This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

HIRE-SNN: Harnessing the Inherent Robustness of Energy-Efficient Deep Spiking Neural Networks by Training with Crafted Input Noise

Souvik Kundu, Massoud Pedram, Peter A. Beerel
University of Southern California, Los Angeles, CA, USA
{souvikku, pedram, pabeerel}@usc.edu

Abstract

Low-latency deep spiking neural networks (SNNs) have become a promising alternative to conventional artificial neural networks (ANNs) because of their potential for increased energy efficiency on event-driven neuromorphic hardware.Neural networks, including SNNs, however, are subject to various adversarial attacks and must be trained to remain resilient against such attacks for many applications. Nevertheless, due to prohibitively high training costs associated with SNNs, an analysis and optimization of deep SNNs under various adversarial attacks have been largely overlooked. In this paper, we first present a detailed analysis of the inherent robustness of low-latency SNNs against popular gradient-based attacks, namely fast gradient sign method (FGSM) and projected gradient descent (PGD). Motivated by this analysis, to harness the model's robustness against these attacks we present an SNN training algorithm that uses crafted input noise and incurs no additional training time. To evaluate the merits of our algorithm, we conducted extensive experiments with variants of VGG and ResNet on both CIFAR-10 and CIFAR-100 dataset. Compared to standard trained direct-input SNNs, our trained models yield improved classification accuracy of up to 13.7% *and* 10.1% *on FGSM and PGD attack generated images, respectively, with negligible loss in clean image accuracy. Our models also outperform inherently-robust SNNs trained on rate-coded inputs with improved or similar classification performance on attack-generated images while having up to* 25× *and* ∼4.6× *lower latency and computation energy, respectively. For reproducibility, we have open-sourced the code at* github.com/ksouvik52/hiresnn2021.

1. Introduction

Artificial neural networks (ANNs) have become enormously successful in various computer vision applications [36, 10, 31, 37, 19]. However, as these applications are often part of safety-critical and trusted systems, concerns about their vulnerability to adversarial attacks have grown rapidly. In particular, well crafted adversarial images with small, often unnoticeable perturbations can fool a well trained ANN to make incorrect and possibly dangerous decisions [25, 1, 39], despite their otherwise impressive performance on clean images. To improve the model performance of ANNs against attacks, training with various adversarially generated images [22, 17] has proven to be very effective. Few other prior art references [40, 33] have applied noisy inputs to train robust models. However, all these training schemes incur non-negligible clean image accuracy drop and require significant additional training time.

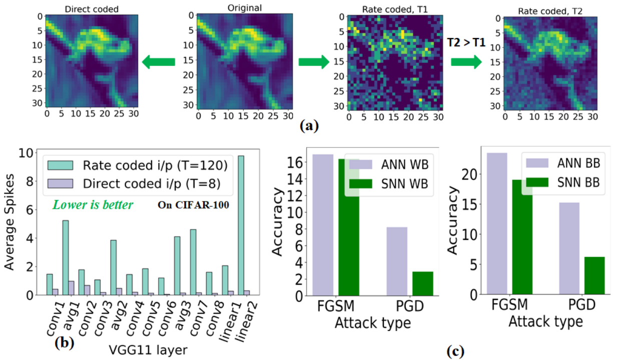

Figure 1. (a) Direct and rate-coded input variants of the original image. (b) Layer wise average spikes for VGG11. (c) Performance of direct-input VGG11 SNN and its equivalent ANN under various white-box (WB) and black-box (BB) attacks on CIFAR-100.

Brain-inspired [23] deep spiking neural networks (SNNs) have also gained significant traction due to their potential for lowering the required power consumption of machine learning applications [13, 28]. The underlying SNN hardware can use binary spike-based sparse processing via accumulate (AC) operations over a fixed number of time steps[1] T which consume much lower power than the traditional energy-hungry multiply-accumulate (MAC) operations that dominate ANNs [8]. Recent advances in SNN training by using approximate gradient [2] and hybrid direct-input-coded ANN-SNN training with joint threshold, leak, and weight optimization [29] have improved the SNN

[1] Here, a time step is the unit of time taken by each input image to be processed through all layers of the model.

accuracy while simultaneously reducing the number of required time steps. This has lowered both their computation cost, which is reflected in their average spike count as shown in Fig. 1(b), and inference latency. However, the trustworthiness of these state-of-the-art (SOTA) SNNs under various adversarial attacks is yet to be fully explored.

Some earlier works have claimed that SNNs may have *inherent* robustness against popular gradient-based adversarial attacks [7, 34, 24]. In particular, *Sharmin et al.* [34] observed that rate-coded input-driven (Fig. 1(a)) SNNs have inherent robustness, which the authors primarily attributed to the highly sparse spiking activity of the model. However, these explorations are mostly limited to small datasets on shallow SNN models, and more importantly, these techniques give rise to high inference latency. This paper extends this analysis, asking two key questions.

1. *To what degree does SOTA low-latency deep SNNs retain their inherent robustness under both black-box and white-box adversarial-attack generated images?*

2. *Can computationally-efficient training algorithms improve the robustness of low-latency deep SNNs while retaining their high clean-image classification accuracy?*

Our contributions are two-fold. We first empirically study and provide detailed observations on inherent robustness claims about deep SNN models when the SNN inputs are directly coded. Interestingly, we observe that despite significant reductions in the average spike count, deep direct-input SNNs have lower classification accuracy compared to their ANN counterparts on various white-box and black-box attack generated adversarial images, as exemplified in Fig. 1(c).

Based on these observations, we present HIRE-SNN, a spike timing dependent backpropagation (STDB) based SNN training algorithm to better harness SNN's inherent robustness. In particular, we optimize the model trainable parameters using images whose pixel values are perturbed using crafted noise across the time steps. More precisely, we partition the training time steps T into $\mathcal{N}$ equal-length periods of length $\lfloor T/\mathcal{N} \rfloor$ and train each image-batch over each period, adding input noise after each period. The key feature of our approach is that, instead of showing the same image repeatedly, we efficiently use the time steps of SNN training to input different noisy variants of the same image. This avoids extra training time and, because we update the weights after each period, requires less memory for the storage of intermediate gradients compared to traditional SNN training methods. To demonstrate the efficacy of our scheme we conduct extensive evaluations with both VGG [35] and ResNet [10] SNN model variants on both CIFAR-10 and CIFAR-100 [14] datasets.

