2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W) | 979-8-3503-2543-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/DSN-W58399.2023.00053

Stochastic Computing as a Defence Against Adversarial Attacks

Florian Neugebauer[1], Vivek Vekariya[2], Ilia Polian[1] and John P. Hayes[3]

[1]*Institute of Computer Architecture and Computer Engineering*
University of Stuttgart, Germany
florian.neugebauer@iti.uni-stuttgart.de
ilia.polian@iti.uni-stuttgart.de

[2]*fortiss GmbH*
Munich, Germany
vekariya@fortiss.org

[3]*Computer Engineering Laboratory*
University of Michigan
Ann Arbor, USA
jhayes@umich.edu

***Abstract*— Neural networks (NNs) are increasingly often employed in safety critical systems. It is therefore necessary to ensure that these NNs are robust against malicious interference in the form of adversarial attacks, which cause an NN to misclassify inputs. Many proposed defenses against such attacks incorporate randomness in order to make it harder for an attacker to find small input modifications that result in misclassification. Stochastic computing (SC) is a type of approximate computing based on pseudo-random bit-streams that has been successfully used to implement convolutional neural networks (CNNs). Some results have previously suggested that such stochastic CNNs (SCNNs) are partially robust against adversarial attacks. In this work, we will demonstrate that SCNNs do indeed possess inherent protection against some powerful adversarial attacks. Our results show that the white-box C&W attack is up to 16x less successful compared to an equivalent binary NN, and Boundary Attack even fails to generate adversarial inputs in many cases.**

Keywords—stochastic computing, neural network, adversarial attack

I. Introduction

Since the discovery of adversarial attacks on neural networks [10] there has been a contest between increasingly strong and successful attack algorithms and corresponding defensive measures. A wide range of different attacks under varying attack scenarios has since been developed, from attacks that assume knowledge about the NN structure and parameters (also called white-box attacks) to attacks that are applied to NNs of unknown structure (black-box attacks). On the other hand, defensive mechanisms have been proposed to counteract these attacks. Among them are input transformations such as rescaling and compression [6], and randomness based defences such as random dropout of neurons [5] and extra randomization layers [13]. However, such defensive measures lead to increased complexity and hardware cost. An architecture that incorporates defensive mechanisms naturally would therefore be highly beneficial.

Stochastic Computing (SC) is a promising candidate for this task. SC employs a number format based on randomized bit streams that encode probabilities. SC provides highly area and power efficient implementations of basic arithmetic functions, most prominently multiplication, which can be implemented with a single AND gate. As a type of approximate computing, SC has been shown to be a viable architecture for diverse applications including digital signal filters [12] and image processing [1]. In recent years, it has further been shown that SC offers efficient hardware implementations for CNNs for both ASIC [7] and FPGA [8] designs. For instance, [8] shows that real-time classification by a small SCNN is possible with an accuracy almost on par with a conventional binary CNN.

Moreover, experiments have shown that implementing parts of a conventional CNN in SC decreases the success rate of adversarial attacks on the network without sacrificing much of the overall classification accuracy [11]. These initial findings suggest that leveraging SC's inherent randomness enables the implementation of small, efficient CNNs that are naturally robust against adversarial attacks. SCNNs are commonly proposed for use in heavily resource constrained environments where implementing additional defensive measures can be too expensive for conventional binary networks.

In this work, we show that an CNN with a first layer implemented in SC severely reduces the effectiveness of C&W attack [4], a powerful white-box attack, by 16x for the fashion MNIST and by more than 10x for the CIFAR10 dataset. Moreover, Boundary Attack [2], a black-box attack that is closest to most real-life attack scenarios for SCNNs in our opinion, even fails to find adversarial inputs in many cases due to the approximate operations in SCNNs. The remainder of this work is structured as follows: Section II presents background information on SC and adversarial attacks. Specifics on our SCNN architecture are given in section III. Simulation setup and results are discussed in section IV, before section V concludes this work.

II. Background

A. Stochastic computing

SC computes arithmetic functions using a stream-based, probabilistic number format, the so-called stochastic numbers (SNs). In unipolar format, the value a of an SN A is defined as $a = n_1/n$ with n_1 being the number of bits with value 1 in A. In other words, the value of a unipolar SN is equal to the probability $P(\{A_i = 1\})$ of any bit i in A being 1. Bipolar format defines the value of A as $a = (n_1 - n_0)/n$ where n_0 is the number of 0s in A. It follows that unipolar SN values lie in the range $[0, 1]$ and have precision $1/n$, while bipolar SNs have values in the range $[-1, 1]$ and precision $2/n$. The position of 1s in an SN has no influence on its value—all 1s have identical weight. Therefore, most values have multiple possible representations.

