Building Efficient Event-driven Networks from Frame-driven Networks with FIT

Yang Zhao, Haibo Wang, Yang Yang*
Computational Aerodynamic Institute China Aerodynamic Research and Development Center (CARDC) Mianyang, China
E-mail:emyong@cardc.cn
*Corresponding author

***Abstract*—Spiking Neural Nnetworks (SNNs), the third generation Artificial Neural Networks (ANNs), are attractive because of their event-driven and sparsely spiking. An efficient train method is converting pre-trained ANNs to SNNs. There have been some approaches to decrease accuracy losses from ANNs to SNNs. However, their theoretical analyses are not enough and their effects still have room for improvement. In this paper, we analyze reasons of accuracy losses from ANNs to SNNs systematically. Then we propose an optimization method that can convert ANNs to SNNs with almost no accuracy loss, which is called finite-input-time (FIT). The MNIST database is employed to verify our analyses and optimization method. Simulation results are consistent with our analyses and show that our optimization method can convert ANNs to SNNs without accuracy loss.**

Keywords- spiking neural networks (SNNs); artificial neural networks (ANNs); conversion; finite-input-time (FIT); accuracy loss

I. INTRODUCTION

Artificial Neural networks (ANNs) have shown their great advantages in pattern recognition, image processing, and so on, and they almost have the start-of-the-art performance [1-5]. However, neural networks needs numerous computation efforts to successfully train them [6]. Efforts have been made to speed up execution or decrease costs of execution. In particular, Spiking Neural Networks (SNNs), the third generation ANNs, are considered as a promising choices to achieve such goals [7-9]. Neurons in SNNs spike sparsely and result in less computation. Moreover, the main operation in SNNs is addition, which means SNNs need less cost than ANNs when they have the same number of operations [6].

There have been some neuromorphic platforms for implementation of SNNs, which are fast and efficient, such as Minitaur, TrueNorth and SpiNNaker [10-12]. Therefore, developing methods to train SNNs with high accuracy is necessary. For SNNs with different coding methods (rate-coding or temporal-coding), there are several approaches to train them [18-22]. A supervised learning algorithm called SpikeProp that can train temporal-coding SNNs is presented by Bohteet al. and some improved methods are proposed later [19]. Some supervised learning algorithms that can train rate-coding SNNs are also proposed [20]. What's more, some unsupervised learning methods based on spiking-time-dependent-plasticity (STDP) are presented, which are more brain-like [21,22]. There is also a method that converts pre-trained ANNs to SNNs, which is more straightforward [6, 23,24]. However, there is still no systematic analysis to answer why these optimization methods can work better and why there are still some accuracy losses from ANNs to SNNs. Effects of these methods still have room for improvement.

In this paper, we analyze systematically to explain reasons of accuracy losses and mechanisms of different optimization methods. Then we propose an optimization method called finite-input-time (FIT), which can convert ANNs to SNNs without accuracy loss. We employ the MNIST database, the fully connected neural networks and the convolutional neural networks that are used in [24] to demonstrate our analyses and optimization method.

II. CONVERSION OF ANNS TO SNNS

The basis that ANNs can be directly converted to SNNs is that the spiking rates of IF neurons in SNNs should be proportional to activations of neurons in ANNs. [23] gives an approach that pre-trained ANNs with ReLUs can be converted to SNNs, which is our basis and will be described as follows.

Assume there are an ANN and a corresponding SNN. The neurons in the ANN and the SNN are ReLUs and IF neurons respectively. And the kinds of information transmitted in the ANN and the SNN are continuous values and spikes respectively. Assume these two neural networks have L layers, the number of neurons in each layer is $N^l(l\in1,2,\cdots,\Lambda)$, $W^l(l\in1,2,\cdots,L-1)$ is the weight matrix between layer l and layer $l+1$, and $b^l(l\in 2,3,\cdots,L)$ is the bias matrix. Note that the bias matrix can be set to 0 if neural networks have no bias. Then the activation of neuron i in layer l in the ANN can be expressed as:

$$a_i^l = max\left(0,\sum_{j=1}^{N^{l-1}} \omega_{ij}^{l-1}a_j^{l-1} + b_i^l\right) \quad (1)$$

