Enhancing the Robustness of Spiking Neural Networks with Stochastic Gating Mechanisms

Jianhao Ding[1], **Zhaofei Yu**[1,2*], **Tiejun Huang**[1], **Jian K. Liu**[3]

[1]School of Computer Science, Peking University
[2]Institution for Artificial Intelligence, Peking University
[3]School of Computer Science, University of Birmingham
djh01998@stu.pku.edu.cn, {yuzf12, tjhuang}@pku.edu.cn, j.liu.22@bham.ac.uk

Abstract

Spiking neural networks (SNNs) exploit neural spikes to provide solutions for low-power intelligent applications on neuromorphic hardware. Although SNNs have high computational efficiency due to spiking communication, they still lack resistance to adversarial attacks and noise perturbations. In the brain, neuronal responses generally possess stochasticity induced by ion channels and synapses, while the role of stochasticity in computing tasks is poorly understood. Inspired by this, we elaborate a stochastic gating spiking neural model for layer-by-layer spike communication, introducing stochasticity to SNNs. Through theoretical analysis, our gating model can be viewed as a regularizer that prevents error amplification under attacks. Meanwhile, our work can explain the robustness of Poisson coding. Experimental results prove that our method can be used alone or with existing robust enhancement algorithms to improve SNN robustness and reduce SNN energy consumption. We hope our work will shed new light on the role of stochasticity in the computation of SNNs. Our code is available at https://github.com/DingJianhao/StoG-meets-SNN/.

Introduction

As a representative of low-energy neural network systems, spiking neural networks (SNN) are being studied by researchers from the perspective of deep learning (Maass 1997; Zenke et al. 2021; Xu et al. 2022; Shen et al. 2023). While traditional artificial neural networks (ANN) use floatpoint values to simulate the rate coding of biological neurons, SNNs take a different approach by directly communicating through spike sequences, offering a simplified representation of the intricate dynamics of the biological system (Gerstner et al. 2014). This simplification presents a remarkable advantage for SNNs, enabling them to carry out computations on neuromorphic hardware with high efficiency (DeBole et al. 2019; Pei et al. 2019). Behind the advantage is the impact that the simplified SNN gradually approaches ANN in terms of computing capability, and does not exhibit the huge advantage of high robustness shown by the biological nervous system. Nonetheless, researchers continue to explore and refine SNNs, aiming to bridge the gap between the computational capabilities of SNNs and the robustness observed in biological neural systems.

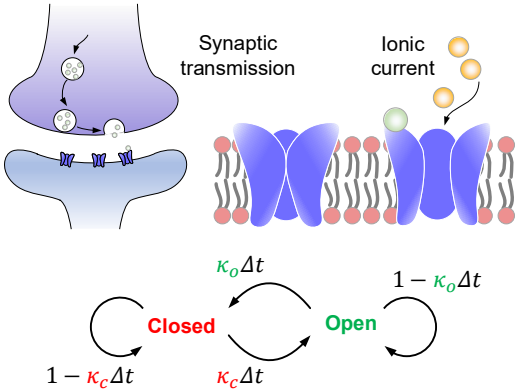


Figure 1: Diagram of spiking neurons with stochastic gating in neuronal synapses and ion channels.

Typical application scenarios for SNNs include robotic control (Bing et al. 2018) and autonomous driving (Yamazaki et al. 2022), where reliable perception is safety-critical. Malicious attacks can make neural networks give wrong predictions (Goodfellow, Shlens, and Szegedy 2015; Szegedy et al. 2014). These attacks are usually constructed from derivatives that are automatically differentiated. While SNNs possess the potential for improved robustness compared to their traditional counterparts (Sharmin et al. 2020), they are not immune to attacks due to their training with surrogate functions (Wu et al. 2018). Recent research has shown that SNNs can still be susceptible to adversarial attacks, emphasizing the need to understand and enhance their robustness against such threats (Liang et al. 2023; Sharmin et al. 2019). Consequently, the quest for understanding and improving the robustness of SNNs has become a critical and pressing issue in the field (Liang et al. 2022). In order to unleash the potential invulnerability of SNN from adversarial attacks and ensure their reliability and safety in real-world applications, researchers are actively exploring innovative techniques and strategies. At present, leading studies mainly consider how to use the wisdom of generalization on toxic samples for defense (Kundu, Pedram, and Beerel 2021), and

*Corresponding author

stay at the discussion of simplified neurons, which is still far from the nature of human perception.

Stochasticity is pervasive in biological coding. At present, using stochasticity to improve the robustness of SNN has become a feasible solution. Sharmin et al. stated that Poisson coding sampling data from a Bernoulli distribution could help SNN improve robustness (Sharmin et al. 2020). This method has been applied to text classification (Lv, Xu, and Zheng 2023) and sparked a discussion about the robustness of the encoding methods (Kim et al. 2022). At the same time, Li et al. believed that SNNs with inherent noise are more robust to input noise than ANNs (Li et al. 2020). The robustness verification of the above works is mainly carried out experimentally, lacking a theoretical explanation. For biological systems, the role of noise is more complex. Faisal et al. reviewed the function of noise at the cellular level. How much these noises contribute beneficially to neuronal processing is the fundamental problem of neural coding (Faisal, Selen, and Wolpert 2008). Thus, how to introduce meaningful stochasticity to provide robustness for deep SNNs becomes our primary motivation. In synapses, the transmission of neurotransmitters and the opening and closing of ion channels are both stochastic, which provoke noise to signals and are believed to improve temporal representation (White, Rubinstein, and Kay 2000). Inspired by this, we propose a stochastic gating model to improve robustness. Our main contributions are summarized as follows:

- We propose a stochastic gating model for spike transmission between layers in SNNs. This model functions similarly to synapses by filtering spikes, while also introducing stochasticity to the network.
- We theoretically demonstrate that our gating model is equivalent to a regularizer through the Taylor expansion of adversarial loss, allowing us to regulate the robustness of SNNs by controlling the probability of gate opening.
- Theoretical findings indicate that our model can be equivalent to reducing the Lipschitz constant of SNN, thereby preventing error amplification. Besides, the gating model can be equivalent to Poisson coding, helping to explain why Poisson coding can promote robustness.
- Experimental results validate that our method can effectively improve the robustness of SNN, and can be used in conjunction with adversarial training and weight regularization techniques. Moreover, our approach also offers the additional benefit of reducing energy consumption during SNN inference.

Related Work

Supervised training algorithms emerged in recent years (Sengupta et al. 2019; Zhou et al. 2021; Duan et al. 2022) influence SNN-specific attack methods (Kundu, Pedram, and Beerel 2021; Liang et al. 2023). Early training development is ANN-to-SNN conversion, which uses the characteristics that ANN activation can simulate the firing rate of SNN (Ding et al. 2021; Bu et al. 2022a,b; Hu et al. 2023; Hao et al. 2023). Sharmin et at. conducted conversion-based attacks by generating an ANN with weights identical to an SNN and producing perturbation with ReLU activation. Another popular progress in SNN learning is backpropagation through time (BPTT) due to its high scalability and performance (Neftci, Mostafa, and Zenke 2019; Wu et al. 2018, 2021). BPTT overcomes the non-differentiable spike activation problem using surrogate functions. Liang et al. successfully constructed effective attacks with BPTT (Liang et al. 2023). BPTT-based attacks have proved to be more powerful than conversion-based ones (Sharmin et al. 2019). Note that the training and attacks for possible adversarial training of SNN in this paper are all based on BPTT.

As for SNN robustness, some works start from the computational components in SNN. El-Allami et al. proposed to improve robustness by searching for neuron leaky parameters and inference time-steps (El-Allami et al. 2021). Kundu et al. constructed a robust weight-tuning strategy by training with noisy data (Kundu, Pedram, and Beerel 2021). Ding et al. performed Lipschitz analysis on SNN and gave a regularized training strategy (Ding et al. 2022). They also introduced a gradient approximation called BPTR that can also threaten SNN. Liang et al. analyzed the certified robustness of SNN and gave the corresponding training methods (Liang et al. 2022). These algorithms are improvements based on simplified LIF neurons and do not take full advantage of biologically-inspired SNNs. Other works originate from an explicit and ubiquitous feature of the nervous system: stochasticity. Dapello et al. explained that the CNN model based on V1 stochasticity could resist perturbations (Dapello et al. 2020). In SNN, Poisson coding encodes data using Bernoulli sampling. Sharmin et al. argued that this coding could bring perturbation immunity to SNNs (Sharmin et al. 2020). Kim et al. discussed the robust superiority of Poisson coding over direct coding (Kim et al. 2022). The current understanding of Poisson coding is mainly experimental, lacking theoretical guidance. Besides, we have little knowledge of the performance of SNNs under adversarial attacks when stochasticity is imposed on the entire network. Therefore, this work tries to introduce stochasticity to SNNs and study the effect.

