Inherent Adversarial Robustness of Deep Spiking Neural Networks: Effects of Discrete Input Encoding and Non-Linear Activations

Saima Sharmin[1][0000−0002−1866−9138], Nitin Rathi[1][0000−0003−0597−064X], Priyadarshini Panda[2][0000−0002−4167−6782], and Kaushik Roy[1][0000−0002−0735−9695]

[1] Purdue University, West Lafayette IN 47907, USA
{ssharmin,rathi2,kaushik}@purdue.edu
[2] Yale University, New Haven CT 06520, USA
priya.panda@yale.edu

Abstract. In the recent quest for trustworthy neural networks, we present Spiking Neural Network (SNN) as a potential candidate for inherent robustness against adversarial attacks. In this work, we demonstrate that adversarial accuracy of SNNs under gradient-based attacks is higher than their non-spiking counterparts for CIFAR datasets on deep VGG and ResNet architectures, particularly in blackbox attack scenario. We attribute this robustness to two fundamental characteristics of SNNs and analyze their effects. First, we exhibit that input discretization introduced by the Poisson encoder improves adversarial robustness with reduced number of timesteps. Second, we quantify the amount of adversarial accuracy with increased leak rate in Leaky-Integrate-Fire (LIF) neurons. Our results suggest that SNNs trained with LIF neurons and smaller number of timesteps are more robust than the ones with IF (Integrate-Fire) neurons and larger number of timesteps. Also we overcome the bottleneck of creating gradient-based adversarial inputs in temporal domain by proposing a technique for crafting attacks from SNN[1].

Keywords: Spiking Neural Networks, Adversarial attack, Leaky-Integrate-Fire neuron, Input discretization

1 Introduction

Adversarial attack is one of the biggest challenges against the success of today's deep neural networks in mission critical applications [16], [10], [32]. The underlying concept of an adversarial attack is to purposefully modulate the input to a neural network such that it is subtle enough to remain undetectable to human eyes, yet capable of fooling the network into incorrect decisions. This malicious behavior was first demonstrated in 2013 by Szegedy *et. al.* [27] and Biggio *et. al.* [4] in the field of computer vision and malware detection, respectively. Since then, numerous defense mechanisms have been proposed to address

[1] https://github.com/ssharmin/spikingNN-adversarial-attack

this issue. One category of defense includes fine-tuning the network parameters like adversarial training [11], [18], network distillation [22], stochastic activation pruning [8] etc. Another category focuses on preprocessing the input before passing through the network like thermometer encoding [5], input quantization [31], [21], compression [12] etc. Unfortunately, most of these defense mechanisms have been proved futile by many counter-attack techniques. For example, an ensemble of defenses based on "gradient-masking" collapsed under the attack proposed in [1]. Defensive distillation was broken by Carlini-Wagner method [6], [7]. Adversarial training has the tendency to overfit to the training samples and remain vulnerable to transfer attacks [28]. Hence, the threat of adversarial attack continues to persist.
In the absence of adversarial robustness in the existing state-of-the-art networks, we feel there is a need for a network with *inherent* susceptibility against adversarial attacks. In this work, we present Spiking Neural Network (SNN) as a potential candidate due to two of its fundamental distinctions from the non-spiking networks:

1. SNNs operate based on discrete binary data (0/1), whereas their non-spiking counterparts, referred as Analog Neural Network (ANN), take in continuous-valued analog signals. Since SNN is a binary spike-based model, input discretization is a constituent element of the network, most commonly done by Poisson encoding.
2. SNNs employ nonlinear activation function of the biologically inspired Integrate-Fire (IF) or Leaky-Integrate-Fire (LIF) neurons, in contrast to the piecewise-linear ReLU activations used in ANNs.

Among the handful of works done in the field of SNN adversarial attacks [19], [2], most of them are restricted to either simple datasets (MNIST) or shallow networks. However, this work extends to complex datasets (CIFAR) as well as deep SNNs which can achieve comparable accuracy to the state-of-the-art ANNs [24], [23]. For robustness comparison with non-spiking networks, we analyze two different types of spiking networks: (1) converted SNN (trained by ANN-SNN conversion [24]) and (2) backpropagated SNN (an ANN-SNN converted network, further incrementally trained by surrogate gradient backpropagation [23]). We identify that converted SNNs fail to demonstrate more robustness than ANNs. Although authors in [25] show similar analysis, we explain with experiments the reason behind this discrepancy and, thereby, establish the necessary criteria for an SNN to become adversarially robust. Moreover, we propose an SNN-crafted attack generation technique with the help of the surrogate gradient method. We summarize our contribution as follows:

- We show that the adversarial accuracies of SNNs are higher than ANNs under a gradient-based blackbox attack scenario, where the respective clean accuracies are comparable to each other. The attacks were performed on deep VGG and ResNet architectures trained on CIFAR10 and CIFAR100 datasets. For whitebox attacks, the comparison is dependent on the relative strengths of the adversary.

- The increased robustness of SNN is attributed to two fundamental characteristics: input discretization through Poisson encoding and non-linear activations of LIF (or IF) neurons.
  - ○ We investigate how adversarial accuracy changes with the number of timesteps (inference latency) used in SNN for different levels of input pre-quantization[2]. In case of backpropagated SNNs (trained with smaller number of timesteps), the amount of discretization as well as adversarial robustness increases as we reduce the number of timesteps. Pre-quantization of the analog input brings about further improvement. However, converted SNNs appear to depend only on the input pre-quantization, but invariant to the variation in the number of timesteps. Since these converted SNNs operate under larger number of timesteps [24], discretization effect is minimized by input averaging, and hence, the observed invariance.
  - ○ We show that piecewise-linear activation (ReLU) in ANN linearly propagates the adversarial perturbation throughout the network, whereas LIF (or IF) neurons diminish the effect of perturbation at every layer. Additionally, the leak factor in LIF neurons offers an extra knob to control the adversarial perturbation. We perform a quantitative analysis to demonstrate the effect of leak on the adversarial robustness of SNNs.