Assume the membrane potential of the corresponding neuron in the SNN is $V_i^l(t)$, which has a change $\Delta V_i^l(t)$ at every time step. $\Delta V_i^l(t)$ can be computed as:

$$\Delta V_i^l(t) = \sum_{j=1}^{N^{l-1}}\left(\omega_{ij}^{l-1}\varepsilon_{t,j}^{l-1} + b_i^l\right) \quad (2)$$

Where$\varepsilon l-1\ t,j$is a step function which indicates whether the neuron j in layer $l-1$generates a spike at time t. The step function can be expressed as:

$$\varepsilon_{t,i}^{l} = \varepsilon\left(V_i^l\left(t-1\right) + \Delta V_i^l\left(t\right) - V_{th}\right) \tag{4}$$

where V_{th} is the threshold. When the membrane potential of an IF neuron exceeds the threshold, the IF neuron generates a spike and resets its membrane potential. There are two kinds of reset methods [6][24]: reset-to-zero and reset-by-subtraction, which are expressed as:

$$V_i^l\left(t\right) = \begin{cases} \left(V_i^l\left(t-1\right) + \Delta V_i^l\left(t\right)\right)\left(1-\varepsilon_{t,i}^l\right) & \textit{reset-to-zero} \\ V_i^l\left(t-1\right) + \Delta V_i^l\left(t\right) - V_{th}\varepsilon_{t,i}^l & \textit{reset-by-subtraction} \end{cases} \tag{5}$$

Assume the SNN is executed with a time step Δt, which means the highest spiking rate of all neurons is $r_{max} = 1/\Delta t$. When the neuron i has generated $g_i^l(t)$ spikes at time t, the spiking rate can be computed as $r_i^l(t) = g_i^l(t)/t$ and its ideal spiking rate is $a_i^l r_{max}$. Then the resulting spiking rates for IF neurons based on reset-to-zero and reset-by-subtraction in the input layer are computed as:

$$r_i^1\left(t\right) = \begin{cases} a_i^1 r_{\max} \cdot \dfrac{V_{th}}{V_{th}+\delta_i^1} - \dfrac{V_i^1\left(t\right)}{t\cdot\left(V_{th}+\delta_i^1\right)} & \textit{reset-to-zero} \\ a_i^1 r_{\max} - \dfrac{V_t^1\left(t\right)}{t\cdot V_{th}} & \textit{reset-by-subtraction} \end{cases} \tag{6}$$

For neurons based on reset-to-zero, assume the input neuron i needs m_i^1time steps to integrate its input currents to exceed the threshold, which means $m_i^1 a_i^1 \geqslant V_{th}$ and $(m_i^1-1)\cdot a_i^1 < V_{th}$. Therefore, the membrane potential will always exceed the threshold a constant, which is noted as $\delta_i^1 = m_i^1\, a_i^1 - V_{th}$. As described in [6], the neuron loses its potential by$\delta_i^1$for every m_i^1time steps, and there is a loss of the membrane potential at the last time step. For neurons based on reset-by-subtraction, the input neuron i only loses the membrane potential at the last time step. Therefore, spiking rates of neurons based on reset-by-subtraction are more approximate to $a_i^1 r_{max}$.

For neurons in other layers, the resulting spiking rates are:

$$r_i^l\left(t\right) = \begin{cases} a_i^l r_{\max} \cdot \dfrac{V_{th}}{V_{th}+\delta_i^l} - \dfrac{\Delta_i^l\left(t\right)}{t\cdot\left(V_{th}+\delta_i^l\right)} & \textit{reset-to-zero} \\ a_i^l r_{\max} - \dfrac{\Delta_t^l\left(t\right)}{t\cdot V_{th}} & \textit{reset-by-subtraction} \end{cases} \tag{7}$$