Preliminaries

Spiking Neural Networks

Neurons in SNN simplify the biological ones into three key procedures: synaptic integration, first-order dynamics, and neuronal firing. The most commonly used model in the field of deep SNN is the leaky integrate-and-fire (LIF) model (Deng et al. 2021; Kim and Panda 2021). The overall modeling of the dynamics of neurons in layer l is presented in Eqs. 1, 2, 3 ($l = 1, 2, 3, \cdots, L$).

$$\boldsymbol{m}^l = \boldsymbol{W}^l \boldsymbol{s}^{l-1}(t) \tag{1}$$

$$\boldsymbol{u}^l(t) = \lambda \boldsymbol{u}^l(t-1) \odot \left(\mathbf{1} - \boldsymbol{s}^l(t-1)\right) + \boldsymbol{m}^l \tag{2}$$

$$\boldsymbol{s}^l(t) = H\left(\boldsymbol{u}^l(t) - V_{th}\right). \tag{3}$$

The injected electrical signal $\boldsymbol{m}^l$ is the summation of spike signals $\boldsymbol{s}^{l-1}$ from the preceding layer $l-1$ scaled by weights $\boldsymbol{W}^l$. $\odot$ performs the element-wise product. The membrane potential $\boldsymbol{u}^l(t)$ receives the injected signal and decays by the

leaky factor of λ, which takes value in $(0, 1]$. The spike signal is produced by comparing the membrane potential with V_{th}, which is accomplished by a Heaviside function $H(\cdot)$. After the spikes are produced for some neurons, the membrane potential of these neurons is set to 0.

Adversarial Attacks for SNN with Differential Surrogates

With the help of surrogate functions, backpropagation is enabled in the network. The backpropagation of the gradient is derived by unfolding the dynamics of LIF neurons and applying surrogate functions to the non-differentiable Heaviside function. Please refer to supplementary materials for detailed gradient formulations. Various surrogate functions have been demonstrated to be effective in previous literature (Neftci, Mostafa, and Zenke 2019; Zheng et al. 2021). In this paper, we adopt the rectangular function suggested by (Zheng et al. 2021).

The gradient-based adversarial attack scheme will look for a feasible attack perturbation $\boldsymbol{\delta}$ in a l_p-constrained neighborhood $B(\boldsymbol{x}, \epsilon)$ centered on the raw data $\boldsymbol{x}$, which can be expressed in Eq. 4.

$$\boldsymbol{\delta} = \underset{\boldsymbol{\delta} \in B(\boldsymbol{x}, \epsilon)}{\arg\max} \mathcal{L}\left(\boldsymbol{x} + \boldsymbol{\delta}, y\right), \tag{4}$$

where $\mathcal{L}$ is the task loss, y is the target label. The maximization problem is typically performed via several iterations with the support of gradient w.r.t. $\boldsymbol{x}$, which have evoked many works such as Projected Gradient Descent (PGD) (Madry et al. 2018), Fast Gradient Sign Method (FGSM) (Goodfellow, Shlens, and Szegedy 2015), Momentum-based Iterative Method (MIM) (Dong et al. 2018) and so on. For SNN, researchers noted that the combination of gradient-based attacks and surrogate functions had posed a significant security problem to SNN (Ding et al. 2022; Bu et al. 2023). To perform an attack on images, one should first encode images into sequences $\varsigma\left(\boldsymbol{x}\right) = \boldsymbol{s}^0$ where $\varsigma : \mathbb{R}^{C \times H \times W} \rightarrow \mathbb{R}^{T \times C \times H \times W}$ .T is the number of time-steps. C, H, W are the number of channel, height, and width, respectively. The most common coding method is to replicate $\boldsymbol{x}$ in the temporal dimension (Kim et al. 2022), such that the gradient w.r.t $\boldsymbol{x}$ can be worked out from the gradient w.r.t $\boldsymbol{s}^0$ as suggested in Eq. 5. Note that we simplify the notation $\nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{x}, y\right)$ to $\nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{x}\right)$ in the following:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}} = \nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{x}, y\right) = \frac{1}{T} \sum_t \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}^0\left(t\right)}. \tag{5}$$

Local Linearization of Robust Training

With the condition that the perturbation $\boldsymbol{\delta}$ is in the neighborhood $B(\boldsymbol{x}, \epsilon)$, unified modeling to the effect of perturbation is to measure $\mathcal{L}\left(\boldsymbol{x} + \boldsymbol{\delta}\right) - \mathcal{L}\left(\boldsymbol{x}\right)$, which is the difference of the loss value before and after perturbation. Theoretical justifications typically process this difference with local linearity techniques (Qin et al. 2019), which can be expressed in Eq. 6:

$$\left|\mathcal{L}\left(\boldsymbol{x} + \boldsymbol{\delta}\right) - \mathcal{L}\left(\boldsymbol{x}\right)\right| \leqslant \left|\boldsymbol{\delta} \odot \nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{x}\right)\right|_1 + g\left(\boldsymbol{\delta}, \boldsymbol{x}\right), \tag{6}$$

where $g\left(\boldsymbol{\delta}, \boldsymbol{x}\right)$ is a residual item, $\left|\cdot\right|_1$ is l_1 norm for vector. In this case, one can consider the adversarial training methods as minimizing the vanilla loss with a regularizer related to a certain attack method, which is shown in Eq. 7.

$$\min \mathcal{L}\left(\boldsymbol{x} + \boldsymbol{\delta}\right) \tag{7}$$

$$\rightarrow \min \mathcal{L}\left(\boldsymbol{x}\right) + \lambda \left|\boldsymbol{\delta} \odot \nabla_{\boldsymbol{x}} \mathcal{L}\left(\boldsymbol{x}\right)\right|_1 + \mu g\left(\boldsymbol{\delta}, \boldsymbol{x}\right), \tag{8}$$

where λ and μ are parameters. This motivates research on exploiting regularization methods to enhance the robustness (Kurakin, Goodfellow, and Bengio 2017; Ross and Doshi-Velez 2018; Roth, Kilcher, and Hofmann 2020).

The situation for SNNs is slightly different from that for ANNs. The perturbed $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\delta}$ is encoded to spike trains on which the perturbation is considered. Suppose that the input can produce spike response for layer l at time-step t through forward-propagation $f_{1:l,t}(\cdot)$, then $\boldsymbol{s}^l\left(t\right) = f_{1:l,t}\left(\boldsymbol{x}\right)$, $\tilde{\boldsymbol{s}}^l\left(t\right) = f_{1:l,t}\left(\tilde{\boldsymbol{x}}\right)$. The spike response for all time-steps can be represented as $\boldsymbol{s}^l = \text{concat}\left(\boldsymbol{s}^l\left(1\right), \boldsymbol{s}^l\left(2\right), \cdots, \boldsymbol{s}^l\left(T\right)\right) \in \{0, 1\}^{T \times N_l}$ where N_l is the number of neurons in layer l. Thus, when we consider the absolute difference of task loss w.r.t. perturbed and clean spike trains for layer l ($\Delta\mathcal{L}_{l:L}$), similar to Eq. 6, we have:

$$\Delta\mathcal{L}_{l:L} = \left|\mathcal{L}_{l:L}\left(\tilde{\boldsymbol{s}}^l\right) - \mathcal{L}_{l:L}\left(\boldsymbol{s}^l\right)\right| \tag{9}$$

$$\leqslant \sum_{t=1}^{T} \left|\boldsymbol{\xi}^l\left(t\right) \odot \nabla_{\boldsymbol{s}} \mathcal{L}_{l:L}\left(\boldsymbol{s}^l\right)\left(t\right)\right|_1 + g_{l:L}\left(\boldsymbol{\xi}^l, \boldsymbol{s}^l\right), \tag{10}$$

where the perturbation for $\boldsymbol{s}^l$ follows $\boldsymbol{\xi}^l\left(t\right) = \tilde{\boldsymbol{s}}^l\left(t\right) - \boldsymbol{s}^l\left(t\right)$, $\mathcal{L}_{l:L}$ is the loss function from layer l to the last layer, $g_{l:L}$ is the residual. As there are only items of 0 and 1 in $\boldsymbol{s}^l(t)$ and $\tilde{\boldsymbol{s}}^l(t)$, items in $\boldsymbol{\xi}^l(t)$ can only take values in $\{-1, 0, 1\}$. We denote $d_{l:L}\left(\boldsymbol{\xi}^l, \boldsymbol{s}^l\right) = \sum_{t=1}^{T} \left|\boldsymbol{\xi}^l\left(t\right) \odot \nabla_{\boldsymbol{s}} \mathcal{L}_{l:L}\left(\boldsymbol{s}^l\right)\left(t\right)\right|_1$ as the item of first-order Taylor expansion of the adversarial loss. We emphasize the essence of this item when the loss function follows the local Lipschitz property such that the residual item can be bounded by the first-order item multiplied by some positive value. In this case we can rewrite Eq. 9 as:

$$\Delta\mathcal{L}_{l:L} \leqslant M d_{l:L}\left(\boldsymbol{\xi}^l, \boldsymbol{s}^l\right), \tag{11}$$

where M is a constant. By looking at $d_{l:L}\left(\boldsymbol{\xi}^l, \boldsymbol{s}^l\right)$, we can gain some insights into improving the robustness of SNNs. First, $\nabla_{\boldsymbol{s}} \mathcal{L}_{l:L}\left(\boldsymbol{s}^l\right)\left(t\right)$ can be expressed as ${\boldsymbol{W}^{l+1}}^T \nabla_{\boldsymbol{u}} \mathcal{L}_{l:L}\left(\boldsymbol{u}^{l+1}\right)\left(t\right)$ according to Eqs. 1, 2. Therefore, by constraining the norm of $\boldsymbol{W}^{l+1}$, the robustness can be improved (Ding et al. 2022). However, the difficulty in doing so lies in how to design an effective regularization to be applied to the weights. Another straightforward and possible solution is to reduce the strength of the perturbation $\boldsymbol{\xi}^l(t)$ by some means, which motivates this work. The nervous system has many mechanisms to regulate the informative coding during signal transmission. The advantage of this solution is that the proposed model can be applied to SNN wherever spiking neurons or neural coding are used.

