AdverQuil: an Efficient Adversarial Detection and Alleviation Technique for Black-Box Neuromorphic Computing Systems

Hsin-Pai Cheng
Department of ECE
Duke University
Durham, NC
hc218@duke.edu

Juncheng Shen
Department of ECE
Duke University
Durham, NC
juncheng.shen@duke.edu

Huanrui Yang
Department of ECE
Duke University
Durham, NC
hy128@duke.edu

Qing Wu
Air Force Research Laboratory
Rome, NY
qing.wu.2@us.af.mil

Hai Li
Department of ECE
Duke University
Durham, NC
hai.li@duke.edu

Yiran Chen
Department of ECE
Duke University
Durham, NC
yiran.chen@duke.edu

ABSTRACT

In recent years, neuromorphic computing systems (NCS) have gained popularity in accelerating neural network computation because of their high energy efficiency. The known vulnerability of neural networks to adversarial attack, however, raises a severe security concern of NCS. In addition, there are certain application scenarios in which users have limited access to the NCS. In such scenarios, defense technologies that require changing the training methods of the NCS, e.g., adversarial training become impracticable. In this work, we propose AdverQuil – an efficient adversarial detection and alleviation technique for black-box NCS. AdverQuil can identify the adversarial strength of input examples and select the best strategy for NCS to respond to the attack, without changing structure/parameter of the original neural network or its training method. Experimental results show that on MNIST and CIFAR-10 datasets, AdverQuil achieves a high efficiency of 79.5 - 167K image/sec/watt. AdverQuil introduces less than 25% of hardware overhead, and can be combined with various adversarial alleviation techniques to provide a flexible trade-off between hardware cost, energy efficiency and classification accuracy.

CCS CONCEPTS

• **Computing methodologies** → **Cognitive science**; **Neural networks**; • **Hardware** → **Neural systems**;

1 INTRODUCTION

Deep neural networks (DNNs) have achieved a remarkable success in real-world applications such as image and audio recognitions, natural language processing, and semantic understanding [1][2][3][4], etc. These deployed DNNs often have a large number of parameters and involve extensive computations. However, performance of conventional computing hardware that based on von Neumann architecture is greatly hindered by the increasing gap between high computing capacity and limited memory bandwidth.

Inspired by the insight of neural science, neuromorphic computing systems (NCS) are proposed to accelerate computations of neural networks. Neurons/axons and synapses are two basic units of a NCS. Similar to biologic neural networks, a NCS is composed of a large amount of parallel, extensively connected neurons/axons. Such a design offers a high computation efficiency and is often able to be reconfigured to various complex DNN models such as convolutional neural networks (CNNs) [5]. As a recent important research outcome, the IBM TrueNorth Neurosynaptic System that built on so-called IBM TrueNorth chip(s) can perform an event-driven hand gesture recognition in real-time with a power consumption less than 200mW [6].

However, recent studies show that many machine learning models including DNNs are vulnerable to adversarial examples – a type of malicious inputs crafted with slight perturbations [7] [8]. From a security perspective, the attack to the machine learning model using adversarial examples is often referred to as adversarial attack [9][10]. In addition, recent studies also show that adversarial attack has a certain level of transferability, i.e., an adversarial attack that is harmful to one model may be also harmful to the others [11]. For instance, [12] demonstrates that adversarial attack has a wide transferability to fool ResNet, VGG and GoogleNet.

To protect machine learning model from being attacked by adversarial examples, many defense methods have been proposed, such as adversarial training, ensemble model and cascade model [7][13][14]. These methods can recover the model accuracy from the degradation caused by the attack. However, there are multiple things are overlooked in these technologies.

First, many of these defense methods such as adversarial training require the changes of training process of the DNN models, including training samples, model parameters, and model configuration. Hence, the applicability of these methods is constrained by the accessibility to the DNN model and the training examples, which are often limited by intellectual property and other considerations;

ASP-DAC 2019, Jan. 2019, Tokyo, Japan

ACM ISBN 978-1-4503-6007-4/19/01…$15.00
https://doi.org/10.1145/3287624.3288753

Second, computation workload, hardware implementation cost, and energy consumption of these defense methods have not been carefully optimized as these performance metrics are usually not the primary concerns of the target applications. Such a design philosophy greatly hinders the deployment of these defense methods in edge and mobile applications.