$$\Delta_i^l(t) = \left(V_{i_l}^l(t) + \sum_{i_{l-1}=1}^{N^{l-1}} \omega_{i_l i_{l-1}}^{l-1} V_{i_{l-1}}^{l-1}(t) + \cdots + \sum_{i_1=1}^{N^1} \omega_{i_2 i_1}^1 V_{i_1}^1(t)\right) / \left(t\cdot V_{th}\right)$$, which means that neurons in higher layers receive spikes with slightly lower rates due to membrane potential losses layer by layer. Analyses for neurons in higher layers are similar to the input layer. Hence, neurons based on reset-by-subtraction can express ideal spiking rates with less loss than neurons based on reset-to-zero. In other words, SNNs based on reset-by-subtraction will lose less accuracy.

III. ANALYSES

One deficiency in [6] is that it does not take the range of activations into account, which can affect accuracy losses significantly. When we use the MNIST database and the 784-1200-1200-10 fully connected neural network employed in [24], the distribution of activations of the second layer is shown in Fig.1. We can see that several thousand activations are greater than 1 though most activations are less than 1. In [6], more activations are greater than 1. Usually, activations that are greater than 1 are significant for feature classification. Therefore, it is necessary to analyze more systematically.

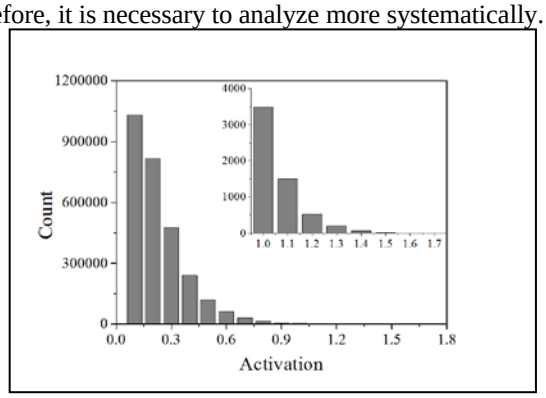

Figure 1. Distribution of activations of the second layer. Only activations greater than zero are counted.

For a ReLU in layer l, its activation is *al i*. The spiking rates of the IF neuron based on reset-to-zero and reset-by-subtraction are noted as *zrl i*(*t*) and *srl i*(*t*) respectively. When *al i* ⩽ 1, we can consider *zrl i*(*t*) as the product of *srl i*(*t*) and a factor $\frac{V_{th}}{V_{th}+\delta_i^l}$ ($\frac{V_{th}}{V_{th}+\delta_i^l} = \frac{V_{th}}{m_i^l \cdot a_i^l} < 1$), which is expressed as $(a_i^l r_{\max} - \frac{\Delta_i^l(t)}{t\cdot V_{th}}) \cdot \frac{V_{th}}{m_i^l \cdot a_i^l}$. Note that here we think $zr_i^l(t)$ and $sr_i^l(t)$ have the same $\Delta^l{}_i(t)$. Therefore, $sr_i^l(t)$ is more approximate to $a_i^l\, r_{max}$ than $zr_i^l(t)$ and the SNN based on reset-by-subtraction loses less accuracy.

When $a_i^l > 1$, $sr_i^l(t)$ will achieve r_{max} at $a_i^l = \alpha_s$ and keep constant even though a_i^l is greater than α_s . Here, α_s is slightly greater than 1 and related with $\Delta_i^l(t)$. The reason for this phenomenon is that the SNN cannot express spiking rates that are greater than r_{max}. This will result in a bad conclusion that spiking rates of IF neurons are not proportional to activations and cannot reflect any difference of different activations, which may cause accuracy losses. Similarly, $zr_i^l(t)$ will achieve r_{max} at $a_i^l = \alpha_z$ and keep constant even though a_i^l is greater than α_z. However, α_z is greater than α_s because of the factor. Therefore, $zr_i^l(t)$ can still reflect differences of different activations between α_s and α_z, which may cause less accuracy losses.