The common method of creating an SN of a desired value is to use a small pseudo random number generator, e.g. a linear feedback shift register (LFSR). In each clock cycle, the LFSR generates a random number $r \in [0, 1]$ that is compared to the desired value d (in conventional binary format). After n clock cycles, this stochastic number generator (SNG) will have generated a unipolar SN of length n and expected value d. Changing the LFSR's starting state or its characteristic polynomial will generate an SN with the same approximate value, but different randomized position of 1s and 0s.

979-8-3503-2543-0/23/$31.00 ©2023 IEEE
DOI 10.1109/DSN-W58399.2023.00053

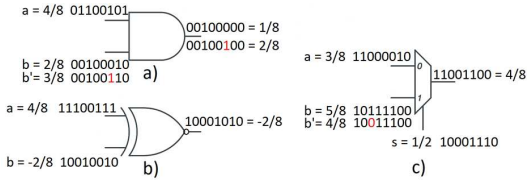

Figure 1: Basic SC operations: a) unipolar multiplication, b) bipolar multiplication, c) (unipolar) scaled addition.

Due to the identical significance of each bit in an SN, bit-parallel operations in conventional number formats can be computed in a bit-serial manner in SC, significantly reducing the hardware cost. For example, unipolar multiplication of two SNs X and X' can be performed via a single AND gate over n clock cycles, as the value of the output SN Y is $y = P(\{Y_i = 1\}) = P(\{X_i = 1\}) \cdot P(\{X'_i = 1\}) = x \cdot x'$. For bipolar multiplication, an XNOR gate is used. Scaled addition can likewise be performed efficiently by a multiplexer. Fig. 1 shows examples for these operations.

Several properties of SC are demonstrated in fig. 1. Fig. 1a shows how a bit flip (marked in red) in input b' changes the value of the output only by $1/n$, while in fig 1c the change in the input value does not change the output value at all, due to the randomized position of 1s. A source of inaccuracy in SC is shown in fig. 1b. The expected result of this bipolar multiplication is $-1/8$, however the SN length of 8 is not sufficient to represent this value. The result of $-1/4$ is the closest possible approximation in this case. While this inaccuracy is usually considered a downside of SC, it can be beneficial with regard to adversarial attacks. These attacks try to make small modifications to an NN's input data that lead to a misclassification. However, when the inputs of the network are SNs, small changes in the inputs generally lead to inconsequential and unpredictable changes in the output and sometimes even to no change at all (see fig. 1c). Especially the latter property should be noted, as it means that even in cases where an adversarial example has been found, it may not lead to a consistent misclassification. This reduces the efficiency of those attacks and makes it harder to find adversarial examples in the first place, without the need for specific defensive operations in the NN.

B. Adversarial attack scenarios

The goal of an adversarial attack is to cause the target NN to misclassify a given input. Attacks achieve this goal by applying small (commonly measured in a distance metric such as L_2) changes, called perturbations, to this input. These perturbations are specifically crafted for each input and attack. For example, an attack might use a backpropagation algorithm, similar to the one used for training an NN, to compute the input perturbations required to cause a misclassification, as opposed to the changes in the weights that are computed during training. If the goal of an attack is a specific output class, it is called "targeted", if any output other than the original class is accepted, it is called "untargeted". Both cases are considered in this work.

Many different types of adversarial attacks with access to varying degrees of information about the target network exist. In general, attacks can be classified according to what is known by the attacker about the target NN. In this work, we use the following attack classification:

- White-box attacks have complete or nearly complete information about the NN architecture, its implementation, and parameters such as weights. This information is used in gradient-based attacks, for example the C&W attack.
- Black-box attacks have no information about network architecture and parameters and can only use the NN as an oracle. Some attacks in this class assume information about the exact output values, while others only use the label of the final output class.

A direct white-box attack on an SCNN is infeasible, as some SC components do not provide exact implementations of their target functions. For example, the commonly used stanh activation function [3] and its variants are based on a finite state machine (FSM) that provides a very close approximation to the hyperbolic tangent function, but they are not mathematically identical. In this work, we therefore consider white-box attacks according to the definition above with nearly complete knowledge about the NN model. They use the underlying arithmetic binary model instead of the hardware-specific, cycle wise SC model in order to find adversarial examples.