In this work, we propose AdverQuil – An efficient adversarial detection and alleviation technique for black-box NCS. Compared to the existing defense methods to adversarial attack, AdverQuil has the following unique properties and advantages:

(1) AverQuil can identify the adversarial strength of input examples and guide the NCS to select the best strategy to respond to the attack. No changes in the parameters, the configuration and the training method of the protected DNN model running on the NCS are needed;
(2) AdverQuil treats the protected DNN model as a black-box during the adversarial strength detection process and does not need any information about the DNN model. This design offers a great flexibility in the deployment of AdverQuil;
(3) AdverQuil is transparent to the protected DNN model and can be combined with other defense methods (either black-box or white-box), e.g., spatial smoothing and adversarial training for an enhanced protection;
(4) AdverQuil enables a flexible design framework to explore the tradeoff between protection effectiveness and the hardware and energy costs.

The rest of this paper is organized as follows: Section 2 gives background about NCS and adversarial attack on DNN model; Section 3 and 4 present the motivation and design details of AdverQuil; Section 5 shows our experimental results and discussions; Section 6 concludes our work.

2 BACKGROUND

2.1 Neuromorphic Computing System

Neuromorphic Computing System (NCS) was originally referred to as the hardware that mimics neuro-biological architectures to implement models of neural systems. The concept was then extended to computing systems that can run bio-inspired computing models, e.g., neural networks. IBM Neurosynaptic System is a famous NCS that is built on a neuromorphic ASIC chip named TrueNorth. Unlike conventional von Neumann architectures where data processing and storage are separated, a TrueNorth chip is composed of 4,096 highly parallel and connected neurosynaptic cores. Every neurosynaptic core consists of 256 input axons and 256 output neurons that are connected with a 256×256 synapse crossbar. The input to the TrueNorth chip is encoded as spikes. By leveraging the ON/OFF state of a pair of neurons, the weight represented at each cross-point of the synapse crossbar can be selected from {-1, 0, 1}. A floating-point input can be approximated by the occurrence probability of the spikes during a time frame, and a floating-point weight can be approximated using multiple hardware copies of the synapse crossbar. The output from the synapse crossbar is processed by an activation function circuitry [15][16][17][18].

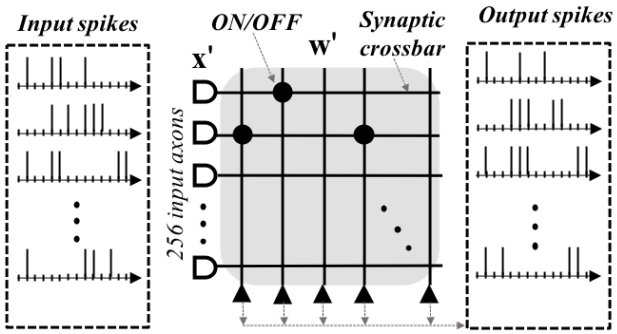


Figure 1: Executing neural Networks on TrueNorth.

An artificial neural network's operation can be generally described as:

$$\begin{aligned} y &= \mathbf{wx} \\ a &= f(y). \end{aligned} \tag{1}$$

Here y is an inner product of a weight vector ($\mathbf{w}$) and an input vector($\mathbf{x}$). f(·) is a nonlinear activation function. As shown in Figure 1, to map an artificial neural network to TrueNorth neurosynaptic cores, we quantize the weight ($\mathbf{w}$) and input vector ($\mathbf{x}$) to **w'** and **x'**, respectively. Assuming McCulloch-Pitts neuron model is adopted, the corresponding stochastic operation on TrueNorth can be represented by Equations (2) - (5) as:

$$y' = \sum_{i=1}^{n-1} w'_i x'_i, \tag{2}$$

$$\begin{aligned} P(w'_i = c_i) &= p_i \\ P(w'_i = 0) &= 1 - p_i, \end{aligned} \tag{3}$$

$$p_i = w_i / c_i, \tag{4}$$

$$\begin{aligned} P(x'_i = 1) &= x_i \\ P(x'_i = 0) &= 1 - x_i. \end{aligned} \tag{5}$$

Here, i is the index of an input and its weight. Equation (3) explains the ON/OFF state of the synapse follows the Bernoulli distributions with probability p_i. That is to say, if the synaptic connection is ON, a value c_i is assigned as its weight, otherwise the synapse is in an OFF connection with a weight of 0. Equation (4) assures that the expected value of the synaptic weight in TrueNorth equals the corresponding weight in the mapped ANN. In Equation (5), the input x is transformed to a bitstream of 0 and 1 and therefore, the probabilities of $x'_i = 1$ and $x'_i = 0$ are x_i and $1 - x_i$, respectively.

