Is Spiking Secure? A Comparative Study on the Security Vulnerabilities of Spiking and Deep Neural Networks

Alberto Marchisio[1], Giorgio Nanfa[1,2], Faiq Khalid[1], Muhammad Abdullah Hanif[1],
Maurizio Martina[2], Muhammad Shafique[1]

[1]*Technische Universität Wien, Vienna, Austria*
[2]*Politecnico di Torino, Turin, Italy*

Email: {alberto.marchisio, faiq.khalid, muhammad.hanif, muhammad.shafique}@tuwien.ac.at
giorgio.nanfa@studenti.polito.it, maurizio.martina@polito.it

Abstract—**Spiking Neural Networks (SNNs) claim to present many advantages in terms of biological plausibility and energy efficiency compared to standard Deep Neural Networks (DNNs). Recent works have shown that DNNs are vulnerable to adversarial attacks, i.e., small perturbations added to the input data can lead to targeted or random misclassifications. In this paper, we aim at investigating the key research question: "Are SNNs secure?" Towards this, we perform a comparative study of the security vulnerabilities in SNNs and DNNs w.r.t. the adversarial noise. Afterwards, we propose a novel black-box attack methodology, i.e., without the knowledge of the internal structure of the SNN, which employs a greedy heuristic to automatically generate imperceptible and robust adversarial examples (i.e., attack images) for the given SNN. We perform an in-depth evaluation for a Spiking Deep Belief Network (SDBN) and a DNN having the same number of layers and neurons (to obtain a fair comparison), in order to study the efficiency of our methodology and to understand the differences between SNNs and DNNs w.r.t. the adversarial examples. Our work opens new avenues of research towards the robustness of the SNNs, considering their similarities to the human brain's functionality.**

Index Terms—**Machine Learning, Neural Networks, Spiking Neural Networks, Security, Adversarial Examples, Attack, Vulnerability, Resilience, SNN, DNN, Deep Neural Network.**

I. Introduction

Spiking Neural Networks (SNNs), the third generation neural network models [15], are rapidly emerging as another design option compared to Deep Neural Networks (DNNs), due to their inherent model structure and properties matching the closest to today's understanding of a brain's functionality. As a result, SNNs have the following key properties.

- **Biologically Plausible**: spiking neurons are very similar to the biological ones because they use discrete spikes to compute and transmit information. For this reason, SNNs are also highly sensitive to the temporal characteristics of the processed data [5][32].
- **Computationally more Powerful than several other NN Models:** a lower number of neurons is required to realize/model the same computational functionality [8].
- **High Energy Efficiency:** spiking neurons process the information only when a new spike arrives. Therefore, they have relatively lower energy consumption compared to complex DNNs, because the spike events are sparse in time [3][21][30]. Such a property makes the SNNs particularly suited for deep learning-based systems where the computations need to be performed at the edge, i.e., in a scenario with limited hardware resources [17].

SNNs have primarily been used for tasks like real-data classification, biomedical applications, odor recognition, navigation and analysis of an environment, speech and image recognition [13][25]. Recently, the work of Fatahi et al. [4] proposed to convert every pixel of the images into spike trains (i.e., the sequences of spikes) according to its intensity. Since SNNs represent a fundamental step towards the idea of creating an architecture as similar as possible to the current understanding of the structure of a human brain, *it is fundamental to study their security vulnerability w.r.t. adversarial attacks. In this paper, we demonstrate that indeed, even a small adversarial perturbation of the input images can modify the spike propagation and increase the probability of the SNN misprediction (i.e., the image is misclassified).*

Adversarial Attacks on DNNs: In recent years, many methods to generate adversarial attacks for DNNs and their respective defense techniques have been proposed [7][12][16]. A minimal and imperceptible modification of the input data can cause a classifier misprediction, which can potentially produce a wrong output with high probability. This scenario may lead to serious consequences in safety-critical applications (e.g., automotive, medical, UAVs and banking) where even a single misclassification can have catastrophic consequences [34].

In the image recognition field, having a wide variety of possible real-world input images [12], with diverse pixel intensity patterns, the classifier cannot recognize if the source of the misclassification is the attacker or other factors [26].

Given an input image x, the goal of an adversarial attack $x^* = x + \delta$ is to apply a small perturbation δ such that the predicted class $C(x)$ is different from the target one $C(x^*)$, i.e., the class in which the attacker wants to classify the example. Inputs can also be misclassified without specifying the target class. This is the case of untargeted attacks, where the target class is not defined a-priori by the attacker. Targeted attacks can be more difficult to apply than the untargeted ones, but they can be relatively more effective in several cases [33]. Another important classification of adversarial attacks is based on the knowledge of the network under attack, as discussed below.