Methods: Stochastic Gating for Robust SNN

In this section, we will introduce the proposed stochastic gating model (StoG) for SNN and give relevant theoretical analysis and insights. Based on our theoretical analysis, a corresponding training strategy with StoG models is proposed.

Regulating Spike Trains with Stochastic Gating

The dynamics of LIF show that the spiking output from the lower layer is directly integrated into the upper layer input signal through weights (Eq. 1). However, in the nervous system, the stochastic release of neurotransmitters and the existence of voltage-gated ion channels composes a nondeterministic signal channel. Neuronal signal transportation relies on releases of synaptic vesicles and results in noisy ionic currents (Faisal, Selen, and Wolpert 2008; White, Rubinstein, and Kay 2000). A series of complex models regarding the calcium, potassium, and sodium channel gating have been proposed (Bicknell and Goodhill 2016; Lema and Auerbach 2006; Peracchia 2004). We view the nondeterministic signal channel as a gating process that can be modeled as a continuous-time Markov process and happens before spatial summation. To simplify the model, only two states of the channel are considered: open or closed state. Suppose $\kappa_{o;i,l}$ and $\kappa_{c;i,l}$ are values indicating the transition probability of opening and closing, then the distribution of open and closed states is given by:

$$\bar{\boldsymbol{P}}\left(i,l\right)=\left[\begin{array}{c}\bar{P}_o\left(i,l\right)\\ \bar{P}_c\left(i,l\right)\end{array}\right]=\left[\begin{array}{c}\kappa_{c;i,l}/\left(\kappa_{c;i,l}+\kappa_{o;i,l}\right)\\ \kappa_{o;i,l}/\left(\kappa_{c;i,l}+\kappa_{o;i,l}\right)\end{array}\right], \tag{12}$$

where $\bar{P}_o\left(i,l\right)$ and $\bar{P}_c\left(i,l\right)$ is the converged stationary probability of open and closed states. The derivation of Eq. 12 refers to the supplementary materials. When modeling a two-state gating model, we perform a Monte Carlo simulation on the stationary distribution. In the model, a parameter is assigned for each neuron, indicating the probability of gate opening. We denote this probability as $\bar{\boldsymbol{P}}_o^l=\left\{\bar{P}_o\left(i,l\right)|i=1,2,\cdots,N_l\right\}$ where N_l is the number of neurons. With $\bar{\boldsymbol{P}}_o^l$, we can sample the gating vector from a Bernoulli distribution for each discrete time-step:

$$\boldsymbol{G}^l\left(t\right)\sim\text{Bernoulli}\left(\bar{\boldsymbol{P}}_o^l\right). \tag{13}$$

The items in $\boldsymbol{G}$ take values in 0 and 1. And the neuronal charge in Eq. 1 can be altered to Eq. 14:

$$\boldsymbol{m}^l=\boldsymbol{W}^l\left(\boldsymbol{G}^l\left(t\right)\odot\boldsymbol{s}^{l-1}\left(t\right)\right). \tag{14}$$

Capability of Linearly Resisting Perturbations

Recall that by linearization techniques, we can obtain the relationship of loss, perturbation, and gradients. After applying the perturbation, the spike signals are modulated by the proposed model, causing the actual perturbation to be indirectly modulated as well, expressed as:

$$\boldsymbol{s}^l\left(t\right)\xrightarrow{gating}\boldsymbol{s}'^l\left(t\right)=\boldsymbol{s}^l\left(t\right)\odot\boldsymbol{G}^{l+1}\left(t\right), \tag{15}$$

$$\boldsymbol{\xi}^l\left(t\right)\xrightarrow{gating}\boldsymbol{\xi}'^l\left(t\right)=\boldsymbol{\xi}^l\left(t\right)\odot\boldsymbol{G}^{l+1}\left(t\right). \tag{16}$$

To simplify the discussion, we only analyze the changes in the adversarial loss when a gating model is added to the layer l and suppose the loss function follows the local Lipschitz property. Then, the first-order expansion item $d_{l:L}$ and $\Delta\mathcal{L}_{l:L}$ can be rewritten as:

$$d_{l:L}\left(\boldsymbol{\xi}^l,\boldsymbol{s}^l\right)\xrightarrow{gating}d_{l:L}\left(\boldsymbol{\xi}^l,\boldsymbol{s}^l;\boldsymbol{G}^{l+1}\right) \tag{17}$$

$$=\sum_{t=1}^{T}\left|\boldsymbol{\xi}^l\left(t\right)\odot\boldsymbol{G}^{l+1}\left(t\right)\odot\nabla_{\boldsymbol{s}'}\mathcal{L}_{l:L}\left(\boldsymbol{s}'^l\right)\left(t\right)\right|_1, \tag{18}$$

where $\boldsymbol{s}'$ replaces $\boldsymbol{s}$ in the original loss function.

From Eq. 11, we can know that $\Delta\mathcal{L}_{l:L}$ is affected by the constraint of $d_{l:L}\left(\boldsymbol{\xi}^l,\boldsymbol{s}^l;\boldsymbol{G}^{l+1}\right)$. And $d_{l:L}\left(\boldsymbol{\xi}^l,\boldsymbol{s}^l;\boldsymbol{G}^{l+1}\right)$ is controlled by $\boldsymbol{G}^{l+1}$ (Eq. 17). Theorem 1 gives the relationship of $d_{l:L}\left(\boldsymbol{\xi}^l,\boldsymbol{s}^l;\boldsymbol{G}^{l+1}\right)$ and $\boldsymbol{G}^{l+1}$ such that by adjusting $\bar{\boldsymbol{P}}_o^{l+1}$, $\Delta\mathcal{L}_{l:L}$ can be constrained.

Theorem 1. *For neurons in layer l, suppose that $\boldsymbol{s}^l$ is the spike response at time-step t ($t=1,2,3,\cdots,T$, $l=1,2,3,\cdots,L$), $\boldsymbol{\xi}^l(t)$ is the vector of perturbations at time-step t, $\boldsymbol{G}^{l+1}\sim\text{Bernoulli}\left(\bar{\boldsymbol{P}}_o^{l+1}\right)$ is the gating vector. $\bar{\boldsymbol{P}}_o^{l+1}$ is the probability of gate opening in which items take values at $[0,1]$. $d_{l:L}\left(\boldsymbol{\xi}^l,\boldsymbol{s}^l;\boldsymbol{G}^{l+1}\right)=\sum_{t=1}^{T}\left|\boldsymbol{\xi}^l\left(t\right)\odot\boldsymbol{G}^{l+1}\left(t\right)\odot\nabla_{\boldsymbol{s}'}\mathcal{L}_{l:L}\left(\boldsymbol{s}'^l\right)\left(t\right)\right|_1$. Then, it satisfies that:*

$$\mathop{\mathbb{E}}_{G\sim B(\bar{P}_o)}\left(d_{l:L}\left(\boldsymbol{\xi}^l,\boldsymbol{s}^l;\boldsymbol{G}^{l+1}\right)\right)\leqslant\Omega T\sum_{i=1}^{N_l}\bar{\boldsymbol{P}}_o^{l+1}, \tag{19}$$

where $\Omega=\max_{\boldsymbol{s}',t}\sqrt{\sum_{i=1}^{N_l}\nabla_{\boldsymbol{s}'}\mathcal{L}_{l:L}\left(\boldsymbol{s}'^l\right)\left(t\right)^2}$.

