A New ANN-SNN Conversion Method with High Accuracy, Low Latency and Good Robustness

Bingsen Wang[1] , **Jian Cao**[1,*] , **Jue Chen**[1] , **Shuo Feng**[1] and **Yuan Wang**[2,*]
[1]School of Software and Microelectronics, Peking University
[2]School of Integrated Circuits, Peking University
nevermore@stu.pku.edu.cn, caojian@ss.pku.edu.cn, chenjue@stu.pku.edu.cn, feng_shuo@pku.edu.cn, wangyuan@pku.edu.cn

Abstract

Due to the advantages of low energy consumption, high robustness and fast inference speed, Spiking Neural Networks (SNNs), with good biological interpretability and the potential to be applied on neuromorphic hardware, are regarded as the third generation of Artificial Neural Networks (ANNs). Despite having so many advantages, the biggest challenge encountered by spiking neural networks is training difficulty caused by the non-differentiability of spike signals. ANN-SNN conversion is an effective method that solves the training difficulty by converting parameters in ANNs to those in SNNs through a specific algorithm. However, the ANN-SNN conversion method also suffers from accuracy degradation and long inference time. In this paper, we reanalyze the relationship between Integrate-and-Fire (IF) neuron model and ReLU activation function, propose a StepReLU activation function more suitable for SNNs under membrane potential encoding, and use it to train ANNs. Then we convert the ANNs to SNNs with extremely small conversion error and introduce leakage mechanism to the SNNs and get the final models, which have high accuracy, low latency and good robustness, and have achieved the state-of-the-art performance on various datasets such as CIFAR and ImageNet.

1 Introduction

With the rapid development of artificial intelligence, deep learning has played a more and more important role in many fields as a representative of AI algorithms. However, Artificial Neural Networks (ANNs) also face many challenges, such as poor robustness and high energy consumption. In recent years, due to the increasing demand for edge computing and energy-saving applications, Spiking Neural Networks (SNNs) have also attracted great attention because of their distinctive advantages from ANNs [Roy *et al.*, 2019]. As a representative of the third-generation neural networks [Maass, 1997], SNN is inspired by neurons in the biological nervous system, whose cells use discrete spike signals to encode and transmit information. It is precisely because the spike signal is asynchronous, event-based and non-differentiable that SNN has excellent robustness and energy efficiency [Diehl and Cook, 2015], but it also faces challenges such as training difficulties [Tavanaei *et al.*, 2019].

Although training the SNN network is very difficult (due to the inability to directly use the backpropagation algorithm), many excellent works have made a lot of contributions to the improvement of SNN's performance [Zhang *et al.*, 2019; Wu *et al.*, 2018a]. In general, there are two mainstream implementation methods of SNNs: ANN-SNN Conversion and direct training SNN.

ANN-SNN conversion. It can be proved mathematically that, when the inference time is long enough, ANN-SNN conversion can obtain the same accuracy as ANN [Sengupta *et al.*, 2019a], which is based on the equivalence of Integrate-and-Fire (IF) neuron and ReLU activation function [Cao *et al.*, 2015a] under firing rate encoding. However, for finite time steps, there will always be errors in the ANN-SNN conversion, such as clipping error, quantization error (flooring error) and unevenness error [Bu *et al.*, 2021a]. Among these three errors, unevenness error is caused by the unevenness of input spikes, while the clipping error and the quantization error are caused by firing rate encoding. In addition to firing rate encoding, we also often use membrane potential encoding [Li *et al.*, 2022a], time encoding and sequence encoding [Jeffares *et al.*, 2021] in SNNs.

Direct training. Due to the great success of gradient descent algorithm and backpropagation algorithm in training ANNs, some works regard SNNs as Recurrent Neural Networks (RNNs) and use Back-Propagation Through Time (BPTT) algorithm to train it. In these works, surrogate gradient methods are used to map discrete spike signals into differentiable mathematical formulas, so that the backpropagation algorithm can be used [Neftci *et al.*, 2019; Wu *et al.*, 2018b; Wunderlich and Pehle, 2021; Mostafa, 2017]. Although surrogate gradient methods can lead to very few time steps, the performance of SNNs trained in this way cannot reach a level comparable to that of ANNs. In addition to the surrogate gradient approach, there are some works using synaptic plasticity [Kheradpisheh *et al.*, 2018] for network training, which is

*Corresponding Author

also considered to be more biologically interpretable.

Difficulties. Generally speaking, no matter which method of building SNNs has its own difficulties and defects. Although the ANN-SNN conversion method can obtain better network performance and can also be used in deep networks, it will cause a relatively high time latency because this method requires more time steps [Sengupta *et al.*, 2019b]. In addition, while the direct training method has the advantages of fast inference speed and good robustness, it also has problems such as difficult training and insufficient performance [Zenke and Vogels, 2021]. That is to say, if we use existing methods, in order to obtain a spiking neural network with better performance, we must make a trade-off between inference speed, network accuracy and energy efficiency, which is also an open hot issue.