  Overall, we show that SNNs employing LIF neurons, trained with surrogate gradient-based backpropagation, and operating at less number of timesteps are more robust than SNNs trained with ANN-SNN conversion that requires IF neurons and more number of timesteps. Hence, the training technique plays a crucial role in fulfilling the prerequisites for an adversarially robust SNN.
- Gradient-based attack generation in SNNs is non-trivial due to the discontinuous gradient of the LIF (or IF) neurons. We propose a methodology to generate attacks based on the approximate surrogate gradients.

2 Background

2.1 Adversarial Attack

Given a clean image x belonging to class i and a trained neural network M, an adversarial image x_{adv} needs to meet two criteria:

1. x_{adv} is visually "similar" to x *i. e.* $|x - x_{adv}| = \epsilon$, where ϵ is a small number.
2. x_{adv} is misclassified by the neural network, *i. e.* $M(x_{adv}) \neq i$

The choice of the distance metric $|.|$ depends on the method used to create x_{adv} and ϵ is a hyper-parameter. In most methods, l_2 or l_∞ norm is used to measure the similarity and the value of ϵ is limited to $\leq \frac{8}{255}$ where normalized pixel intensity $x \in [0, 1]$ and original $x \in [0, 255]$.

In this work, we construct adversarial examples using the following two methods:

[2] Full-precision analog inputs are quantized to lower bit precision values before undergoing the discretization process by the Poisson encoder of SNN

Fast Gradient Sign Method (FGSM) This is one of the simplest methods for constructing adversarial examples, introduced in [11]. For a given instance x, true label y_{true} and the corresponding cost function of the network $J(x, y_{true})$, this method aims to search for a perturbation δ such that it maximizes the cost function for the perturbed input $J(x + \delta, y_{true})$, subject to the constraint $|\delta|_\infty < \epsilon$. In closed form, the attack is formulated as,

$$x_{adv} = x + \epsilon \times sign\Big(\nabla_x J\left(x, y_{true}\right)\Big) \tag{1}$$

Here, ϵ denotes the strength of the attack.

Projected Gradient Descent (PGD) This method, proposed in [18], produces more powerful adversary. PGD is basically a *k-step* variant of FGSM computed as,

$$x_{adv}^{(k+1)} = \mathit{\Pi}_{x+\epsilon}\Bigg\{\Bigg(x_{adv}^{(k)} + \alpha \times sign\Big(\nabla_x\big(J(x_{adv}^{(k)}, y_{true})\big)\Big)\Bigg)\Bigg\} \tag{2}$$

where $x_{adv}^{(0)} = x$ and $\alpha(\leq \epsilon)$ refers to the amount of perturbation used per iteration or step, k is the total number of iterations. $\mathit{\Pi}_{x+\epsilon}\{.\}$ performs a projection of its operand on an ϵ-ball around x, *i. e.*, the operand is clipped between $x+\epsilon$ and $x-\epsilon$. Another variant of this method adds a random perturbation of strength ϵ to x before performing PGD operation.

2.2 Spiking Neural Network (SNN)

The main difference between an SNN and an ANN is the concept of time. The incoming signals as well as the intermediate node inputs/outputs in an ANN are static analog values, whereas in an SNN, they are binary spikes with a value of 0 or 1, which are also functions of time. In the input layer, a Poisson event generation process is used to convert the continuous valued analog signals into binary spike train. Suppose the input image is a 3-D matrix of dimension $h \times w \times l$ with pixel intensity in the range $[0, 1]$. At every time step of the SNN operation, a random number (from the normal distribution $\mathcal{N}(0, 1)$) is generated for each of these pixels. A spike is triggered at that particular time step if the corresponding pixel intensity is greater than the generated random number. This process continues for a total of T timesteps to produce a spike train for each pixel. Hence, the size of the input spike train is $T \times h \times w \times l$. For a large enough T, the timed average of the spike train will be proportional to its analog value. Every node of the SNN is accompanied with a neuron. Among many neuron models, the most commonly used ones are Integrate-Fire (IF) or Leaky-Integrate-Fire (LIF) neurons. The dynamics of the neuron membrane potential at time $t+1$ is described by,

$$V(t+1) = \lambda V(t) + \sum_i w_i x_i(t) \tag{3}$$

Here $\lambda = 1$ for IF neurons and < 1 for LIF neurons. w_i denotes the synaptic weight between current neuron and *i-th* pre-neuron. $x_i(t)$ is the input spike from the *i-th* pre-neuron at time t. When $V(t+1)$ reaches the threshold voltage V_{th}, an output spike is generated and the membrane potential is reset to 0, or in case of soft reset, reduced by the amount of the threshold voltage. At the output layer, inference is performed based on the cumulative membrane potential of the output neurons after the total time T has elapsed.

One of the main shortcomings of SNNs is that they are difficult to train, especially for deeper networks. Since the neurons in an SNN have discontinuous gradients, standard gradient-descent techniques do not directly apply. In this work, we use two of the supervised training algorithms [24], [23], which achieve ANN-like accuracy even for deep neural networks and on complex datasets.

