Adversarial Training for Probabilistic Spiking Neural Networks

Alireza Bagheri
†*ECE Department*
New Jersey Institute of Technology
Newark, NJ 07102, USA
Email: *ab745@njit.edu*

Osvaldo Simeone†
Department of Informatics
King's College London
London, WC2R 2LS, UK
Email: *osvaldo.simeone@kcl.ac.uk*

Bipin Rajendran
ECE Department
New Jersey Institute of Technology
Newark, NJ 07102, USA
Email: *bipin@njit.edu*

***Abstract*—Classifiers trained using conventional empirical risk minimization or maximum likelihood methods are known to suffer dramatic performance degradations when tested over examples adversarially selected based on knowledge of the classifier's decision rule. Due to the prominence of Artificial Neural Networks (ANNs) as classifiers, their sensitivity to adversarial examples, as well as robust training schemes, have been recently the subject of intense investigation. In this paper, for the first time, the sensitivity of spiking neural networks (SNNs), or third-generation neural networks, to adversarial examples is studied. The study considers rate and time encoding, as well as rate and first-to-spike decoding. Furthermore, a robust training mechanism is proposed that is demonstrated to enhance the performance of SNNs under white-box attacks.**

***Index Terms*—Spiking Neural Networks (SNNs), adversarial examples, adversarial training, Generalized Linear Model (GLM)**

I. INTRODUCTION

The classification accuracy of Artificial Neural Networks (ANNs) trained over large data sets from the problem domain has attained super-human levels for many tasks including image identification [1]. Nevertheless, the performance of classifiers trained using conventional empirical risk minimization or Maximum Likelihood (ML) is known to decrease dramatically when evaluated over examples adversarially selected based on knowledge of the classifier's decision rule [2]. To mitigate this problem, robust training strategies that are aware of the presence of adversarial perturbations have been shown to improve the accuracy of classifiers, including ANNs, when tested over adversarial examples [2]–[4].

ANNs are known to be energy-intensive, hindering their implementation on energy-limited processors such as mobile devices. Despite the recent industrial efforts around the production of more energy-efficient chips for ANNs [5], the gap between the energy efficiency of the human brain and that of ANNs remains significant [6], [7]. A promising alternative paradigm is offered by Spiking Neural Networks (SNNs), in which synaptic input and neuronal output signals are sparse asynchronous binary spike trains [5]. Unlike ANNs, SNNs are hybrid digital-analog machines that make use of the temporal dimension, not just as a neutral substrate for computing, but as a means to encode and process information [7].

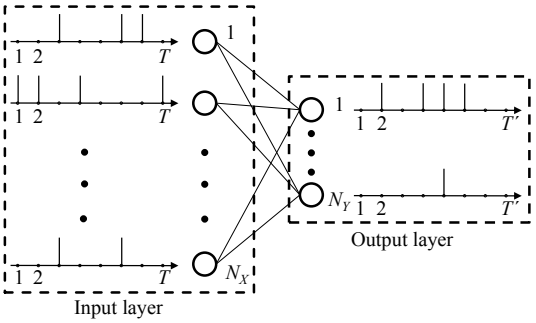

Fig. 1. Two-layer SNN for supervised learning.

Training methods for SNNs typically assume deterministic non-linear dynamic models for the spiking neurons, and are either motivated by biological plausibility, such as the spike-timing-dependent plasticity (STDP) rule [5], [8], or by an attempt to mimic the operation of ANNs and associated learning rules (see, e.g., [9] and references therein). Deterministic models are known to be limited in their expressive power, especially as it pertains prior domain knowledge, uncertainty, and definition of generic queries and tasks. Training for probabilistic models of SNNs has recently been investigated in, e.g., [10]–[13] using ML and variational inference principles.

In this paper, for the first time, the sensitivity of SNNs trained via ML is studied under white-box adversarial attacks, and a robust training mechanism is proposed that is demonstrated to enhance the performance of SNNs under adversarial examples. Specifically, we focus on a two-layer SNN (see Fig. 1), and consider rate and time encoding, as well as rate and first-to-spike decoding [13]. Our results illuminate the sensitivity of SNNs to adversarial example under different encoding and decoding schemes, and the effectiveness of robust training methods.