The remainder of this paper is arranged as follows. In Section 2 and 3 we present the necessary background and provide analysis of inherent robustness of direct-input SNNs, respectively. Section 4 presents our training scheme. We provide our experimental results and discussion on Section 5 and finally conclude in Section 6.

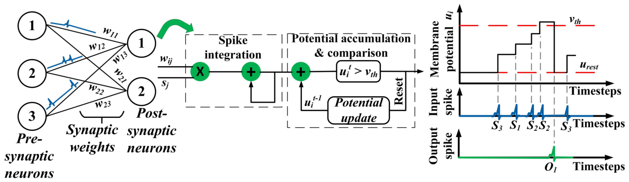

Figure 2. SNN fundamental operations.

2. Background

2.1. SNN Fundamentals

In ANN training, updating weights involve a single forward-backward pass transferring multi-bit weights and gradients through layers of a network. In contrast, in SNN training, updating weights involve T forward and backward passes, each pass propagating either binary spikes or a notion of their gradients. Note that T is known as the SNN's inference latency and the spiking dynamics of an SNN layer are typically defined with either the Integrate-Fire (IF) [21] or Leaky-Integrate-Fire (LIF) [20] neuron model. Interestingly, the LIF model introduces a non-linearity in the model that can be compared to the rectilinear (ReLU) operation in conventional ANNs. The discrete time [38] iterative version of the LIF neuron dynamics is defined by the following equation

$$u_i^{t+1} = \lambda u_i^t + \sum_j w_{ij} O_j^t - v_t O_i^t \tag{1}$$

$$O_i^t = \begin{cases} 1, & \text{if } z_i^t > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $z_i^t = (\frac{u_i^t}{v_t} - 1)$ is the normalized membrane potential and v_t is current layer firing threshold. The decay factor λ=1 for IF and $\lambda < 1$ for LIF neuron models. u_i^{t+1} represents the membrane potential of the i^{th} neuron at time step $t+1$, and O_i^t and O_j^t represent the output spikes of current neuron i and one of its pre-synaptic neurons j, respectively. w_{ij} represents the weight between the two. The inference output is obtained by comparing the total number of spikes generated by each output neuron over T time steps.

However, supervised training of SNNs faces the challenge of backpropagating gradients of binary spike trains which are undefined. This issue has been addressed using approximate gradient computations [2] at the cost of slow convergence and high memory requirements. In this paper, we refer to this step as *traditional* SNN training.

ANN-to-SNN conversion.

A popular alternative to traditional SNN training *from scratch* for deep SNNs involves first training a constrained ANN model [32] and then converting it into an SNN by computing layer thresholds [6, 32]. The SNN models yielded through this technique, however, require high latency T to perform well on complex vision tasks. We thus adopt a more recently developed hybrid training technique that leverages the benefits of the ANN-to-SNN conversion technique followed by a few epochs of direct-input driven traditional SNN training to reduce inference latency [29].

2.2. Adversarial Attacks

Various gradient-based adversarial attacks have been proposed to generate adversarial images, which have barely-visible perturbations from the original images but still manage to fool a trained neural network. One such attack is the fast gradient sign method (FGSM) [9]. Let $f(\boldsymbol{x},\boldsymbol{y})$ represents the function of an ANN, implicitly parameterized by network parameters $\boldsymbol{\theta}$, that accepts a vectorized input image $\boldsymbol{x}$ and generates a corresponding label $\boldsymbol{y}$. FGSM perturbs each element x in $\boldsymbol{x}$ along the sign of the gradient of the inference loss w.r.t. $\boldsymbol{x}$

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \epsilon * sign(\nabla_x \mathcal{L}(f(\boldsymbol{x},\boldsymbol{y}))) \tag{3}$$

where the scalar ϵ is the perturbation parameter that determines the severity of the attack.

Another well-known attack is projected gradient descent (PGD) [22]. It is a multi-step variant of FGSM and is known to be one of the most powerful first-order attacks [1]. Assuming $\hat{\boldsymbol{x}}^{k=1} = \boldsymbol{x}$ the iterative update of the perturbed data $\hat{\boldsymbol{x}}$ in k^{th} step of PGD is given in Eq. 4.

$$\hat{\boldsymbol{x}}^k = Proj_{P_\epsilon(\boldsymbol{x})}[\hat{\boldsymbol{x}}^{k-1} + \alpha * sign(\nabla_x \mathcal{L}(f(\hat{\boldsymbol{x}}^{k-1}, \boldsymbol{y})))] \tag{4}$$

Here, *Proj* projects the updated adversarial sample onto the projection space $P_\epsilon(\boldsymbol{x})$, the ϵ-L_∞ neighbourhood of the benign sample[2] $\boldsymbol{x}$, and α is the attack step size.

Note that for both attack techniques we consider two scenarios: 1) white-box (WB) attack in which the attacker has complete access to the model parameters, and 2) black-box (BB) attack in which the attacker has no knowledge of the model's trainable parameters and thus produces weaker perturbations than the white-box alternative.

3. Initial Study: SNN Robustness Analysis

To motivate our novel training algorithm to harness robustness, this section describes an empirical analysis into the robustness of traditionally-trained SNNs on gradient-based adversarial attacks. We performed traditional SNN training with the initial weights and thresholds set to that of a trained equivalent ANN and generated through the conversion process, respectively[3].