- *White-box attack:* an attacker has the complete access and knowledge of the architecture, the network parameters, the training data and the existence of a possible defense.
- *Black-box attack:* an attacker does not know the architecture, the network parameters, the training data and a possible defense, but it can only access to the input and output of the network (which is treated as a black-box), and may be the testing dataset [24].

Our Approach towards Adversarial Attacks on SNNs: *In this paper, we aim at generating, for the first time, imperceptible and robust adversarial examples for SNNs under the black-box settings.* Bagheri et al. [1] studied the vulnerabilities of SNNs under white-box assumptions, while we consider a black-box scenario, which makes the attacker stronger under a wide range of real-world scenarios. For the evaluation, we apply these attacks to a Spiking Deep Belief Network (SDBN) and a DNN having the same number of layers and neurons, to obtain a fair comparison. As per our knowledge[1], this kind of black-box attack was previously applied **only** to a DNN model [14]. This method is efficient for DNNs because it is able to generate adversarial noise which is imperceptible to the human eye.

As shown in Figure 1, we investigate the vulnerability of SDBNs to random noise and adversarial attacks, aiming at identifying the similarities / differences w.r.t. DNNs. Our experiments show that, when applying a random noise to a given SDBN, its classification accuracy decreases, by increasing the noise magnitude. Moreover, when applying our attack to SDBNs, we observe that, in contrast to the case of DNNs, the output probabilities follow a different behavior, i.e., while the adversarial image remains imperceptible, the misclassification is not always guaranteed.

In short, we make the following Novel Contributions:

1) We analyze the variation in the accuracy of a Spiking Deep Belief Network (SDBN) when a random noise is added to the input images. (**Section III**)
2) We evaluate the improved generalization capabilities of the SDBN when adding a random noise to the training images. (**Section III-C**)

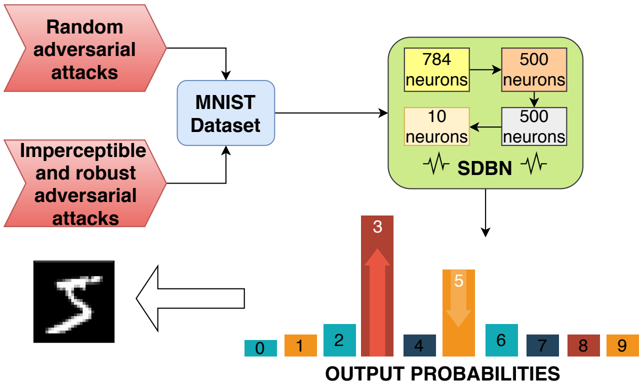


Fig. 1: Overview of our proposed approach.

3) We develop a methodology to automatically create imperceptible adversarial examples for SNNs. (**Section IV**)
4) We apply our methodology to an SDBN *(it is the first attack of this type applied to SDBNs)* and a DNN for generating adversarial examples, and evaluate their imperceptibility and robustness. (**Section V**)

Before proceeding to the technical sections, in **Section II**, we briefly discuss the background and the related work, focusing on SDBNs and adversarial attacks on DNNs.

II. Background and Related Work

A. *Spiking Deep Belief Networks*

Deep Belief Networks (DBNs) [2] are multi-layer networks that are widely used for classification problems and have been implemented in many areas such as visual processing, audio processing, images and text recognition [2]. DBNs are implemented by stacking pre-trained Restricted Boltzmann Machines (RBMs), energy-based models consisting in two layers of neurons, one hidden and one visible, symmetrically and fully connected, i.e., without connections between the neurons inside the same layer (this is the main difference w.r.t. the standard Boltzmann machines). RBMs are typically trained with unsupervised learning, to extract the information saved in the hidden units, and then a supervised training is performed to train a classifier based on these features [9].

Spiking DBNs (SDBNs) improve the energy efficiency and computation speed, as compared to DBNs. Such a behavior has already been observed by O'Connor et al. [23]. That work proposed a DBN model composed of 4 RBMs of 784-500-500-10 neurons, respectively. It has been trained offline and transformed in an event-based domain to increase the processing efficiency and the computational power. The RBMs are trained with the Persistent Contrastive Divergence (CD) algorithm, an unsupervised learning rule using Gibbs sampling, a Markov-Chain Monte-Carlo algorithm, with optimizations for fast weights, selectivity and sparsity [6][20][31]. Once every RBM is trained, the information is stored in the hidden units to

[1] A previous version of this work is available in [19].

use it as an input for the visible units of the following layer. Afterwards, a supervised learning algorithm [10], based on the features coming from the unsupervised training, is performed. The RBMs of this model use the *Siegert* function [28] in their neurons. It allows to have a good approximation of firing rate of Leaky Integrate and Fire (LIF) neurons [5], used for CD training. Hence, the neurons of an SDBN generate Poisson spike trains, according to the *Siegert* formula.