2.2 Adversarial Attack on a NCS

Adversarial attack denotes the method of adding an imperceptible perturbation to input examples (i.e., adversarial examples) to mislead a learning-based model. An adversarial example $\hat{x}$ can be generated by injecting perturbation η to the original input sample x such as $\hat{x} = x + \eta$ [7]. The linear transformation of $\hat{x}$ with respect to a given weight vector $\mathbf{w}$ can then be expressed as:

$$\mathbf{w}\hat{\mathbf{x}} = \mathbf{wx} + \mathbf{w}\eta. \tag{6}$$

For a fixed adversarial perturbation, $\mathbf{w}\hat{\mathbf{x}}$ increases linearly with the dimensionality of $\mathbf{w}$. Since the dimensionality of $\mathbf{w}$ is high in practical problems, a minor perturbation η could induce a large change of $\mathbf{w}\hat{\mathbf{x}}$. The adversarial example, hence, may be mistakenly recognized as a wrong class. Adversarial attacks can also be implemented on a NCS. In TrueNorth, for example, a perturbation $w_i\eta_i$ can be added to each $w_i x_i$. By injecting a perturbation, the summation term, $\hat{y}$, is therefore perturbed to $\hat{y'}$ as:

$$\hat{y'} = \sum_{i=0}^{n-1} w_i' x_i' + w_i' \eta_i. \tag{7}$$

Now we have to prove that the expected value of $\hat{y'}$ equals $\mathbf{wx} + \mathbf{w}\eta$. Suppose w' and x' are independent, the expectation of $\hat{y'}$ can be computed as:

$$\mathbb{E}\{\hat{y'}\} = \mathbb{E}\{\sum_{i=0}^{n-1} w_i' x_i'\} + \mathbb{E}\{\sum_{i=0}^{n-1} w_i' \eta_i'\}. \tag{8}$$

Since w_i equals $p_i c_i$ based on Equation (4), we have:

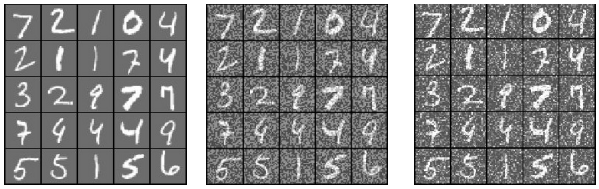
$$\begin{aligned} \mathbb{E}\{\hat{y'}\} &= \sum_{i=0}^{n-1} p_i c_i x_i + \sum_{i=0}^{n-1} p_i c_i \eta_i \\ &= \sum_{i=0}^{n-1} w_i x_i + \sum_{i=0}^{n-1} w_i \eta_i \\ &= \mathbf{wx} + \mathbf{w}\eta. \end{aligned} \tag{9}$$

The above explanation indicates that added perturbation would also be implemented on the IBM Neurosynaptic System, i.e., an adversarial example can be represented as a series of spikes that disguise as a legitimate data but cause TrueNorth chips unable to classify it correctly. This conclusion will be experimentally validated in Section 3.

2.3 Existing Detection and Defense Methods

Many approaches have been proposed to defend adversarial attack. Here we list three methods most relevant to our work.

(1) *Spatial Smoothing*: Spatial Smoothing is one kind of image pre-processing techniques that blurs the input image in order to suppress the noises on the image [19]. The low computation cost of spatial smoothing make it ideal for low-power and cost-sensitive applications.
(2) *Cascaded Structure*: Recent studies show that cascading multiple base classifier can improve the overall classification accuracy [20]. Theoretically, similar structure can also be applied to improve the robustness on a NCS by cascading multiple subnetworks. However, such a cascaded structure inevitably increases the hardware overhead and prolong the data processing latency.
(3) *Adversarial Training*: It has been shown that the resilience of learning models to adversarial attack can be enhanced by being trained with adversarial examples [7]. Grouping normal and adversarial examples into the training dataset, for example, can enhance the resilience of the learning model [21]. Since the training dataset has been changed, adversarial training indeed alters the parameters of the learning model from the original ones that are obtained from the original training dataset.