Contributions. In this work, we address two core challenges by combining the ANN-SNN conversion with the RNNs network structure, resulting in high-accuracy and low-latency SNNs with improved robustness. Specifically, we carefully analyzed the causes of conversion errors in the ANN-SNN conversion method, replaced firing rate encoding with membrane potential encoding, and proposed a new activation function StepReLU which is equivalent to IF neurons. This enables our SNNs to achieve a high accuracy close to that of ANNs at only one time step. In addition, we also use the network structure of RNNs and introduce a leakage mechanism, which makes our SNNs highly robust while maintaining high accuracy. The contributions of this work are summarized as follows:

- We reanalyzed the equivalence between IF neurons and ReLU activation function, and proposed a StepReLU activation function more suitable for spiking neural networks on the basis of discrete spike signals and membrane potential encoding. This allows SNNs to achieve state-of-the-art accuracy of ANNs over only one time step.
- We use the RNNs' network structure and introduce a leakage mechanism, so that the IF neurons in the last layer of the network have the same effect as the Leaky Integrate-and-Fire (LIF) neurons. This makes our model more robust and more tolerant to noisy signals.
- We conduct experiments on large datasets such as CIFAR and ImageNet, and the experimental results show that our method is effective on different network structures. The results show that our models are more robust while achieving state-of-the-art accuracy in a very short inference time.

2 Related Work

ANN-SNN conversion is a relatively mature method at present, which can obtain SNNs with the same accuracy as ANNs, and some of the latest works have also obtained models with less inference time than previous works. The ANN-SNN conversion method was first proposed and used in practical work by Cao et al. [2015b]. Then data-based and model-based normalization was proposed by Diehl et al. [2015].

Under firing rate coding, soft-reset mechanism (also called reset-by-subtraction mechanism) is proposed to solve the information loss problem caused by hard-reset mechanism [Rueckauer *et al.*, 2016; Diehl *et al.*, 2016]. Layer-based and neuron-based parameter normalization are proposed to solve the problem of different thresholds of neurons in different layers of SNNs, so that deeper networks can be realized [Sengupta *et al.*, 2019b; Ding *et al.*, 2021; Li *et al.*, 2021]. Rueckauer et al. [2017] indicated that, after selecting appropriate parameters, we can essentially eliminate clipping errors in ANN-SNN conversion through parameter normalization. After that, a lot of following works are studying how to choose a more appropriate normalization scale. Spiking-YOLO with channel-wise data-based normalization was proposed by Kim et al. [2020] to solve the problem of fast object detection in SNNs.

There are also some new neuron models proposed. In order to eliminate the quantization error in ANN-SNN conversion, Bu et al. [2021b] proposed the quantization clip-floor-shift activation function under average membrane potential encoding. Wang et al. [2022] proposed a neuron-wise parameter normalization method and a signed neuron with memory function under membrane potential encoding to eliminate ANN-SNN conversion errors, and their method achieved state-of-the-art results on multiple datasets. In order to solve the problems of high energy consumption and large time latency of traditional methods, Li and Zeng [2022] proposed a neuron model for releasing burst spikes.

In recent years, people have begun to pay more attention to the inference time of SNNs, and some works have made considerable progress. Deng and Gu [2021], Li et al. [2021], Meng et al. [2022] and Li and Zeng [2022] all achieved excellent results with 16 and 32 timesteps. Bu et al. [2021b] achieved excellent inference accuracy within 10 time steps or even 2 time steps. Besides, Massa et al. [2020] and Singh et al. [2021] deployed converted SNNs on the Loihi Neuromorphic Processor [Davies *et al.*, 2018] and evaluated the performance. Back-Propagation Through Time (BPTT) typically achieves very short inference time [Wu *et al.*, 2019; Lee *et al.*, 2016; Lee *et al.*, 2020], but results in high computational consumption. On the other hand, BPTT is only suitable for simple datasets such as MNIST and CIFAR10 [Kheradpisheh and Masquelier, 2020; Zhang and Li, 2020]. Rathi et al. [2020] initialized SNNs with conversion method and tuned SNNs with STDP to get a shorter inference time. Li et al. [2022b] proposed a new neuromorphic data augmentation method to improve the inference speed of SNNs.

3 Method

In this section, we use mathematical formulations to describe the ANN-SNN conversion process of our method. Based on the reanalysis of ANN-SNN conversion errors, we propose a StepReLU activation function to train ANNs with membrane potential encoding. Then we solve the neurodynamic equations of LIF neuron model to obtain the conditions that the leakage mechanism needs to fulfill, and introduce the leakage mechanism into SNNs. Further, we propose a new ANN-SNN conversion method based on the StepReLU function.

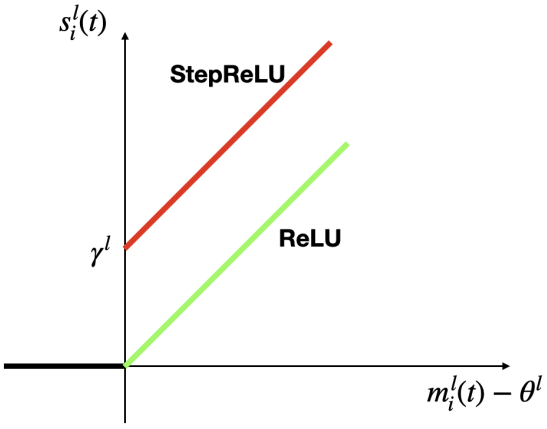

Figure 1: The Difference between StepReLU and ReLU

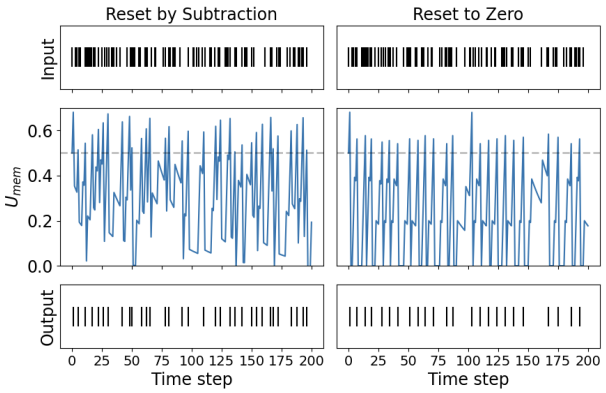

Figure 2: The Difference between Hard-reset and Soft-reset

3.1 Neuron Model

Neuron model for ANNs. For ANNs, the activity of neurons can be simplified to the following formula:

$$\boldsymbol{a}^l = \mathrm{ReLU}(W^l \boldsymbol{a}^{l-1} - \boldsymbol{b}^l), \quad l = 1, 2, \cdots, M \tag{1}$$

where the vector $\boldsymbol{a}^l$ represents the outputs of all neurons in l-th layer, W^l is the weight matrix between layer l and layer $l-1$, the vector $\boldsymbol{b}^l$ is a bias item, which also represents the thresholds of each neuron in layer l, and ReLU is a common activation function in ANNs.

Neuron model for SNNs. Similar to previous works [Han *et al.*, 2020], we use the Integrate-and-Fire (IF) neuron model to describe the ANN-SNN conversion process, because the IF neuron is mathematically equivalent to the ReLU activation function. At each time step t, the IF neurons in each layer receive the inputs from the neurons in the previous layer and emit spikes as outputs according to certain conditions. This process can be expressed by Eq. (2):

$$\boldsymbol{m}^l(t) = \boldsymbol{v}^l(t-1) + W^l \boldsymbol{x}^{l-1}, \tag{2}$$

where the vector $\boldsymbol{v}^l(t)$ is the membrane potential of layer l at time step t, vector $\boldsymbol{m}^l(t)$ is an intermediate variable for recording the membrane potential and $W^l \boldsymbol{x}^{l-1}$ can be regarded as the inputs received by neurons in layer l. For IF neurons, when the membrane potential of the i-th neuron in layer l exceeds the threshold θ^l, the neuron will fire a spike $s_i^l(t)$ to neurons in the next layer. Since we used membrane potential encoding instead of firing rate encoding, $s_i^l(t)$ can be expressed as Eq. (3):

$$s_i^l(t) = \begin{cases} \mathrm{ReLU}[m_i^l(t) - \theta^l] + \gamma^l, & \text{if} \quad m_i^l(t) > \theta^l \\ 0 \qquad\qquad , & \text{if} \quad m_i^l(t) \le \theta^l \end{cases} \tag{3}$$

here γ^l is a spike parameter that we can set, which is caused by different excitation thresholds of neurons in different layers. In ANNs, the minimum of γ^l can be 0, then Eq. (3) becomes the ReLU activation function. However, in SNNs, we noticed that as long as a neuron fires a spike, the output (membrane potential) of the neuron must be a positive number, and it cannot be equal to 0. Furthermore, unlike firing rate encoding, the output of neurons under membrane potential encoding can exceed 1. That is to say, $\gamma^l > 0$. Based on this, we propose the StepReLU activation function, as shown in the following Eq. (4):

$$\begin{aligned} \mathrm{StepReLU}(x, \gamma) &= \mathrm{ReLU}(x) + \gamma \\ &= \begin{cases} x + \gamma, & \text{if} \quad x > 0 \\ 0, & \text{if} \quad x \le 0 \end{cases} \end{aligned} \tag{4}$$

In this way, we can rewrite the spike firing process of a neuron in SNNs as Eq. (5):

$$s_i^l(t) = \mathrm{StepReLU}(m_i^l(t) - \theta^l, \gamma^l). \tag{5}$$

The difference between the StepReLU activation function and the ReLU activation function is shown in Fig. 1. After firing a spike, the membrane potential of the neuron is reset. In order to preserve more information of a spike, we choose the hard-reset mechanism, in which we reset the membrane potential to 0. This process can be represented by Eq. (6).

$$v_i^l(t) = \begin{cases} m_i^l(t), & \text{if} \quad m_i^l(t) \le \theta^l \\ 0, & \text{if} \quad m_i^l(t) > \theta^l \end{cases}. \tag{6}$$

The difference between hard-reset and soft-reset is shown in Fig. 2, and the figure was made with the help of the snntorch package [Eshraghian *et al.*, 2021]. The short black vertical lines in the upper and lower parts of the figure represent the input and output spike trains of neurons, respectively. The blue line in the middle of the figure shows how the neuron's membrane potential changes over time. The only difference between hard-reset mechanism and soft-reset mechanism is that, when the membrane potential of a neuron reaches the threshold and fires a spike, hard-reset will cause the membrane potential to return to 0 directly, while soft-reset will only reduce the membrane potential by a specific value.