This represents a great advantage in terms of power consumption and speed, as compared to the classical DBNs, which are based on a discrete-time model [23]. *Since there has been no prior work on studying the security vulnerabilities of SNNs / SDBNs, we aim at investigating these aspects in a black-box setting, which is important for their real-world applications in security/safety-critical systems.*

B. Adversarial Attacks for DNNs

The robustness and self-healing properties of DNNs have been thoroughly investigated in the recent researches [27]. As demonstrated for the first time by Szegedy et al. [29], adversarial attacks can misclassify an image by changing its pixels with small perturbations. Kurakin et al. [12] defined adversarial examples as *a sample of input data which has been modified very slightly in a way that is intended to cause a machine learning classifier to misclassify it*. Luo et al. [14] proposed a method to generate attacks by maximizing their noise tolerance and taking into account the human perceptual system in their distance metric. A similar attack is able to mislead even more complex DNNs, like Capsule Networks [18], which are notoriously more robust against adversarial attacks. This methodology has strongly inspired our algorithm. The human eyes are more sensitive to the modifications of the pixels in low variance areas. Hence, to maintain the imperceptibility as much as possible, the modification of pixels in only the high variance areas is preferable.

Moreover, a robust attack aims at increasing *its ability to stay misclassified to the target class after the transformations due to the physical world*. For example, considering a crafted sample, after an image compression or a resizing, its output probabilities can change according to the types of the applied transformations. Therefore, the attack can be ineffective if it is not robust enough to those variations.

Motivated by the above-discussed considerations, *we propose an algorithm to automatically generate imperceptible and robust adversarial examples for SNNs, and study their differences w.r.t. the adversarial examples generated for DNNs using the same technique.*

III. Analysis: Applying Random Noise to SDBNs

A. Experimental Setup

For a case study, we consider an SDBN [23] composed of four fully-connected layers of 784-500-500-10 neurons, respectively. We implement this SDBN in Matlab, for analyzing the MNIST database, a collection of 28×28 gray scale images of handwritten digits, divided into 60,000 training images and 10,000 test images. Each pixel intensity is encoded as a value between 0 and 255. To maximize the spike firing, the input data are scaled to the range [0,0.2], before converting them into spikes. In our experiments, the pixel intensities are represented as the probability that a spike occurs.

B. Understanding the Impact of Random Noise Addition to Inputs on the Accuracy of an SDBN

We test the accuracy of the SDBN for different noise magnitudes, applied to three different combinations of images:

- to all the training images only.
- to all the test images only.
- to both the training and test images.

To test the vulnerability of the SDBN, we apply two different types of noises: *normally-distributed* and *uniformly-distributed* random noise.

The results of our experiments are shown in Table I and Figure 2. The initial "clean-case" accuracy, obtained without applying noise, is 96.2%. When the noise is applied to the test images, the accuracy of the SDBN decreases accordingly with an increase in the noise magnitude, more evidently in the case of the normally-distributed random noise. This behavior is due to the fact that the standard normal distribution contains a wider range of values, compared to the uniform distribution. For both noise distributions, the accuracy decreases more when the noise magnitude applied is around 0.15 (see the red-colored values in Table I).

TABLE I: Evaluation of the SDBN accuracy applying two different types of random noise with different values of noise magnitude. The red and blue values are helping the reader to identify the accuracy results that are discussed in the text. (ACC stands for Accuracy, TR+TST stands for Training and Test Datasets)

ACC	TRAIN	TEST	TR+TST	TRAIN	TEST	TR+TST
δ	NORMALLY			UNIFORMLY		
0.02	**96.65**	94.73	96.54	**96.8**	96.02	96.81
0.05	95.19	94.42	94.99	96.7	95.64	96.72
0.08	92.99	82.73	73.64	95.89	94.64	95.56
0.1	76.01	77.07	10.39	94.34	93.36	92.8
0.15	24.61	**48.23**	10.32	47.03	**82.76**	10.51
0.2	10.26	33.34	10.05	14.64	60.79	10.16
0.3	10.31	21.52	9.88	9.59	34.9	10.16
0.4	10.27	17.05	10.34	9.98	23.16	10.03

When the noise is applied to the training images, the accuracy of the SDBN does not decrease as much as in the previous case, as long as the noise magnitude (δ) is lower than 0.1. On the contrary, for $\delta = 0.02$, the accuracy increases (see the blue-colored values in Table I) w.r.t. the baseline (i.e., without noise). Indeed, adding noise in training samples improves the generalization capabilities of the neural network. Hence, its capability to correctly classify new unseen

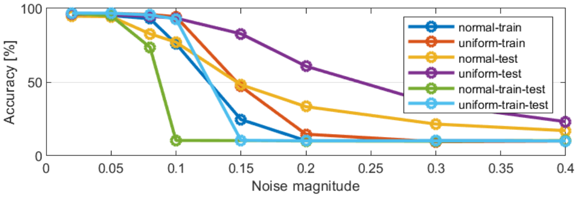


Fig. 2: Normal and uniform random noise applied to all the pixels of the MNIST dataset.