3.1. Performance Analysis

We first performed SNN training with direct-coded inputs and evaluated the robustness of the trained models under various white-box and black-box attacks. Interestingly, as shown in Table 1, the generated deep SNNs, i.e., VGG11 and ResNet12, consistently provide inferior performance against various black-box attacks compared to their ANN counterparts. For example, we observe that the VGG11 SNN provides only 6.2% accuracy on the PGD black-box attack, while its ANN equivalent provides an accuracy of 15.25%. These results imply that traditional SNN training appears to be insufficient to harness the inherent robustness of low-latency direct-input deep SNNs.

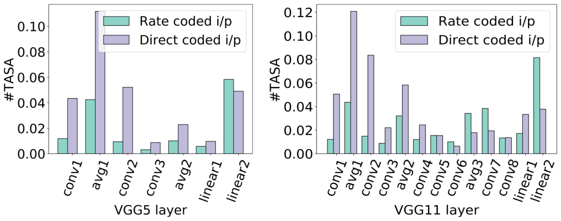

Figure 3. Per layer TASAs of VGG5 and VGG11 on CIFAR-10 and CIFAR-100, respectively.

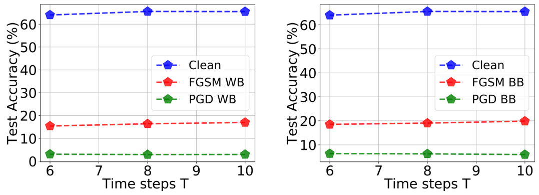

Figure 4. Classification performance of VGG11 on CIFAR-100 as number of time steps T varies.

It is important to note that [34] observed that, for rate-coded SNNs, spike based sparse activation maps correlates with adversarial success. To extend this analysis to direct-input SNNs, we examine two distinct metrics of the SNN's spiking activity, as defined below.

Definition 1. *Spiking activity (SA)*: We define a layer's spiking activity as the ratio of number of spikes produced over all the neurons accumulated across all time units T of a layer to the total number of neurons present in that layer. We also define a layer's SA divided by T as the *time averaged spiking activity* (TASA).

Observation 1. *Compared to rate-coded SNNs, deep SNNs with direct-coded inputs and lower latency generally exhibit lower SA but higher TASA, especially in the initial convolution layers.*

Particularly, this distinction can be seen for VGG11 SNN in Figs. 3(b) and 1(b) and suggests that the observation that sparse SA correlates well with success against adversarial images [34] can extend to low-latency direct-coded SNNs if the spiking activity is quantified using TASA. VGG5 also shows lower sparsity level of spikes at the early layers (Fig. 3(a)) compared to its rate-coded counterpart.

Observation 2. *Direct-input coded SNNs yield lower clean accuracy and no significant improvement in adversarial image classification accuracy as latency T is reduced.*

Earlier research has shown that robustness to adversarial images of SNNs trained on rate-coded inputs improves

[2] It is noteworthy that the generated $\hat{\boldsymbol{x}}$ are clipped to a valid range which for our experiments is [0, 1].

[3] The description of our training hyperparameter settings is given in Section 5.1 for all the experiments in this section.

Model- Attack category	Accuracy (%) with ANN			Accuracy (%) with high latency SNN-BP [34]			Accuracy (%) with low latency SNN-BP		
	Clean	FGSM	PGD	Clean	FGSM	PGD	Clean	FGSM	PGD
Dataset : CIFAR-10									
VGG5-WB	90.2	13.3	2.0	*89.3*	*15.0*	*3.8*	87.9	35.5	5.3
VGG5-BB	90.2	24.0	6.4	*89.3*	*21.5*	*16*	87.9	38.3	6.7
ResNet12-WB	92.6	19.9	2.0	–	–	–	91.9	21.1	0.2
ResNet12-BB	92.6	28.6	4.3	–	–	–	91.9	24.7	0.6
Dataset : CIFAR-100									
VGG11-WB	69.5	16.9	8.2	*64.4*	*15.5*	*6.3*	65.6	16.4	2.9
VGG11-BB	69.5	23.5	15.3	*64.4*	*21.4*	*16.5*	65.6	19.0	6.2
ResNet12-WB	61.5	13.5	2.8	–	–	–	61.9	10.5	0.6
ResNet12-BB	61.5	23.2	12.0	–	–	–	61.9	14.1	2.0

Table 1. Comparison of model performances under various white-box and black-box attacks on both CIFAR-10 and CIFAR-100. Note that *italicized* values are taken directly from the original paper.

with the reduction in training time steps [34]. Motivated by this we performed a similar analysis on VGG11 using direct input CIFAR-100. Interestingly, as shown in Fig. 4, as T reduces the classification performance on both black box and white box attack generated images does not improve. Intuitively, these attacks are more effective on direct-coded inputs because of the lack of approximation at the inputs, unlike for Poisson generated rate-coded inputs. However, SNNs with rate-coded inputs generally require larger training time and memory footprint [29] to reach competitive accuracy. In Fig. 4 we relate the reduction in network performance on clean images to the aggressive reduction in the number of training time steps.

Definition 2. *Perturbation distance (PD):* We define perturbation distance as the L_2-norm of the absolute difference of pixel values between a real image and its adversarially-perturbed variant. Similarly, for an intermediate layer with spike based activation maps [15, 16], we define *spike PD* as the L_2-norm of the absolute difference of the normalized spike-based activation maps generated at a layer output when fed with an original and its perturbed variant, respectively.

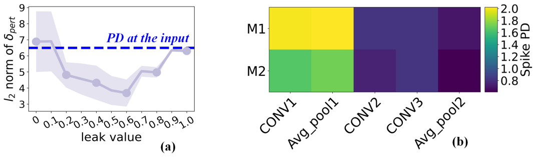

Figure 5. (a) PD vs LIF leak parameter for a fixed threshold (0.8) and latency ($T = 10$) averaged over two randomly chosen input images that are perturbed with PGD-1. (b) Intermediate layer spike PD for VGG5 fed with a randomly-selected CIFAR-10 clean image and its perturbed variant.