(a) original images **(b) image with adversarial noise** **(c) images with Gaussian noise**

Figure 2: The MNIST dataset with different types of noises. The examples with adversarial noise and Gaussian noise share some similarities.

3 MOTIVATION

Although many approaches have been proposed to defend adversarial attack, almost all of these approaches will either degrade the model accuracy of classifying the non-adversarial testing examples, or drastically increase hardware overhead. Therefore, when to apply these approaches becomes the essential to achieve a successful defense. In this section, we will present the motivation of our work through the analysis of the limitations of two existing adversarial defense methods – spatial smoothing and adversarial training.

3.1 Spatial Smoothing's Limitations

Because adversarial examples are generated by injecting perturbations into the original examples, the adversarial examples appear as the images contaminated by noises, as depicted in Figure 2.

Spatial smoothing is a nonlinear noise suppression technique that amortizes the noise through averaging the pixel values of the images. In spatial smoothing, every image pixel is replaced with the median value of its neighbor pixels. Spatial smoothing has been proven capable of reducing Gaussian noise on the images [19].

By considering the perturbation injected to the adversarial examples as a type of noises, spatial smoothing can be also applied to filter out such (adversarial) noises before the images are sent to the classifier for classification. Such a mechanism does not require any changes on the classifier and hence, is applicable to black-box adversarial defense scenario. During the application of spatial smoothing, we need to find out the best filter size that results in the most effective protection of the model against the adversarial attack (e.g., using training dataset).

Figure 3 illustrates the classification accuracies of the test examples with different adversarial strengths without ('Original') and with ('Spatial smoothing') spatial smoothing for both MNIST and CIFAR-10. When the adversarial strength is high, adding spatial smoothing produces a better accuracy than the original model, implying that the adversarial noises are effectively filtered out. When the adversarial strength is low, however, the accuracy of applying spatial smoothing becomes lower than that of the original model. This is because that in such a case, spatial smoothing blurs the images that being classified and leads to an accuracy degradation larger than the one recovered from the adversarial attack when the adversarial strength is small.

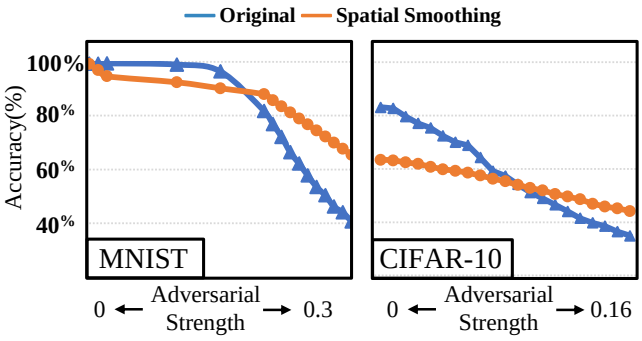


Figure 3: The classification accuracies before and after applying spatial smoothing under adversarial attacks with different adversarial strengths on a) MNIST and b) CIFAR-10. Spatial smoothing harms the model accuracy when adversarial strength is low.

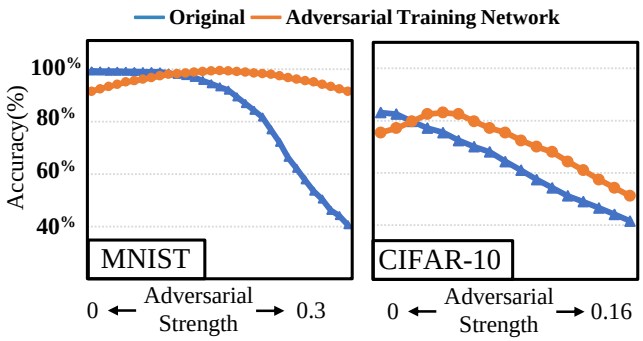


Figure 4: The classification accuracies of original model and adversarial trained model under adversarial attacks with different adversarial strengths on a) MNIST and b) CIFAR-10. The adversarial strengths adopted during the adversarial training network ε=0.05 (for MNIST) and ε=0.03 (for CIFAR-10), respectively. Adversarial training performs badly when the adversarial strength is low.