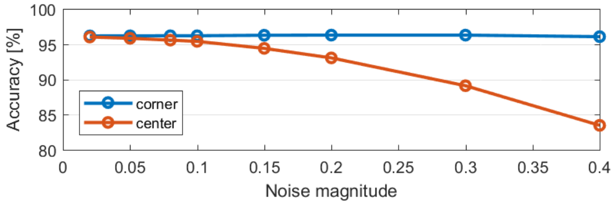


Fig. 3: Normal random noise applied to some pixels of the MNIST test images.

samples also increases. This observation, as was analyzed in several other scenarios for Deep Neural Networks with back-propagation training [11], is also valid for our SDBN model. However, if the noise is equal to or greater than 0.1, the accuracy drops significantly. This behavior means that the SDBN is unable to learn input features due to the inserted noise, thus it is unable to correctly classify the inputs.

When the noise is applied to both the training and test images, we notice that the behavior observed for the case of noise applied to only the training images is accentuated. For low noise magnitudes (mostly in the uniform noise case), the accuracy is similar or higher than the baseline. For noise magnitudes greater than 0.1 (more precisely, 0.08 for the case of normal noise applied), the accuracy decreases more sharply than in the case of noise applied only to the training images. Such a value of noise magnitude represents a threshold of tolerable noise for the SDBN. Hence, when the noise is too high, the network cannot classify well.

C. Applying Noise to a Restricted Window of Pixels

In this analysis, we add a normally distributed random noise to a restricted window of pixels of the test images. Considering a rectangle of 4×5 pixels, we analyze two scenarios:

- The noise is applied to 20 pixels at the top-left corner of the image. The variation of the accuracy is represented by the blue-colored line of Figure 3. As expected, the accuracy remains almost constant, because the noise affects irrelevant pixels. The resulting image, when the noise is equal to 0.3, is shown in Figure 4b.
- The noise is applied to 20 pixels in the middle of the image, with coordinates $(x, y) = ([14\ \ 17], [10\ \ 14])$. The accuracy descreases more significantly (orange-colored line of Figure 3), as compared to the previous case, because some white pixels representing the handwritten digits (and therefore the important ones for the classification) are affected by the noise. The resulting image, when the noise is equal to 0.3, is shown in Figure 4c. This analysis shows that the location of noise insertion impacts the accuracy, thereby unleashing a potential vulnerability of SNNs that can be exploited by the adversarial attacks.

D. Key Observations from our Analyses

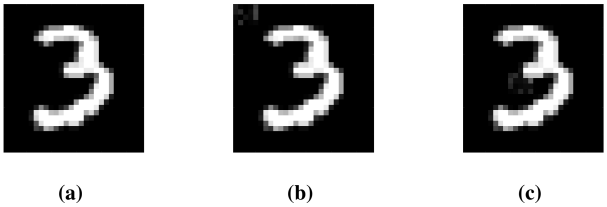


Fig. 4: Comparison between images with normally distributed random noise (with magnitude 0.3) applied to the corner and to the left center of the image. (a) Without noise. (b) Noise applied to the top-left corner. (c) Noise applied to the center.

From the analyses performed in the above Sections III-B and III-C, we derive the following key observations that can be exploited by an adversarial example generation methodology.

- The normal noise is more powerful than the uniform counterpart, since the accuracy decreases more sharply.
- For a low noise magnitude applied to the training images, we notice a small accuracy improvement, due to the improved generalization capability of SDBNs.
- When applying the noise to a restricted window of pixels, the perturbation is more effective if the window is in the center of the image (or generally speaking, in the input regions belonging to the features that are key for the correct classification), as compared to the corner. This is due to the fact that the noise is applied to the pixels which play an important role for accurate feature learning and consequently for the correct classification.

IV. Our Novel Methodology to Generate Imperceptible and Robust Adversarial Examples

Similar to the case of DNNs, the scope of a good attack on SNNs is also to generate adversarial images, which are difficult to be detected by human eyes (i.e., imperceptible) and resistant to physical transformations (i.e., robust). Therefore, for better understanding, we first discuss these two aspects.

