Contents lists available at

The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)

TDSNN: From Deep Neural Networks to Deep Spike Neural Networks with Temporal-Coding

Lei Zhang,[1,2,3] **Shengyuan Zhou,**[1,2,3] **Tian Zhi,**[1,3] **Zidong Du,**[1,3] **Yunji Chen***[1,2]

[1]Institute of Computing Technology, Chinese Academy of Sciences

[2]University of Chinese Academy of Sciences

[3]Cambricon Tech. Ltd

{zhanglei03, zhousy, zhitian, duzidong, cyj}@ict.ac.cn

Abstract

Continuous-valued deep convolutional networks (DNNs) can be converted into accurate rate-coding based spike neural networks (SNNs). However, the substantial computational and energy costs, which is caused by multiple spikes, limit their use in mobile and embedded applications. And recent works have shown that the newly emerged temporal-coding based SNNs converted from DNNs can reduce the computational load effectively. In this paper, we propose a novel method ...s, called *TDSNN*. ...aky integrate-and-...iew coding princi-...*ing Neuron* mech-...proposed method ...n average in large ...ore than 0.5% ac-...DSNN may prove ...adoption of SNNs ...ation of neural net-...ssing of both spa-...97). However, it is ...ies of works have ...o fit for the SNN ...ee, Delbruck, and ...atisfactory results in simple tasks like MNIST (Lecun et al. 1998), the training methods can hardly be scaled into deep SNN models to solve more complex tasks like ImageNet (Russakovsky et al. 2014). Meanwhile, the researches on the biological training methods also meet the same problem. These works have put many existing brain mechanisms into the modeling of SNN including spike timing dependent plasticity (STDP) and long-term potentiation or depression, short-term facilitation or depression, hetero-synaptic plasticity, etc. Recently, Zhang et al. (Zhang et al. 2018) proposed a novel multi-layer SNN model which applies biological mechanisms and achieves 98.52% on MNIST dataset, but its performance is unknown when applied on larger datasets.

*Yunji Chen (cyj@ict.ac.cn) is the corresponding author.

Unlike SNNs, deep neural networks (DNNs) have been able to perform the state-of-the-art results on many complex tasks such as image recognition (Krizhevsky, Sutskever, and Hinton 2012; Krizhevsky 2009; Simonyan and Zisserman 2014; He et al. 2015), speech recognition (Abdel-Hamid et al. 2012; Sainath et al. 2013; Hinton et al. 2012), natural language processing (Kim 2014; Severyn and Moschitti 2015) and so on. But heavy computation load promotes researchers to find more efficient approach to deploy them in mobiles or embedded systems. This inspires the SNN researchers that a fully-trained DNN might be slightly tuned to be directly converted to a SNN without complicated training procedure. Beginning with the work of (Perezcarrasco et al. 2013), where DNN units were translated into biologically inspired spiking units with leaks and refractory periods, continuous efforts have been made to realize this idea. After a series of success in transferring deep networks like Lenet and VGG-16 (Cao, Chen, and Khosla 2015; Diehl et al. 2015; Rueckauer et al. 2017), now the rate-coding based SNN can achieve state-of-the-art performance with minor accuracy loss even in the conversion of complicated layers like Max-Pool, BatchNorm and SoftMax.

However, weak points in rate-coding based SNN are obvious. Firstly, the rate-coding based SNN could become more accurate with the increasement of the simulation duration and the average firing rate of its neurons. But it may lose potential performance advantage over the DNNs as the firing rates increase (Rueckauer and Liu 2018). Secondly, part of the accuracy loss in the conversion occurs at the parameter determination procedure. In rate-coding based SNN, determination of important parameters like firing threshold seriously affect the final accuracy while non deterministic methods have been proposed to eliminate the loss.