Through Theorem 1, we can realize that adding a gating model to SNN and adjusting the parameter $\bar{\boldsymbol{P}}_o^{l+1}$, we can expect to achieve linear constraints for adversarial loss, formally expressed as:

$$\mathop{\mathbb{E}}_{G\sim B(\bar{P}_o)}\left(\Delta\mathcal{L}_{l:L}\right)\leqslant M\mathop{\mathbb{E}}_{G\sim B(\bar{P}_o)}\left(d_{l:L}\left(\boldsymbol{\xi}^l,\boldsymbol{s}^l;\boldsymbol{G}^{l+1}\right)\right) \tag{20}$$

$$\leqslant M\Omega T\sum_{i=1}^{N_l}\bar{\boldsymbol{P}}_o^{l+1}. \tag{21}$$

When reducing the value of $\text{sum}\left(\bar{\boldsymbol{P}}_o^{l+1}\right)$, the robustness of the model is expected to improve. A special case is when $\text{sum}\left(\bar{\boldsymbol{P}}_o^{l+1}\right)\to 0$, we have $\mathop{\mathbb{E}}_{G\sim B(\bar{P}_o)}\left(\Delta\mathcal{L}_{l:L}\right)\to 0$. In this case, no matter what perturbation is applied, the value of the adversarial loss will not be modified. Note that $\text{sum}\left(\bar{\boldsymbol{P}}_o^{l+1}\right)$ should not be too small, as this will also make the probability of input data passing gating low, making the signal difficult to be transmitted. Therefore, to improve the robustness of SNNs, in addition to adding stochastic gating before the

synaptic transmission, an adaptive mechanism for acquiring gate opening probabilities is also required, which will be detailed in a joint training scheme.

Impact on Local Lipschitz Property of Neurons

We have considered the impact of stochastic gating from a global perspective through the difference between perturbed and clean loss. Another perspective is to consider the local properties of the forwarding process. Researchers have deduced the Lipschitz property of neural networks to illustrate the problem of error amplification in the forward process (Cisse et al. 2017; Ding et al. 2022; Lin, Gan, and Han 2019). Here, we can also analyze the influence of channel gating on the forward process. First, we select the metric of the two spike sequences as $\mathcal{D}(\boldsymbol{s}^l, \tilde{\boldsymbol{s}}^l) = \|\boldsymbol{s}^l - \tilde{\boldsymbol{s}}^l\|_2^2$, where $\|\cdot\|_2^2$ in $\mathcal{D}$ is calculated on the spatiotemporal dimensions. Theorem 2 shows that in a feed-forward SNN, neurons with StoG models can alleviate the error amplification in layer-by-layer forwarding communication.

Theorem 2. *For neurons in layer l, suppose that $\boldsymbol{s}^l$ and $\tilde{\boldsymbol{s}}^l$ is the clean and perturbed spike response at time-step t ($t = 1, 2, 3, \cdots, T$, $l = 1, 2, 3, \cdots, L$). The metric of the two spike sequences is defined as $\mathcal{D}(\boldsymbol{s}^l, \tilde{\boldsymbol{s}}^l) = \|\boldsymbol{s}^l - \tilde{\boldsymbol{s}}^l\|_2^2$. N_l is the number of neurons in layer l. $\boldsymbol{G}^{l+1} \sim \text{Bernoulli}\left(\bar{\boldsymbol{P}}_o^{l+1}\right)$ is the gating vector for layer l. $\bar{\boldsymbol{P}}_o^{l+1}$ is the probability of gate opening in which items take values at $[0, 1]$. V_{th} is the threshold of neurons. Then, it satisfies that:*

$$\mathop{\mathbb{E}}_{G\sim B(\bar{P}_o)}\left[\mathcal{D}(\boldsymbol{s}^L, \tilde{\boldsymbol{s}}^L)\right] \tag{22}$$

$$\leqslant \mathop{\mathbb{E}}_{G\sim B(\bar{P}_o)}\left[\mathcal{D}(\boldsymbol{s}^l, \tilde{\boldsymbol{s}}^l)\right] \cdot \prod_{j=l}^{L-1}\left[\frac{T\rho^{j+1}}{V_{th}^2}\sum_{i=1}^{N_j}\bar{\boldsymbol{P}}_o^{j+1}\right] + C, \tag{23}$$

where ρ^l is a Lipschitz constant induced by weights for layer l, and C is a constant.

From Theorem 2, we can find that as the signal propagates layer by layer, the influence of the perturbation is not only controlled by the weights but also affected by $\bar{\boldsymbol{P}}_o^l$, and the two effects with no interference to each other. In other words, stochastic gating can be simultaneously applied to SNN with weight Lipschitz constraints.

Poisson Coding as a Special Form of Stochastic Gating

From Eq. 15, s becomes s' after the sampling of the Bernoulli distribution of open probability. Interestingly, Poisson coding commonly used in SNN is also sampled by Bernoulli distribution (Diehl et al. 2015; Sharmin et al. 2020). Compared with Eq. 15, the process of Poisson coding can be described as:

$$\text{Poisson coding}: \boldsymbol{s}'^0(t) = 1 \odot \boldsymbol{X}, \boldsymbol{X} \sim \text{Bernoulli}(\boldsymbol{x}), \tag{24}$$

where $\boldsymbol{X}$ is the encoding of the input image $\boldsymbol{x}$. $\boldsymbol{s}'^0$ represents the output of the final Poisson coding. Poisson coding is equivalent to using $\boldsymbol{x}$ as the probability of gate opening $\bar{\boldsymbol{P}}_o^1$ in the input layer. In this case, the original $\boldsymbol{s}^0$ is an input that is always 1.

Prior to this work, Poisson coding was considered to be able to bring inherent robustness to SNNs (Kim et al. 2022; Sharmin et al. 2019, 2020), but it is only explained experimentally. Our work shows that the robustness of Poisson coding also has a theoretical basis from Theorems 1 and 2 by constructing the equivalence between Poisson coding and StoG model. The adversarial loss can be affected by the mean value of input data $\boldsymbol{x}$.

Joint Training with Stochastic Gating

Effectively training SNNs with StoG involves two issues, one is how to train the parameters in StoG, and the other is how to penalize the StoG parameters for the purpose of achieving robustness.

For the first issue, we use the Gumbel-Max reparameterization trick presented in (Maddison, Mnih, and Teh 2017), and set the temperature to 0.5. When forwarding each layer of StoG model, we sample $\boldsymbol{U} \sim \text{Uniform}(0, 1)$ such that $\boldsymbol{G} = H\left(\log\left(\bar{\boldsymbol{P}}_o/\left(1-\bar{\boldsymbol{P}}_o\right)\right) + \log \boldsymbol{U} - \log\left(1-\boldsymbol{U}\right)\right)$ where H is the Heaviside function. The gradient of $\bar{\boldsymbol{P}}_o$ is induced by the corresponding binary concrete random variable. Note that $\bar{\boldsymbol{P}}_o$ cannot be too small, thus, we do not allow $\bar{\boldsymbol{P}}_o$ to be less than 0.1 during training.

For the second issue, we consider that the training loss function is given by three parts, the task loss followed by two regularizers, which are respectively responsible for punishing the weight $\boldsymbol{W} = \left\{\boldsymbol{W}^i, i = 1, 2, 3, \cdots, L-1\right\}$ and StoG parameters $\bar{\boldsymbol{P}}_o = \left\{\bar{\boldsymbol{P}}_o^i, i = 1, 2, 3, \cdots, L-1\right\}$. The overall loss can be expressed as:

$$\mathcal{L} = \mathcal{L}_{task}\left(\boldsymbol{x}; \boldsymbol{W}, \bar{\boldsymbol{P}}_o\right) + \gamma \text{Reg}_P\left(\bar{\boldsymbol{P}}_o\right) + \beta \text{Reg}_W\left(\boldsymbol{W}\right). \tag{25}$$

where Reg_P is the penalty of $\bar{\boldsymbol{P}}_o$, and γ is its strength. Reg_W is the penalty for the weight, and β is its strength. In this work, we apply l_2 penalty to $\bar{\boldsymbol{P}}_o$.

Experiments

Experimental Setup

To verify the effectiveness of our method, we conduct experiments on classification tasks on the CIFAR-10 and CIFAR-100 datasets (Krizhevsky, Hinton et al. 2009). The SNN architecture used is VGG-11 (Simonyan and Zisserman 2015) and Wide-ResNet-16-4 (Zagoruyko and Komodakis 2016) (WRN-16-4 for short). The SNN accepts directly encoded input, and the inference time-step is 8. The experiments are conducted on GPU devices of the NVIDIA RTX 3090 with PyTorch (v1.12.1).