ANN-SNN Conversion This training algorithm was originally outlined in [9] and subsequently improved in [24] for deep networks. Note that the algorithm is suited for training SNNs with IF neurons only. They propose a threshold-balancing technique that sequentially tunes the threshold voltage of each layer. Given a trained ANN, the first step is to generate the input Poisson spike train for the network over the training set for a large enough time-window so that the timed average accurately describes the input. Next, the maximum value of $\sum_i w_i x_i$ (term 2 in Eq. 3) received by layer 1 is recorded over the entire time range for several minibatches of the training data. This is referred as the maximum activation for layer 1. The threshold value of layer 1 neuron is replaced by this maximum activation keeping the synaptic weights unchanged. Such threshold tuning operation ensures that the IF neuron activity precisely mimics the ReLU function in the corresponding ANN. After balancing layer 1 threshold, the method is continued for all subsequent layers sequentially.

Conversion & Surrogate-Gradient Backpropagation In order to take advantage of the standard backpropagation-based optimization procedures, authors in [3], [33], [30] introduced the surrogate gradient technique. The input-output characteristics of an LIF (or IF) neuron is a step function, the gradient of which is discontinuous at the threshold point (Fig. 1). In surrogate gradient technique, the gradient is approximated by pseudo-derivatives like linear or exponential functions. Authors in [23] proposed a novel approximation function for these gradients by utilizing the spike time information in the derivative. The gradient at timestep t is computed as follows:

$$\frac{\partial o^t}{\partial u^t} = \alpha e^{-\beta \Delta t} \tag{4}$$

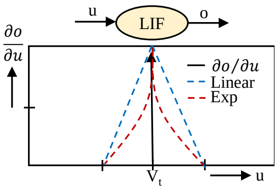


Fig. 1. Surrogate gradient approximation of an LIF neuron.

Here, o^t is the output spike at time t, u^t is the membrane potential at t, Δt is the difference between current timestep and the last timestep post-neuron generated a spike. α and β are hyperparameters. Once the neuron gradients are approximated, backpropagation through time (BPTT) [29] is performed using the chain rule of derivatives. In BPTT, the network is unrolled over all timesteps. The final output is computed as the cumulation of outputs at every timestep and eventually, loss is defined on the summed output. During backward propagation of the loss, the gradients are accumulated over time and used in gradient-descent optimization. Authors in [23] proposed a hybrid training procedure in which the surrogate-gradient training is preceded by an ANN-SNN conversion to initialize the weights and thresholds of the network. The advantage of this method over ANN-SNN conversion is twofold: one can train a network with both IF and LIF neurons and the number of timesteps required for training is reduced by a factor of 10 without losing accuracy.

3 Experiments

3.1 Dataset and Models

We conduct our experiments on VGG5 and ResNet20 for CIFAR10 dataset and VGG11 for CIFAR100. The network topology for VGG5 consists of conv3,64-avgpool-conv3,128 (×2)-avgpool-fc1024 (×2)-fc10. Here conv3,64 refers to a convolutional layer with 64 output filters and 3×3 kernel size. fc1024 is a fully-connected layer with 1024 output neurons. VGG11 contains 11 weight layers corresponding to the configuration A in [26] with maxpool layers replaced by average pooling. ResNet20 follows the proposed architecture for CIFAR10 in [13], except the initial 7×7 non-residual convolutional layer is replaced by a series of two 3×3 convolutional layers. For ANN-SNN conversion of ResNet20, threshold balancing is performed only on these initial non-residual units (as demonstrated by [24]). The neurons (in both ANN and SNN) contain no bias terms, since they have an indirect effect on the computation of threshold voltages during ANN-SNN conversion. The absence of bias eliminates the use of batch normalization [14] as a regularizer. Instead, a dropout layer is used after every ReLU (except for those which are followed by a pooling layer).

3.2 Training Procedure

The aim of our experiment is to compare adversarial attack on 3 networks: 1) ANN, 2) SNN trained by ANN-SNN conversion and 3) SNN trained by backpropagation, with initial conversion. These networks will be referred as ANN, SNN-conv and SNN-BP, respectively from this point onward.

For both CIFAR10 and CIFAR100 datasets, we follow the data augmentation techniques in [17]: 4 pixels are padded on each side, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. Testing is performed on the original 32×32 images. Both training and testing data are normalized to $[0, 1]$. For training the ANNs, we use cross-entropy loss with stochastic gradient descent optimization (weight decay=0.0001, mometum=0.9). VGG5 (and ResNet20) are trained for a total of 200 epochs, with an initial learning rate of 0.1 (0.05), which is divided by 10 at 100-th (80-th) and 150-th (120-th) epoch. VGG11 with CIFAR100 is trained for 250 epochs with similar learning schedule. During training SNN-conv networks, a total of 2500 timesteps are used for all VGG and ResNet architectures. SNN-BP networks are trained for 15 epochs with cross-entropy loss and adam [15] optimizer (weight decay=0.0005). Initial learning rate is 0.0001, which is halved every 5 epochs. A total of 100 timesteps is used for VGG5 and 200 timesteps for ResNet20 & VGG11. Training is performed with either linear surrogate gradient approximation [3] or spike time dependent approximation [23] with $\alpha = 0.3$, β=0.01 (in Eq. 4). Both techniques yield approximately similar results. Leak factor λ is kept at 0.99 in all cases, except in the analysis for the leak effect.

In order to analyze the effect of input quantization (with varying number of timesteps) and leak factors, only VGG5 networks with CIFAR10 dataset is used.