Observation 3. *Leaky integrate and fire (LIF) nonlinearity applying layers contribute to the inherent robustness of rate-coded input driven SNNs by diminishing the perturbation distance [34]. Unfortunately, this observation does not generally hold for direct-coded SNNs in which the LIF layers may increase or degrade the perturbation distance, suggesting that the impact of the leak parameter must be considered jointly with other factors, including related weights and thresholds.*

The LIF operation in SNNs yields non-linear dynamics that can be contrasted to the piecewise linear ReLU operation in traditional ANNs. To analyze their impact on image perturbation distance, we fed an LIF layer the clean images taken from a digit classification dataset [5] along with their perturbed variants, sweeping the leak parameter value and measuring the impact on the perturbation distance. As depicted in Fig. 5(a), the leak factor helps reduce the perturbation distance only if its value falls in a certain range.

To further study the impact of LIF layers, we analyzed the spike PD of the models. In particular, we fed two VGG5 SNN models trained with two different seeds ($M1$ and $M2$) with a randomly-sampled CIFAR-10 clean image $\boldsymbol{x}_C$ and its black-box attack generated variant $\boldsymbol{x}_P$ and computed the corresponding intermediate layer spike PDs. Both $M1$ and $M2$ classified $\boldsymbol{x}_C$ correctly, however, $M2$ failed to correctly classify $\boldsymbol{x}_P$. Interestingly, as shown in Fig. 5(b) despite the presence of the LIF layers, the spike PD values do not always reduce as we progress from layer to layer through the network. Moreover, this degree of unpredictability seems to be irrespective of whether the model classifies the image correctly. We conclude that despite LIF's promise to reduce input perturbation, its impact is also a function of other parameters, including the trainable weights, leak, threshold, and time steps.

Based on these empirical observations, we assert that the majority of the reasons that make rate-coded SNN inherently robust are either absent or need careful tuning for direct-input SNN models, as presented in the next section.

4. HIRE-SNN Training

This section presents our training algorithm for robust SNNs. As shown in Eq. 2 the LIF neuron functional output at each time step recursively depends on its state in previous time steps [26]. Each input pixel in traditional SNN training using direct-coded inputs, is fed into the network as a multi-bit value that is *fixed* over the T time steps and yield an order of magnitude reduction in latency compared to rate-coded alternatives. However, our approach is different than direct coding because we partition the training time steps T into

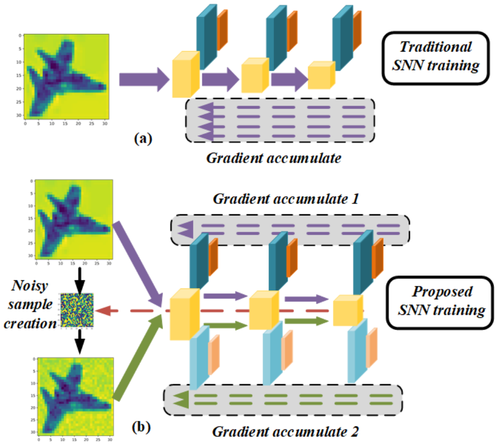

Figure 6. (a) and (b) represent the traditional and proposed training schemes, respectively. Here the green and orange blocks represent activation maps and the gradients that are generated after passing the input image. For the proposed training scheme we use two color variants deep and light, respectively, to highlight the sets of activation maps and gradients from an image and its noisy variant during two different periods. The yellow blocks represent the weight tensors that get updated from accumulated gradients. In (b) we compute the input gradient with these updated weights to craft the noise. Here, we assumed $T = 4$ and $\mathcal{N} = 2$.

$\mathcal{N}$ equal-length periods and feed in a different perturbed variant of the image during each period of $\lfloor T/\mathcal{N} \rfloor$ steps.

To be more precise, consider an SNN model defined by the function $g(\boldsymbol{x}, \boldsymbol{y}; T)$ implicitly parameterized by $\boldsymbol{\theta}$. Assume an input batch $\mathcal{B}$ of size $h_i \times w_i \times c_i \times n_{\mathcal{B}}$, where h_i, w_i, and c_i represent spatial height, width, and channel length of an image, respectively, with $n_{\mathcal{B}}$ as the number of images in the batch. In contrast to traditional approaches, where weight update happens only after T steps, we allow different perturbed image variants generation and weight update to happen at small interval of $\lfloor T/\mathcal{N} \rfloor$ steps within the window of T, for an image batch. This important modification allows us to train the model with different adversarial image variants without costing any additional training time. As the exact gradient of the binary spike trains is undefined, we use a linear surrogate gradient (Eq. 5) approximation [2] to allow backpropagation (BP) and gradient-based parameter update in SNNs.

$$\frac{\partial O_i^t}{\partial z_i^t} = \gamma * max\{0, 1 - |z_i^t|\} \tag{5}$$

where γ is a damping factor that controls the approximate back-propagation error, to update the trainable parameters. We also compute the gradient of the loss with respect to each input pixel x to craft the perturbation for next period. Through an abuse of notation, we define ϵ_s and ϵ_t as the pixel noise step and bound, respectively, and generate perturbation scalar for each of the $h_i \times w_i \times c_i$ pixels of an image as

$$\kappa = clip[\kappa + \epsilon_s \times sign(\nabla_x \mathcal{L}), -\epsilon_t, +\epsilon_t] \tag{6}$$