Membrane potential encoding. The reason why firing rate encoding is currently used more frequently is mainly because people generally regard the SNNs as binary networks, in which neurons' firing a spike represents 1, and non-firing represents 0. In order to obtain excellent performance without sacrificing inference time, we no longer use binary firing rate encoding but full-precision membrane potential encoding. Under membrane potential encoding, a spike fired by a

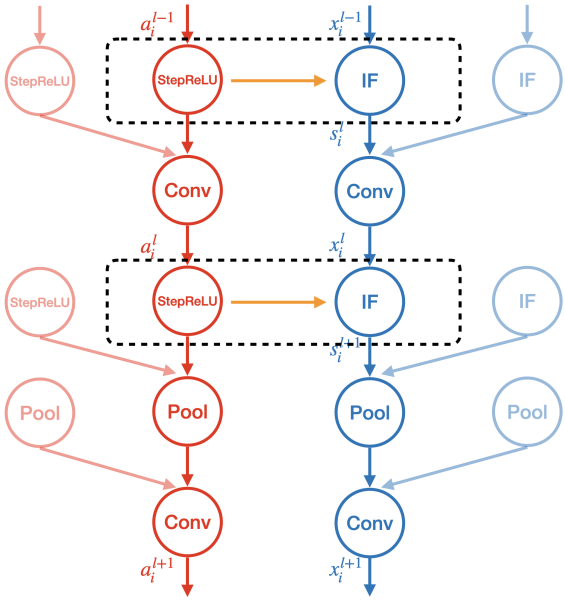

Figure 3: ANN-SNN Conversion Process

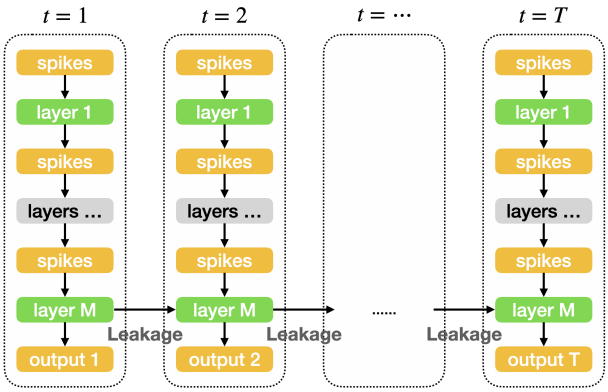

Figure 4: Spiking Neural Network Architecture with Leakage

neuron no longer represents 0 or 1, but represents the membrane potential of the neuron when the spike is fired. Then we consider the case that neuron i sends a spike to neuron j. If we take the synaptic weight between i and j as 1, the spike sent by i is consistent with the spike received by j. Although the duration of the spike signal is very short, it is still the transmission of electrical signals in essence. In other words, when the connection weight between i and j is 1, the charge of the spike sent by i is equal to the charge of the spike received by j. We know that an IF neuron is physically equivalent to a Resistance-Capacitance (R-C) circuit with a battery. Due to the difference between the circuit constants R and C, the spike signals of the same charge represent different values for different neurons under membrane potential encoding, as shown in Eq. (7):

$$q_i = q_j = C_i s_i = C_j x_j, \quad x_j = \frac{C_i}{C_j} \cdot s_i = W_{ij} \cdot s_i \tag{7}$$

where C represents the capacitance constant of the neurons, s_i represents the value of the spike signal for neuron i, x_j represents the value of the spike signal for neuron j and q represents the charge of the spike. W_{ij} is the conversion weight between neuron i and neuron j (the connection weight is 1), and is also an element of the weight matrix W in Eq. (2). In this way, we have clarified the definition of the weights between neurons under membrane potential encoding.

3.2 Conversion from ANN to SNN

Conversion error analysis. Different from the conversion error under firing rate encoding and soft reset mechanism, the conversion error generated by our method is much simpler. As can be seen from Fig. 1, in theory, when we use the StepReLU activation function to train the network, the output value of StepReLU and the output value of ReLU only differ by a constant γ^l. In other words, we can use the StepReLU activation function in the training of ANNs, in which we will not introduce additional conversion errors as long as we make reasonable adjustments to the bias term. The key points are around 0. We artificially introduced and amplified the difference between spike firing and non-spike firing, which will increase the tolerance of the network to noise interference, and may also cause some activated points near 0 to not be activated. This is the only source of ANN-SNN conversion error in our proposed method. For the trade-off between conversion error and robustness, we will show it in the Experiments part. Overall, if we choose appropriate parameters, we can achieve a robustness improvement with little drop in network inference accuracy.

ANN-SNN conversion. According to our method, the cores of ANN-SNN conversion are using StepReLU function to train ANNs and mapping the outputs of ANNs' neurons to the membrane potential of SNNs' neurons. The ANN-SNN conversion in one time step t can be expressed by Eq. (8)-(9):

$$\boldsymbol{a}^l = \text{StepReLU}(W^l \boldsymbol{a}^{l-1} - \boldsymbol{b}^l, \gamma^l), \quad l = 1, 2, \cdots, M \tag{8}$$

$$\boldsymbol{x}^{l-1}(t) = \boldsymbol{a}^{l-1}, \boldsymbol{\theta}^l = \boldsymbol{b}^l, \boldsymbol{s}^l(t) = \boldsymbol{a}^l. \tag{9}$$

The specific process of ANN-SNN conversion is shown in Fig. 3. In the figure, red circles and blue circles in the middle represent the neurons in ANNs and SNNs respectively, and the horizontal arrows indicate that the corresponding neurons are equivalent. Semi-transparent circles along the edges of the figure indicate other non-target neurons in the network.