3.2 Adversarial Training's Limitations

We also perform adversarial training on the learning models adopted in Figure 3 and compare the results to the ones of original models, as shown in Figure 4. Here the adversarial strengths adopted during the adversarial training ε=0.05 (for MINIST) and ε=0.03 (for CIFAR-10), respectively. Similar to the spatial smoothing, adversarial training performs worse when the adversarial strength of the adversarial attack is low. It is because that adversarial training harms the intrinsic classification accuracy of the learning model on the original test examples.

The results in Figure 3 and Figure 4 demonstrate that many existing defense methods work effectively over only limited ranges of the adversarial strength under the adversarial attack and generally perform badly on the non-adversarial test examples. To solve this problem, we have to accurately identify when these defense methods should or should not be applied.

4 PROPOSED METHOD

In this section, we explain how to utilize the advantage of a NCS and combine with our proposed method to defend against adversarial attack. The following sections are organized as follows. First, we set up the adversarial environment in a real-world scenario by adopting various types of adversarial attacks. Second, we introduce the workflow of AdverQuil. Finally, we explain the function and outcome of each component.

4.1 Robustness in a Real-world Scenario

Previous studies only consider an ideal scenario of adversarial attack [22]. However, in a real-world scenario, an adversarial attack can be any type. Therefore, in out experimental setup, we consider multiple adversarial attacks to simulate the real-world scenario. In particular, we adopt the following three attacking methods with a wide range of strength to testify the robustness in different aspects.

(1) *Gaussian noise attack*: To test whether the model is robust to natural noise, we adopt Gaussian noise as one of the attack methods. Gaussian noise is commonly exist and continuous. This adversarial example is easy to be generated and we assume NCSs are robust to Gaussian noise to some extent.

(2) *Fast gradient sign method (FGSM)*: The main idea of FGSM is that the adversarial example can be crafted by adding a perturbation orthogonal to the classification model's decision boundary. This perturbation causes the example to cross to another class. Since most of the machine learning model has a similar shape of decision boundary, it is very likely to transfer adversarial example from one model to another. We use this method to test whether a model is robust to adversarial examples transferred from a different model.

(3) *Carlini & Wagner method (CW)*: CW attack is a recently proposed effective attacking method [23]. It iteratively construct the adversarial example by solving the minimization problem as:

$$\begin{aligned} &minimize \quad D(x, x+\eta) \\ &s.t. \quad C(x+\eta) = t \end{aligned} \tag{10}$$

Here D is a distance measurement metric that measures the distance between legitimate example, x, and adversarial $x+\eta$. By iteratively minimizing the distance and also satisfying the constrain, this method demonstrates a strong attack to the machine learning model

In this paper, the above three attack methods are utilized to testify whether a NCS is robust to natural noise, common adversarial examples, and strongest adversarial examples, respectively.

4.2 Structure of AdverQuil

The proposed AdverQuil framework can be conceptually depicted in Figure 5. AdverQuil is composed of three main components – spike-efficient encoding, adversarial detector, and alleviation. These components are serially connected to perform the function.

(1) *Spike-efficient encoding*: Spike-efficient encoding is a novel spike encoding method that specifically designed for the adversarial detector allowing the adversarial detector using

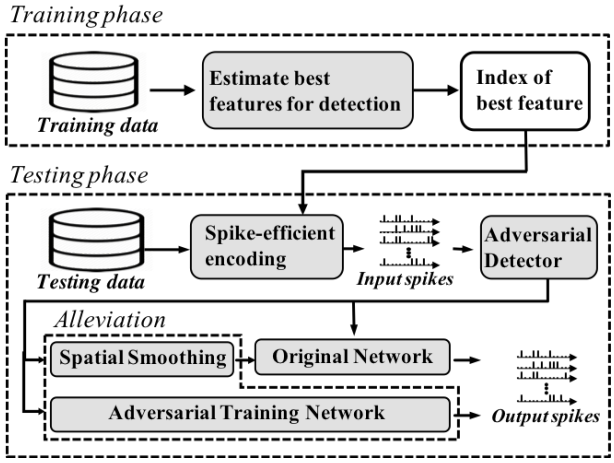


Figure 5: The workflow of AdverQuil.

less spike to efficiently distinguish adversarial attacks. This method can be combined with other existing input coding methods such as rate coding and time coding;

(2) *Adversarial detector*: This component detects whether the input is a strong adversarial example or a weak adversarial example.