Then we analyze further to explain why the optimization methods in [24] can work better. Here we only consider the data normalization method due to its better performance. The data normalization optimization method finds the maximum activation of each layer, and then rescales all weights by the maximum activation of each layer to make sure there is no activation that exceeds 1. We rename the optimization method to max-normalization. The max-normalization can be expressed as

$$W^{\max\_l} = W^l \cdot f^l \tag{8}$$

where $f^l = 1/f^{l-1}a^l_{max}$ and $f^l = 1$. After rescaling, spiking rates of IF neurons are proportional to corresponding activations and SNNs based on reset-by-subtraction result in less accuracy losses than SNNs based on reset-to-zero according to analyses above.

IV. FIT

Though max-normalization can decrease accuracy losses by making all activations less than 1, some losses still cannot be avoided because spiking rates of IF neurons are not approximately equal to ideal rates due to rescaling. We want to achieve a goal that no accuracy is lost is required.

We can know that the main reason that results in accuracy losses for reset-by-subtraction is that IF neurons cannot spike with rates approximately equal to ideal rates when activations are greater than α_s, which can be called saturation. It seems that the parts of activations greater than α_s are lost. In fact, they are not lost but stored. When we test the 784-1200-1200-10 fully connected neural network and the 28 × 28-16c5-2s-16c5-2s-10 convolutional neural network in [24], membrane potentials of output neurons are shown in Fig.2. Here we just show membrane potentials that are greater than zero which are valuable for classification. We can find that most membrane potentials are much greater than 1 in Fig. 2(a) and Fig. 2(b), which means that many spikes that should be fired are not fired. When we consider (6) again, we can find that the analytic reason is that $\Delta^l_i(t)$ is too large. It is obvious that the spiking rate can be expressed correctly when $\Delta^l_i(t)$ is small enough, in other words, spikes that are stored are fired.

In [6, 23, 24], input neurons receive input currents at every time step, which means that neurons corresponding to activations that are greater than α_s have no time to fire spikes stored in membrane potentials. Here we present an optimization method called finite-input-time (FIT) to solve this problem. Our method is to give finite time steps for input and give extra time steps to fire stored spikes. Therefore, the total execution times of SNNs consist of two parts: one part with input currents and the other without input currents. Firstly, give input currents to SNNs for T_1 time steps to make all neurons that are significant for classification are activated enough. Then, SNNs are executed with extra T_2 time steps to make all stored spikes are fired and all membrane potentials remained are less than the threshold, which means $\Delta^l_i(t)$ is small enough to has little influence on final rates. Hence, our optimization method can make spiking rates of IF neurons approximately equal to activations of ReLUs without rescaling and convert ANNs to SNNs with almost no accuracy loss.

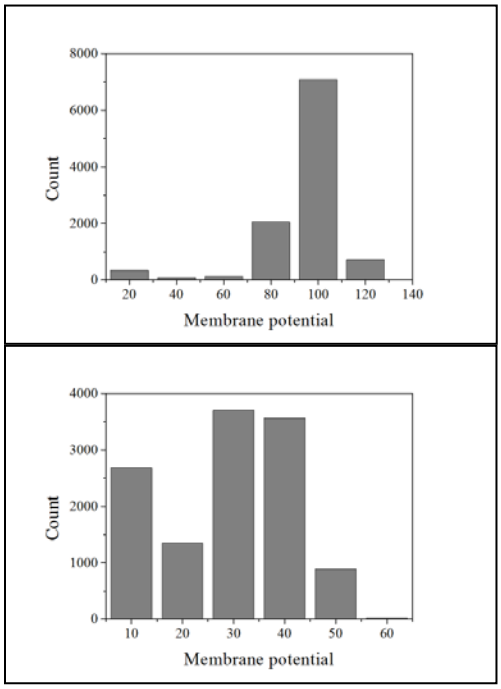

Figure 2. Distribution of membrane potentials of a) the fully connected SNN and b) the convolutional SNN.