A. Imperceptibility of Adversarial Examples

Creating an imperceptible example means to add perturbations to the pixels, while making sure that humans do not notice them. We consider an area A=N·N of pixels x, and we compute the standard deviation (SD) of a pixel $x_{i,j}$ as in Equation (1).

$$SD(x_{i,j}) = \sqrt{\frac{\sum_{k=1}^{N}\sum_{l=1}^{N}(x_{k,l}-\mu)^2 - (x_{i,j}-\mu)^2}{N \cdot N}} \quad (1)$$

Here, μ is the average value of pixels belonging to the N·N area. If a pixel has a high standard deviation, it means that a perturbation added to this pixel is more likely to be hardly detected by the human eye, compared to a pixel with a low standard deviation. The sum of all the perturbations δ added to the pixels of the area A allows to compute the distance ($D(X^*, X)$) between the adversarial example X^* and the original one X. Its formula is shown in Equation (2).

$$D(X^*, X) = \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\delta_{i,j}}{SD(x_{i,j})} \quad (2)$$

Such value can be used to monitor the imperceptibility. Indeed, the distance $D(X^*, X)$ indicates how much perturbation is added to the pixels in the area A. Hence, the maximum perturbation tolerated by the human eye can be associated to a certain value of the distance, D_{MAX}. The value of D_{MAX} can vary among different datasets or images, because it depends on the resolution and the contrast between neighboring pixels.

B. Robustness of adversarial examples

Many adversarial attack methods used to maximize the probability of target class to ease the classifier misclassification of the image. The main problem of these methods is that they do not account for the relative difference between the class probabilities, i.e., the gap, defined in Equation (3).

$$Gap(X^*) = P(target\ class) - max\{P(other\ classes)\} \quad (3)$$

Therefore, after an image transformation, a minimal modification of the probabilities can make the attack ineffective. To improve the robustness, it is desirable to increase the difference between the probability of the target class and the highest probability of the other classes, i.e., to maximize the gap function.

C. How to Automatically Generate Attacks for SNNs?

Obtaining both the imperceptibility and robustness at the same time is complicated. Typically, a robust attack would require perceptible changes of the input, while an imperceptible attack does not change the classification much. We designed a heuristic algorithm to automatically generate imperceptible yet robust adversarial examples for SNNs. Our technique is also applicable to DNNs, as we will demonstrate in the result section. Note that, leveraging the same methodology to generate adversarial examples for both SNNs and DNNs enables a fair comparison. Our algorithm is based on the black-box model, i.e., the attacks are performed on some pixels of the image, without having insights of the network. Given the maximum allowed distance D_{MAX} such that human eyes cannot detect perturbations, the problem can be expressed as in Equation (4).

$$\arg\max_{X^*} Gap(X^*) \ | \ D(X^*, X) \leq D_{MAX} \quad (4)$$

In summary, *the purpose of our iterative algorithm is to perturb a set of pixels, to maximize the gap function, thus making the attack robust, while keeping the distance between the samples below the desired threshold, in order to remain imperceptible.*

Based on the key observations of our analysis in Section III-D, our iterative methodology (see Algorithm 1) perturbs only a window of pixels of the image. We choose a certain value N, which corresponds to an area of N·N pixels, performing the attack on a subset M of pixels.

Algorithm 1 : Methodology for Generating Adversarial Examples for SNNs and DNNs

```
Given: network (SNN or DNN), original sample X, maximum human perceptual distance $D_{max}$, noise magnitude $\delta$, area of A pixels, target class, M
while $D(X^*, X) < D_{MAX}$ do
    -Compute Standard Deviation SD for every pixel of A
    -Compute $Gap(X^*)$, $Gap^-(X^*)$, $Gap^+(X^*)$
    if $Gap(X^*)^- > Gap(X^*)^+$ then
        $VariationPriority(x_{i,j}) =$
        $[Gap^-(X^*) - Gap(X^*)] \cdot SD(x_{i,j})$
    else
        $VariationPriority(x_{i,j}) =$
        $[Gap^+(X^*) - Gap(X^*)] \cdot SD(x_{i,j})$
    end if
    -Sort in descending order $VariationPriority$
    -Select M pixels with highest $VariationPriority$
    if $Gap(X^*)^- > Gap(X^*)^+$ then
        Subtract noise with magnitude $\delta$ from the pixel
    else
        Add noise with magnitude $\delta$ to the pixel
    end if
    -Compute $D(X^*, X)$
    -Update the original example with the adversarial one
end while
```

After computing the standard deviation for the selected N·N pixels, we compute the gap function, i.e., the difference between the probability of the target class and the highest probability between the other classes. Then, the algorithm decides whether to apply a positive or a negative noise to the pixels. Therefore, we compute two parameters for each pixel, $Gap^+(X^*)$ and $Gap^-(X^*)$. $Gap^+(X^*)$ is the value of the gap function computed by adding a perturbation unit to the single pixel, while $Gap^-(X^*)$ is its counterpart, computed subtracting a perturbation unit. According to the