In black-box attacks, the target NN is used as a so-called oracle, i.e., the attacker can query the NN by providing inputs and receives an answer from the NN. This answer can be in the form of a class label or the output vector of the final NN layer. We consider this model to be more realistic than white-box attacks for SCNNs, as it does not depend on the specific hardware design. Furthermore, SCNNs are intended for specialized, low-cost hardware implementations with limited communication capabilities, for example in sensor nodes. Consequentially, we focus our simulations on attacks that we consider to be powerful and/or close to likely real life scenarios for SCNNs, namely C&W [4] (white-box, targeted) and Boundary Attack [2] (black-box, untargeted).

III. SCNN Architecture

A. SCNN Implementation

In recent years, a de facto standard architecture for SCNNs and SC neurons has established itself, with varying circuit specifics. In convolutional layers, SC implements the main operations of multiply-accumulate (MAC), activation function and max-pooling in one combined hardware component as shown in fig. 2. Each XNOR gate multiplies a

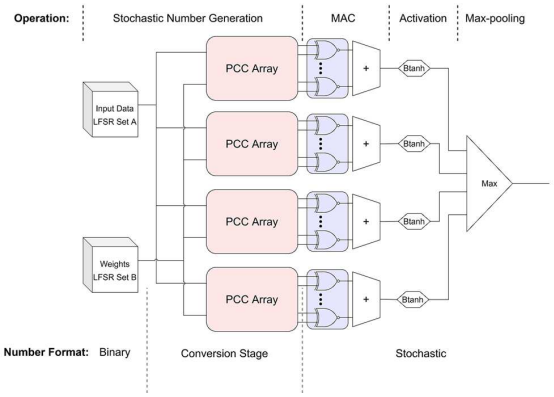

Figure 2: Basic SC MAC-activation-pooling component with LFSR and probability conversion circuit (PCC), e.g. comparator.

weight with an activation value. All products of a single kernel (the set of weights used in a specific convolution operation) are then added, whereby the particular implementation of the adder can vary. Some SC designers opt for an exact parallel counter, which results in a stream of integers equal to the sum of products. We employ an approximate version of such a counter, producing a slightly inaccurate sum, but reducing hardware cost in the process.

The subsequent activation function commonly computes either a hyperbolic tangent (tanh) or clipped ReLU function. Clipping is necessary to stay within the representable range of SNs, which cannot exceed 1. The tanh function provides an inherent clipping, as its image is $[-1, 1]$. It is therefore the most widely used activation function in SCNNs. In either case, the function is implemented with an FSM based on the design in [3]. As the output format of the addition circuit can vary slightly depending on the implementation details, the FSM has to be adjusted accordingly. MAC and activation functions are often combined into a single hardware component in SC. Finally, a stochastic maximum circuit implements max-pooling. Like the stochastic adder, its implementation details can differ between designs, we employ the design from [9].

B. Effects of randomness in SCNNs

The core concepts behind SC as a defence against adversarial attacks are its probabilistic data format and operations. In our implementation, SNs are generated according to the method described in section II.*A*, i.e., using an LFSR with specific sequence length as a random number source, leading to SNs with exact values except for rounding errors. Randomness is therefore introduced into the SCNN only through arithmetic operations, specifically MAC operations and the activation function. The main benefit of this approach is that the randomization is entirely invisible from the outside. No additional operations or layers are needed and the NN model does not need to be modified.

If an SCNN receives the same input twice, its computations and output values will be identical only if the LFSR starting states are identical. If the LFSR starting states are changed however, computations and output values will differ even for identical inputs. We illustrate this effect by generating the same feature map three times with identical input data but different LFSR starting states. Fig. 3 shows these feature maps using the fashion MNIST dataset. In all cases, the object is clearly recognizable and looks the same. However, individual pixel values vary slightly, as is best seen in the background pixels. Since the object itself is accurately preserved, the network is still able to classify the images correctly in all cases. Smaller perturbations caused by an attacker may however not be transmitted consistently through layers as intended.