The rest of the paper is organized as follows. In Sec. II, we describe the architecture of the two-layer SNN with Generalized Linear Model (GLM) neuron, as well as information encoding and decoding mechanisms. The design of adversarial perturbations is covered in Sec. III, while a robust training is presented in Sec. IV. Sec. V presents numerical results, and closing remarks are given in Sec. VI.

II. SNN-BASED CLASSIFICATION

In this section, we introduce the classification task and the SNN architecture under study.

Network Architecture: We consider the problem of classification using the two-layer SNN illustrated in Fig. 1. The SNN is fully connected and has N_X presynaptic neurons in the input, or sensory layer, and N_Y neurons in the output layer. Each output neuron is associated with a class. In order to feed the SNN, an input example, e.g., a gray scale image, is encoded into a set of N_X discrete-time spike trains, each with T samples. The input spike trains are fed to the N_Y postsynaptic GLM neurons, which output discrete-time spike trains. A decoder then selects the image class on the basis of the spike trains emitted by the output neurons.

Information Encoding: We consider two encoding mechanisms.

1) Rate encoding: With the conventional rate encoding method (see, e.g., [14]), each entry of the input signal is converted into a discrete-time spike train by generating an independent and identically distributed (i.i.d.) Bernoulli vector. The probability of generating a "1", i.e., a spike, is proportional to the value of the entry. In the experiments in Sec. V, we use gray scale images of USPS dataset with pixel intensities normalized between 0 and 1 that yield a proportional spike probability between 0 and $1/2$.

2) Time encoding: With the time encoding method, each entry of the input signal is converted into a spike train having only one spike, whose timing depends on the entry value. In particular, assuming intensity-to-latency encoding [14]–[16], the spike timing in the time interval $[1, T]$ depends linearly on the entry value, such that the maximum value yields a spike at the first time sample $t = 1$, and the minimum value is mapped to a spike in the last time sample $t = T$.

GLM Neuron Model: The relationship between the input spike trains from the N_X presynaptic neurons and the output spike train of any postsynaptic neuron i follows a Bernoulli GLM with canonical link function (see, e.g., [13], [17]). To elaborate, we denote as $x_{j,t}$ and $y_{i,t}$ the binary signal emitted by the j-th presynaptic and the i-th postsynaptic neurons, respectively, at time t. Also, we let $\mathbf{x}_{j,a}^{b} = (x_{j,a}, ..., x_{j,b})$ be the vector of samples from spiking process of the presynaptic neuron j in the time interval $[a, b]$. Similarly, the vector $\mathbf{y}_{i,a}^{b} = (y_{i,a}, ..., y_{i,b})$ contains samples from the spiking process of the neuron i in the interval $[a, b]$. The membrane potential of postsynaptic neuron i at time t is given by

$$u_{i,t} = \sum_{j=1}^{N_X} \boldsymbol{\alpha}_{j,i}^T \mathbf{x}_{j,t-\tau_y}^{t-1} + \boldsymbol{\beta}_i^T \mathbf{y}_{i,t-\tau_y'}^{t-1} + \gamma_i, \tag{1}$$

where $\boldsymbol{\alpha}_{j,i} \in \mathbb{R}^{\tau_y}$ is a vector that defines the *synaptic kernel* (SK) applied on the $\{j, i\}$ synapse between presynaptic neuron j and postsynaptic neuron i; $\boldsymbol{\beta}_i \in \mathbb{R}^{\tau_y'}$ is the *feedback kernel* (FK); and γ_i is a bias parameter. Note that τ_y and τ_y' denote the lengths of the SK and FK, respectively. The vector of variable parameters $\boldsymbol{\theta}_i$ includes the bias γ_i and the parameters that define the SK and FK filters, which are discussed below.