We adopted three training strategies to determine the effectiveness of the method. The first is a vanilla training scheme (BPTT), directly using raw images for training. The second is an adversarial training strategy, which uses samples from white-box (WB) FGSM attacks ($\epsilon = 2/255$) for training (Goodfellow, Shlens, and Szegedy 2015) (abbreviated as AT). The third is to add a Lipschitz penalty proposed

CIFAR-10, VGG-11					
BPTT Attack	Clean	FGSM-UT	PGD-l_∞-UT	FGSM-RT	PGD-l_∞-RT
Vanilla	93.05	12.63/86.44	0.04/99.96	28.82/71.75	12.89/86.87
Vanilla+StoG	91.64	**16.22/82.32**	**0.28/99.69**	**33.38/66.40**	**16.97/82.67**
AT	91.13	41.74/54.20	23.37/74.36	60.37/37.47	54.54/41.78
AT+StoG	90.13	**45.75/49.29**	**27.74/69.26**	**65.54/30.78**	**59.47/36.73**
RAT	90.39	48.46/46.39	28.08/68.94	63.69/33.60	62.24/33.08
RAT+StoG	88.02	**55.98/36.28**	**39.58/54.86**	**70.63/23.98**	**68.52/25.35**
CIFAR-100, VGG-11					
	Clean	FGSM-UT	PGD-l_∞-UT	FGSM-RT	PGD-l_∞-RT
Vanilla	73.38	5.30/92.82	0.04/99.95	11.96/88.16	3.59/95.67
Vanilla+StoG	70.44	**8.27/88.34**	**0.49/99.31**	**15.93/81.54**	**8.57/89.17**
AT	67.29	20.58/69.42	11.41/83.04	35.20/54.38	40.68/43.48
AT+StoG	66.37	**24.45/63.03**	**14.42/78.15**	**40.00/45.51**	**41.50/42.43**
RAT	67.41	29.23/56.65	16.52/75.49	47.51/36.14	51.59/28.54
RAT+StoG	62.26	**33.40/46.55**	**23.15/62.99**	**55.95/18.10**	**53.40/21.45**

Table 1: Performance under attacks. Items in the table follow the format of "robust accuracy/ASR". "UT" and "RT" denote "Untargeted" and "Random Targeted" attacks, respectively.

EOT-10-AutoPGD	CIFAR-10 VGG-11	CIFAR-10 WRN-16-4	CIFAR-100 VGG-11	CIFAR-100 WRN-16-4
Vanilla	2.11/6.07	0.09/0.67	1.04/2.55	0.25/0.75
Vanilla+StoG	**11.58/18.70**	**3.35/4.79**	**9.20/11.10**	**6.19/7.69**
AT	51.33/57.09	56.40/62.73	26.18/30.95	31.34/37.26
AT+StoG	**62.14/65.99**	**62.71/67.34**	**38.33/40.77**	**36.60/40.75**
RAT	55.88/63.06	59.73/69.06	33.41/41.41	34.73/47.74
RAT+StoG	**60.21/65.15**	59.61/66.29	**39.13/46.03**	34.60/41.87

Table 2: Performance of EOT-AutoPGD attacks for different SNN architectures. Items in the table follow the format of "BPTT-based/BPTR-based robust accuracy".

in (Ding et al. 2022) to the weights under the adversarial training setting (abbreviated as RAT). For all three strategies, we test their robustness with and without the proposed model. To punish $\bar{\boldsymbol{P}}_o$, we set $\gamma = 5 \times 10^{-6}$ by default. We train our model with white-box FGSM adversarial examples on each mini-batch of images. The perturbation boundary is $2/255$ (Kundu, Pedram, and Beerel 2021). See our code for the rest of the training setups.

Robustness verification employs a comprehensive approach. The basic attack method is chosen between the fast gradient sign method (FGSM) (Goodfellow, Shlens, and Szegedy 2015) and project gradient descent (Madry et al. 2018), which are the most common in SNN robustness verification. We implement l_2 and l_∞ versions of PGD attacks, respectively. We also extend our analyses with the AutoPGD attack, which can produce optimal perturbation (Croce and Hein 2020) and augment the gradients for attack with expectation-over-transformation (EOT). The EOT step is set to 10 by default, as recommenced in the literature (Athalye et al. 2018).

We consider several aspects of attacks: untargeted and targeted settings; black-box (BB) and white-box settings; and BPTT- and BPTR-based attack settings (suggested by Ding et al.). We use the BB strategy from literature (Kundu, Pedram, and Beerel 2021): The attacker does not know the defense model's parameters. The attacker trains an SNN model with the same dataset and architecture as the defense model, using BPTT with a new random seed. The black-box attacks use BPTT-based attacks.

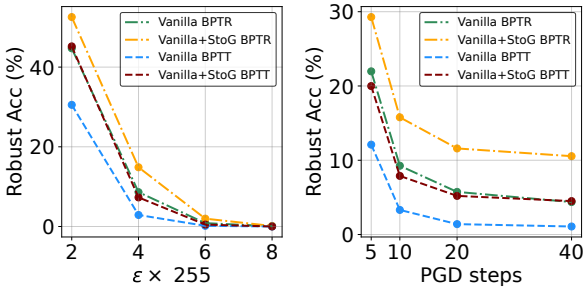
We use attack classification accuracy (robust accuracy) and attack success rate (ASR) as metrics of robustness. The attack success rate here refers to the proportion of misclassifications in the originally correctly classified data. The smaller the ASR, the better the robustness.

BPTT Attack	Untargeted	Targeted
Vanilla	2.11/97.73	1.10/98.82
EOT-1 Vanilla+StoG	19.33/80.24	22.36/77.11
EOT-10 Vanilla+StoG	11.56/88.54	14.63/85.48

Table 3: Performance of AutoPGD attacks with different EOT steps. Items in the table follow the format of "robust accuracy/ASR".

Performance

Table 1 shows the robustness of CIFAR-10 and CIFAR-100 under FGSM and PGD-l_∞ attacks. For experimental fairness, the models shown in the table are trained and reasoned with the same random seeds. The intensity of the FGSM



(a) Effect of intensity (b) Effect of PGD steps

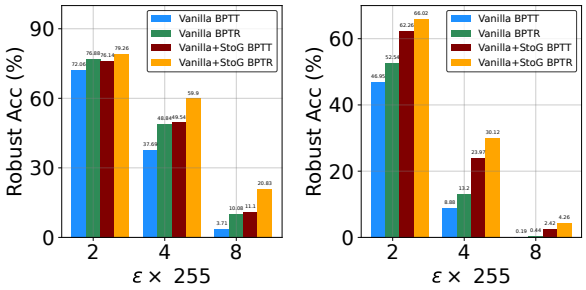
Figure 2: Performance against EOT-PGD attacks with increasing intensity and steps.

attack is $8/255$. For the PGD-l_∞ attack, the overall intensity, step number, and step size are fixed to $8/255$, 7, and 0.01, respectively. The results show that for the three training strategies, our proposed StoG can reduce the ASR and improve robust accuracy. The robustness performance of the proposed model is similar under untargeted and random-targeted attacks. This verifies that our method can work with and without adversarial training.

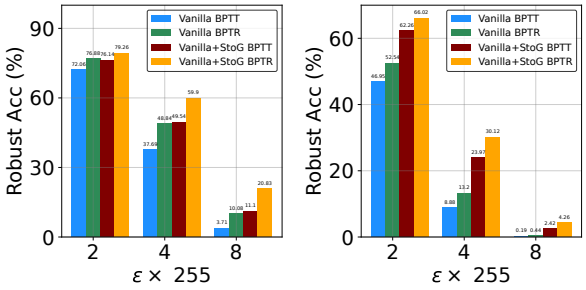
Due to randomness in our proposed method, we further use EOT-AutoPGD attacks with 10 EOT steps, 10 PGD steps, and an intensity of $4/255$. We report our results in Table 2, which records the robustness of different architectures under untargeted EOT-AutoPGD attacks. The reported accuracies are either from BPTT- or BPTR-based attacks. In the table, the robust accuracy of most of the SNNs that added the StoG model has increased. This implies that StoG modules have the potential to increase robustness for various SNN architectures. We also test the effect of EOT steps and list the results in Table 3. Compared with attacks with one EOT step, EOT-10-AutoPGD poses a stronger threat to vanilla+StoG models. However, the performances are still higher than in models without StoG modules.

We observe that no clear gradient obfuscation is invoked by adding stochastic gating. In previous works (Kundu, Pedram, and Beerel 2021; Ding et al. 2022) showed no obvious obfuscation with BPTT- or BPTR-based attacks on SNNs. Here, we use randomly initialized PGD attacks with 10 EOT steps to check for gradient obfuscation. We apply perturbations to vanilla and vanilla+StoG VGG-11 models trained on CIFAR-10, respectively. We display the perturbation results by increasing PGD intensity and steps in Fig. 2. Our results show that model robustness decreases with more PGD steps and reaches an asymptote after 40 steps. Also, higher attack intensities can fail the SNN models for both BPTT- and BPTR-based attacks.

Finally, we test how the accuracy changes when the models encounter l_2 and BB attacks. We use EOT-PGD as an attack model and change the attack intensity. The PGD step size α satisfies $\alpha = 2.5\epsilon/K$ where K is the number of PGD steps. The results of BPTT- and BPTR-based attacks are shown in Fig. 3. For VGG and WRN architectures, they both show similar trends in robustness. For CIFAR-10 VGG-11 and $\epsilon = 8/255$, the BB BPTR-based robust accuracy

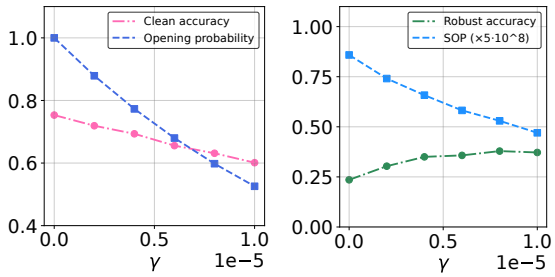


(a) l_2, WB, VGG-11 (b) l_2, WB, WRN-16-4



(c) l_∞, BB, VGG-11 (d) l_∞, BB, WRN-16-4

Figure 3: Performance under EOT-PGD attacks with l_2 and black-box settings.