(3) *Alleviation*: We alleviate the impact of the adversarial attack by adopting existing defense methods such as spatial smoothing or adversarial-training.

The whole workflow of AdverQuil can be summarized as the follows: An input example first passes through the adversarial detector and is classified as either a weak adversarial example or strong adversarial example. When the adversarial detector identifies the input example as a strong one, it will be passed to the *alleviation* part; Otherwise, it will be directly sent to the original classifier without any special handling; It is worth noting that AdverQuil is transparent to the embedded adversarial alleviation technique. AdverQuial can effectively overcome the limited working range issue of many adversarial alleviation techniques by guiding the application of these techniques based on the adversarial strength of the input examples. In addition, AdverQuil offers great scalability and expandability by classifying the input examples into more than one group, each of which that can be properly handled by different adversarial alleviation techniques.

4.3 Spike-efficient Encoding

Spike-efficient encoding can be divided to a training phase where it searches the adversarial examples' saliency map, and a testing phase which reorders the spikes according to their importance ranks. This encoding method provides the adversarial detector an efficient scheme to encode the input examples with spikes, enabling less communication cost and faster detection of the adversarial strength of the input examples.

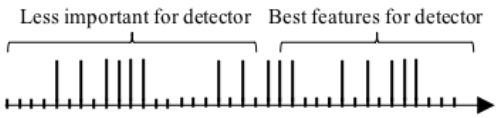


Figure 6: The best features have higher priority to be transmitted

Without loss of generality, we denote the pixels of an input example's saliency map as the features of the input example. Suppose we use a saliency map with a width of x_t, we will have x_t^2 features. With coding rate C, the total coding cost would be $s(x_t) = C \cdot x_t^2$. Therefore the saliency map width can be serve as a good metric of the coding cost.

The goal of the proposed spike-efficient encoding scheme is to find an optimum saliency map width (x^*) of the input examples which would lead to a minimum coding cost. We can write the optimization objective as:

$$\begin{aligned} x^* &= \arg\min_{x_t} s(x_t) \\ &= \arg\min_{x_t} C \cdot x_t^2. \end{aligned} \tag{11}$$

During the search of the optimum spike number, we also need apply a constraint to ensure that the adversarial detector's accuracy of $A(x_t)$ does not fall lower than its original accuracy $A(x_0)$. The goal of the constrained optimization can then be rewritten as:

$$\begin{aligned} x^* = &\arg\min_{x_t} C \cdot x_t^2 \\ &s.t.\ A(x_0) - A(x_t) \le 0. \end{aligned} \tag{12}$$

In practical, we may tolerate a small amount of accuracy loss by penalizing the objective function with an additional term of $\lambda(A(x_0) - A(x_t))$. Here λ is a positive constant. The objective function can then be written in a Lagrangian as:

$$x^* = \arg\min_{x_t} C \cdot x_t^2 + \lambda(A(x_0) - A(x_t)),\ \lambda > 0. \tag{13}$$

In the objective of Equation (13), the accuracy term $A(\cdot)$ may not be convex and is hard to be optimized directly. Hence, we use greedy search to find the optimal x_t in the training phase of spike-efficient encoding. As depicted in Algorithm 1, it greedily searches x_t by iteratively discarding the least important feature in the original input. Here the importance of a feature is measured by weight connection method [24].

We first use all the pixels as the features of an input example. Since features that have small pixel variances generally carry limited information, we discard 10% of the features with the least variances; We then train the adversarial detector using the rest 90% features. If the accuracy degradation of the trained adversarial detector is still within the allowable range, we will repeat the above steps; Otherwise, we will stop and use the feature set before the 10% reduction of the current iteration as the final choice x_t.

In the testing phase, we reorder the sequence of the pixels based on the choice of x_t: the pixels belonging to saliency map of a width of x_t are place in the front while the other pixels are placed to the tail of the bitstream, as shown in Figure 6. This encoding method allows the adversarial detector to quickly identify adversarial examples

Table 1: The contribution of spike-efficient encoding.