According to the GLM, the log-probability of the output spike train $\mathbf{y}_i = [y_{i,1}, ..., y_{i,T}]^T$ conditioned on the input spike trains $\mathbf{x} = \{\mathbf{x}_j\}_{j=1}^{N_X}$ can be written as

$$\log p_{\boldsymbol{\theta}_i}(\mathbf{y}_i | \mathbf{x}) = \sum_{t=1}^{T} [y_{i,t} \log g(u_{i,t}) + \bar{y}_{i,t} \log \bar{g}(u_{i,t})], \tag{2}$$

where $g(\cdot)$ is an activation function, such as the sigmoid function $g(x) = \sigma(x) = 1/(1 + \exp(-x))$, and we defined $\bar{y}_{i,t} = 1 - y_{i,t}$ and $\bar{g}(u_{i,t}) = 1 - g(u_{i,t})$. As per (2), each sample $y_{i,t}$ is Bernoulli distributed with spiking probability $g(u_{i,t})$.

As in [13], we adopt the parameterized model of [17] for the SK and FK filters. Accordingly, we write the SK $\boldsymbol{\alpha}_{j,i}$ and the FK $\boldsymbol{\beta}_i$ as

$$\boldsymbol{\alpha}_{j,i} = \sum_{k=1}^{K_{\boldsymbol{\alpha}}} w_{j,i,k} \mathbf{a}_k = \mathbf{A} \mathbf{w}_{j,i}, \tag{3}$$

and

$$\boldsymbol{\beta}_i = \sum_{k=1}^{K_{\boldsymbol{\beta}}} v_{i,k} \mathbf{b}_k = \mathbf{B} \mathbf{v}_i, \tag{4}$$

respectively, where we have defined the fixed basis matrices $\mathbf{A} = [\mathbf{a}_1, ..., \mathbf{a}_{K_{\boldsymbol{\alpha}}}]$ and $\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_{K_{\boldsymbol{\beta}}}]$ and the vectors $\mathbf{w}_{j,i} = [w_{j,i,1}, ..., w_{j,i,K_{\boldsymbol{\alpha}}}]^T$ and $\mathbf{v}_i = [v_{i,1}, ..., v_{i,K_{\boldsymbol{\beta}}}]^T$; $K_{\boldsymbol{\alpha}}$ and $K_{\boldsymbol{\beta}}$ denote the respective number of basis functions; $\mathbf{a}_k = [a_{k,1}, ..., a_{k,\tau_y}]^T$ and $\mathbf{b}_k = [b_{k,1}, ..., b_{k,\tau_y'}]^T$ are the basis vectors; and $\{w_{j,i,k}\}$ and $\{v_{i,k}\}$ are the learnable weights for the kernels $\boldsymbol{\alpha}_{j,i}$ and $\boldsymbol{\beta}_i$, respectively. For the experiments discussed in Sec. V, we adopt the raised cosine basis functions introduced in [17, Sec. Methods].

Information Decoding: We also consider two alternative decoding methods, namely rate decoding and first-to-spike decoding. *1) Rate decoding*: With rate decoding, decoding is carried out by selecting the output neuron with the largest number of spikes. *2) First-to-spike decoding*: With first-to-spike decoding, the class that corresponds to the neuron that spikes first is selected.

ML training: Conventional ML training is performed differently under rate and first-to-spike decoding methods, as briefly reviewed next.

1) Rate decoding: With rate decoding, the postsynaptic neuron corresponding to the correct label $c \in \{1, ..., N_Y\}$ is assigned a desired output spike train $\mathbf{y}_c$ containing a number of spikes, while an all-zero vector $\mathbf{y}_i$, $i \neq c$, is assigned to the other postsynaptic neurons. Using the ML criterion, one hence maximizes the sum of the log-probabilities (2) of the desired output spikes $\mathbf{y}(c) = \{\mathbf{y}_1, ..., \mathbf{y}_{N_Y}\}$ for the given N_X input spike trains $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_{N_X}\}$. The log-likelihood function for a given training example $(\mathbf{x}, c)$ can be written as

$$L_{\boldsymbol{\theta}}(\mathbf{x}, c) = \sum_{i=1}^{N_Y} \log p_{\boldsymbol{\theta}_i}(\mathbf{y}_i | \mathbf{x}), \tag{5}$$

where the parameter vector $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{V}, \boldsymbol{\gamma}\}$ includes the parameters $\mathbf{W} = \{\mathbf{W}_i\}_{i=1}^{N_Y}$, $\mathbf{V} = \{\mathbf{v}_i\}_{i=1}^{N_Y}$ and $\boldsymbol{\gamma} = \{\gamma_i\}_{i=1}^{N_Y}$.