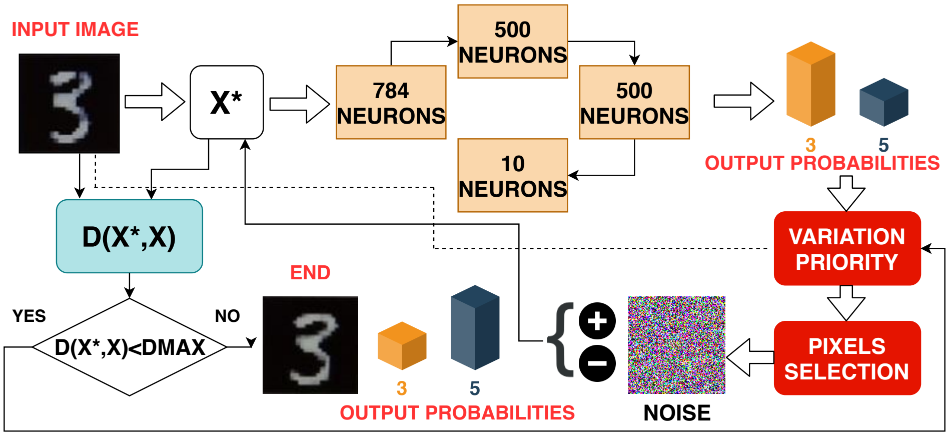


Fig. 5: Our methodology for generating adversarial examples, illustrating with the example of the considered networks.

difference between these values and the gap function, and considering also the standard deviation, we compute the variation priority, a function that indicates the effectiveness of the pixel perturbation. For example, if $Gap^{-}(X^{*})$ is greater than $Gap^{+}(X^{*})$, it means that, for the pixel under consideration, subtracting the noise will be more effective than adding it to the pixel, since the difference between $P(target\ class)$ and $max[P(other\ classes)]$ will increase more. Once computed the vector *VariationPriority*, its values are sorted, and the highest M values are perturbed. Note, according to the previous considerations, the noise is added to, or subtracted from, the selected M pixels depending on the highest value between $Gap^{+}(X^{*})$ and $Gap^{-}(X^{*})$. The algorithm starts the next iteration by replacing the original input image with the created adversarial one. The iterations terminate when the distance between original and adversarial examples overcomes the maximum perceptual distance. Figure 5 summarizes the operational flow of our methodology, applied to the SDBN, for generating adversarial examples.

V. Evaluating our Attack Methodology on SDBNs and DNNs

A. Experimental Setup

Using the methodology of Section IV-C, we attack two different networks: the same SDBN as the one analyzed in Section III and a DNN. To achieve a fair comparison, we design the DNN for our experiments having the same set of parameters as the SDBN, i.e., composed of four fully-connected layers of 784-500-500-10 neurons, respectively. The DNN is trained with the scaled conjugate gradient backpropagation algorithm [22], and after training, its achieved classification accuracy on the MNIST dataset is 97.13%.

For discussion, we start with a test sample, labeled as "five" (see Figure 6). It is classified correctly by both networks, but with different output probabilities. We use a value of δ equal to the 10% of the pixel intensity scale range and a D_{MAX} equal to 22 to compare the attacks. We distinguish two cases, having different search window sizes:

(I) Figure 6a: N=5 and M=10. Based on the analysis in Section III, we define the search window in a central area of the image, as shown by the red square, which is affected by high variation.

(II) Figure 6b: N=7 and M=10. It can be interesting to observe the difference w.r.t. the case I: in this situation we perturb the same amount M of pixels, selected from a search window which contains 24 more pixels.

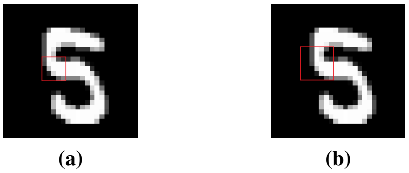


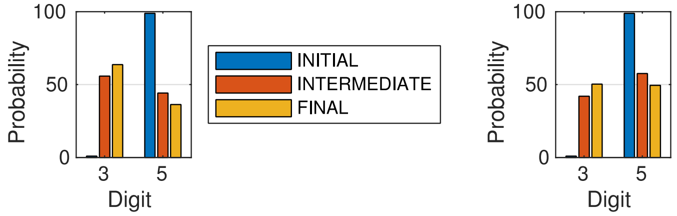
Fig. 6: Selected area of pixels to attack

B. DNN Under Attack

The baseline DNN classifies our test sample as a "five" with its associated probability equal to 98.79%, as shown in the blue-colored bars of Figure 7. The selected target class is "three" for both the cases. The classification results of their respective adversarial images are shown in Figure 7 for both the cases. From the results in Table II, we can observe that, having a small search window leads to obtaining a more robust attack, as compared to larger search windows. The generated adversarial examples are shown in Figure 8.