3.3 Adversarial Input Generation Methodology

For the purpose of whitebox attacks, we need to construct adversarial samples from all three networks (ANN, SNN-conv, SNN-BP). The ANN-crafted FGSM and PGD attacks are generated using the standard techniques described in Eq. 1 and 2, respectively. We carry out non-targeted attacks with $\epsilon = 8/255$. PGD attacks are performed with iteration steps $k = 7$ and per-step perturbation $\alpha = 2/255$. FGSM or PGD method cannot be directly applied to SNN due to its discontinuous gradient problem (described in Sec. 2.2). To that end, we outline a surrogate-gradient based FGSM (and PGD) technique. In SNN, analog input X is converted to Poisson spike train X_{spike} which is fed into the $1st$ convolutional layer. If X_{rate} is the timed average of X_{spike}, the membrane potential of the $1st$ convolutional layer X_{conv1} can be approximated as

$$X_{conv1} \approx Conv(X_{rate}, W_{conv1}) \tag{5}$$

W_{conv1} is the weight of the $1st$ convolutional layer. From this equation, the sign of the gradient of the network loss function J w.r.t. X_{rate} or X is described by (detailed derivation is provided in *supplementary*),

$$sign\Big(\frac{\partial J}{\partial X}\Big) \approx sign\Big(\frac{\partial J}{\partial X_{rate}}\Big) = sign\Big(Conv\big(\frac{\partial J}{\partial X_{conv1}}, W_{conv1}^{180rotated}\big)\Big) \tag{6}$$

Surrogate gradient technique yields $\frac{\partial J}{\partial X_{conv1}}$ from SNN, which is plugged into Eq. 6 to calculate the sign of the input gradient. This sign matrix is later used to compute X_{adv} according to standard FGSM or PGD method. The algorithm is summarized in 1.

```
Algorithm 1 SNN-crafted $X_{adv}$ : $FGSM$
Require: Input $(X, y_{true})$, Trained SNN $(N)$ with loss function J.
Ensure: $\frac{\partial J}{\partial X_{conv1}} \leftarrow 0$
  for timestep $t$ in total time $T$ do
    forward: Loss $J(X, y_{true})$
    backward : Accumulate gradient $\frac{\partial J}{\partial X_{conv1}} += X_{conv1}.grad$
  end for
  post-processing: $sign(\frac{\partial J}{\partial X}) = sign\Big(Conv(\frac{\partial J}{\partial X_{conv1}}, W_{conv1}^{180rotated})\Big)$
  SNN-crafted adversary: $X_{adv}^{SNN} = X + \epsilon \times sign(\frac{\partial J}{\partial X})$
```

4 Results

4.1 ANN vs SNN

Table 1 summarizes our results for CIFAR10 (VGG5 & ResNet20) and CIFAR100 (VGG11) datasets in whitebox and blackbox settings. For each architecture, we start with three networks: ANN, SNN-conv and SNN-BP, trained to achieve comparable baseline clean accuracy. Let us refer them as M_{ANN}, $M_{SNN\text{-}conv}$ and $M_{SNN\text{-}BP}$, respectively. During blackbox attack, we generate an adversarial test dataset x_{adv} from a separately trained ANN of the same network topology as the target model but different initialization. It is clear that the adversarial accuracy of SNN-BP during FGSM and PGD blackbox attacks is higher than the corresponding ANN and SNN-conv models, irrespective of the size of the dataset or network architecture (the highest value of the accuracy for each attack case is highlighted by *orange text*). The amount of improvement in adversarial accuracy, compared to ANN, is listed as Δ in the Table. If M_{ANN} and $M_{SNN\text{-}BP}$ yield adversarial accuracy of $p_{ANN}\%$ and $p_{SNN\text{-}BP}\%$, respectively, the value of Δ amounts to $p_{SNN\text{-}BP}\% - p_{ANN}\%$. On the other hand, during whitebox attack, we generate three sets of adversarial test dataset: $x_{adv,ANN}$ (generated from M_{ANN}), $x_{adv,SNN\text{-}conv}$ (generated from $M_{SNN\text{-}conv}$) and so on. Since ANN and SNN have widely different operating dynamics and constituent elements, the strength of the constructed adversary varies significantly from ANN to SNN during whitebox attack (demonstrated in Sec. 4.2). SNN-BP shows significant improvement in whitebox adversarial accuracy (Δ ranging from 2% to 4.6%) for both VGG and Resnet architectures with CIFAR10 dataset. In contrast, VGG11 ANN with CIFAR100 manifests higher whitebox accuracy than SNN-BP. We attribute this discrepancy to the difference in adversary-strength of ANN & SNN for different dataset and network architectures. From Table 1, it is evident that SNN-BP networks exhibit the highest amount of adversarial accuracy (*orange text*) among the three networks in all blackbox attack cases (attacked by a common adversary), whereas SNN-conv and ANN demonstrate comparable accuracy, irrespective of the dataset, network topology or attack generation method. Hence, we conclude that SNN-BP is inherently more robust compared to their non-spiking counterpart as well as SNN-conv models, when all three networks are attacked by identical adversarial inputs. It is important to

Table 1. A comparison of the clean and adversarial (FGSM and PGD) test accuracy (%) among ANN, SNN-conv and SNN-BP networks. Highest value of the accuracy for each attack case is marked in *orange text*. FGSM accuracy is calculated at $\epsilon = 8/255$. For PGD, $\epsilon = 8/255$, α (per-step perturbation) = 2/255, k (number of steps) = 7. The blackbox attacks are generated from a separately trained ANN of the same network topology as the target model but different initialization