The sum in (5) is further extended to all examples in the training set. The negative log-likelihood (NLL) $-L_{\boldsymbol{\theta}}$ is convex with respect to $\boldsymbol{\theta}$ and can be minimized via SGD [13].

2) First-to-spike decoding: With first-to-spike decoding, the class that corresponds to the neuron that spikes first is selected. The ML criterion hence maximizes the probability to have the first spike at the output neuron corresponding to the correct label. The logarithm of this probability for a given example $(\mathbf{x}, c)$ can be written as

$$L_{\boldsymbol{\theta}}(\mathbf{x}, c) = \log\left(\sum_{t=1}^{T} p_t(\boldsymbol{\theta})\right), \tag{6}$$

where

$$p_t(\boldsymbol{\theta}) = \prod_{i=1, i\neq c}^{N_Y} \prod_{t'=1}^{t} \bar{g}(u_{i,t'}) g(u_{c,t}) \prod_{t'=1}^{t-1} \bar{g}(u_{c,t'}), \tag{7}$$

is the probability of having the first spike at the correct neuron c at time t. In (7), the potential $u_{i,t}$ for all i is obtained from (1) by setting $y_{i,t} = 0$ for all i and t. The minimization of the log-likelihood function $L_{\boldsymbol{\theta}}$ in (6), which is not concave, can be tackled via SGD as proposed in [13].

III. Designing Adversarial Examples

In this work, we consider white-box attacks based on full knowledge of the model, i.e., of the parameter vector $\boldsymbol{\theta}$, as well as of the encoding and decoding strategies. Accordingly, given an example $(\mathbf{x}, c)$, an adversarial spike train $\mathbf{x}^{\text{adv}}$ is obtained as a perturbed version of the original input $\mathbf{x}$, where the perturbation is selected so as to cause the classifier to be more likely to predict an incorrect label $c' \neq c$, while being sufficiently small.

We consider the following types of perturbations: (i) *Remove attack*: one or more spikes are removed from the input $\mathbf{x}$; (ii) *Add attack*: one or more spikes are added to the input $\mathbf{x}$; and (iii) *Flip attack*: one or more spikes are added or removed. The size of the disturbance is measured for all attacks by the number of spikes that are added and/or removed. Mathematically, this can be expressed as the Hamming distance

$$d_H\left(\mathbf{x}, \mathbf{x}^{\text{adv}}\right) = \sum_{j=1}^{N_X} \sum_{t=1}^{T} 1\left(x_{j,t} \neq x_{j,t}^{\text{adv}}\right), \tag{8}$$

where $1(\cdot)$ is the indicator function, i.e., $1(a) = 1$ if condition a is true and $1(a) = 0$ otherwise.

In order to select the adversarial perturbation of an input $\mathbf{x}$, we consider the maximization of the likelihood of a given incorrect target class $c' \neq c$. According to [18], an effective way to choose the target class c' is to find the class $c^{\text{LL}} \neq c$ that is the least likely under the given model $\boldsymbol{\theta}$. Mathematically, for a given training example $(\mathbf{x}, c)$, the least likely class is obtained by solving the problem

$$c^{\text{LL}} = \underset{c' \neq c}{\operatorname{argmin}} \quad L_{\boldsymbol{\theta}}(\mathbf{x}, c'), \tag{9}$$

where the log-likelihood $L_{\boldsymbol{\theta}}(\mathbf{x}, c')$ is given by (5) for rate decoding and (6) for first-to-spike decoding.