This poses challenges to researchers to find conversion methods based on another coding scheme—temporal coding. Neurons based on temporal-coding make full use of the spike time to complete the transmission of information, and the number of spikes is significantly reduced. Also, temporal coding has been proved to be efficient in computing even in biological brains (Van and Thorpe 2001). The temporal-coding based neurons apply a so-called time-to-first-spike (TTFS) scheme and each neuron fires at most once during the forward inference (Thorpe, Delorme, and Rullen 2001). Obviously, temporal-coding based SNN is more com-

putationally efficient than rate-coding based SNN. Recently Rueckauer and Liu (Rueckauer and Liu 2018) proposed a conversion method using sparse temporal coding, but it is only suitable for shallow neural network models.

In this paper, we put forward a novel conversion method called *TDSNN*, transferring DNNs to temporal-coding SNNs with only trivial accuracy loss. We also propose a new encoding scheme *Reverse Coding* along with a novel *Ticking Neuron* mechanism. We address three challenges during the process of converting as follows. Firstly, we maintain the coding consistency in the entire network after mapping. Secondly, we eliminate the conversion error from DNNs to SNNs. Thirdly, we maintain accuracy in SNN with spike-once neurons. The details will be discussed later in the *Reverse Coding* part. According to our evaluation, we achieve 42% total operations reduction on average in large networks comparing with DNNs with no more than 0.5% accuracy loss. The experimental results show that our method is an efficient and high-performance conversion method for temporal-coding based SNN.

Background

In this section, we will introduce two common neural models. One is integrate-and-fire(IF) model, the other is leaky integrate-and-fire (LIF) model.

Integrate-and-fire (IF) model

We introduce integrate-and-fire(IF) model at first, which is widely used in previous rate-coding based conversion approaches. In IF models, each neuron will accumulate potential from input current and fire a spike when its potential reaches the threshold. Although successful conversions have been made from DNN neurons with ReLU activations to IF neurons, the process of the transformation still exists unsatisfactory problems. For example, it is difficult to accurately determine thresholds in these conversion methods. Although Diehl et al. (Diehl et al. 2015) has proposed data-based and model-based threshold determination methods, the converted SNN still suffers unstable accuracy loss comparing with the re-trained DNN model. What's worse, IF neurons implement no time-dependent memory. Once a neuron receives a below-threshold action potential at some time, it will keep retaining that voltage until it fires. Obviously, it is not in line with the neural behavior in neural science.

Leaky Integrate-and-fire (LIF) model

To better model neural behavior in the spike-threshold model, researchers in neural science field proposed a simplified model—Leaky Integrate-and-fire (LIF) model (Koch and Segev 1998), which is derived from the famous Hodgkin–Huxley model (Hodgkin and Huxley 1990). Generally, it is defined as following:

$$I(t) - \frac{V(t)}{R} = C \cdot \frac{dV(t)}{dt}, \tag{1}$$

where R is a leaky constant, C is the membrane capacitance, $V(t)$ is the membrane potential and $I(t)$ is considered as the input stimulus.

Considering that the inputs are instantaneous currents that are generated from discrete-time neural spikes in an SNN model, the equation (1) turns to:

$$\frac{V(t)}{L} + \frac{dV(t)}{dt} = \sum_i \omega_i \cdot I_i(t). \tag{2}$$

where ω_i denote the synapse strength of the connections in SNN, L is the leak-related constant. If there exists no continuous spikes, the neuron potential will decay as time goes by. The LIF model processes the time information by adding a leaky term, reflecting the diffusion of ions that occurs through the membrane when some equilibrium is not reached in the cell. If no spike occurs during the time interval from t_1 to t_2, the potential of the LIF neuron will change as following:

$$V(t_2) = V(t_1) \cdot exp(-\frac{t_2 - t_1}{L}). \tag{3}$$

And if no no spike occurs in this neuron, the final potential accumulation P at time T_{il} will be:

$$P(T_{il}) = \sum_i \omega_i \cdot exp(-\frac{T_{il} - t_i}{L}). \tag{4}$$

This implies that the pre-synaptic neuron's potential contribution to post-synaptic neurons is positively correlated with the firing time of pre-synaptic spike. This nonlinear characteristic has never been put good use in previous conversion methods. We make full use of these features to build a conversion method for temporal-coding SNN.