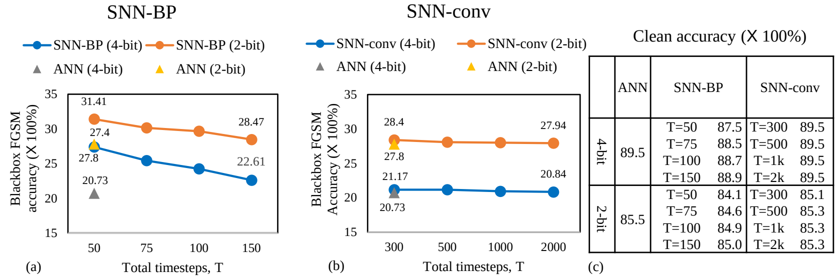
		Whitebox				Blackbox			
		ANN	SNN-conv	SNN-BP	$\Delta^\dagger$	ANN	SNN-conv	SNN-BP	$\Delta^\dagger$
CIFAR10									
VGG5	Clean	90%	89.9%	89.3%	—	90%	89.9%	89.3%	—
	FGSM	10.4%	7.7%	15%	**4.6%**	18.9%	19.3%	21.5%	**2.6%**
	PGD	1.8%	1.7%	3.8%	**2.0%**	9.3%	9.6%	16.0%	**6.7%**
ResNet20	Clean	88.0%	87.5%	86.1%	—	88.0%	87.5%	86.1%	—
	FGSM	28.9%	28.8%	31.3%	**2.4%**	56.7%	56.8%	56.8%	**0.1%**
	PGD	1.9%	1.4%	4.9%	**3.0%**	41.5%	41.6%	46.5%	**5.0%**
CIFAR100									
VGG11	Clean	67.1%	66.8%	64.4%	—	67.1%	66.8%	64.4%	—
	FGSM	17.1%	10.5%	15.5%	**-1.6%**	21.2%	21.4%	21.4%	**0.2%**
	PGD	8.5%	4.1%	6.3%	**-2.2%**	15.6%	15.8%	16.5%	**0.9%**

$^\dagger\Delta$ = Adversarial accuracy (SNN-BP) - Adversarial accuracy (ANN)

mention here that our conclusion is validated for VGG & ResNet architectures and gradient-based attacks only.

In the next two subsections, we explain two characteristics of SNNs contributing towards this robustness, as well as the reason for SNN-conv not being able to show similar behavior.

Effect of input quantization and number of timesteps The main idea behind non-linear input pre-processing as a defense mechanism is to discretize continuous-valued input signals so that the network becomes non-transparent to adversarial perturbations, as long as they lie within the discretization bin. SNN is a binary spike-based network, which demands encoding any analog valued input signal into binary spike train, and hence, we believe it has the inherent robustness. In our SNN models, we employ Poisson rate encoding, where the output spike rate is proportional to the input pixel intensity. However, the amount of discretization introduced by the Poisson encoder varies with the number of timesteps used. Hence, the adversarial accuracy of the network can be controlled

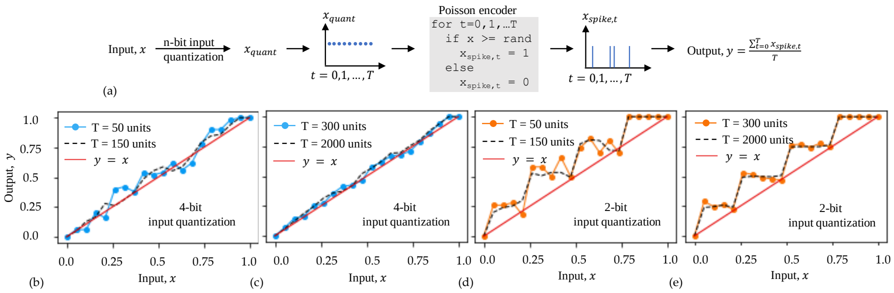


Clean accuracy (X 100%)

	ANN	SNN-BP		SNN-conv	
4-bit	89.5	T=50	87.5	T=300	89.5
		T=75	88.5	T=500	89.5
		T=100	88.7	T=1k	89.5
		T=150	88.9	T=2k	89.5
2-bit	85.5	T=50	84.1	T=300	85.1
		T=75	84.6	T=500	85.3
		T=100	84.9	T=1k	85.3
		T=150	85.0	T=2k	85.3

Fig. 2. Blackbox FGSM accuracy(%) versus total number of timesteps (T) plot with 4-bit (*blue*) and 2-bit (*orange*) input quantization for (a) SNN-BP and (b) SNN-conv networks. SNN-BP adversarial accuracy increases drastically with decreased number of timesteps, whereas SNN-conv is insensitive to it. (c) A table summarizing the clean accuracy of ANN, SNN-conv and SNN-BP for different input quantizations and number of timesteps.

by varying the number of timesteps as long as the clean accuracy remains within reasonable limit. This effect can be further enhanced by quantizing the analog input before feeding into the Poisson encoder. In Fig. 2(a), we demonstrate the FGSM adversarial accuracy of an SNN-BP network (VGG5) trained for 50, 75, 100 and 150 timesteps with CIFAR10 dataset. As number of timesteps drop from 150 to 50, accuracy increases by $\sim 5\%$ (*blue line*) for 4-bit input quantization. Note that clean accuracy drops by only 1.4% within this range, from 88.9% (150 timesteps) to 87.5% (50 timesteps), as showed in the table in Fig. 2(c). Additional reduction of the number of timesteps leads to larger degradation of clean accuracy. The adversarial accuracy for corresponding ANN (with 4-bit input quantization) is showed in *gray triangle* in the same plot for comparison. Further increase in adversarial accuracy is obtained by pre-quantizing the analog inputs to 2-bits (*orange line*) and it follows the same trend with number of timesteps. Thus varying the number of timesteps introduces an extra knob for controlling the level of discretization in SNN-BP in addition to the input pre-quantizations. In contrast, in Fig. 2(b), similar experiments performed on SNN-conv network demonstrate little increase in adversarial accuracy with the number of timesteps. Only pre-quantization of input signal causes improvement of accuracy from $\sim 21\%$ to $\sim 28\%$. Note that the range of the number of timesteps used for SNN-conv (300 to 2000) is much higher than SNN-BP, because converted networks have higher inference latency. The reason behind the invariance of SNN-conv towards the number of timesteps is explained in Fig. 3. We plot the input-output characteristics of the Poisson-encoder for 4 cases: (b) 4-bit input quantization with smaller number of timesteps (50 and 150), (c) 4-bit quantization, larger number of timesteps (300 and 2000) and their 2-bit counterparts in (d) and (e), respectively. It is evident from (c) and (e) that larger number of timesteps introduces more of an averaging effect, than quantization, and hence, varying the number of timesteps has negligible effect on the transfer plots (*solid*