Algorithm 1 Greedy Design $(\boldsymbol{\theta}, T_A, \epsilon)$

Input: $\mathbf{x}$, $\boldsymbol{\theta}$, T_A, ϵ

Compute c^{LL} from (9)
Initialize: $\mathbf{x}^{\text{adv}}(0) \leftarrow \mathbf{x}$
for $i = 1$ **to** $\lfloor \epsilon N_X T \rfloor$ **do**
  $\mathbf{x}^{\text{adv}}(i) \leftarrow \mathbf{x}^{\text{adv}}(i-1) + \mathbf{p}$, where $\mathbf{p}$ is obtained by solving problem (10) with $\mathbf{x}^{\text{adv}}(i-1)$ in lieu of $\mathbf{x}$ and $p_{j,t} = 0$ for all $t > T_A$.
end for

Output: $\mathbf{x}^{\text{adv}}$

Algorithm 2 Adversarial Training (T_A, ϵ_A)

Input: Training set, basis functions $\mathbf{A}$ and $\mathbf{B}$, learning rate η, T_A, and ϵ_A

Initialize: $\boldsymbol{\theta}$

for each iteration **do**
  Choose example $(\mathbf{x}, c)$ from the training set
  Compute $\mathbf{x}^{\text{adv}}$ and c^{LL} from Algorithm 1 with input $\boldsymbol{\theta}$, T_A and ϵ_A
  Update $\boldsymbol{\theta}$: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \nabla_{\boldsymbol{\theta}} L_{\boldsymbol{\theta}}\left(\mathbf{x}^{\text{adv}}, c\right)$
end for

Output: $\boldsymbol{\theta}$

Then, in order to compute the adversarial perturbation $\mathbf{p}$, we maximize the likelihood of class c^{LL} under model $\boldsymbol{\theta}$ by tackling the following optimization problem

$$\begin{aligned} \max_{\mathbf{p} \in \mathcal{C}} \quad & L_{\boldsymbol{\theta}}\left(\mathbf{x} + \mathbf{p}, c^{\text{LL}}\right) \\ \text{s.t.} \quad & \|\mathbf{p}\|_0 \leq \epsilon N_X T, \end{aligned} \tag{10}$$

where $\|\mathbf{p}\|_0$ denotes the number of non-zero elements of $\mathbf{p}$. In (10), the perturbation $\epsilon > 0$ controls the adversary strength. In particular, the adversary is allowed to add or remove spikes from a fraction ϵ of the $N_X T$ input samples, i.e., T samples for each input neuron. The constraint set $\mathcal{C}$ in problem (10) is given by the set of binary perturbations, i.e., $\mathcal{C} = \{0, 1\}^{N_X T}$, for add attacks, since spikes can only be added; $\mathcal{C} = \{0, -1\}^{N_X T}$ for remove attacks; and $\mathcal{C} = \{0, \pm 1\}^{N_X T}$ for flip attacks.

The exact solution of problem (10) requires an exhaustive search over all possible perturbations of $\epsilon N_X T$ samples. In the worst case of flip attacks, the resulting search space is hence exponential in N_X and T. Therefore, here we resort to a greedy search method. As detailed in Algorithm 1, at each of the $\lfloor \epsilon N_X T \rfloor$ steps, the method looks for the best spike to add, remove or flip, depending on the attack type. We further reduce complexity by searching only among the first $T_A \leq T$ samples across all input neurons. As a results, the complexity of each step of Algorithm 1 is at most $N_X T_A$.

IV. Robust Training

In order to increase the robustness of the trained SNN to adversarial examples, in this section, we propose a robust training procedure. Accordingly, in a manner similar to [4], during the SGD-based training phase, each training example