Algorithm 1: HIRE-SNN Training Algorithm

```
Input: Training examples (X, Y), noise bound [-$\epsilon_t$, $\epsilon_t$], noise step
  $\epsilon_s$, learning rate $\eta$, SNN training t-steps $T$, total training epochs
  $N_{ep}$, iteration $\mathcal{N}$.
// Initialize parameters
$\kappa \leftarrow 0$
for $l \leftarrow 1$ to $L$ do
    $\mathbf{W}^l \leftarrow$ ANN trained $\mathbf{W}^l$
    $v_t^l \leftarrow initThreshold(\mathbf{W}^l, X)$
    $l_k^l \leftarrow 1.0$
end
for $n \leftarrow 1$ to $N_{ep}$ do
    for each batch $\mathcal{B} \subset (X, Y)$ do
        for $p \leftarrow 1$ to $\mathcal{N}$ do
            // Compute gradients through STDB
            $\delta_{\mathbf{W}} \leftarrow \mathbb{E}_{(x,y)\in\mathcal{B}}[\nabla_{\mathbf{W}}\mathcal{L}(g(x+\kappa, y; \frac{T}{\mathcal{N}}))]$
            $\delta_{v_t} \leftarrow \mathbb{E}_{(x,y)\in\mathcal{B}}[\nabla_{v_t}\mathcal{L}(g(x+\kappa, y; \frac{T}{\mathcal{N}}))]$
            $\delta_{l_k} \leftarrow \mathbb{E}_{(x,y)\in\mathcal{B}}[\nabla_{l_k}\mathcal{L}(g(x+\kappa, y; \frac{T}{\mathcal{N}}))]$
            // Compute perturbation
            $\delta_x \leftarrow [\nabla_x\mathcal{L}(g(x+\kappa, y; \frac{T}{\mathcal{N}}))]$
            $\kappa \leftarrow \text{clip}(\kappa + \epsilon_s * sign(\delta_x), -\epsilon_t, \epsilon_t)$
            // Update trainable parameters
            $\mathbf{W} \leftarrow \mathbf{W} - \eta * \delta_{\mathbf{W}}$
            $v_t \leftarrow v_t - \eta * \delta_{v_t}$
            $l_k \leftarrow l_k - \eta * \delta_{l_k}$
        end
    end
end
```

where κ represents the perturbation for an input pixel x of a batch $\mathcal{B}$ computed at the p^{th} period. Note that for current batch, we initialize κ in the first period with the perturbation computed at the last period of the previous batch. In contrast, the computation of the perturbation of other periods is based on the computed perturbation from the corresponding previous period. It is noteworthy that ϵ_s is not necessarily the same as ϵ of the FGSM or PGD attacks, and we generally choose ϵ_s to be sufficiently small to not lose significant classification accuracy on clean images. We include weights $\mathbf{W}$, threshold $\boldsymbol{v}_t$ and leak $\boldsymbol{l}_k$ parameters in the trainable parameters $\boldsymbol{\theta}$ to retain clean image accuracy at low latencies [29]. Our detailed training algorithm called HIRE-SNN is presented in Algorithm 1. It is noteworthy that, apart from noise crafted inputs, our training framework can easily be extended to support various input encoding [4, 34] as well as image augmentation techniques [30] that can improve classification performance.

5. Experiments

5.1. Experimental Setup

Dataset and ANN training. For our experiments we used two widely accepted image classification datasets, namely CIFAR-10 and CIFAR-100. For both ANN and direct-input SNN training, we use the standard data-augmented (horizontal flip and random crop with reflective padding) input. For rate-coded input based SNN training, we produce a spike train with rate proportional to the input pixel via a Poisson generator function [15]. We performed ANN training for 240 epochs with an initial learning rate (LR) of 0.01 that decayed by a factor of 0.1 after 150, 180, and 210 epochs.

ANN-SNN conversion and SNN training. We performed the ANN-SNN conversion as recommended in [29] to generate initial thresholds for the SNN training. We then train the converted SNN for only 30 epochs with batch-size of 32 starting with the trained ANN weights. We set starting LR to 10^{-4} and decay it by a factor of 5 after 60%, 80%, and 90% completion of the total training epochs. Unless stated otherwise, we used training time steps T of 6, 8, and 10 for VGG5, VGG11, and ResNet12, respectively. To avoid overfitting and perform regularization we used a dropout of 0.2 to train the models. The ϵ_s is chosen to be 0.013 and 0.025 (apart from the ϵ_s sweep test) to train with VGG5 and VGG11, respectively, with ϵ_t equal to ϵ_s. For ResNet12 we chose ϵ_s to be 0.008 and 0.015 on CIFAR-10 and CIFAR100, respectively. Also, $\mathcal{N}$ is set to 2 unless otherwise mentioned. The basic motivation to pick hyperparameters $\mathcal{N}$, ϵ_s, and ϵ_t is to ensure there is only an insignificant drop in the clean image accuracy while still improving the adversarial performance. We conducted all the experiments on a NVIDIA 2080 Ti GPU having 11 GB memory with the models implemented using PyTorch [27]. Further training and model details along with analysis on the hyperparameters are provided in the supplementary material.

Adversarial test setup. For PGD, we set ϵ for the L_∞ neighborhood to $8/255$, the attack step size $\alpha = 0.01$, and the number of attack iterations K to 7, the same values as in [34]. For FGSM, we choose the same ϵ value as above.

5.2. Performance Against WB and BB Attacks

To perform this evaluation, for each model variant we use three differently trained networks: ANN equivalent Φ_{ANN}, hybrid traditionally trained SNN Φ^T_{SNN}, and SNN trained with proposed technique Φ^P_{SNN}, all trained to have comparable clean-image classification accuracy. We compute Δ_d as the difference in clean-image classification performance between Φ^P_{SNN} and Φ^T_{SNN}. We define Δ_a[4] as the accuracy difference between Φ^P_{SNN} and either of Φ^T_{SNN} or Φ_{ANN} while classifying on perturbed image. Note, both Φ^P_{SNN} and Φ^T_{SNN} are trained with direct inputs. Table 2 shows the absolute and relative performances of the models generated through our training framework on white-box attack generated images using both FGSM and PGD attack techniques. In particular, we observe that with negligible performance compromise on clean images, Φ^P_{SNN} consistently outperforms Φ^T_{SNN} for all the models on both datasets. Specifically, we observe that the perturbed image classification can have an improved performance of up to 12.2% and 8.8%, on CIFAR-10 and CIFAR-100 respectively. Compared to Φ_{ANN} we observe improved performance of up to 25% on WB attacks.