IV. Simulation Setup and Results

A. Network and simulation setup

The structure of our simulated NN is a slightly modified version of the model used by Carlini and Wagner in the analysis of their attack [4]; details are given in table 1. The number of kernels in each layer varies with the type of input image (RGB or greyscale). Hyperbolic tangent is used as activation function in the first layer, as it lends itself better for implementation in SCNNs. The networks were implemented in python using the tensorflow framework, attacks were performed using the code provided by their respective authors

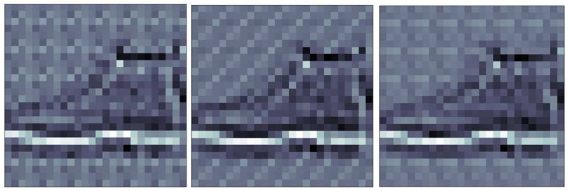

Figure 3: Feature map from the first layer of an SCNN using identical inputs and weights but different LFSR starting states.

[14][15]. We restrict the use of SC to the first NN layer to preserve as much classification accuracy as possible.

The only necessary adaptation from conventional to SCNN is to scale the weighs of the layers that are implemented in SC to the interval $[-1, 1]$ during training. This can be easily achieved by scaling all weights w^l in an affected layer l by the maximum absolute weight value in layer l after the regular weight update step Δw^l:

$$w^l = \frac{w^l + \Delta w^l}{\max_l |w^l + \Delta w^l|} \quad (1)$$

B. C&W attack results

C&W attack was used to generate adversarial examples in a white-box scenario for both datasets. In the case of fashion MNIST, 900 adversarial examples, and in the case of CIFAR10, 450 adversarial examples were generated. Targeted attacks using L_2 norm were performed in both cases and only images that were initially correctly classified by the network were considered for the attack. Classes are represented equally in attacks with 10% of all initial images and targets per class. We consider an attack to be successful if the network classifies the adversarial input as the target class and unsuccessful otherwise. Unsuccessful attacks are further split into adversarial inputs that were classified wrongly as some class other than the target or initial class, and correctly classified inputs.

Table 2 shows the success rates of the attack on the binary reference network and the SCNN. In case of the binary network, the C&W attack on fashion MNIST has a success rate of 81.6%; for CIFAR10 the success rate is 98%. The SCNN on the other hand shows a remarkable resilience against the attack. For fashion MNIST, only 5.1% of attacks are successful, while 55.2% of adversarial inputs are still classified correctly. Many inputs end up being classified incorrectly. We consider these cases a partially successful

Table 1: Network structure with layer ID (from input to output) used in simulations. Kernel sizes vary with input type (greyscale or RGB).

ID	Layer type	Kernel size	AF
1	2DConvolution	3x3x32/3x3x3x64	tanh
2	2D Convolution	3x3x32/3x3x3x64	ReLU
3	Max-pooling	2x2	–
4	2D Convolution	3x3x64/3x3x92	ReLU
5	2D Convolution	3x3x64/3x3x92	ReLU
6	Max-pooling	2x2	–
7	2D Convolution	3x3x128	ReLU
8	2D Convolution	3x3x128	ReLU
9	Max-pooling	2x2	–
10	Dense 20/100	–	–
11	Dense 10	–	softmax

Table 2: Classification results of C&W attack on fashion MNIST and CIFAR10 datasets.

	Binary NN		**SCNN**	
	#Inputs	Ratio	#Inputs	Ratio
Fashion MNIST output classes				
Target	734	81.6%	46	5.1%
Correct	42	4.7%	497	55.2%
Other	124	13.8%	357	39.7%
CIFAR10 output classes				
Target	441	98.0%	41	9.1%
Correct	1	0.2%	94	20.9%
Other	8	1.8%	315	70.0%

defence, as even though the network's output is wrong, the attack does also not achieve its goal. Similar results can be observed for CIFAR10, where only 9.1% of attacks on the SCNN are successful. While most adversarial examples are not classified correctly, the SCNN still provides a very good defence against the attack. In summary, SC reduces the success rate of the C&W attack by a factor of 16 for the fashion MNIST dataset and a factor of 10.8 for CIFAR10.

C. Boundary attack results

The execution of Boundary Attack takes significantly longer than C&W attack, as it is performed on the SCNN directly and thus has to simulate all bitwise operations for every one of its iterations. We therefore initially only tried to create adversarial examples for 100 images of fashion MNIST. After 10,000 iterations for each image however, adversarial examples had only been created successfully for 22 of them. In the other 78 cases, no adversarial examples within the specified L_2 limit had been found. A look at the development of L_2 distances over the course of the attack's iterations showed that for most images this distance did not decrease at all, as should be the case according to [2]. In an example given by the authors, the distance decreases by more than one order of magnitude between iterations 1,053 and 1,828: from $8.0 \cdot 10^{-3}$ to $5.6 \cdot 10^{-4}$. In our attack on the SCNN however, the average L_2 distance stayed almost constant between iterations. For instance, iteration 1,053 had an average L_2 distance of 2.90; in iteration 1,828 the average distance was 2.86. Only a few samples reached distances in the order of 10^{-3} and led eventually to the creation of successful adversarial examples. Further iterations showed that the distances for other samples fluctuated, but did not show a gradual decrease.