Leakage mechanism. We usually think that the reason why SNNs have good robustness and sparsity is that there is a leakage mechanism between different time steps. The leakage mechanism is derived from the Leaky Integrate-and-Fire (LIF) neuron model, which means that the membrane potential of neurons that do not fire a spike at the current time step will gradually decrease over time. In neurobiology, the LIF neuron model is an approximation to the Hodgkin-Huxley (H-H) model, and its dynamic process can be expressed by the differential equation shown in Eq. (10):

$$\tau_m \frac{du}{dt} = -[u(t) - u_{\text{rest}}] + Ri(t), u_{\text{rest}} = 0, \tag{10}$$

where $u(t)$ represents the membrane potential, $Ri(t)$ represents the input signals, τ_m is the temporal constant and $\tau_m = RC$. Here we take $u_{\text{rest}} = 0$ because the value of u_{rest} does not affect the result under the hard-reset mechanism. In addition to the differential equation in Eq. (10), the LIF neuron also satisfies the hard-reset mechanism shown in Eq. (11):

$$\lim_{t \to t_f^+} u(t) = 0, \tag{11}$$

here t_f represents the moment when the LIF neuron fires a spike signal, which can be determined by $t_f = \{t | u(t) = \vartheta\}$. And $\lim_{t \to t_f^+}$ represents the right limit to t_f. ϑ is the membrane potential threshold of the LIF neuron. When the membrane potential $u(t)$ of the LIF neuron reaches the threshold ϑ, the neuron will fire a spike $s(t)$, and $s(t)$ satisfies Eq. (12):

$$s(t) = \begin{cases} \lim\limits_{t \to t_f^-} u(t), & \text{if} \quad t = t_f \\ 0, & \text{if} \quad t \neq t_f \end{cases}, \tag{12}$$

here we have considered the fact that the membrane potential may exceed the threshold in the actual solution, and we have already used the membrane potential code to describe the spike. The above three equations completely describe all the activities of a LIF neuron, and the solution to the differential equation in Eq. (10)-(12) can be written as:

$$\begin{aligned} u(t) &= -\sum_{t_f} s(t_f) \cdot e^{-\frac{t-t_f}{\tau_m}} + \frac{R}{\tau_m} \int_0^\infty e^{-\frac{s}{\tau_m}} \cdot i(t-s) ds \\ &= \int_0^\infty \eta(s) S(t-s) ds + \int_0^\infty \kappa(s) \cdot i(t-s) ds \end{aligned} \tag{13}$$

where $S(t) = s_{f1}(t) + s_{f2}(t) + \cdots$ represents the output sequence and $i(t)$ represents the input sequence, $\kappa(s)$ represents the parameter related to leakage and $\eta(s)$ represents the parameter related to firing sequence. Since the leakage only occurs in the input sequence, we neglect the firing item $\int_0^\infty \eta(s) S(t-s) ds$ and discretize the convolution of $i(s)$ and $\kappa(s)$ in Eq. (13) to get:

$$\boldsymbol{m}^l(T) = \boldsymbol{v}^l(0) + \sum_{t=1}^{T} \kappa(T-t) \cdot W^l \boldsymbol{s}^{l-1}(t). \tag{14}$$

In Eq. (14), we have assumed that the neurons in the l-th layer do not send spikes to the next layer within T time steps. Obviously, this assumption is true for the last layer of neurons in the network. Based on this, we introduce a leakage mechanism on the neurons of the last layer (also called the inference layer) of the Spiking Neural Networks, hoping to make the entire networks more robust in a few time steps. In this way, the SNNs network structure we finally converted from ANNs is shown in Fig. 4. In the figure, similar to the structure of RNNs, the vertical arrows indicate the transmission direction of spike signals, and the horizontal arrows indicate the time steps. At each time step from 1 to T, the spike signals are transmitted along the direction of the network and the outputs are obtained according to Eq. (8) and Eq. (9), and then these outputs are summed along the time direction according to Eq. (14) to obtain the final output.

4 Experiments

In order to verify the effectiveness of our method, we use VGG and ResNet network structures to conduct experiments on CIFAR10 [LeCun *et al.*, 1998], CIFAR100[1] [Krizhevsky, 2009] and ImageNet[2] [Deng *et al.*, 2009] datasets, and compare with other state-of-the-art results on image classification tasks. In the experiment, we use the Kaiming normal initialization method to initialize the ANNs network parameters, and use the SGD optimizer with 0.9 momentum. The L2 penalty with a value of 5e-4 is also added. In addition, we set an initial learning rate of 0.1 and update the learning rate with the CosineAnnealingLR strategy. The training epoch is 300 and the batch size is 128. Specific to a single experiment, we will make targeted adjustments to various parameters.

In this part, we verify our method from two aspects, one is the inference accuracy of ANN-SNN conversion, and the other is the robustness of the converted SNNs, which is brought by StepReLU activation function and leakage mechanism respectively. Our code can be seen at https://github.com/QuelThalasGrace/aNewANN-SNNConversionMethod.

4.1 ANN-SNN Conversion in One Time Step

Following the aforementioned method, we use the StepReLU activation function to train an ANN and convert it to an SNN at one time step. We test the performance of the converted SNN on different datasets, and the experimental results are shown in Table 1. Here, we compare our method with the state-of-the-art ANN-SNN conversion methods, including Quantization Clip-floor-shift Activation(QCA) from Bu et al. [2021b], Burst+LIPooling(BLIP) from Li and Zeng [2022], SNN Conversion with Advanced Pipeline(SNNC-AP) from Li et al. [2021], Signed Neuron with Memory and Neuron-Wise Normalization(SNMNN) from Wang et al. [2022] and Differentiation on Spike Representation(DSR) from Meng et al. [2022]. Experimental results show that our method can make the performance of SNNs close to that of ANNs in only one time step, and the optimal accuracy of transformed SNNs depends on the accuracy of ANNs. In other words, when using our method for ANN-SNN conversion, the better the initial performance of the ANNs, the better the performance that our resulting SNNs can achieve at one time step.