Model	Accuracy (%) with proposed SNN training			Δ_a over traditional SNN training		Δ_a over ANN equivalent	
	Clean(Δ_d)	FGSM	PGD	FGSM	PGD	FGSM	PGD
Dataset : CIFAR-10							
VGG5	87.5 (-0.4)	38.0	9.1	+2.5	**+3.8**	**+25**	**+7.1**
ResNet12	90.3 (-1.6)	33.3	3.8	**+12.2**	+3.5	+13.4	+1.8
Dataset : CIFAR-100							
VGG11	65.1 (-0.4)	22.0	7.5	+5.7	+4.6	+5.1	-0.7
ResNet12	58.9 (-3.0)	19.3	5.3	**+8.8**	**+4.7**	**+5.8**	**+2.5**

Table 2. Performance comparison of SNN models generated using the proposed training scheme on clean and adversarially-generated images under a white-box attack.

Model	Accuracy (%) with proposed SNN training			Δ_a over traditional SNN training		Δ_a over ANN equivalent	
	Clean	FGSM	PGD	FGSM	PGD	FGSM	PGD
Dataset : CIFAR-10							
VGG5	87.5	42.1	14.9	+3.9	**+8.3**	**+18.1**	**+8.5**
ResNet12	90.3	38.4	7.8	**+13.7**	+7.2	+9.7	+3.5
Dataset : CIFAR-100							
VGG11	65.1	29.1	16.1	+10.0	+9.9	**+5.6**	**+0.9**
ResNet12	58.9	24.5	12.1	**+10.4**	**+10.1**	+1.3	$\sim$0

Table 3. Performance comparison of SNN models generated using the proposed training scheme on clean and adversarially-generated images under a black-box attack.

Table 3 shows the model performances and comparisons on black-box attack generated images using both FGSM and PGD. For this evaluation, for each model variant we used the same model trained with a different seed to generate the perturbed images. For all the models on both the datasets we observe Φ^P_{SNN} yields higher accuracy on the perturbed images generated through BB attack compared to those generated through WB attack, primarily because of BB attacks yield weaker perturbations [1, 3]. Importantly, we observe superior performance of Φ^P_{SNN} over both Φ^T_{SNN} and Φ^T_{SNN} under this weaker form of attack. In particular, Φ^P_{SNN} provides an improvement Δ_a of up to 13.7% and 10.4% on CIFAR-10 and CIFAR-100, compared to Φ^T_{SNN}.

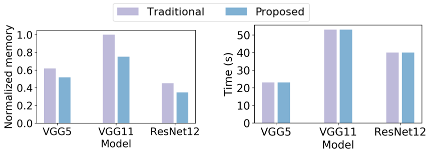

Figure 7. Normalized GPU memory usage and average training time for a batch of 200 images for VGG5, VGG11, and ResNet12 when trained with the traditional and proposed approaches.

Fig. 7 shows the normalized random access memory (RAM) memory and average training time for 200 batches for both the traditional and presented SNN training. Interestingly, due to the shorter update interval the proposed approach require less memory by up to ∼25% while incurring no extra GPU training time.

[4] Δ_a between model M1 and M2 is $Acc_{M1}\% - Acc_{M2}\%$.

Checks to identify gradient obfuscation	Fail	Pass
i) Single-step attack performs better compared to iterative attacks		✓
ii) Black-box attacks performs better compared to white-box attacks		✓
iii) Increasing perturbation bound can't increase attack strength		✓
iv) Unbounded attacks can't reach ∼100% success		✓
v) Adversarial example can be found through random sampling		✓

Table 4. Checklist set of tests for characteristic behaviors caused by obfuscated and masked gradients [1].

5.3. Discussion

Here, we evaluate the potential presence of obfuscated gradients through experiments with the HIRE-SNN trained models under different attack strengths. We then study the efficacy of noise crafting and performance under no trainable threshold-leak condition. Finally, we evaluate the impact of the new knob ϵ_s in trading off clean and perturbed image accuracy.

Gradient obfuscation analysis. We conducted several experiments to verify whether the inherent robustness of the presented HIRE-SNNs come from an incorrect approximation of the true gradient based on a single sample. In particular, the performance of generated models was checked against the five tests (Table 4) proposed in [1] that can identify potential gradient obfuscation.

As shown in Table 2 and 3, for all the models on both datasets the single-step FGSM performs poorly compared to its iterative counterpart PGD. This certifies the success of Test (i), as listed in Table 4. Test (ii) passes because our black-box generated perturbations in Table 3 yield weaker attacks[5] than their white-box counterparts shown in Table 2. To verify Tests (iii) and (iv) we analyzed VGG5 on CIFAR-10 with increasing attack bound ϵ. As shown in Fig. 8(a), the classification accuracy decreases as we increase ϵ and finally reaches an accuracy of $\sim 0\%$. Test (v) can fail only if gradient based attacks cannot provide adversarial examples for the model to misclassify. It is clear from our experiments, however, that FGSM and PGD, both variants of gradient based attacks, can sometimes fool the network despite our training.

We also evaluated the VGG5 performance with increased attack strength by increasing the number of iterations K of PGD and found that the model's robustness decreases with increasing K. However, as Fig. 8(b) shows, after $K = 40$, the robustness of the model nears an asymptote. In contrast, if the success of the HIRE-SNNs arose from the incorrect gradient of a single sample, increasing the attack iterations would have broken the defense completely [11].

Thus, based on these evaluations we conclude that even if the models include obfuscated gradients, they are not significant source of the robustness for the HIRE-SNNs.

Importance of careful noise crafting. To evaluate the merits of the presented noise crafting technique, we also trained VGG11 with a version of our training algorithm with the perturbation introduced via Gaussian noise. In particular, we pertubed the image pixels using Gaussian noise with zero mean and standard deviation equal to ϵ_s. It is clear from Fig. 9 that compared to the traditional training, the proposed training with perturbation generated through Gaussian noise (GN) fails to provide any noticeable improvement on the adversary-generated images both under white-box and black-box attacks. In contrast, training with carefully crafted noise significantly improves the performance over that with GN against adversary by up to 6.5% and 9.7%, on WB and BB attack-created images, respectively.

