Rate Gradient Approximation Attack Threats Deep Spiking Neural Networks Supplementary Materials

1. Static R-I Curves

In section 4.2, we introduce our Rate Gradient Approximation Attack. The key idea of this attack is to approximate the SNN backward pass of SNNs using only the average firing rate over time-steps to generate more effective gradients. However, the gradient $\frac{\partial r_i}{\partial I_i}$ can not be directly calculated, so we need to determine the surrogate functions for approximation.

The definition of the static R-I curve is the function that maps the constant input current to the output average firing rate. We used this static R-I curve to estimate the relationship between actual firing rates and neuronal inputs. To calculate the static R-I curve, we consider the time it takes for a single neuron to fire a spike from its resting state. During this process, the behavior of the neuron can be described as

$$u_i(t) = \lambda u_i(t-1) + I_i. \tag{S1}$$

Static R-I curve for IF neurons

For IF neurons, we have $\lambda = 1$, thus we rewrite this equation as

$$u_i(t) = u_i(t-1) + I_i. \tag{S2}$$

This is an arithmetic sequence. Supposing that the firing threshold is θ, we can directly calculate the static R-I curve as

$$r_i = \frac{I_i}{\theta}. \tag{S3}$$

Static R-I curve for LIF neurons

For LIF neurons, we have $\lambda < 1$. The membrane potential $\{u_i(t)\}_{t=0,1,\ldots}$ can be viewed as a sequence. And we can then transform this formula to

$$u_i(t) + \frac{I_i}{\lambda - 1} = \lambda(u_i(t-1) + \frac{I_i}{\lambda - 1}). \tag{S4}$$

This is a geometric sequence. Thus, we can finally calculate the function of $u_i(t)$.

$$u_i(t) = \frac{I_i}{\lambda - 1}\lambda^t - \frac{I_i}{\lambda - 1}. \tag{S5}$$

The spike firing time is the time for membrane potential $u_i(t)$ to accumulate from resting value to the threshold θ, which should be

$$t^{(f)} = \left\lceil \log_\lambda \frac{I_i + \theta(\lambda - 1)}{I_i} \right\rceil. \tag{S6}$$

Then the firing rate is the multiplicative inverse of the spike firing time

$$r_i = \frac{1}{t^{(f)}} = \left\lceil \log_\lambda \frac{I_i + \theta(\lambda - 1)}{I_i} \right\rceil^{-1}. \tag{S7}$$

So far, we have obtained the static R-I curves for IF and LIF neurons.

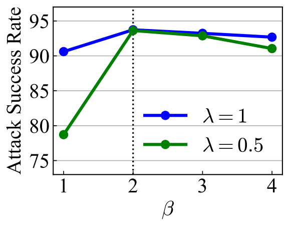
2. Actual R-I Curves

In section 4.3, we also draw the actual R-I curve from experimental data, shown by the light blue scatter in Fig.4 and Fig. 5 of the main text. Here we introduce how to get the actual R-I curve in detail.

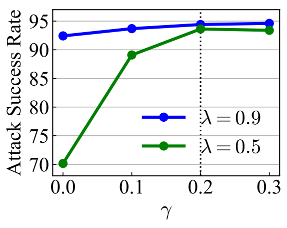
The actual R-I curve is the relationship between the actual average firing rate and the actual average input current. Therefore, we can obtain the input and output firing rate on the validation set, and then draw a scatter plot to represent the actual relationship between R-I. In this experiment, we use the VGG-11 architecture with the CIFAR-10 dataset. We choose the neurons in the second layer and random sample 8000 data points. We average the inputs and outputs over time for each neuron to get the average inputs and outputs. We then scatter these 8000 data pairs as the actual R-I curves. From this, we can better adjust the surrogate function to fit in practicality.

3. Hyperparameter Studies

We add two parameters to the surrogate function for Rate Gradient Attack. The final surrogate gradient can be used to approximate the gradient of $\frac{\partial r_i}{\partial I_i}$ for LIF neurons with arbitrary leaky parameters λ. The general expression for



(a) Effect of β



(b) Effect of γ

Figure S1. Effect of hyperparameters

this function can be written as

$$\frac{\partial r_i}{\partial I_i} = \begin{cases} \gamma, & 0 \leqslant I_i \leqslant (1-\lambda)\theta \\ \frac{1-\gamma\theta+\gamma\theta\lambda}{(\beta+\lambda-1)\theta}, & (1-\lambda)\theta < I_i \leqslant \beta\theta \ . \\ 0, & I_i > \beta\theta \ \text{ or } \ I_i < 0 \end{cases} \tag{S8}$$

We then conduct further experiments to determine both hyperparameters with VGG-16 structure on the CIFAR-10 dataset. We change the value of hyperparameters and test the attack success rate of the RGA-based attack. The results are in Fig. S1a and Fig. S1b.

Effect of parameter β

To test the effectiveness of β on different neurons, we use the LIF neuron with inference time $T = 8$ and leaky parameters $\lambda = 1.0, 0.5$. Candidate beta values include 1, 2, 3, and 4. From Fig. S1a, we can see that the best value for β is around 2. Thus, we choose $\beta = 2$ for all RGA-based attacks.

Effect of parameter γ

Following the experiment above, we make similar experiments to test the effectiveness of the smooth parameter γ. We choose LIF neurons with inference time $T = 8$ and leaky parameters $\lambda = 0.9, 0.5$. Since γ only works on leaky neurons, we do not test the case that $\lambda = 1$. We compare the attack success rate with different γ ($\gamma = 0, 0.1, 0.2, 0.3$) and show the results in Fig. S1b. When $\gamma = 0.2$, the RGA-based attack performs the most significant results. Thus, γ is set to 0.2 for all RGA-based attacks.

4. Network Training Configuration

As for preprocessing, we first normalize images to [0, 1] and then random crop and horizontalflip the images. We also add Cutout [1] as extra data augmentation. As for neuron configurations, the resting potential is set to 0, and the threshold is set by default 1. Besides, we use the triangle-shaped surrogate gradient [2] when doing both supervised STBP training and STBP-based attacks. We use the Stochastic Gradient Descent with momentum to optimizer all networks and use cosine annealing [3] for warming up. The loss function is the cross entropy loss for both CIFAR datasets and CIFAR10-DVS dataset. Other hyperparameters in training can be checked from Tab. S1.

Initial learning rate	Momentum	Weight decay	Batchsize	Epochs
0.1	0.9	5e-4	64	200

Table S1. Training Configuration

5. Cost Analysis

In this section, we discuss the time and memory cost of the RGA attack. Here we evaluate the computational time of adversarial attacks. We fix the mini-batch size as 32 and perform an adversarial attack on the validation set of CIFAR-10. We use NVIDIA 3090 GPU and CentOS Linux platform. We compare the time and memory cost for RGA and STBP attacks and present the results in Tab. S2. We find that the RGA attack requires less memory and computational cost than STBP. This also makes this attack method more potential in adversarial training.

Attack Method	RGA	STBP
Time Cost (s)	6.273	10.683
Memory Cost (Mb)	3299	4053

Table S2. Effectiveness of the RGA Attack

References

[1] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 2

[2] Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 2016. 2

[3] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 2