(a) Clean accuracy (b) Spike Operation

Figure 4: Effect of stochastic gating models on WRN-16-4 architectures.

for vanilla+StoG SNN is higher than that of vanilla SNN by 10.75%.

Effect of Stochastic Gating

The StoG model brings robustness by filtering spikes, thus potentially having low power consumption. We visualize the mean of $\bar{\boldsymbol{P}}_o$, clean and robust accuracy (attacked by FGSM($\epsilon = 2/255$)), and estimate of energy consumption for different γ settings ($[0, 1 \times 10^{-5}]$). We estimate the SNN energy consumption by the number of Spike Operations (SOP) suggested in (Kundu et al. 2021). Compared to the improvement in robust accuracy, γ has little effect on clean accuracy. At the same time, a larger γ results in a smaller $\bar{\boldsymbol{P}}_o$ and fewer SOP. We display results for WRN-16-4 in Fig. 4. For VGG-11, $\gamma = 1 \times 10^{-5}$ reduces the average opening probability to 0.52 and the SOP by 1.428×.

Conclusions and Discussions

Conclusion: To improve SNN robustness, we propose a simple yet effective method to simulate the stochastic gating process and filter the perturbed spike sequence. Theoretical analyses show that our method can be regarded as adding regularization to SNN training. At the same time, our theorems can be used to explain why Poisson coding can bring natural robustness to SNN. We hope that this method will be used in SNN applications with low energy budgets and high robustness.

Limitation: The limitation of the designed method is that it sacrifices a little accuracy in order to enhance robustness. When there is low opening probabilities, it is conceivable that almost all signals are filtered out, which makes it difficult to improve the model's performance. It is important to find a method that is more adaptable, allowing for the joint optimization of accuracy, robustness, and performance. Also, the SNN-specific gradient attacks are not very well studied, which points out another future direction. Robustness-enhanced methods evaluated using large EOT steps should be investigated.

Broader Impact: In neuroscience, the stochastic channel is commonly seen as a cause of neural noise. However, the dynamics of neurons can counteract this noise. Our research indicates that channel gating, despite introducing noise to the signal, can actually enhance the resilience of complex cognitive processes like recognition and decision-making. By employing SNNs as research tools, we can contribute valuable insights to ongoing discussions in neuroscience regarding robustness.

Supplementary Material

Proofs

Proof for Theorem 1 is given as follows.

Proof. By observing the equation of $d_{l:L}$, the l_1 norm can be expressed as multiplying two vectors and taking their absolute value, and can be rewritten using Cauchy's inequality.

$$\begin{aligned} & d_{l:L}\left(\boldsymbol{\xi}^l, \boldsymbol{s}^l, \boldsymbol{G}^{l+1}\right) \\ &= \sum_{t=1}^{T}\left|\boldsymbol{\xi}^l(t) \odot \boldsymbol{G}^{l+1}(t) \odot \nabla_{\boldsymbol{s}'} \mathcal{L}_{l:L}\left(\boldsymbol{s}'^l\right)(t)\right|_1, \\ &\leqslant \sum_{t=1}^{T} \sqrt{\sum_{i=1}^{N_l} \boldsymbol{\xi}_i^l(t)^2 \boldsymbol{G}_i^{l+1}(t)^2} \sqrt{\sum_{i=1}^{N_l} \nabla_{\boldsymbol{s}'} \mathcal{L}_{l:L}\left(\boldsymbol{s}'^l\right)(t)^2}, \\ &= \sum_{t=1}^{T} \sqrt{\sum_{i=1}^{N_l}\left|\boldsymbol{\xi}_i^l(t)\right| \boldsymbol{G}_i^{l+1}(t)} \sqrt{\sum_{i=1}^{N_l} \nabla_{\boldsymbol{s}'} \mathcal{L}_{l:L}\left(\boldsymbol{s}'^l\right)(t)^2}. \end{aligned} \tag{26}$$

In the above formula, $\boldsymbol{G}^{l+1}(t)^2$ can replace $\boldsymbol{G}^{l+1}(t)$ as the items in $\boldsymbol{G}^{l+1}(t)$ can only take values at 0 or 1. $\left|\boldsymbol{\xi}_i^l(t)\right|$ can replace $\boldsymbol{\xi}_i^l(t)^2$ as the items in $\boldsymbol{\xi}_i^l(t)$ can only take values at -1 or 1.

Treat the second square root term in Eq. 26 on the right as a whole, that is, $\sqrt{\sum_{i=1}^{N_l} \nabla_{\boldsymbol{s}'} \mathcal{L}_{l:L}\left(\boldsymbol{s}'^l\right)(t)^2}$. In this term, there are two variables. One is $\boldsymbol{s}'^l$, which is the output of layer l after gating, $\boldsymbol{s}'^l \in \{0,1\}^{T \times N_l}$. The second is the time-step t, whose value is between 1 and T. Thus, we consider the term bounded, and this upper bound must be greater than 0. Here we denote the upper bound as Ω, which can be formulated as:

$$\Omega = \max_{\boldsymbol{s}', t} \sqrt{\sum_{i=1}^{N_l} \nabla_{\boldsymbol{s}'} \mathcal{L}_{l:L}\left(\boldsymbol{s}'^l\right)(t)^2}. \tag{27}$$

In this case, we have

$$d_{l:L}\left(\boldsymbol{\xi}^l, \boldsymbol{s}^l, \boldsymbol{G}^{l+1}\right) \leqslant \Omega \sum_{t=1}^{T} \sqrt{\sum_{i=1}^{N_l}\left|\boldsymbol{\xi}_i^l(t)\right| \boldsymbol{G}_i^{l+1}(t)}. \tag{28}$$

Considering that $\left|\boldsymbol{\xi}_i^l(t)\right|$ is binary. This means that any sequence multiplied by $\left|\boldsymbol{\xi}_i^l(t)\right|$ and then summed will not be greater than the direct addition of the sequence. Then, Eq. 28 can continue to be transformed into:

$$d_{l:L}\left(\boldsymbol{\xi}^l, \boldsymbol{s}^l, \boldsymbol{G}^{l+1}\right) \leqslant \Omega \sum_{t=1}^{T} \sqrt{\sum_{i=1}^{N_l} \boldsymbol{G}_i^{l+1}(t)}. \tag{29}$$

Take the expectation of the Bernoulli distribution on both sides of Eq. 29, it gives:

$$\mathbb{E}\left(d_{l:L}\left(\boldsymbol{\xi}^l, \boldsymbol{s}^l, \boldsymbol{G}^{l+1}\right)\right) \leqslant \Omega \cdot \mathbb{E}\left(\sum_{t=1}^{T} \sqrt{\sum_{i=1}^{N_l} \boldsymbol{G}_i^{l+1}(t)}\right). \tag{30}$$

Apparently,

$$\mathbb{E}\left(\sqrt{\sum_{i=1}^{N_l} \boldsymbol{G}_i^{l+1}(t)}\right) < \mathbb{E}\left(\sum_{i=1}^{N_l} \boldsymbol{G}_i^{l+1}(t)\right). \tag{31}$$

Thus,

$$\begin{aligned} \mathbb{E}\left(d_{l:L}\left(\boldsymbol{\xi}^l, \boldsymbol{s}^l, \boldsymbol{G}^{l+1}\right)\right) &\leqslant \Omega \cdot \mathbb{E}\left(\sum_{t=1}^{T} \sqrt{\sum_{i=1}^{N_l} \boldsymbol{G}_i^{l+1}(t)}\right) \\ &\leqslant \Omega \cdot \mathbb{E}\left(\sum_{t=1}^{T} \sum_{i=1}^{N_l} \boldsymbol{G}_i^{l+1}(t)\right) = \Omega T \sum_{i=1}^{N_l} \bar{\boldsymbol{P}}_o^{l+1}. \end{aligned} \tag{32}$$

□

Proof for Theorem 2 is given as follows.