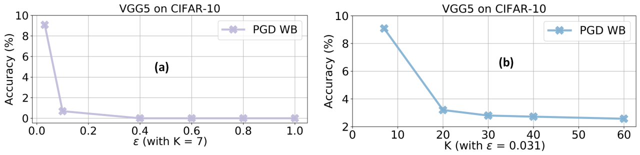

Figure 8. White-box PGD attack performance as a function of (a) bound ϵ and (b) attack iterations K with VGG5 on CIFAR-10.

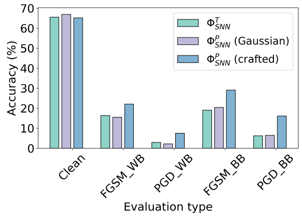

Figure 9. Comparison of traditional SNN vs. proposed training with both GN and crafted input noise. All the training were performed with direct-input VGG11 on CIFAR-100.

Efficacy of proposed training when threshold and leak parameters are not trainable. To further evaluate the efficacy of proposed training scheme, we trained VGG5 on CIFAR-10 using our technique but with threshold and leak parameters fixed to their initialized values. As shown in Table 5, our generated models still consistently outperform traditionally trained models under both white-box and black-box attacks with negligible drop in clean image accuracy. Interestingly, fixing the threshold and leak parameters yields higher robustness at the cost of lower clean-image accuracy. This may be attributed to the difference in adversarial strength of the perturbed images and is a useful topic of future research.

Impact of the noise-step knob ϵ_s. To analyze the impact of the introduced hyperparameter ϵ_s, we performed experiments with VGG5 and VGG11, training the models with various $\epsilon_s \in [0.01, 0.03]$. As depicted in Fig. 10 with increased ϵ_s the models show a consistent improvement on both white-box and black-box attack generated perturbed images with only a small drop in clean image performance of up to $\sim 2\%$. Note, here $\epsilon_s = 0$ corresponds to traditional SNN training. With the optimal choice of ϵ_s our models outperform the state-of-the-art inherently robust SNNs trained

[5] Note that here we say an attack is weaker than other when the classification accuracy on that attack-generated images is higher compared to the images generated through the other.

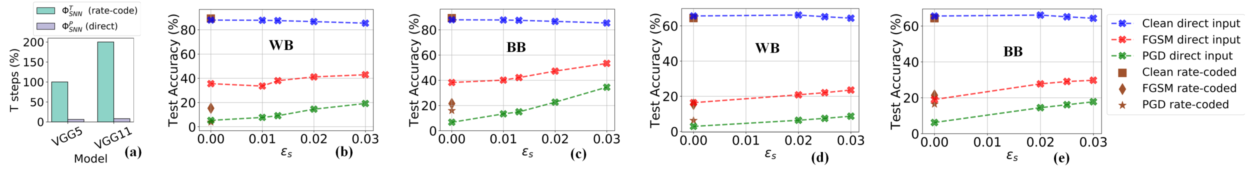

Figure 10. (a) Inference T steps for rate-coded vs direct input trained SNNs, (b-e) Accuracy vs. ϵ_s plot for both clean and adversarially generated images (both with WB and BB attack settings) with VGG5 (b, c) and VGG11 (d, e) on CIFAR-10 and CIFAR-100, respectively.

Model	Dataset	Training Method	Clean Acc. (%)	Acc. % on WB		Acc. % on BB	
				FGSM	PGD	FGSM	PGD
VGG5	CIFAR-10	Traditional	**87.2**	33.0	4.5	40.4	8.8
		Proposed	86.8	**40.5**	**13.6**	**46.2**	**21.9**

Table 5. Performance comparison of proposed with traditional SNN training when threshold-leak parameters are frozen to their initialized values.

on rate-coded inputs [34] maintaining similar clean image accuracy with an improved inference latency of up to 25× as shown in Fig. 10.

5.4. Computation Energy

Model	FLOPs of a CONV layer l	
	Variable	Value
ANN [18]	FL^l_{ANN}	$(k^l)^2 \times H^l_o \times W^l_o \times C^l_o \times C^l_i$
SNN [15]	FL^l_{SNN}	$(k^l)^2 \times H^l_o \times W^l_o \times C^l_o \times C^l_i \times \zeta^l$

Table 6. Convolutional layer FLOPs for ANN and SNN models

Let us assume a convolutional layer l with weight tensor $\mathbf{W}^l \in \mathbb{R}^{k^l \times k^l \times C^l_i \times C^l_o}$ taking an input activation tensor $\mathbf{A}^l \in \mathbb{R}^{H^l_i \times W^l_i \times C^l_i}$, with H^l_i, W^l_i, k^l, C^l_i and C^l_o to be the input height, width, filter height (and width), channel size, and number of filters, respectively. Table 6 presents the FLOPs requirement for an ANN and corresponding SNN for this layer to produce an output activation tensor $\mathbf{O}^l \in \mathbb{R}^{H^l_o \times W^l_o \times C^l_o}$. ζ^l represents the associated spiking activity for layer l. Now, for an L-layer SNN with rate-coded and direct inputs, the inference computation energy is,

$$E^{rate}_{SNN} = (\sum_{l=1}^{L} FL^l_{SNN}) \cdot E_{AC} \tag{7}$$

$$E^{direct}_{SNN} = FL^1_{SNN} \cdot E_{MAC} + (\sum_{l=2}^{L} FL^l_{SNN}) \cdot E_{AC} \tag{8}$$

where E_{AC} and E_{MAC} represent the energy cost of AC and MAC operation, respectively. For our evaluation we use their values as shown in Table 7. In particular, as exemplified in Fig. 11(a), the computation energy benefit of HIRE-SNN VGG11 over its inherently robust rate-coded SNN and ANN counterpart is as high as 4.6× and 10×, respectively, considering 32-b floating point (FP) representation. For a 32-b integer (INT) implementation, this advantage is as much as 3.9× and 53×, respectively (Fig. 11(b)).