Theoretical Analysis

In this section, we present details of our conversion method. We first introduce the *Reverse Coding* guideline and *ticking neuron* mechanism, respectively. Then, we analyze theoretically on all the requirements of conversion method, including temporal-coding function and important parameters.

Reverse Coding

There has been massive number of influential works in the practice of encoding input into a single spike before our work. For example, Van et al. (Van and Thorpe 2001) proposed a rank order based encoding scheme, in which the inputs are encoded into specific spiking order. The potential contribution of later-spike neurons will be punished with a delay factor so that the ones fire earlier will contribute a major part of potential accumulation to post-synaptic neurons. Combined with LIF neurons, this coding method has been proven to achieve a Gaussian-difference-filtering effect, which is helpful for extracting features. But these guidelines failed to serve well when considering the conversion of DNN to SNN.

Unlike these works in which significant inputs are encoded into earlier spike times, we follow an opposite coding guideline based on characteristics of LIF neurons and we name it *Reverse Coding*:

The stronger the input stimulus is, the later the corresponding neuron fires a spike.

This is consistent with the characteristics of leaky-IF neurons, where post-synaptic neurons will assert most impressive contribution to the recent-spike neurons.

Determining the principle of coding schemes moves one step closer to build a conversion method for SNN with temporal-coding. Following *Reverse Coding*, we still need to tackle the following challenges to build a conversion method for SNN with temporal-coding:

- **How to maintain coding consistency in the entire network after mapping.** Even if the input of the first layer is encoded according to *Reverse Coding*, there is no guarantee that the neurons in the subsequent layers will also spike exactly in the same way.
- **How to eliminate the conversion error from DNNs to SNNs.** A bunch of parameters(threshold, spike frequency, presentation time etc.) need to be determined in previous rate-coding based conversion methods and it still lacks strict theoretical evidence to minimize the error caused by parameter selection.
- **How to maintain accuracy in SNN with spike-once neurons.** This requires the converted SNN makes full use of the nonlinear characteristics of neuron model and also the adjustments in DNN should be well designed.

Next, we will tackle these challenges by introducing the *Ticking Neuron* mechanism below.

Ticking Neuron Mechanism

Converting the weights of DNN into that of SNN directly is the most straightforward way, which also occurred in previous rate-coding based conversion methods. However, since none inhibitory mechanisms are applied, the post-synaptic neurons would spike at any time once their potentials exceed the threshold. In such case, it would be difficult to control the spiking moment and spike times along with the appropriate threshold. What's worse, those neurons in SNN, whose corresponding neuron output in DNN is large, may issue spikes at an earlier time. It violates the *Reverse Coding* principle we introduced above.

To address the challenges above, we propose an auxiliary neuron called *ticking neuron* and a corresponding spike processing mechanism. As it shows in figure 1, the post-synaptic neurons will be inhibited to fire once the process begins, reflecting a common refractory period in SNN. The inhibition will last until time T_{il}(T_{il} must be large enough to cover all the pre-synaptic spikes, at least it could be the firing time of the neuron firing the last spike). During that time interval, each of the pre-synaptic neurons will only fire one spike following *Reverse Coding*. The *ticking neuron* will record the time T_{il} and update its connections' weight ω_{tick} to $f(T_{il})$ (which means ω_{tick} is only dependent on T_{il} and if T_{il} is a pre-determined constant it could also be pre-determined). After that, *ticking neuron* begins to issue spikes at each time step until each of the post-synaptic neurons has fired one spike. Those neurons has already fired one spike will be inhibited to fire(could be considered as being in a very long self-inhibitory period).

To make the LIF neuron whose corresponding neuron in DNN outputs a large value fire at a later time point, the original weights in DNN layers need to be negated. According to equation (4), for those LIF neurons whose corresponding output value in DNN is negative, they will spike immediately once the inhibition is released. While for those post-synaptic neurons whose corresponding output in DNN is positive, their potential will be at a negative point when all the pre-synaptic neurons fires. After that, the *ticking neuron* begins to fire, increasing their potential until reach the given threshold. Clearly, the larger the output in DNN, the later the corresponding neurons in SNN will fire a spike.