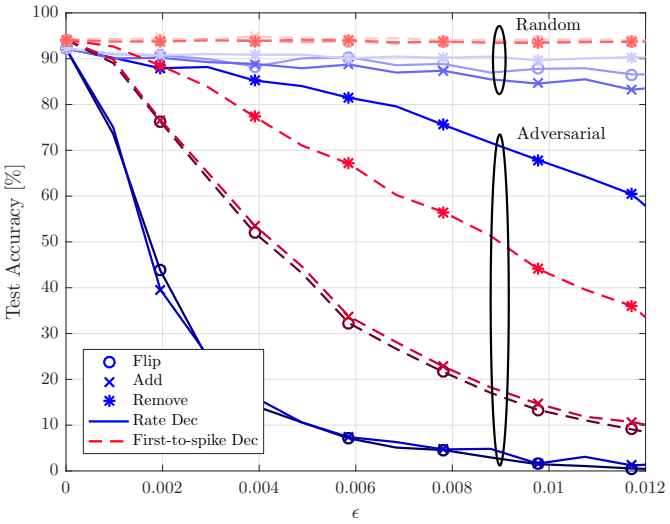

Fig. 2. Test accuracy for ML training under adversarial and random changes versus ϵ with rate encoding for both rate and first-to-spike decoding rules ($T = K = 16$).

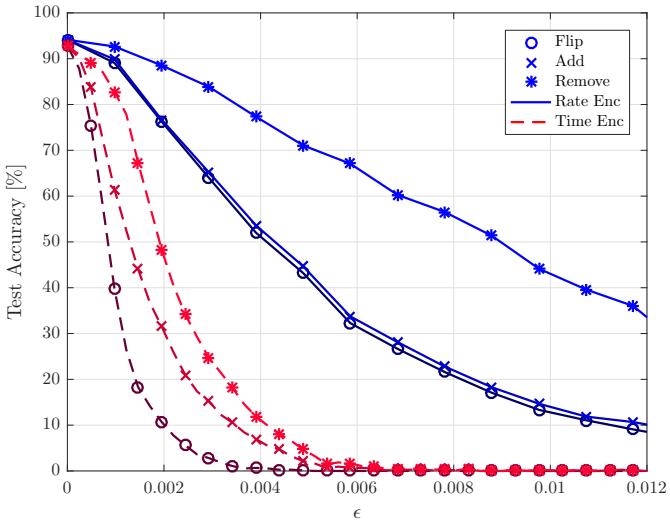

Fig. 3. Test accuracy for ML training under adversarial attacks versus ϵ with both rate and time encoding rules for first-to-spike decoding ($T = K = 16$).

$(\mathbf{x}, c)$ is substituted with the adversarial example $\mathbf{x}^{\text{adv}}$ obtained from Algorithm 1 for the current iterate $\boldsymbol{\theta}$. The training algorithm is detailed in Algorithm 2. Note that, the robust training algorithm is parameterized by T_A and ϵ_A, which determine the parameters of the assumed adversary during training.

V. Numerical Results

In this section, we numerically study the performance of the described probabilistic SNN under the adversarial attacks. We use the standard USPS dataset as the input data. As a result, we have $N_X = 256$, with one input neuron per pixel of the 16×16 images. Unless stated otherwise, we focus solely in the classes $\{1, 5, 7, 9\}$ and we set $T = K = 16$. We assume the worst-case $T_A = T$ for the adversary during the test phase. For rate decoding, we use a desired spike train with one spike after every three zeros. SGD is applied for 200 training epochs and early stopping is used for all schemes. Holdout validation with 20% of training samples is applied to select between 10^{-3} and 10^{-4} for the constant learning rate η. The model parameters $\boldsymbol{\theta}$ are randomly initialized with uniform distribution between -1 and 1.

We first evaluate the sensitivity of different encoding and decoding schemes to adversarial examples obtained as explained in Sec. III. For reference, we consider also perturbations obtained by randomly and uniformly adding, removing and flipping spikes. Fig. 2 illustrates the test accuracy under adversarial and random perturbations when performing standard ML training. The accuracy is plotted versus the adversary's power ϵ assuming rate encoding and both rate and first-to-spike decoding rules. The results highlight the notable difference in performance degradation caused by random perturbations and adversarial attacks. In particular, adversarial changes can cause a significant drop in classification accuracy even with small values of ϵ, particularly when the most powerful flip attacks are used.

First-to-spike decoding is seen to be more resistant to add and flip attacks, while it is more vulnerable than rate decoding to remove spike attacks. The resilience of first-to-spike decoding can be interpreted as a consequence of the fact that the log-likelihood (6), unlike (5) for rate decoding, associates multiple outputs to the correct class, namely all of those with the correct neuron spiking first. Nevertheless, removing properly selected spikes can be more deleterious to first-to-spike decoding as it may prevent spiking by the correct neuron.