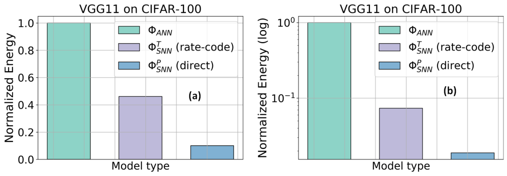

Figure 11. Comparison of normalized compute energy computed assuming (a) 32-bit FP and (b) 32-bit INT implementations.

Serial No.	Operation	Energy (pJ)	
		32-b INT	32-b FP
1.	32-bit multiplication	3.1	3.7
2.	32-bit addition	0.1	0.9
3.	32-bit MAC (#1 + #2)	3.2	4.6
4.	32-bit AC (#2)	0.1	0.9

Table 7. Estimated energy costs for various operations in a 45 nm CMOS process at 0.9 V [12]

6. Conclusions

In this paper we first analyzed the inherent robustness of low-latency SNNs trained with direct inputs to provide insightful observations. Motivated by these observations we then present a training algorithm that harnesses the inherent robustness of low-latency SNNs without incurring any additional training time cost. We conducted extensive experimental analysis to evaluate the efficacy of our training along with experiments to understand the contribution of the carefully crafted noise. Particularly, compared to traditionally trained direct input SNNs, the generated SNNs can yield accuracy improvement of up to 13.7% on black-box FGSM attack generated images. Compared to the SOTA inherently robust VGG11 SNN trained on rate-coded inputs (CIFAR-100) our models perform similarly or better on clean and perturbed image classification performance while providing an improved performance of up to 25× and ∼4.6×, in terms of inference latency and computation energy, respectively. We believe that this study is a step in making deep SNNs a practical energy-efficient solution for safety-critical inference applications where robustness is a need.

7. Acknowledgments

This work was partly supported by USC Annenberg fellowship and NSF including grant number 1763747.

References

[1] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Confer-*

ence on Machine Learning, pages 274–283. PMLR, 2018. 1, 3, 6, 7

[2] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. *arXiv preprint arXiv:1803.09574*, 2018. 1, 2, 5

[3] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017. 6

[4] Gourav Datta, Souvik Kundu, and Peter A Beerel. Training energy-efficient deep spiking neural networks with single-spike hybrid input encoding. *arXiv preprint arXiv:2107.12374*, 2021. 5

[5] Li Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 4

[6] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. Liu, and M. Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 1–8, 2015. 2

[7] Rida El-Allami, Alberto Marchisio, Muhammad Shafique, and Ihsen Alouani. Securing deep spiking neural networks against adversarial attacks through inherent structural parameters. *arXiv preprint arXiv:2012.05321*, 2020. 2

[8] Clément Farabet, Rafael Paz, Jose Pérez-Carrasco, Carlos Zamarreño, Alejandro Linares-Barranco, Yann LeCun, Eugenio Culurciello, Teresa Serrano-Gotarredona, and Bernabe Linares-Barranco. Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing. *Frontiers in neuroscience*, 6:32, 2012. 1

[9] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 3

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2

[11] Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 588–597, 2019. 7

[12] Mark Horowitz. 1.1 Computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014. 8

[13] Giacomo Indiveri and Timothy Horiuchi. Frontiers in neuromorphic engineering. *Frontiers in Neuroscience*, 5:118, 2011. 1

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2

[15] Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A. Beerel. 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 (WACV)*, pages 3953–3962, January 2021. 4, 6, 8

[16] Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A Beerel. Towards low-latency energy-efficient deep snns via attention-guided compression. *arXiv preprint arXiv:2107.12445*, 2021. 4

[17] Souvik Kundu, Mahdi Nazemi, Peter A Beerel, and Massoud Pedram. Dnr: A tunable robust pruning framework through dynamic network rewiring of dnns. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, pages 344–350, 2021. 1

[18] Souvik Kundu, Mahdi Nazemi, Massoud Pedram, Keith M Chugg, and Peter Beerel. Pre-defined sparsity for low-complexity convolutional neural networks. *IEEE Transactions on Computers*, 2020. 8

[19] Souvik Kundu and Sairam Sundaresan. Attentionlite: Towards efficient self-attention models for vision. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2225–2229. IEEE, 2021. 1

[20] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 14:119, 2020. 2

[21] Sen Lu and Abhronil Sengupta. Exploring the connection between binary and spiking neural networks. *arXiv preprint arXiv:2002.10064*, 2020. 2

[22] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 1, 3

[23] Zachary F Mainen and Terrence J Sejnowski. Reliability of spike timing in neocortical neurons. *Science*, 268(5216):1503–1506, 1995. 1

[24] Alberto Marchisio, Giorgio Nanfa, Faiq Khalid, Muhammad Abdullah Hanif, Maurizio Martina, and Muhammad Shafique. Is spiking secure? A comparative study on the security vulnerabilities of spiking and deep neural networks. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020. 2

[25] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016. 1

[26] E. O. Neftci, H. Mostafa, and F. Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019. 4

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[28] Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, 12:774, 2018. 1

[29] Nitin Rathi and Kaushik Roy. DIET-SNN: Direct input encoding with leakage and threshold optimization in deep

spiking neural networks. *arXiv preprint arXiv:2008.03658*, 2020. 1, 3, 4, 5, 6

[30] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021. 5

[31] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 1

[32] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience*, 13:95, 2019. 2

[33] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019. 1

[34] Saima Sharmin, Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations. In *European Conference on Computer Vision*, pages 399–414. Springer, 2020. 2, 3, 4, 5, 6, 8

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 1

[37] Hu Tao, Weihua Li, Xianxiang Qin, and Dan Jia. Image semantic segmentation based on convolutional neural network and conditional random field. In *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, pages 568–572. IEEE, 2018. 1

[38] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018. 2

[39] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1369–1378, 2017. 1

[40] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. *arXiv preprint arXiv:1905.00877*, 2019. 1