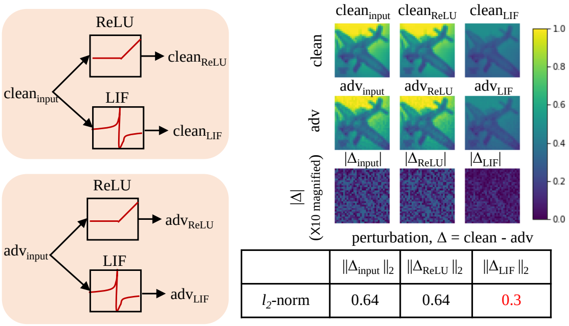


```
for t=0,1,…T
  if x >= rand
    x_spike,t = 1
  else
    x_spike,t = 0
```

Fig. 3. The input-output characteristics of Poisson encoder to demonstrate the effect of the total number of timseteps T used to generate the spike train with different levels of pre-quantization of the analog input. When T is in the low value (between 50 to 150) regime (subplots (b) and (d)), the amount of quantization significantly changes for varying the number of timesteps (*solid dotted and dashed lines*). But in the high value regime of T (plots (c) and (e)), *solid dotted* and *dashed* lines almost coincide due to the averaging effect. The flow of data from output y to input x is showed in the schematic in (a)

dotted and *dashed* lines coincide), which is not true for (b) and (d). Due to this averaging effect, Poisson output y for SNN-conv tends to follow the trajectory of x_{quant} (quantized ANN input), leading to comparable adversarial accuracy to the corresponding ANN over the entire range of timesteps in Fig. 2(b). Note that, in these plots input x refers to the analog input signal, whereas output y is the timed average of the spike train (as showed in the schematic in Fig. 3(a)).

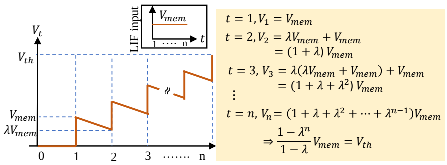
Effect of LIF (or IF) neurons and the leak factor Another major contributing factor towards SNN robustness is their highly nonlinear neuron activations (Integrate-Fire or Leaky-Integrate-Fire), whereas ANNs use mostly piecewise linear activations like ReLU. In order to explain the effect of this nonlinearity, we perform a proof of concept experiment. We feed a clean and corresponding adversarial input to a ReLU and an LIF neuron ($\lambda = 0.99$ in Eq. 3). Both of the inputs are 32×32 images with pixel intensity normalized to $[0, 1]$. Row 1, 2 and 3 in Fig. 4 present the clean image, corresponding adversarial image and their absolute difference (amount of perturbation), respectively. Note, the outputs of the LIF neurons are binary at each timestep, hence, we take an average over the entire time-window to obtain corresponding pixel intensity. ReLU passes both clean and adversarial inputs without any transformation, hence the l_2-norm of the perturbation is same at the input and ReLU output (bottom table of the figure). However, the non-linear transformation in LIF reduces the perturbation of 0.6 at input layer to 0.3 at its output. Basically, the output images of LIF (*column 3*) neurons is a low pixel version of the input images, due to the translation of continuous analog values into a binary spike representation. This behavior helps diminish the propagation of adversarial perturbation through the network. IF neurons also demonstrate this


ReLU
$\text{clean}_{\text{ReLU}}$
$\text{clean}_{\text{input}}$
LIF
$\text{clean}_{\text{LIF}}$
ReLU
adv_{ReLU}
$\text{adv}_{\text{input}}$
LIF
adv_{LIF}
clean
$\text{clean}_{\text{input}}$ $\text{clean}_{\text{ReLU}}$ $\text{clean}_{\text{LIF}}$
adv
$\text{adv}_{\text{input}}$ adv_{ReLU} adv_{LIF}
$|\Delta|$ (X10 magnified)
$|\Delta_{\text{input}}|$ $|\Delta_{\text{ReLU}}|$ $|\Delta_{\text{LIF}}|$
1.0
0.8
0.6
0.4
0.2
0.0
perturbation, Δ = clean - adv

	$\|\Delta_{\text{input}}\|_2$	$\|\Delta_{\text{ReLU}}\|_2$	$\|\Delta_{\text{LIF}}\|_2$
l_2-norm	0.64	0.64	0.3

Fig. 4. The input and output of ReLU and LIF neurons for each of clean and adversarial image. Column 1 shows clean image, adversarial image and the absolute value of the adversarial perturbation before passing through the neurons. Column 2 and 3 depict the corresponding images after passing through a ReLU and an LIF, respectively. The bottom table contains the l_2-*norm* of the perturbation at the input, ReLU output and LIF output