Proof. Due to the addition of the StoG model, the spike input is modulated to:

$$\boldsymbol{s}'^l(t) = \boldsymbol{s}^l(t) \odot \boldsymbol{G}^{l+1}(t). \tag{33}$$

We assume that $\boldsymbol{G}^{l+1}$ takes the same matrix when passing the noisy spike response as it did without noise. According

to the implication from previous work (Lin, Gan, and Han 2019; Ding et al. 2022), the layer-by-layer error amplification of SNN can be characterized as:

$$\mathcal{D}(\boldsymbol{s}^{l+1}, \tilde{\boldsymbol{s}}^{l+1}) \leqslant \frac{\rho^{l+1}}{V_{th}^2} \mathcal{D}(\boldsymbol{s}^{l} \odot \boldsymbol{G}^{l+1}, \tilde{\boldsymbol{s}}^{l} \odot \boldsymbol{G}^{l+1}) + \Gamma^{l+1}. \tag{34}$$

In Eq. 34, ρ^{l+1} is the Lipschitz constant for layer $l+1$. Its value only relates to the weights in layer $l+1$, and satisfies:

$$\rho^{l+1} \leqslant \underset{\boldsymbol{s} \in \{-1,0,1\}^{N_l}}{\arg\max} \left\| \boldsymbol{W}^{l+1} \boldsymbol{s} \right\|_2^2 / \left\| \boldsymbol{s} \right\|_2^2. \tag{35}$$

To obtain the relation between $\mathcal{D}(\boldsymbol{s}^{l+1}, \tilde{\boldsymbol{s}}^{l+1})$ and $\mathcal{D}(\boldsymbol{s}^{l}, \tilde{\boldsymbol{s}}^{l})$, we take a deeper look at $\mathcal{D}(\boldsymbol{s}^{l} \odot \boldsymbol{G}^{l+1}, \tilde{\boldsymbol{s}}^{l} \odot \boldsymbol{G}^{l+1})$. According to the inequality of norms, we have:

$$\begin{aligned} &\mathcal{D}(\boldsymbol{s}^{l} \odot \boldsymbol{G}^{l+1}, \tilde{\boldsymbol{s}}^{l} \odot \boldsymbol{G}^{l+1}) = \left\| \left(\boldsymbol{s}^{l} - \tilde{\boldsymbol{s}}^{l} \right) \odot \boldsymbol{G}^{l+1} \right\|_2^2 \\ &\leqslant \left\| \left(\boldsymbol{s}^{l} - \tilde{\boldsymbol{s}}^{l} \right) \right\|_2^2 \left\| \boldsymbol{G}^{l+1} \right\|_2^2 = \mathcal{D}(\boldsymbol{s}^{l}, \tilde{\boldsymbol{s}}^{l}) \left\| \boldsymbol{G}^{l+1} \right\|_2^2. \end{aligned} \tag{36}$$

Since each item in the matrix $\boldsymbol{G}^{l+1}$ can only take a value of 0, 1, $\left\| \boldsymbol{G}^{l+1} \right\|_2^2$ is actually calculating:

$$\left\| \boldsymbol{G}^{l+1} \right\|_2^2 = \sum_{t=1}^{T} \sum_{i=1}^{N_l} \boldsymbol{G}_i^{l+1}(t). \tag{37}$$

Substituting Eq. 36 and Eq. 37 into Eq. 34, and taking expectations on both sides of the formula, we get:

$$\begin{aligned} &\mathbb{E}\left[\mathcal{D}(\boldsymbol{s}^{l+1}, \tilde{\boldsymbol{s}}^{l+1}) \right] \\ &\leq \frac{\rho^{l+1}}{V_{th}^2} \mathbb{E}\left[\mathcal{D}(\boldsymbol{s}^{l}, \tilde{\boldsymbol{s}}^{l}) \right] \mathbb{E}\left(\sum_{t=1}^{T} \sum_{i=1}^{N_l} \boldsymbol{G}_i^{l+1}(t) \right) + \Gamma^{l+1} \\ &= \frac{T\rho^{l+1}}{V_{th}^2} \mathbb{E}\left[\mathcal{D}(\boldsymbol{s}^{l}, \tilde{\boldsymbol{s}}^{l}) \right] \sum_{i=1}^{N_l} \bar{\boldsymbol{P}}_o^{l+1} + \Gamma^{l+1}. \end{aligned} \tag{38}$$

We can simplify the above formula to:

$$\mathbb{E}\left[\mathcal{D}(\boldsymbol{s}^{l+1}, \tilde{\boldsymbol{s}}^{l+1}) \right] \leq A^{l} \mathbb{E}\left[\mathcal{D}(\boldsymbol{s}^{l}, \tilde{\boldsymbol{s}}^{l}) \right] + B^{l}, \tag{39}$$

where $A^l = \frac{T\rho^{l+1}}{V_{th}^2} \sum_{i=1}^{N_l} \bar{\boldsymbol{P}}_o^{l+1}$, $B^l = \Gamma^{l+1}$. Thus,

$$\begin{aligned} &\mathbb{E}\left[\mathcal{D}(\boldsymbol{s}^{l+2}, \tilde{\boldsymbol{s}}^{l+2}) \right] \leqslant A^{l+1} \mathbb{E}\left[\mathcal{D}(\boldsymbol{s}^{l+1}, \tilde{\boldsymbol{s}}^{l+1}) \right] + B^{l+1} \\ &= A^{l+1} \cdot A^{l} \cdot \mathbb{E}\left[\mathcal{D}(\boldsymbol{s}^{l}, \tilde{\boldsymbol{s}}^{l}) \right] + A^{l+1} B^{l} + B^{l+1}. \end{aligned} \tag{40}$$

By analogy, we have:

$$\mathbb{E}\left[\mathcal{D}(\boldsymbol{s}^{L}, \tilde{\boldsymbol{s}}^{L}) \right] \leqslant \mathbb{E}\left[\mathcal{D}(\boldsymbol{s}^{l}, \tilde{\boldsymbol{s}}^{l}) \right] \cdot \prod_{j=l}^{L-1} A^{j} + C. \tag{41}$$

where C is a constant related to $A^{l+1}, \cdots, A^{L-1}, B^{l}, \cdots, B^{L-1}$.

□

The derivation of Eq. 12 is given as follows.

$\boldsymbol{K}_{i,l}$ is the transition probability matrix between open and closed states. $\boldsymbol{P}(t; i, l)$ is the vector of the probability of open and closed states for neuron i in layer l.

$$\boldsymbol{P}(t + \Delta t; i, l) = \boldsymbol{K}_{i,l} \boldsymbol{P}(t; i, l), \tag{42}$$

$$\boldsymbol{K}_{i,l} = \begin{bmatrix} 1 - \kappa_{o;i,l} \Delta t & \kappa_{c;i,l} \Delta t \\ \kappa_{o;i,l} \Delta t & 1 - \kappa_{c;i,l} \Delta t \end{bmatrix} \tag{43}$$

The state transition matrix reveals the Markov chain's irreducibility, allowing access to another state. The Renewal Theorem (Alsmeyer 1994) can be used to calculate stationary probabilities:

$$\bar{\boldsymbol{P}}(i, l) = \left[\bar{P}_o(i, l), \bar{P}_c(i, l) \right]^T, \tag{44}$$

$$\bar{\boldsymbol{P}}(i, l) = \boldsymbol{K}_{i,l} \bar{\boldsymbol{P}}(i, l), \tag{45}$$

$$\bar{P}_o(i, l) + \bar{P}_c(i, l) = 1. \tag{46}$$

Thus, we can obtain:

$$\bar{P}_o(i, l) = \kappa_{c;i,l} / \left(\kappa_{c;i,l} + \kappa_{o;i,l} \right), \tag{47}$$

$$\bar{P}_c(i, l) = \kappa_{o;i,l} / \left(\kappa_{c;i,l} + \kappa_{o;i,l} \right). \tag{48}$$

Adversarial Attacks for SNN with Differential Surrogates

The network's backpropagation of gradient is achieved by unfolding LIF neuron dynamics and applying surrogate functions to the non-differentiable Heaviside function, as described in Eqs. 1, 2, and 3 in the main text.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_{ij}^{l}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_i^{l}(t)} \boldsymbol{s}_j^{l-1}(t), \tag{49}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_i^{l}(t)} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_i^{l}(t)} h\left(\boldsymbol{u}_i^{l}(t) \right) + \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_i^{l}(t+1)} \frac{\partial \boldsymbol{u}_i^{l}(t+1)}{\partial \boldsymbol{u}_i^{l}(t)}. \tag{50}$$

where $\mathcal{L}$ is the task loss, $\boldsymbol{W}_{ij}^{l}$ is the synaptic weight that connects neuron i in layer l and neuron j in layer $l-1$. $h\left(\boldsymbol{u}_i^{l}(t) \right)$ is the surrogate gradient of the Heaviside function, which is used to substitute $\frac{\partial \boldsymbol{s}_i^{l}(t)}{\partial \boldsymbol{u}_i^{l}(t)}$ to overcome the non-differentiable problem.

Acknowledgments

This work was supported by the National Natural Science Foundation of China(62088102, 62176003).

References

Alsmeyer, G. 1994. On the Markov renewal theorem. *Stochastic processes and their applications*, 50(1): 37–56.

Athalye, A.; Engstrom, L.; Ilyas, A.; and Kwok, K. 2018. Synthesizing robust adversarial examples. In *Proceedings of the 38th International Conference on Machine Learning*, 284–293. PMLR.

Bicknell, B. A.; and Goodhill, G. J. 2016. Emergence of ion channel modal gating from independent subunit kinetics. *Proceedings of the National Academy of Sciences*, 113(36): E5288–E5297.