The comparison between rate and time encoding in terms of sensitivity to adversarial examples is considered in Fig. 3 under the assumption of first-to-spike decoding. Time encoding is seen to be significantly less resilient than rate encoding. This is due to the fact that time encoding, in the form considered here of intensity-to-latency encoding, which associated a single spike per input neuron [14], can be easily made ineffective by removing selected spikes.

We then evaluate the impact of robust adversarial training as compared to standard ML. To this end, in Fig. 4, we plot the test accuracy for the case of flip and remove attacks for both ML and adversarial training when $T = K = 8$. Here we also focus solely on the two classes $\{5, 7\}$. We recall that the adversarial training scheme is parametrized by the time support T_A of the attacks considered during training, here $T_A = 8$, and by its power ϵ_A, here $\epsilon_A = 5/2048$ and $\epsilon_A = 10/2048$. It is observed that robust training can significantly improve the robustness of the SNN classifier, even when ϵ_A is not equal to the value ϵ used by the attacker during the test phase. Furthermore, increasing ϵ_A enhances the robustness of the trained SNN at the cost of a higher computational complexity. For instance, for an attacker in the test phase

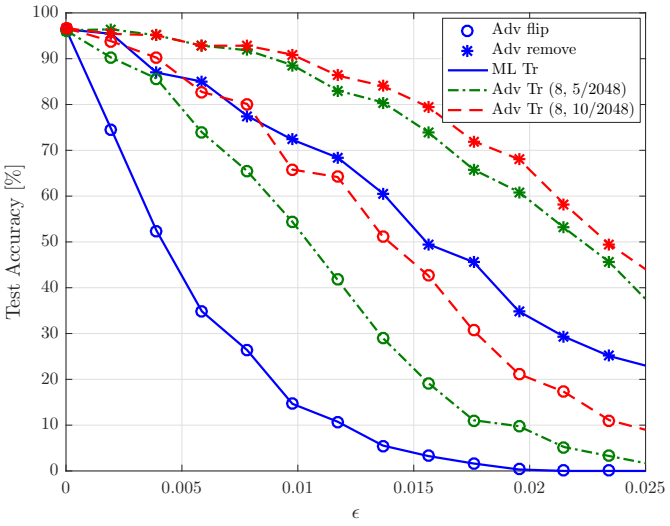

Fig. 4. Test accuracy under adversarial attacks versus ϵ with rate encoding and rate decoding with ML and adversarial training ($T = K = 8$).

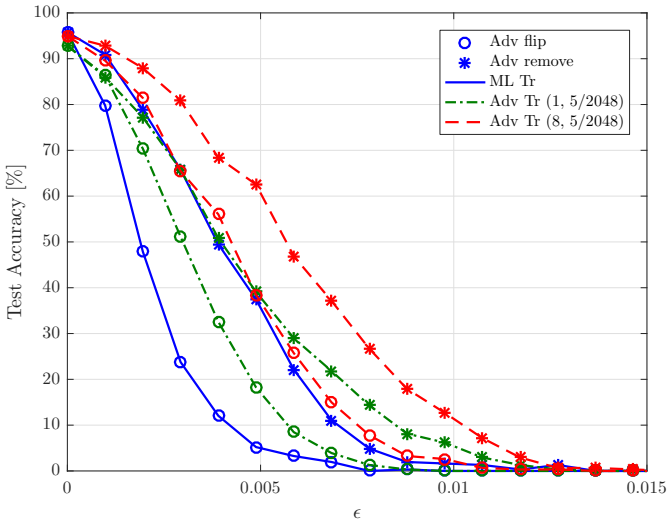

Fig. 5. Test accuracy under adversarial attacks versus ϵ with time encoding and rate decoding with ML and adversarial training ($T = K = 8$).

with $\epsilon = 10/2048$, i.e., with 10 bit flips, conventional ML achieves an accuracy of 45%, while adversarial training with $\epsilon_A = 10/2048$ (i.e., 10 bit flips) achieves an accuracy of 87%. Finally, the results show that the classifier remains resilient against other type of attacks, despite being trained assuming the flip attack.