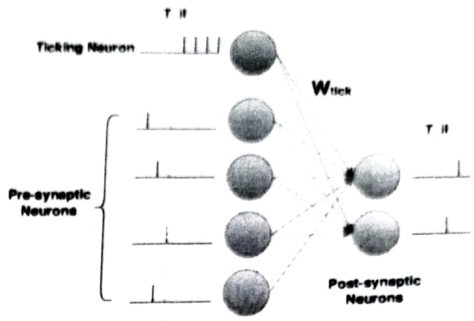

Figure 1: A Layer with a ticking neuron in SNN.

Why *ticking neuron* is necessary? Because it guarantees that all of the neurons can fire spikes. Supposing that if we remove the neuron, those post-synaptic neurons with a negative potential will slowly approach the reset potential. It will never fire a spike unless a threshold below reset potential is set for it, which is not consistent with existing SNN findings and neural science evidence.

Now all that is required is the strict calculations of spike timing to achieve a lossless conversion.

Mapping Synapse-based Layer

Synapse-based layers refer to those layers with connections of exact weight in the DNN, including convolutional layers (*CONV*), inner-product layers (*IP*), pooling layers (*POOL*), etc. These layers are the fundamental layers of DNN and also the most computationally intensive layers. We will show how to convert these layers to corresponding layers in SNN applied with mechanisms mentioned above.

Supposing that the firing threshold of post-synaptic neuron is θ, the last firing time of pre-synaptic neurons is T_{il} and the current below threshold potential is P. Clearly, the spike timing T_o (considering the output layer's inhibition is released at time 0) of the post-synaptic neuron is the minimum positive integer that obeys the following equation:

$$\sum_{t=0}^{T_o} \omega_{\text{tick}} \cdot \exp(-\frac{T_o - t}{L}) + P \cdot \exp(-\frac{T_o}{L}) \geq \theta, \qquad (5)$$

where L is the leaky constant described before. Solving this equation, we obtain:

$$T_o = \lceil L \cdot \ln(\frac{(-P) \cdot (1 - \exp(-1/L)) + \omega_{\text{tick}}}{\omega_{\text{tick}} - \theta \cdot (1 - \exp(-1/L))}) \rceil. \qquad (6)$$

In order to reduce the number of parameters to be determined, the above equation can be simplified by setting $\theta = 0$. Finally, the equation (6) turns out to be:

$$T_o = \lceil L \cdot \ln(\frac{(-P) \cdot (1 - \exp(-1/L))}{\omega_{tick}} + 1) \rceil \quad (7)$$

Note that the output positive value of DNN neurons A will be mapped into a much smaller negative potential P due to the leaky effect of leaky-IF neurons, which is:

$$-P = A \cdot \exp(-\frac{T_{il}}{L}). \quad (8)$$

Combining equation (7) with equation (8), we will get the connection weight of ticking neuron is as following:

$$\omega_{tick} = (1 - \exp(-\frac{1}{L})) \cdot \exp(-\frac{T_{il}}{L}). \quad (9)$$

This is in line with the constraints we mentioned in the previous section that ω_{tick} is only dependent on T_{il}. If T_{il} is a predetermined large constant, ω_{tick} becomes a constant too. If T_{il} is set as the last firing time of pre-synaptic neurons, then ω_{tick} dynamically updated. Also, the temporal-coding function $T(x)$ will be obtained by combining equation (7)(8)(9):

$$T(x) = \lceil L \cdot \ln(x + 1) \rceil. \quad (10)$$

Correspondingly, a new activate function is needed to be deployed right before these synapse-based layers:

$$G(x) = \begin{cases} \exp(\frac{1}{L} \cdot \lceil L \cdot \ln(x+1) \rceil) & x \geq 0, \\ 1 & x < 0. \end{cases} \quad (11)$$

Note that negative input stimulus is avoided in our work just like other conversion methods did, as *negative neurons* are neither necessary nor in line with existing neural science.