The low success rate of Boundary Attack is due to the fuzziness of decision boundaries in an SCNN illustrated in fig. 4. A conventional binary network has a clearly defined, sharp decision boundary between classes, which Boundary Attack uses as a guideline to generate adversarial examples. However, an SCNN does not have such a sharp boundary. It instead has something more akin to a "boundary space". Within this space, the SCNN does not always place the same input in the same class, but only classifies it with a certain probability. Boundary Attack can therefore not orient itself along a decision boundary and fails to converge.

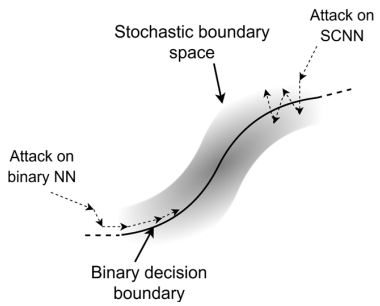

Figure 4: Illustration of binary decision boundary vs. stochastic boundary space and Boundary Attack iterations.

V. Conclusion

We have investigated the defensive capabilities of SCNNs regarding selected adversarial attacks. Our simulations show that SCNNs possess inherent robustness without the need for additional NN layers or defensive operations that are required in conventional binary NNs. The inherent randomness of SC prevents even powerful targeted attacks such as the C&W attack from reliably achieving their desired misclassifications. Additionally, we showed that not only is Boundary Attack similarly well defended against, but an SCNN also makes it very difficult for this specific black-box attack to find adversarial examples in the first place. In future work, our goal is to investigate if SC's robustness is attack-dependent, and to analyze the trade-off between classification accuracy and maximizing defensive capabilities.

Acknowledgements: This research was supported in part by Deutsche Forschungsgemeinschaft (DFG) project number PO1220/12-1. J.P. Hayes was supported by U.S. National Science Foundation grant CCF-2006704.

References

[1] A. Alaghi, C. Li and J.P. Hayes. Stochastic circuits for real-time image-processing applications. *Proc. DAC,* art. 236, 2013.

[2] W. Brendel, J. Rauber and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *preprint arXiv:1712.04248*, 2017.

[3] B. D. Brown and H. C. Card. Stochastic neural computation I: computational elements. *IEEE Trans. Comp.*, (50) 891-905, 2001.

[4] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. *Proc. IEEE Symp. Security and Privacy,* 39-57, 2017.

[5] G. S. Dhillon et al. Stochastic activation pruning for robust adversarial defense. *preprint arXiv:1803.01442*, 2018.

[6] C. Guo, M. Rana, M. Cisse and L. Van Der Maaten. Countering adversarial images using input transformations, *arXiv: 1711.00117*, 2017

[7] V. T. Lee, A. Alaghi, J. P. Hayes et al. Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing. *Proc. DATE,* 13-18, 2017.

[8] P. K. Muthappa, F. Neugebauer, I. Polian and J. P. Hayes. Hardware-based fast real-time image classification with stochastic computing. *Proc. ICCD,* 340-347, 2020.

[9] F. Neugebauer, I. Polian and J. P. Hayes. On the maximum function in stochastic computing. *Proc. Intl. Conf. Computing Frontiers*, 59-66, 2019.

[10] C. Szegedy et al. Intriguing properties of neural networks. *preprint arXiv:1312.6199*, 2013.

[11] P. Ting and J. P. Hayes. Exploiting randomness in stochastic computing. *Proc. ICCAD,* 1-6, 2019.

[12] R. Wang, J. Han, B. F. Cockburn and D. G. Elliott. Design, evaluation and fault-tolerance analysis of stochastic FIR filters. *Microelectronics Reliability* (57), 111-127, 2016.

[13] C. Xie et al. Mitigating adversarial effects through randomization. *preprint arXiv:1711.01991*, 2017.

[14] Github: https://github.com/carlini/nn_robust_attacks

[15] Github: https://github.com/greentfrapp/boundary-attack