In addition, we also verify the robustness improvement of the network brought by the proposed StepReLU activation function. Specifically, we feed ANNs and converted SNNs with noisy inputs without introducing a leakage mechanism (in one time step), and observe the change of network's inference accuracy. Since SNNs use spike signals to transmit information, we choose to use salt and pepper noise, which is more similar to spike signals, as the experimental noise. We use salt and pepper noise signals with different signal-to-noise ratios (p=0.5) to process the input images and send them to ANNs and SNNs respectively, and get inference results. We have done the experiments on CIFAR10, CIFAR100 and ImageNet datasets, and the experimental results are shown in Fig. 5(a)-(c). The results show that as the input signal becomes more and more noisy, the inference accuracy at

[1]https://www.cs.toronto.edu/~kriz/cifar.html

[2]https://image-net.org/challenges/LSVRC/2012/

Method	Net. Arch.	ANN Acc.	T=1	T=16	T=20	T=32	T=64	T=128	T≥256
				CIFAR10					
QCA	VGG16	95.52	-	95.40	-	95.54	95.55	-	95.59
BLIP	VGG16	95.74	-	-	-	95.58	95.66	95.69	95.72
SNNC-AP	VGG16	95.72	-	-	-	93.71	95.14	-	95.79
Ours(StepReLU)	VGG16	95.91	**95.83**	-	-	-	-	-	-
SNMNN	ResNet18	95.39	-	-	-	94.03	94.03	95.19	95.44
QCA	ResNet18	96.04	-	95.92	-	96.08	96.06	-	96.06
BLIP	ResNet20	96.56	-	-	-	96.11	96.49	96.45	96.36
Ours(StepReLU)	ResNet20	96.64	**96.59**	-	-	-	-	-	-
				CIFAR100					
DSR	VGG16	78.12	-	-	78.62	-	-	-	-
BLIP	VGG16	78.49	-	-	-	74.98	78.26	78.66	78.65
Ours(StepReLU)	VGG16	78.27	**78.05**	-	-	-	-	-	-
SNMNN	ResNet18	78.26	-	-	-	74.48	77.59	77.97	78.30
BLIP	ResNet20	80.69	-	-	-	76.39	79.83	80.52	80.57
Ours(StepReLU)	ResNet20	80.60	**80.52**	-	-	-	-	-	-
				ImageNet					
BLIP	VGG16	74.27	-	-	-	70.61	73.32	73.99	74.25
QCA	VGG16	74.29	-	50.97	-	68.47	72.85	73.97	74.32
SNNC-AP	VGG16	75.36	-	-	-	63.64	70.69	73.32	75.32
Ours(StepReLU)	VGG16	74.40	**74.31**	-	-	-	-	-	-

Table 1: ANN-SNN Conversion in One Time Step

one time step decreases for both ANNs and converted SNNs. However, the performance decrease of converted SNNs is less than that of ANNs. Overall, the proposed StepReLU activation function can bring about 1% to 3% improvement in inference accuracy for SNNs compared to ANNs in noisy environments, when the parameter γ^l is properly chosen.

4.2 ANN-SNN Conversion with Leakage Mechanism

According to our derivation and analysis, both StepReLU activation function and the introduction of leakage mechanism will improve the robustness of the network. However, in this part, we pay more attention to the robustness improvement brought by introducing leakage mechanism to SNNs. We still use the salt and pepper noise signal as our experimental noise, and in order to maintain a fast inference speed, we set the time step of SNNs with the leakage mechanism to be 10. In experiments, we still feed the inputs processed with noisy signal directly into ANNs, however this is different in SNNs. Since the inference process of SNNs has ten time steps, we will feed the same input signal to the SNNs every time step, ten times in total. Considering a more realistic situation, we randomly select one of the ten identical input signals, process it with noise signal, and sequentially feed these new input signals into SNNs. Then we select an appropriate leakage parameter κ, and sum the ten output values as shown in Eq. (14) as the final output value, and the category corresponding to the neuron with the largest output value is the inference result. In this way, we observe the inference accuracy of SNNs with leakage mechanism over ten time steps, and the experimental results are shown in Fig. 5(d)-(f). The curves in these figures show the accuracy changes of ANNs and SNNs with leakage mechanism as the proportion of noise signals increases on CIFAR10, CIFAR100 and ImageNet. The results show that the leakage mechanism can greatly improve the robustness of SNNs, and the leakage mechanism makes the inference accuracy of SNNs about 31% higher than that of ANNs under noisy conditions.

4.3 Sources of Robustness Brought by Our Method

As we have verified before, both StepReLU activation function and leaky mechanism can bring robustness improvement to SNNs. Combining the derivation and experimental results, we believe that the StepReLU activation function artificially sets the difference between the activation and inactivation of neurons through the parameter γ^l to achieve robustness improvement, while the leakage mechanism improves robustness by improving the fault tolerance of SNNs through the parameter κ.