Bias term was explicitly excluded in most of the rate-coding based conversion methods. Rueckauer et al. (Rueckauer et al. 2017) proposed the bias with an external spike input of constant rate proportional to the DNN bias. In temporal-coding SNN, determining the firing timing for bias neuron is easy. Here the bias neuron is considered as an input with a numerical value of 1, thus it will be encoded into a spike timing according to equation (10). The connection weights of bias neurons will also be negated after re-training of DNN.

Applied the mechanisms and coding functions above, the synapse-based layers in DNN can be accurately mapped to spike-time-based ones in SNN.

Mapping Max-Pool

Max-Pool layer is a commonly used down-sampling layer in DNN, replacing it with *Average-Pool* will inevitably decrease the DNN accuracy. But it is non-trivial to compute maxima with spiking neurons in SNN. In the rate-coding based SNN (Rueckauer et al. 2017), the authors proposed a simple mechanism for spiking max-pooling, in which output units contain gating functions that only let spikes from the maximally firing neuron pass, while discarding spikes from other neurons. Their methods prove to be efficient in rate-coding based SNN but can not be applied here in temporal-coding SNN.

Here we show how the lossless mapping of the $Max\text{-}Pool$ layer can be done assuming that the weight of the ticking neuron ω_{tick} and the weights $-\omega_k$ in the SNN max-pool kernel (the weights in this layer are also negative). Supposing that the size of the pooling kernel is N (usually equals $KX \cdot KY$, which stands for the kernel width in x and y direction respectively), firing threshold of post-synaptic neurons is 0, and the weights of the mapped kernel are equally distributed.

To determine the weights of ω_{tick} and ω_k, two extreme scenarios should be considered. One is that post-synaptic neurons will accumulate a strong negative potential if all the pre-synaptic neurons in the kernel fire at time T_m ($T_m \leq T_{il}$). The other is that post-synaptic neurons will accumulate a weaker negative potential if there is only one pre-synaptic neurons fires at time T_m. It must be guaranteed that in both extreme cases, the spike timing T_o of a post-synaptic neuron equals T_m.

Taking the potential firing time of post-synaptic neurons in both cases into the equation (7), we obtain the equations 12 and derive the constraints shown as 13, where $C_L = (1 - \exp(-1/L))$ is a constant.

$$\begin{cases} L \cdot \ln(\frac{\omega_k \cdot (1 + (N-1) \cdot \exp(-\frac{T_m}{L}))}{\omega_{tick}/C_L} + 1) > T_m - 1 \\ L \cdot \ln(\frac{N \cdot \omega_k}{\omega_{tick}} \cdot C_L + 1) < T_m \end{cases} \quad (12)$$

$$\begin{cases} \frac{\omega_{tick}}{\omega_k} > \frac{C_L \cdot N}{\exp(\frac{T_m}{L}) - 1} \\ \frac{\omega_{tick}}{\omega_k} < \frac{C_L \cdot (1 + (N-1) \cdot \exp(-\frac{T_m}{L}))}{\exp(\frac{T_m - 1}{L}) - 1} \end{cases} \quad (13)$$

Notice that $\frac{\omega_{tick}}{\omega_k}$ is only dependent on T_m. If $T_m = 1$, only the left side of the above inequality is needed. Considering that the inequality holds if the left side is smaller than the right side, the constraints on L when $T_m > 1$ is obtained:

$$\frac{N}{\exp(T_m/L - 1)} < \frac{1 + (N-1) \cdot \exp(-T_m/L)}{\exp((T_m - 1)/L) - 1} \quad (14)$$

Because ω_k, L and N are predetermined, ω_{tick} is only dependent on the presentation time T_{il} of pre-synaptic neurons. As a result, the lossless mapping of the *Max-Pool* layer can be done by selecting parameters according to equations (13)(14).

However solving such constraints for various N and L would be complicated and the existence of solutions needs to be determined under many circumstances. Here we present a simplified solution for mapping *Max-Pool* layer under our mechanism. By setting the firing threshold $\theta = N$, $\omega_k = 1$ and canceling the use of ticking neuron and leaky effect, the firing moment of the output neuron is equal to the spiking moment of the latest firing neuron in the input kernel. In this way, *Max-Pool* is realized in SNN with *Reverse Coding*.