C. SDBN Under Attack

Our baseline SDBN, without attack, classifies our test sample as a "five" with a probability equal to 82.69%. The



(a) (b)

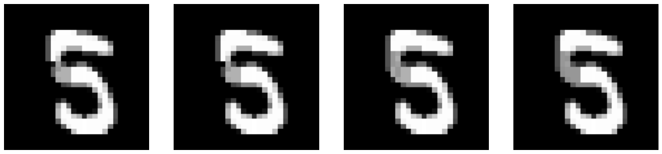
Fig. 7: Output probabilities (% format) of the DNN. (a) Attack using the search window of case I. (b) Attack using the search window of case II.

TABLE II: Results of our simulations for the DNN.

(Case I) After 14 iterations, the probability of the target class has overcome the one of the initial class. Figure 8a shows the sample at this stage (denoted as *intermediate* in Figure 7a). In the following iteration, the gap between the two classes increases, thus increasing the robustness, but also increasing the distance. The sample at this point (denoted as *final* in Figure 7a) corresponds to the attack output, since at the iteration 16 the distance falls above the threshold.

(Case II) After 11 iterations (denoted as *final* in Figure 7b), the sample (in Figure 8d) is classified as a "three". Since at the iteration 12 the distance is already higher than D_{MAX}, Figure 8c shows the sample at the 10^{th}, whose output probabilities are denoted as *intermediate* in Figure 7b.

CASE	ITER	P MAX CLASS	P TARGET CLASS	DISTANCE
I	0	**98.79**	**0.89**	0
I	14	**44.16**	**55.74**	20.18
I	15	**36.25**	**63.67**	21.77
II	0	**98.79**	**0.89**	0
II	10	**57.53**	**42.01**	16.29
II	11	**49.45**	**50.32**	21.19



(a) **(b)** **(c)** **(d)**

Fig. 8: Adversarial samples applied to the DNN. (a) 14^{th} iteration of case I. (b) 15^{th} iteration of case I. (c) 10^{th} iteration of case II. (d) 11^{th} iteration of case II.

complete set of initial "clean-case" output probabilities is shown in Figure 9. We select the "three" as the target class.

The results in Table III show that, in contrast to the attack applied to the DNN, for the case I:

- The SDBN output probabilities do not change monotonically when increasing the iterations of our algorithm.
- At the 20^{th} iteration, the SDBN classifies the target class with a probability of 31.08%, while $D(X^{*},X) = 7.79$.
- At the other iterations, before and after iteration 20, the output probability of classifying the image as the original class still dominates.

Meanwhile, for the case II, we observe that:

- At the 9^{th} iteration, the SDBN misclassifies the image. The probability of classifying the image as a "three" is 50.60%, with a distance $D(X^{*},X) = 10.91$. As a side note, the probability of classifying the image as an "eight" is 49.40%.
- At the other iterations, before and after the iteration 7, the output probability of classifying the image as a "five" is higher than 50%.

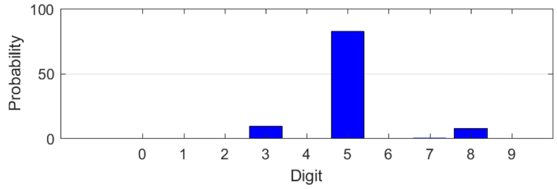


Fig. 9: Output probabilities of the SDBN for the original sample.

TABLE III: Results of our simulations for the SDBN.

CASE	ITER	P MAX CLASS	P TARGET CLASS	DISTANCE
I	0	**82.69**	**7.64**	0
I	20	**60.29**	**31.08**	7.79
I	21	**66.21**	**11.80**	8.15
II	0	**82.69**	**7.64**	0
II	9	**0**	**50.60**	10.91
II	10	**64.94**	**12.03**	11.76

D. *Comparative Discussion between SDBN and DNN*

We can observe how the DNN is vulnerable to the attacks generated by our algorithm, while the SDBN shows a very different response to the attack. The output probabilities of the SDBN do not follow the expected trend, but may sporadically lead to a misclassification if other conditions are satisfied as well. Each pixel of the image is converted to a spike train, thus a slight modification of the pixel intensity can have unexpected consequences, like a wrong feature detection. The SNN sensitivity of the targeted attack is clearly different from the DNN sensitivity for the similar case. Such a difference of robustness should be studied more carefully in future works.

VI. Conclusions

In this work, we studied the security vulnerabilities of SNNs, and compared them to DNNs under our attack methodology. However, there is still a long road for research to follow for analyzing and building robust/secure SNNs. Towards the conclusion of this work, we raise several new research questions like: "What is hidden inside the SNNs that makes them more robust to targeted attacks, as compared to DNNs?" "Can certain specific properties of human brain's functionality be leveraged to build robust and self-healing machine learning algorithms?" An extensive in-depth study of SNNs w.r.t. different security threats is crucial before adopting SNNs in safety-critical applications.