V. RESULTS

To demonstrate our analyses and optimization method, the MNIST database, the fully connected neural networks and the convolutional neural networks in [24] are employed. The architecture of the fully connected neural network is 784-1200-1200-10. It is trained to achieve an accuracy of 98.84% on test set. The architecture of the convolutional neural network is 28 × 28-16c5-2s-16c5-2s-10. It has 16 convolutional kernels of size 5×5 in the second layer and the forth layer. A 2×2 averaging subsampling window follows each convolutional layer. It is trained to achieve an accuracy of 99.14% on test set.

Accuracies of SNNs based on reset-to-zero and reset-by-subtraction are shown in Fig. 3. As is shown, accuracies of SNNs based on reset-by-subtraction are both lower than SNNs based on reset-to-zero.

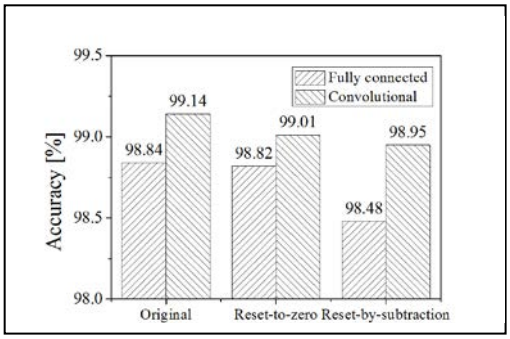

Figure 3. Accuracies of SNNs based on reset-to-zero and reset-by-subtraction.

Accuracies of SNNs optimized by max-normalization based on reset-to-zero and reset-by-subtraction are shown in Fig. 4. Obviously, accuracies of the fully connected SNN and the convolutional SNN based on reset-by-subtraction are both higher than SNNs based on reset-to-zero, which means reset-by-subtraction with max-normalization results in less accuracy losses than reset-to-zero. However, there are still some accuracy losses for both the fully connected SNN and the convolutional SNN compared to original ANNs.

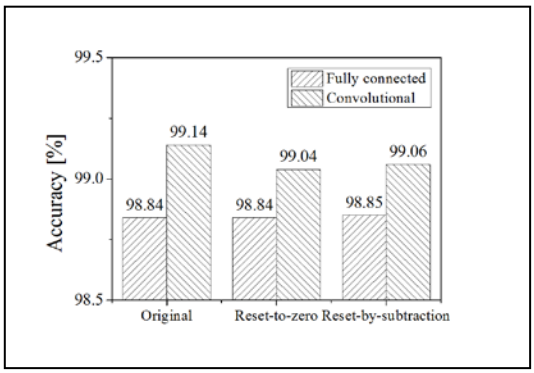

Figure 4. Accuracies of SNNs based on reset-to-zero and reset-by-subtraction with max-normalization.

Fig. 5 shows accuracies of SNNs based on reset-by-subtraction with FIT. As expected, there is no accuracy loss compared to original ANNs. Results also show that 5 (12) time steps are enough for input for the fully connected (convolutional) SNN and extra 4 (25) time steps are enough to achieve the expected accuracy.

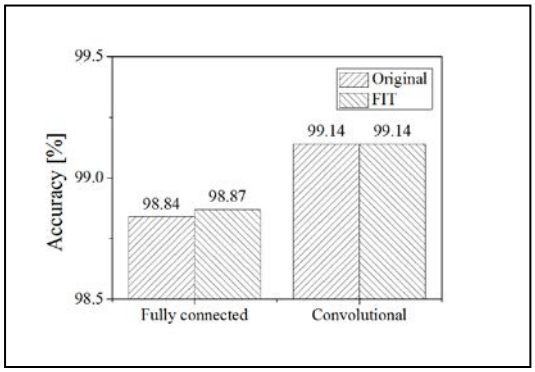

Figure 5. Accuracies of SNNs with FIT.

VI. CONCLUSION

In this paper, we analyze reasons of accuracy losses from ANNs to SNNs systematically and explain why previous methods can get better results. An optimization method called FIT is presented to convert ANNs to SNNs without accuracy loss, which limits input time steps and gives extra time steps to fire stored spikes. Results based on the MNIST database, the fully connected neural network and the convolutional neural network are consistent with our analyses.