For StepReLU activation function, it is more suitable for SNNs under the membrane potential encoding while retaining the characteristics of ReLU activation function to the greatest extent. As we discussed in Section 3.2, the StepReLU activation function has a jump at zero point, which of course makes some neurons that can be activated in ANNs (the membrane potential just reaches the threshold) cannot be activated in SNNs, resulting in a decrease in accuracy. But more importantly, this will also make the SNNs more tolerant to the noise signals near the zero point, because some weaker noise

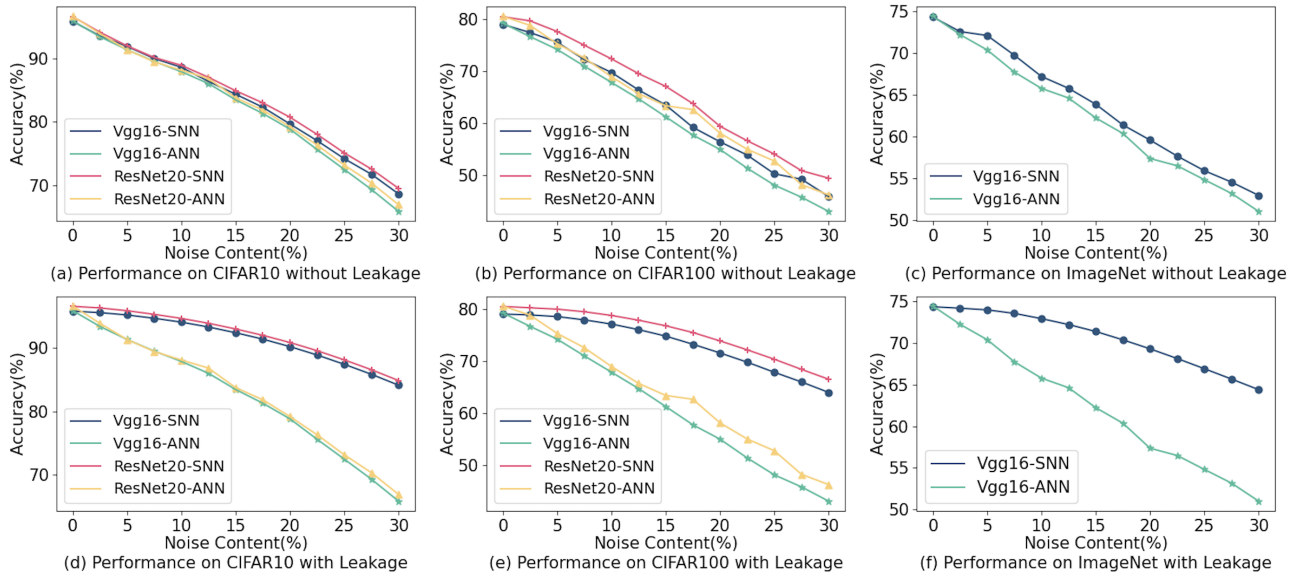

Figure 5: Effects of StepReLU and Leakage on Network Robustness

signals will not exceed the difference of the StepReLU function at the zero point. This point has also been verified in experiments. When we choose the appropriate parameters, the StepReLU function can make SNNs achieve the accuracy close to ANNs (this usually comes down a little bit) in one time step, and the robustness will also increase by 1%-3%, which is mainly produced under weak noise signals.

For leakage mechanism, the reason why it can bring about a huge improvement (about up to 31%) in the robustness of the network is its tolerance to noisy signals. Taking the 10 time steps we chose as an example, since the influence of the output of each time step on the final result will gradually decrease over time (leakage), this leads to the fact that only the output of the last time step will have a substantial impact on the final result. In other words, under the condition of ten time steps, when we assume that the noise signal randomly affects the input of one of the ten time steps, it will only affect the final output when the noise signal affects the input of the last time step. Theoretically, other situations will not have a great impact on the final output, which is the core reason why the leakage mechanism can bring great improvement on robustness of SNNs. If we set a larger time step, theoretically we can get more robust SNNs, but this requires a trade-off between inference time and robustness. This point has also been well verified in experiments.

5 Conclusion

In this paper, we propose a new ANN-SNN conversion method, derived from the StepReLU activation function, membrane potential encoding and leakage mechanism, which can achieve accurate, robust and low-latency inference. we reanalyze the relationship between Integrate-and-Fire (IF) neuron model and ReLU activation function, then we propose a StepReLU activation function more suitable for SNNs, and use it to train ANNs. After finishing the training, we convert the ANNs to SNNs and introduce the leakage mechanism to obtain a model with excellent performance. Our method can make SNNs achieve the inference accuracy close to ANNs in one time step and obtain a robustness improvement (about 1%-3%), and also enable SNNs to achieve a much higher robustness than ANNs (up to about 30%) in a very short inference time (10 time steps).

There are two core foundations in our method, the equivalence between the StepReLU activation function and the Integrate-and-Fire (IF) neuron model under membrane potential encoding, and the solution of the dynamic equations of the Leaky Integrate-and-Fire (LIF) neuron model. We have proved and solved the above two conclusions mathematically, and verified through experiments that our proposed method can effectively improve the performance of SNNs. Since the neurons in the inference layer of the network do not fire spikes, we ignore the output term in Eq. (13) and get Eq. (14), and only introduce a leakage mechanism in the last layer of neurons in the SNNs. In fact, Eq. (13) is valid for any LIF neuron, but since different neurons fire spikes at different moments, it is very difficult to simplify Eq. (13). This is also the core difficulty of reducing the energy consumption of SNNs and realizing asynchronous SNNs in the true sense.