Finally, under the same conditions as in Fig. 5, we study the effect of limiting the power of the adversary assumed during training by considering $T_A = 1$ and $T_A = 8$ with the same $\epsilon_A = 5/2048$. We assume time encoding and rate decoding. It is observed that robust training can still improve the robustness of the SNN classifier, even when $T_A \ll T$ during training. For instance, for an attacker in the test phase with $\epsilon = 5/2048$, i.e., 5 bit flips, conventional ML achieves an accuracy of 34.2%, while adversarial training with $\epsilon_A = 5/2048$ and $T_A = 1$ and 8 achieves accuracy levels of 60.3% and 77.5%, respectively.

VI. Conclusions

In this paper, we have studied for the first time the sensitivity of a probabilistic two-layer SNN under adversarial perturbations. We considered rate and time encoding, as well as rate and first-to-spike decoding. We have proposed mechanisms to build adversarial examples, as well as a robust training method that increases the resilience of the SNN. Additional work is needed in order to generalize the results to multi-layer networks.

VII. Acknowledgment

This work was supported by the U.S. NSF under grant ECCS #1710009. O. Simeone has also received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program (grant agreement #725731).

References

[1] R. Ranjan, S. Sankaranarayanan, A. Bansal, N. Bodla, J.-C. Chen, V. M. Patel, C. D. Castillo, and R. Chellappa, "Deep learning for understanding faces: Machines may be just as good, or better, than humans," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 66–83, 2018.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Int. Conf. on Learn. Repr. (ICLR)*, 2015.

[3] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard, "The robustness of deep networks: A geometrical perspective," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 50–62, 2017.

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[5] H. Paugam-Moisy and S. Bohte, "Computing with spiking neuron networks," *Handbook of natural computing*, pp. 335–376, 2012.

[6] J. Vincent, "Intel investigates chips designed like your brain to turn the AI tide," *https://www.theverge.com/2017/9/26/16365390/intel-investigates-chips-designed-like-your-brain-to-turn-the-ai-tide*, Accessed: Sept. 26, 2017.

[7] J. E. Smith, "Research agenda: Spacetime computation and the neocortex," *IEEE Micro*, vol. 37, no. 1, pp. 8–14, 2017.

[8] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting," *Neural Comput*, vol. 22, no. 2, pp. 467–510, 2010.

[9] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *arXiv preprint arXiv:1802.02627*, 2018.

[10] B. Gardner and A. Grüning, "Supervised learning in spiking neural networks for precise temporal encoding," *PloS one*, vol. 11, no. 8, pp. 1–28, 2016.

[11] S. Guo, Z. Yu, F. Deng, X. Hu, and F. Chen, "Hierarchical bayesian inference and learning in spiking neural networks," *IEEE Trans. Cybern.*, vol. PP, no. 99, pp. 1–13, 2017.

[12] D. J. Rezende, D. Wierstra, and W. Gerstner, "Variational learning for recurrent spiking networks," *Adv Neural Inf Process Syst*, pp. 136–144, 2011.

[13] A. Bagheri, O. Simeone, and B. Rajendran, "Training probabilistic spiking neural networks with first-to-spike decoding," *arXiv preprint arXiv:1710.10704*, 2017.

[14] E. Stromatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco, "An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data," *Front Neurosci*, vol. 11, pp. 1–17, 2017.

[15] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS Comput. Biol.*, vol. 3, no. 2, pp. 247–257, 2007.
[16] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep neural networks for object recognition," *arXiv preprint arXiv:1611.01421*, 2016.
[17] J. W. Pillow, J. Shlens, L. Paninski, A. Sher, A. M. Litke, E. Chichilnisky, and E. P. Simoncelli, "Spatio-temporal correlations and visual signaling in a complete neuronal population," *Nature*, vol. 454, no. 7207, p. 995, 2008.
[18] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.