REFERENCES

[1] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition , Boston, 2015. 1-9.

[2] Simonyan K, and Zisserman A. Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations, Banff, 2014. 1-14.

[3] Russakovsky O, Deng J, Su H, et al. ImageNet large scale visual recognition challenge. Int J Comput Vis, 2015, 115: 211-252.

[4] Wang Y, Xie Z, Xu K, et al. An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning. IEEE Trans Circuits Syst II Exp Briefs, 2016, 174: 988-998.

[5] Lan Q, Wang Z, Wen M, et al. High Performance Implementation of 3D Convolutional Neural Networks on a GPU. Comput Intel Neurosc, 2017, 2017: 1-8.

[6] Rueckauer B, Lungu I A, Hu Y, et al. Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification. Front Neurosci, 2017, 11: 682.

[7] Maass W. Lower bounds for the computational power of networks of spiking neurons. Neural Comput, 1997, 8: 1-40.

[8] Farabet, C, Paz R, Prez-Carrasco J, et al. Comparison between frame-constrained fix-pixelvalue and frame-free spiking-dynamic-pixel convNets for visual processing. Front Neurosci, 2012, 6: 32.

[9] Neil D, Pfeiffer M, and Liu S C. Learning to be efficient: algorithms for training low-latency, low-compute deep spikingneural networks. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing , Pisa, 2016. 293-298.

[10] Neil D, and Liu S C. Minitaur, an event-driven FPGA-based spiking network accelerator. IEEE Trans Very LargeScale Integr Syst, 2014, 22: 2621-2628.

[11] Merolla P A, Arthur J V, Alvarez-Icaza R, et al. A million spiking-neuron integrated circuit with a scalable commu-nication network and interface. Science, 2014, 345: 668-673.

[12] Furber S B, Galluppi F, Temple S, and Plana L A. The SpiNNakerproject. Proc. IEEE, 2014, 102: 652-665.

[13] Diehl P U, Pedroni B U, Cassidy A, et al. TrueHappiness: neuromorphic emotion recognition on TrueNorth. In: Proceedings of the International Joint Conference on Neural Networks, Vancouver, 2016. 4278-4285.

[14] Diehl P U, Zarrella G, Cassidy A, et al. Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware. In: IEEE International Conference on Rebooting Computing, San Diego, 2016. 1-8.

[15] Merkel C, Hasan R, Soures N, et al. Neuromemristive Systems: Boosting Efficiency through Brain-Inspired Computing. Computer, 2016, 49: 56-64.

[16] Sheridan P, Ma W, Lu W. Pattern recognition with memristor networks. In: IEEE International Symposium on Circuits and Systems, Melbourne VIC, 2014. 1078-1081.

[17] Yao P, Wu H, Gao B, et al. Neuromorphic Hardware System for Visual Pattern Recognition With Memristor Array and CMOS Neuron. IEEE Trans Ind Electron, 2015, 62: 2410-2419.

[18] Bohte S M, Poutre J A L, Kok J N. Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing, 2002, 48: 17-37.

[19] Xu Y, Zeng X, Han L, et al. A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. Neural Networks, 2013, 43:99-113.

[20] Ponulak F, Kasinski A. Supervised learning in spiking neural networks with ReSuMe-Sequence learning, classification, and spike shifting. Neural Comput, 2010, 22: 467-510.

[21] Diehl P U, Cook M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Front Comput Neurosc, 2015, 9: 99.

[22] Zeng Y, Zhang T, Xu B. Improving multi-layer spiking neural networks by incorporating brain-inspired rules. Sci China Inf Sci, 2017, 60: 052201.

[23] Cao Y, Chen Y, and Khosla D. Spiking deep convolutional neural networks for energy-efficient object recognition. Int J Comput Vis, 2015, 113: 54-66.

[24] Diehl P U, Neil D, Binas J, et al. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: Proceedings of the International Joint Conference on Neural Networks, Killarney, 2015. 1-8.