non-linear transformation. However, the quantization effect is minimized due to their operation over longer time-window (as explained in the previous section). Unlike SNN-conv, SNN-BP networks can be trained with LIF neurons. The leak factor in an LIF neuron provides an extra knob to manipulate the adversarial robustness of these networks. In order to investigate the effect of leak on the amount of robustness, we develop a simple expression relating the leak factor with neuron spike rate in an LIF neuron. In this case, the membrane potential V_t at timestep t is updated as $V_t = \lambda V_{t-1} + V_{input,t}$ given the membrane potential has not reached threshold yet, and hence, reset signal $= 0$. Here, λ (< 1) is the leak factor and $V_{input,t}$ is the input to the neuron at timestep t. Let us consider the scenario, where a constant voltage V_{mem} is fed into the neuron at every timestep and the membrane potential reaches the threshold voltage V_{th} after n timesteps. As explained in Fig. 5, membrane potential follows a geometric progression with time. After replacing $\frac{V_{th}}{V_{mem}}$ with a constant r, we obtain the following relation between the rate of spike $(1/n)$ and leak factor (λ):



$$t = 1, V_1 = V_{mem}$$
$$t = 2, V_2 = \lambda V_{mem} + V_{mem} = (1+\lambda)\, V_{mem}$$
$$t = 3, V_3 = \lambda(\lambda V_{mem} + V_{mem}) + V_{mem} = (1+\lambda+\lambda^2)\, V_{mem}$$
$$\vdots$$
$$t = n, V_n = (1+\lambda+\lambda^2+\cdots+\lambda^{n-1})V_{mem}$$
$$\Rightarrow \frac{1-\lambda^n}{1-\lambda} V_{mem} = V_{th}$$

Fig. 5. Output of an LIF neuron for constant input voltage.

$$\text{Spike rate,} \quad \frac{1}{n} = \frac{log\lambda}{log[r\lambda - (r-1)]}, \lambda < 1 \tag{7}$$

In Fig. 6(a), we plot the spike rate as a function of leak factor λ for different values of V_{mem} according to Eq. 7, where λ is varied from 0.9999 to 0.75. In

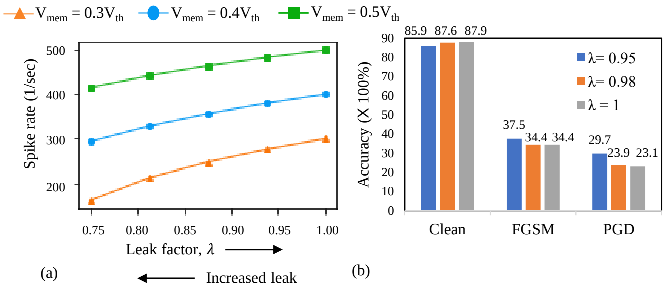


Fig. 6. (a) Spike rate versus leak factor λ for different values of $\frac{V_{mem}}{V_{th}}$. Smaller value of λ corresponds to more leak. (b) A bar plot showing the comparison of clean, FGSM and PGD($\epsilon = 8/255$) accuracy for a VGG5 SNN-BP network trained on CIFAR10 for different values of λ. These are blackbox attacks crafted from a VGG5 ANN model

every case, spike rate decreases (*i. e.* sparsity and hence, robustness increases) with increased amount of leak (smaller λ). The plot in Fig. 6(b) justifies this idea where we show the adversarial accuracy of an SNN-BP network (VGG5 with CIFAR10) trained with different values of leak. For both FGSM and PGD attacks, adversarial accuracy increases by $3 \sim 6\%$ as λ is decreased to 0.95. Note, the clean accuracy of the trained SNNs with different leak factors lies within a range of $\sim 2\%$. In addition to sparsity, leak makes the membrane potential (in turn, the output spike rate) dependent on the temporal information of the incoming spike train [20]. Therefore, for a given input, while the IF neuron produces a deterministic spike pattern, the input-output spike mapping is non-deterministic in an LIF neuron. This effect gets enhanced with increased leak. We assume that this phenomenon is also responsible to some extent for the increased robustness of backpropagated SNNs with increased leak. It is worth mentioning here that Eq. 7 holds when input to the neuron remains unchanged with the leak factor. In our experiments, we train SNN-BP with different values of λ starting from the same initialized ANN-SNN converted network. Hence, the parameters of SNN-BP trained with different leak factors do not vary much from one another. Therefore, the assumption in the equation applies to our results.

4.2 ANN-Crafted vs SNN-Crafted Attack

Lastly, we propose an attack-crafting technique from SNN with the aid of the surrogate gradient calculation. The details of the method is explained in Sec. 3.3. Table 2 summarizes a comparison between ANN-crafted and SNN-crafted (our proposed technique) attacks. Note, these are blackbox attacks, *i. e.*, we train two separate and independently initialized models for each of the 3 networks (ANN, SNN-conv, SNN-BP). One of them is used as the source (*marked as ANN-I, SNN-conv-I etc.*) and the other ones as the target (*marked as ANN-II, SNN-conv-II etc.*). It is clear that SNN-BP adversarial accuracy (last row) is the

Table 2. A comparison of the blackbox adversarial accuracy for ANN-crafted *versus* SNN-crafted attacks. ANN-I and ANN-II are two separately trained VGG5 networks with different initializations. The same is true for SNN-conv and SNN-BP.

	FGSM			PGD		
Source / Target	ANN-I	SNN-conv-I	SNN-BP-I	ANN-I	SNN-conv-I	SNN-BP-I
ANN-II	18.9%	32.7%	31.3%	4.7%	31.7%	13.8%
SNN-conv-II	19.2%	33.0%	31.4%	11.6%	32.4%	14.3%
SNN-BP-II	21.5%	38.8%	32.9%	9.7%	43.6%	17.0%

highest for both SNN-crafted and ANN-crafted inputs. Moreover, let us analyze row 1 of Table 2 for FGSM attack. When ANN-II is attacked by ANN-I, FGSM accuracy is 18.9%, whereas, if attacked by an SNN-conv-I (or SNN-BP-I), the accuracy is 32.7% (or 31.3%). Hence, these results suggest that ANN-crafted attacks are stronger than the corresponding SNN counterparts.