Although our method has achieved excellent performance on different network structures and datasets, there are still some shortcomings that need our improvement. Essentially, the SNNs model we obtained in this work is still synchronous networks, not real asynchronous, event-driven networks, and these points are just considered to be the key to SNNs' low energy consumption. In the follow-up research, we will try to build a truly asynchronous SNNs' model, which can achieve the advantages of high accuracy, low latency, high robustness and low energy consumption at the same time.

Acknowledgements

This work is supported by the Joint Funds of the National Natural Science Foundation of China (U20A20204).

Contribution Statement

The first author and the second author contributed equally to this paper.

References

[Bu *et al.*, 2021a] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2021.

[Bu *et al.*, 2021b] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2021.

[Cao *et al.*, 2015a] 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.

[Cao *et al.*, 2015b] 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.

[Davies *et al.*, 2018] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

[Deng and Gu, 2021] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

[Diehl *et al.*, 2015] 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 *2015 International joint conference on neural networks (IJCNN)*, pages 1–8. ieee, 2015.

[Diehl *et al.*, 2016] Peter U Diehl, Bruno U Pedroni, Andrew Cassidy, Paul Merolla, Emre Neftci, and Guido Zarrella. Truehappiness: Neuromorphic emotion recognition on truenorth. In *2016 international joint conference on neural networks (ijcnn)*, pages 4278–4285. IEEE, 2016.

[Ding *et al.*, 2021] Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. *arXiv preprint arXiv:2105.11654*, 2021.

[Eshraghian *et al.*, 2021] Jason K Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *arXiv preprint arXiv:2109.12894*, 2021.

[Han *et al.*, 2020] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13558–13567, 2020.

[Jeffares *et al.*, 2021] Alan Jeffares, Qinghai Guo, Pontus Stenetorp, and Timoleon Moraitis. Spike-inspired rank coding for fast and accurate recurrent neural networks. *arXiv preprint arXiv:2110.02865*, 2021.

[Kheradpisheh and Masquelier, 2020] Saeed Reza Kheradpisheh and Timothée Masquelier. Temporal backpropagation for spiking neural networks with one spike per neuron. *International Journal of Neural Systems*, 30(06):2050027, 2020.

[Kheradpisheh *et al.*, 2018] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J Thorpe, and Timothée Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018.

[Kim *et al.*, 2020] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11270–11277, 2020.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[Lee *et al.*, 2020] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in neuroscience*, page 119, 2020.

[Li and Zeng, 2022] Yang Li and Yi Zeng. Efficient and accurate conversion of spiking neural network with burst spikes. *arXiv preprint arXiv:2204.13271*, 2022.

[Li *et al.*, 2021] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, pages 6316–6325. PMLR, 2021.

[Li *et al.*, 2022a] Wenshuo Li, Hanting Chen, Jianyuan Guo, Ziyang Zhang, and Yunhe Wang. Brain-inspired multilayer perceptron with spiking neurons. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 783–793, 2022.

[Li *et al.*, 2022b] Yuhang Li, Youngeun Kim, Hyoungseob Park, Tamar Geller, and Priyadarshini Panda. Neuromorphic data augmentation for training spiking neural networks. *arXiv preprint arXiv:2203.06145*, 2022.

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

[Massa *et al.*, 2020] Riccardo Massa, Alberto Marchisio, Maurizio Martina, and Muhammad Shafique. An efficient spiking neural network for recognizing gestures with a dvs camera on the loihi neuromorphic processor. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

[Meng *et al.*, 2022] Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Training high-performance low-latency spiking neural networks by differentiation on spike representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12444–12453, 2022.

[Mostafa, 2017] Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7):3227–3235, 2017.

[Neftci *et al.*, 2019] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[Rathi *et al.*, 2020] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv preprint arXiv:2005.01807*, 2020.

[Roy *et al.*, 2019] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[Rueckauer *et al.*, 2016] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, and Michael Pfeiffer. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052*, 2016.

[Rueckauer *et al.*, 2017] 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.

[Sengupta *et al.*, 2019a] 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.

[Sengupta *et al.*, 2019b] 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.

[Singh *et al.*, 2021] Sonali Singh, Anup Sarma, Sen Lu, Abhronil Sengupta, Vijaykrishnan Narayanan, and Chita R Das. Gesture-snn: co-optimizing accuracy, latency and energy of snns for neuromorphic vision sensors. In *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6. IEEE, 2021.

[Tavanaei *et al.*, 2019] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

[Wang *et al.*, 2022] Yuchen Wang, Malu Zhang, Yi Chen, and Hong Qu. Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion. In *International Joint Conference on Artificial Intelligence*, 2022.

[Wu *et al.*, 2018a] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

[Wu *et al.*, 2018b] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

[Wu *et al.*, 2019] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1311–1318, 2019.

[Wunderlich and Pehle, 2021] Timo C Wunderlich and Christian Pehle. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports*, 11(1):1–17, 2021.

[Zenke and Vogels, 2021] Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4):899–925, 2021.

[Zhang and Li, 2020] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in Neural Information Processing Systems*, 33:12022–12033, 2020.

[Zhang *et al.*, 2019] 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 *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1319–1326, 2019.