Acknowledgments

This work has been partially supported by the Doctoral College Resilient Embedded Systems which is run jointly by TU Wien's Faculty of Informatics and FH-Technikum Wien.

References

[1] A. Bagheri, O. Simeone, and B. Rajendran. Adversarial training for probabilistic spiking neural networks. In *SPAWC*, 2018.

[2] Y. Bengio et al. Greedy layer-wise training of deep networks. In *NIPS*. 2007.

[3] M. Davies et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[4] M. Fatahi et al. evt_mnist: A spike based version of traditional MNIST. *CoRR*, abs/1604.06751, 2016.

[5] W. Gerstner and W. Kistler. *Spiking Neuron Models: An Introduction*. Cambridge University Press, 2002.

[6] H. Goh, N. Thome, and M. Cord. Biasing Restricted Boltzmann Machines to Manipulate Latent Selectivity and Sparsity. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[7] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[8] T. Heiberg, B. Kriener, T. Tetzlaff, G. T. Einevoll, and H. E. Plesser. Firing-rate models for neurons with a broad repertoire of spiking behaviors. *BMC Neuroscience*, 14: P317 – P317, 2013.

[9] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science (New York, N.Y.)*, pages 504–7, 2006.

[10] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006.

[11] L. Holmstrom and P. Koistinen. Using additive noise in back-propagation training. *IEEE Transactions on Neural Networks*, 3(1):24–38, 1992.

[12] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial examples in the physical world. In *ICLR Workshop Track Proceedings*, 2017.

[13] D. Lisitsa and A. A. Zhilenkov. Prospects for the development and application of spiking neural networks. In *EIConRus*, pages 926–929, 2017.

[14] B. Luo, Y. Liu, L. Wei, and Q. Xu. Towards imperceptible and robust adversarial example attacks against neural networks. In *AAAI*, 2018.

[15] W. Maas. Networks of spiking neurons: The third generation of neural network models. *Trans. Soc. Comput. Simul. Int.*, 1997.

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[17] A. Marchisio, M. A. Hanif, F. Khalid, G. Plastiras, C. Kyrkou, T. Theocharides, and M. Shafique. Deep learning for edge computing: Current trends, cross-layer optimizations, and open research challenges. In *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 553–559, 2019.

[18] A. Marchisio, G. Nanfa, F. Khalid, M. A. Hanif, M. Martina, and M. Shafique. Capsattacks: Robust and imperceptible adversarial attacks on capsule networks. *ArXiv*, abs/1901.09878, 2019.

[19] A. Marchisio, G. Nanfa, F. Khalid, M. A. Hanif, M. Martina, and M. Shafique. Snn under attack: are spiking deep belief networks vulnerable to adversarial examples? *ArXiv*, abs/1902.01147, 2019.

[20] E. R. Merino, F. M. Castrillejo, J. D. Pin, and D. B. Prats. Weighted contrastive divergence. *CoRR*, abs/1801.02567, 2018.

[21] P. A. Merolla et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 2014.

[22] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.

[23] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, 7:178, 2013.

[24] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *ASIA CCS*, 2017.

[25] F. Ponulak and A. J. Kasinski. Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologiae experimentalis*, 71 4: 409–33, 2011.

[26] A. Shafahi et al. Are adversarial examples inevitable? In *ICLR*, 2019.

[27] M. Shafique, M. Naseer, T. Theocharides, C. Kyrkou, O. Mutlu, L. Orosa, and J. Choi. Robust machine learning systems: Challenges,current trends, perspectives, and the road ahead. *IEEE Design Test*, 37(2):30–57, 2020.

[28] A. J. F. Siegert. On the first passage time probability problem. *Phys. Rev.*, 81:617–623, 1951.

[29] C. Szegedy et al. Intriguing properties of neural networks. In *ICLR*, 2014.

[30] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida. Deep learning in spiking neural networks. *Neural Networks*, 2019.

[31] T. Tieleman and G. Hinton. Using fast weights to improve persistent contrastive divergence. In *ICML*, 2009.

[32] J. Vreeken. Spiking neural networks, an introduction. Technical report, 2003.

[33] J. Zhang and X. Jiang. Adversarial examples: Opportunities and challenges. *CoRR*, abs/1809.04790, 2018.

[34] J. J. Zhang, K. Liu, F. Khalid, M. A. Hanif, S. Rehman, T. Theocharides, A. Artussi, M. Shafique, and S. Garg. Building robust machine learning systems: Current progress, research challenges, and opportunities. *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019.