5 Conclusions

The current defense mechanisms in ANNs are incapable of preventing a range of adversarial attacks. In this work, we show that SNNs are inherently resilient to gradient-based adversarial attacks due to the discrete nature of input encoding and non-linear activation functions of LIF (or IF) neurons. The resiliency can be further improved by reducing the number of timesteps in the input-spike generation process and increasing the amount of leak of the LIF neurons. SNNs trained using ANN-SNN conversion technique (with IF neurons) require larger number of timesteps for inference than the corresponding SNNs trained with spike-based backpropagation (with LIF neurons). Hence, the latter technique leads to more robust SNNs. Our conclusion is validated only for gradient-based attacks on deep VGG and ResNet networks with CIFAR datasets. Future analysis on more diverse attack methods and architectures is necessary. We also propose a method to generate gradient-based attacks from SNNs by using the surrogate gradients.

Acknowledgement

The work was supported in part by, Center for Brain-inspired Computing (C-BRIC), a DARPA sponsored JUMP center, Semiconductor Research Corporation, National Science Foundation, Intel Corporation, the DoD Vannevar Bush Fellowship and U.S. Army Research Laboratory.

References

1. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: *Proceedings of the International Conference on Machine Learning (ICML)* (2018)
2. Bagheri, A., Simeone, O., Rajendran, B.: Adversarial training for probabilistic spiking neural networks. In: *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)* (2018)
3. Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., Maass, W.: Long short-term memory and learning-to-learn in networks of spiking neurons. In: *Advances in Neural Information Processing Systems*. pp. 787–797 (2018)
4. Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., Roli, F.: Evasion attacks against machine learning at test time. ECML PKDD, part III pp. 387–402 (2013), springer
5. Buckman, J., Roy, A., Raffel, C., Goodfellow, I.: Thermometer encoding: One hot way to resist adversarial examples. In: *International Conference on Learning Representations (ICLR)* (2018)
6. Carlini, N., Wagner, D.: Defensive distillation is not robust to adversarial examples (2016), arXiv:1607.04311
7. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: *2017 IEEE Symposium on Security and Privacy (SP)*. pp. 39–57 (2017)
8. Dhillon, G.S., Azizzadenesheli, K., Lipton, Z.C., Bernstein, J., Kossaifi, J., Khanna, A., Anandkumar, A.: Stochastic activation pruning for robust adversarial defense. In: *International Conference on Learning Representations (ICLR)* (2018)
9. Diehl, P.U., Neil, D., Binas, J., Cook, M., Liu, S.C., Pfeiffer, M.: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: *2015 International Joint Conference on Neural Networks (IJCNN)*. pp. 1–8 (2015)
10. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical world attacks on deep learning visual classifications. In: *CVPR* (2018)
11. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: *International Conference on Learning Representations (ICLR)* (2015)
12. Guo, C., Rana, M., Cisse, M., van der Maaten, L.: Countering adversarial images using input transformations (2018)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. Tech. rep., Microsoft Research, arXiv: 1512.03385 (2015)
14. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *Proceedings of International Conference on Machine Learning (ICML)*. pp. 448–456 (2015)
15. Kingma, D.P., Ba., J.L.: Adam: A method for stochastic optimization. In: *International Conference on Learning Representations (ICLR)* (2014)
16. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. In: *International Conference on Learning Representations (ICLR)* (2017)
17. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-Supervised nets (2014)
18. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: *International Conference on Learning Representations (ICLR)* (2018)
19. Marchisio, A., Nanfa, G., Khalid, F., Hanif, M.A., Martina, M., Shafique, M.: SNN under attack: are spiking deep belief networks vulnerable to adversarial examples? (2019)

20. Olin-Ammentorp, W., Beckmann, K., Schuman, C.D., Plank, J.S., Cady, N.C.: Stochasticity and robustness in spiking neural networks (2019)
21. Panda, P., Chakraborty, I., Roy, K.: Discretization based solutions for secure machine learning against adversarial attacks. *IEEE Access* **7**, 70157–70168 (2019)
22. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: Security and Privacy (SP), 2016 IEEE Symposium on. pp. 582–597 (2016)
23. Rathi, N., Srinivasan, G., Panda, P., Roy, K.: Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In: *International Conference on Learning Representations (ICLR)* (2020)
24. Sengupta, A., Ye, Y., Wang, R., Liu, C., Roy, K.: Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in neuroscience* **13:95** (2019)
25. Sharmin, S., Panda, P., Sarwar, S.S., Lee, C., Ponghiran, W., Roy, K.: A comprehensive analysis on adversarial robustness of spiking neural networks. In: *2019 International Joint Conference on Neural Networks (IJCNN)* (2019)
26. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: *International Conference on Learning Representations (ICLR)* (2015)
27. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., I.Goodfellow, Fergus, R.: Intriguing properties of neural networks. In: *International Conference on Learning Representations (ICLR)* (2014)
28. Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., McDaniel, P.: Ensemble adversarial training: Attacks and defenses. In: *International Conference on Learning Representations (ICLR)* (2018)
29. Werbos, P.J.: Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* **78**(10), 1550–1560 (1990)
30. Wu, Y., Deng, L., Li, G., Zhu, J., Shi, L.: Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience* **12** (2018)
31. Xu, W., Evans, D., Qi, Y.: Feature squeezing: Detecting adversarial examples in deep neural networks. In: *Network and Distributed Systems Security Symposium (NDSS)* (2018)
32. Xu, W., Qi, Y., Evans, D.: Automatically evading classifiers. In: *Network and Distributed Systems Security Symposium (NDSS)* (2016)
33. Zenke, F., Ganguli, S.: Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation* **30**(6), 1514–1541 (2018)