Bing, Z.; Meschede, C.; Röhrbein, F.; Huang, K.; and Knoll, A. C. 2018. A survey of robotics control based on learning-inspired spiking neural networks. *Frontiers in Neurorobotics*, 12: 35.

Bu, T.; Ding, J.; Hao, Z.; and Yu, Z. 2023. Rate Gradient Approximation Attack Threats Deep Spiking Neural Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7896–7906.

Bu, T.; Ding, J.; Yu, Z.; and Huang, T. 2022a. Optimized Potential Initialization for Low-latency Spiking Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 11–20.

Bu, T.; Fang, W.; Ding, J.; Dai, P.; Yu, Z.; and Huang, T. 2022b. Optimal ANN-SNN Conversion for High-accuracy and Ultra-low-latency Spiking Neural Networks. In *International Conference on Learning Representations*, 1–19.

Cisse, M.; Bojanowski, P.; Grave, E.; Dauphin, Y.; and Usunier, N. 2017. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 38th International Conference on Machine Learning*, 854–863.

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 38th International Conference on Machine Learning*, 2206–2216. PMLR.

Dapello, J.; Marques, T.; Schrimpf, M.; Geiger, F.; Cox, D.; and DiCarlo, J. J. 2020. Simulating a primary visual cortex at the front of CNNs improves robustness to image perturbations. *Advances in Neural Information Processing Systems*, 33: 13073–13087.

DeBole, M. V.; Taba, B.; Amir, A.; Akopyan, F.; Andreopoulos, A.; Risk, W. P.; Kusnitz, J.; Otero, C. O.; Nayak, T. K.; Appuswamy, R.; et al. 2019. TrueNorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 52(5): 20–29.

Deng, S.; Li, Y.; Zhang, S.; and Gu, S. 2021. Temporal Efficient Training of Spiking Neural Network via Gradient Reweighting. In *International Conference on Learning Representations*.

Diehl, P. U.; Neil, D.; Binas, J.; Cook, M.; Liu, S.-C.; and Pfeiffer, M. 2015. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *International Joint Conference on Neural Networks*, 1–8.

Ding, J.; Bu, T.; Yu, Z.; Huang, T.; and Liu, J. 2022. SNN-RAT: Robustness-enhanced Spiking Neural Network through Regularized Adversarial Training. *Advances in Neural Information Processing Systems*, 35: 24780–24793.

Ding, J.; Yu, Z.; Tian, Y.; and Huang, T. 2021. Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks. In *International Joint Conference on Artificial Intelligence*, 2328–2336.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9185–9193.

Duan, C.; Ding, J.; Chen, S.; Yu, Z.; and Huang, T. 2022. Temporal Effective Batch Normalization in Spiking Neural Networks. *Advances in Neural Information Processing Systems*, 35: 34377–34390.

El-Allami, R.; Marchisio, A.; Shafique, M.; and Alouani, I. 2021. Securing deep spiking neural networks against adversarial attacks through inherent structural parameters. In *Design, Automation & Test in Europe Conference & Exhibition*, 774–779.

Faisal, A. A.; Selen, L. P.; and Wolpert, D. M. 2008. Noise in the nervous system. *Nature reviews neuroscience*, 9(4): 292–303.

Gerstner, W.; Kistler, W. M.; Naud, R.; and Paninski, L. 2014. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*.

Hao, Z.; Bu, T.; Ding, J.; Huang, T.; and Yu, Z. 2023. Reducing ANN-SNN Conversion Error through Residual Membrane Potential. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 11–21.

Hu, Y.; Zheng, Q.; Jiang, X.; and Pan, G. 2023. Fast-SNN: Fast Spiking Neural Network by Converting Quantized ANN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (01): 1–17.

Kim, Y.; and Panda, P. 2021. Revisiting Batch Normalization for Training Low-Latency Deep Spiking Neural Networks From Scratch. *Frontiers in Neuroscience*, 15: 773954–773954.

Kim, Y.; Park, H.; Moitra, A.; Bhattacharjee, A.; Venkatesha, Y.; and Panda, P. 2022. Rate Coding Or Direct Coding: Which One Is Better For Accurate, Robust, And Energy-Efficient Spiking Neural Networks? In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 71–75.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Kundu, S.; Datta, G.; Pedram, M.; and Beerel, P. A. 2021. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 3953–3962.

Kundu, S.; Pedram, M.; and Beerel, P. A. 2021. Hire-SNN: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5209–5218.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2017. Adversarial Machine Learning at Scale. In *International Conference on Learning Representations*.

Lema, G. M.; and Auerbach, A. 2006. Modes and models of GABAA receptor gating. *The Journal of Physiology*, 572(1): 183–200.

Li, C.; Chen, R.; Moutafis, C.; and Furber, S. 2020. Robustness to noisy synaptic weights in spiking neural networks. In *International Joint Conference on Neural Networks*, 1–8.

Liang, L.; Hu, X.; Deng, L.; Wu, Y.; Li, G.; Ding, Y.; Li, P.; and Xie, Y. 2023. Exploring adversarial attack in spiking neural networks with spike-compatible gradient. *IEEE Transactions on Neural Networks and Learning Systems*, 34(5): 2569–2583.

Liang, L.; Xu, K.; Hu, X.; Deng, L.; and Xie, Y. 2022. Toward Robust Spiking Neural Network Against Adversarial Perturbation. In *Advances in Neural Information Processing Systems*, volume 35, 10244–10256.

Lin, J.; Gan, C.; and Han, S. 2019. Defensive Quantization: When Efficiency Meets Robustness. In *International Conference on Learning Representations*.

Lv, C.; Xu, J.; and Zheng, X. 2023. Spiking Convolutional Neural Networks for Text Classification. In *International Conference on Learning Representations*.

Maass, W. 1997. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9): 1659–1671.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

Neftci, E. O.; Mostafa, H.; and Zenke, F. 2019. 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.

Pei, J.; Deng, L.; Song, S.; Zhao, M.; Zhang, Y.; Wu, S.; Wang, G.; Zou, Z.; Wu, Z.; He, W.; et al. 2019. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature*, 572(7767): 106–111.

Peracchia, C. 2004. Chemical gating of gap junction channels: roles of calcium, pH and calmodulin. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1662(1-2): 61–80.

Qin, C.; Martens, J.; Gowal, S.; Krishnan, D.; Dvijotham, K.; Fawzi, A.; De, S.; Stanforth, R.; and Kohli, P. 2019. Adversarial robustness through local linearization. *Advances in Neural Information Processing Systems*, 32: 13842—-13853.

Ross, A.; and Doshi-Velez, F. 2018. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Roth, K.; Kilcher, Y.; and Hofmann, T. 2020. Adversarial training is a form of data-dependent operator norm regularization. *Advances in Neural Information Processing Systems*, 33: 14973–14985.

Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; and Roy, K. 2019. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience*, 13: 95.

Sharmin, S.; Panda, P.; Sarwar, S. S.; Lee, C.; Ponghiran, W.; and Roy, K. 2019. A comprehensive analysis on adversarial robustness of spiking neural networks. In *International Joint Conference on Neural Networks*, 1–8.

Sharmin, S.; Rathi, N.; Panda, P.; and Roy, K. 2020. Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations. In *European Conference on Computer Vision*, 399–414.

Shen, J.; Xu, Q.; Liu, J. K.; Wang, Y.; Pan, G.; and Tang, H. 2023. ESL-SNNs: An Evolutionary Structure Learning Strategy for Spiking Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 86–93.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

White, J. A.; Rubinstein, J. T.; and Kay, A. R. 2000. Channel noise in neurons. *Trends in neurosciences*, 23(3): 131–137.

Wu, H.; Zhang, Y.; Weng, W.; Zhang, Y.; Xiong, Z.; Zha, Z.-J.; Sun, X.; and Wu, F. 2021. Training spiking neural networks with accumulated spiking flow. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 10320–10328.

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; and Shi, L. 2018. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12: 331.

Xu, Q.; Li, Y.; Shen, J.; Zhang, P.; Liu, J. K.; Tang, H.; and Pan, G. 2022. Hierarchical spiking-based model for efficient image classification with enhanced feature extraction and encoding. *IEEE Transactions on Neural Networks and Learning Systems*.

Yamazaki, K.; Vo-Ho, V.-K.; Bulsara, D.; and Le, N. 2022. Spiking neural networks and their applications: A Review. *Brain Sciences*, 12(7): 863.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference*.

Zenke, F.; Bohté, S. M.; Clopath, C.; Comşa, I. M.; Göltz, J.; Maass, W.; Masquelier, T.; Naud, R.; Neftci, E. O.; Petrovici, M. A.; et al. 2021. Visualizing a joint future of neuroscience and neuromorphic engineering. *Neuron*, 109(4): 571–575.

Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; and Li, G. 2021. Going Deeper With Directly-Trained Larger Spiking Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11062–11070.

Zhou, S.; Li, X.; Chen, Y.; Chandrasekaran, S. T.; and Sanyal, A. 2021. Temporal-coded deep spiking neural network with easy training and robust performance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11143–11151.