] | SN [19] |
|---|---|---|---|---|---|
| MNIST | | | | | |
| Rate [7], [8] | 99.10 | 200 | 0.100 | 1.000 | 1.000 |
| Phase [11] | 99.20 | **16** | 3.000 | 12.048 | 19.228 |
| Burst [10] | 99.25 | 87 | 0.251 | 1.265 | 1.763 |
| Reverse [12] | 99.08 | - | - | - | - |
| **Our Method** | **99.33** | 40 | **0.002** | **0.128** | **0.085** |
| CIFAR-10 | | | | | |
| Rate [7], [8] | 91.14 | 10,000 | 61.949 | 1.000 | 1.000 |
| Phase [11] | 91.21 | 1,500 | 35.196 | 0.317 | 0.418 |
| Burst [10] | 91.41 | 1,125 | 6.920 | 0.112 | 0.112 |
| **Our Method** | **91.43** | **680** | **0.069** | **0.041** | **0.025** |
| CIFAR-100 | | | | | |
| Rate [7], [8] | 66.50 | 10,000 | 81.525 | 1.000 | 1.000 |
| Phase [11] | 68.66 | 8,950 | 258.408 | 1.805 | 2.351 |
| Burst [10] | 68.77 | 3,100 | 25.074 | 0.309 | 0.308 |
| **Our Method** | **68.79** | **680** | **0.084** | **0.041** | **0.025** |

energy consumption based on measured latency and spikes with dynamic and static energy consumption data from neuromorphic architecture, TrueNorth (TN) [18] and SpiNNaker (SN) [19] as in [10]. The estimated energy is defined as (# of spikes)$E_{\text{dyn}}$ + (latency)$E_{\text{sta}}$, where $E_{\text{dyn}}$ and $E_{\text{sta}}$ are dynamic and static energy parameters depeneding on neuromorphic architecture. The energy parameters ($E_{\text{dyn}}$, $E_{\text{sta}}$) are set to (0.4, 0.6) and (0.64, 0.36) for TrueNorth and SpiNNaker, respectively. According to the estimation, our method was able to reduce energy consumption to about 6% and 16% on average compared to rate and burst coding, respectively.

## V. Discussion

The TDSNN (reverse coding), which is a previous study in the line of our work, did not report the number of spikes and latency [12]. Instead of direct comparison to TDSNN, we compared the estimated computational cost as stated in Table III. The rate coding only requires accumulation of incoming spikes. The phase and burst coding, on the other hand, need additional non-linear functions during encoding and decoding procedure. Such extra overheads are alleviated by using a lookup table due to the limited input range of the non-linear function. Thus, these neural coding methods require multiplication and addition operations proportional to the number of spikes.

The TDSNN used auxiliary neurons called ticking neurons for implementing reverse coding. The ticking neurons generate spikes frequently, which causes numerous accumulation operations. In addition, the TDSNN adopted leaky IF neurons, which require an exponential operation at every time step. Thus, required computations are proportional to the time step and number of neurons. The estimated computational cost based on the reported data [12] is stated in Table III. In contrast, the proposed T2FSNN does not require auxiliary neurons, the computational cost of kernel function in T2FSNN can be reduced by replacing the kernel with a lookup table. According to our analysis, the T2FSNN requires significantly lower operations compared to various neural coding schemes including TDSNN.

TABLE III: Analysis of computational cost (million operations, VGG-16 on CIFAR-100)

| | DNN | Rate [7], [8] | Phase [11] | Burst [10] | TDSNN [12] | **T2FSNN** |
|---|---|---|---|---|---|---|
| Mult | 146.50 | - | 258.408 | 25.074 | 14.84 | **0.084** |
| Add | 146.50 | 81.525 | 258.408 | 25.074 | 154.21 | **0.084** |

## VI. Conclusion

In this paper, we proposed the T2FSNN with gradient-based optimization and early firing for utilizing the TTFS coding in deep SNNs. By the extensive experiments on various tasks, we demonstrated our methods improve the inference efficiency of deep SNNs in terms of both latency and number of spikes. We expect that our methods pave the way for energy-efficient inference with deep SNNs with temporal coding.

## Acknowledgments

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) [2016M3A7B4911115, 2018R1A2B3001628], the Brain Korea 21 Plus Project in 2019, AIR Lab (AI Research Lab) in Hyundai Motor Company through HMC-SNU AI Consortium Fund, and Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT1901-12.

## References

[1] S. Park *et al.*, "Quantized memory-augmented neural networks," in *AAAI*, 2018.

[2] S. Han *et al.*, "Learning both weights and connections for efficient neural network," in *NIPS*, 2015.

[3] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019.

[4] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[5] Y. Jin *et al.*, "Hybrid macro/micro level backpropagation for training deep spiking neural networks," in *NIPS*, 2018.

[6] Y. Wu *et al.*, "Direct training for spiking neural networks: Faster, larger, better," in *AAAI*, 2019.

[7] P. U. Diehl *et al.*, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *IJCNN*, 2015.

[8] B. Rueckauer *et al.*, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, p. 682, 2017.

[9] S. Kim *et al.*, "Spiking-yolo: Spiking neural network for real-time object detection," in *AAAI*, 2020.

[10] S. Park *et al.*, "Fast and efficient information transmission with burst spikes in deep spiking neural networks," in *DAC*, 2019.

[11] J. Kim *et al.*, "Deep neural networks with weighted spikes," *Neurocomputing*, vol. 311, pp. 373–386, 2018.

[12] L. Zhang *et al.*, "Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding," in *AAAI*, 2019.

[13] B. Rueckauer and S. Liu, "Conversion of analog to spiking neural networks using sparse temporal coding," in *ISCAS*, 2018.

[14] E. D. Adrian, "The impulses produced by sensory nerve endings," *The Journal of Physiology*, vol. 61, no. 1, pp. 49–72, 1926.

[15] D. A. Butts *et al.*, "Temporal precision in the neural code and the timescales of natural vision," *Nature*, vol. 449, no. 7158, p. 92, 2007.

[16] M. A. Montemurro *et al.*, "Phase-of-firing coding of natural visual stimuli in primary visual cortex," *Current Biology*, vol. 18, no. 5, pp. 375–380, 2008.

[17] S. Thorpe *et al.*, "Spike-based strategies for rapid processing," *Neural Networks*, vol. 14, no. 6-7, pp. 715–725, 2001.

[18] P. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[19] S. B. Furber *et al.*, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.