Dataset	MNIST	CIFAR-10
spike saved	**71.31%**	**43.75%**

Algorithm 1 Spike-efficient encoding

```
procedure TRAINING PHASE(feature, n)
    Discard the features have lowest variance
    // Use the rest of the features to train
    // Decrease these features to n
    while len(rest) > n do
        features = features[rest]
        // train the features with detector model
        detector.train(features, y)
        // Rank the remaining features
        if detector.coed.ndim > 1 then
            x_t = sort(detector.coefs)
            Discard the lowest rank features and repeat
    return x_t
procedure TESTING PHASE(feature, x_t)
    sort the features according to importance
    encode with rate coding
```

without reading all the input spikes. Table 1 illustrates the saving of the number of spikes for adversarial detection using the proposed spike-efficient encoding scheme and adversarial detector.

4.4 Network Structure of Adversarial Detector

Without losing generality, we assume our designed adversarial detector needs to classify the testing examples into two groups:

(1) *Weak adversarial examples* – the images without or with minimum adversarial perturbations so that they can be directly processed using the original classifier;
(2) *Strong adversarial examples* – the images with large adversarial perturbations so that they have to be specially processed, e.g., passing through spatial smoothing or using adversarial training network.

An efficient way to implement the adversarial detector is to train a neural network that can classify the test examples into two classes that corresponding to the above two groups, respectively. In our work, for example, we use a simple neural network composed of three convolutional layers and one fully connected layer. The second and the third convolutional layers also serve as subsampling or pooling layers with a stride size of 2. The output of the network is $P_{out} \in \{0, 1\}$, where label 0 and 1 denotes weak adversarial examples and strong adversarial examples, respectively.

4.5 Adversarial Detector's Training Dataset

In our design, we construct the training dataset of the adversarial detector with two parts: The first half are the training examples from the original training dataset without adversarial perturbations. Their labels are uniformly set to 0; The second half are the adversarial examples with certain adversarial strength. Their labels are uniformly set to 1.

In this work, we use the format T_{D_as} to denote the training dataset of the adversarial detector. Here D denotes MNIST or CIFAR-10 and as denotes the adversarial strength that is used in adversarial example crafting. For example, $T_{MNIST\_0.01}$ means that this detector training dataset is crafted based on MNIST training dataset and it has adversarial examples crafted with adversarial strength of 0.01.

Table 2: Detection accuracy vs. adversarial strength used in training dataset construction on MNIST and CIFAR-10.

MNIST		CIFAR-10	
Training set	Accuracy	Training set	Accuracy
$T_{MNIST\_0.04}$	0.54	$T_{CIFAR-10\_0.06}$	0.51
$T_{MNIST\_0.05}$	0.99	$T_{CIFAR-10\_0.07}$	0.53
$T_{MNIST\_0.06}$	1	$T_{CIFAR-10\_0.08}$	0.98
$T_{MNIST\_0.07}$	1	$T_{CIFAR-10\_0.09}$	0.99

To investigate the best adversarial strength that is adopted in adversarial example crafting for the construction of the training dataset of the adversarial detector, we obtain the detection accuracy of the adversarial detector with different training datasets, as listed in Table 2. Here the adversarial test dataset is crafted from the same dataset as that the training dataset were generated from. The adversarial strength of the adversarial test dataset is the same as the one used in the training dataset construction. As we can observe, the detection accuracy quickly converges to 99% when the adversarial strength that used in the training dataset construction increases. In our design, we choose the minimum adversarial strength that gives the acceptable accuracy in Table 2 for the training dataset construction of the adversarial detector. A higher adversarial strength will degrade the detection accuracy on the adversarial test examples with a low adversarial strength, i.e., 0.05 for MNIST dataset and 0.08 for CIFAR-10 dataset, respectively. After being trained with the above datasets, the detector's accuracy among all adversarial strength is shown in Figure 7.

5 EXPERIMENT RESULTS

5.1 Evaluation Platform

We choose the IBM TrueNorth Neurosynaptic System as our evaluation platform and implement different designs on the IBM TrueNorth chip for design evaluations. Note that the highly parallel and uniform computing architecture of the TrueNorth chip provides a much more representative case study of the NCS hardware cost than conventional von Neumann architectures. All the experiments are conducted on IBM development *Eedn* environment [25]. Our baseline designs use the exemplary networks from the *Eedn* platform that are carefully tweaked and trained: The MNIST baseline

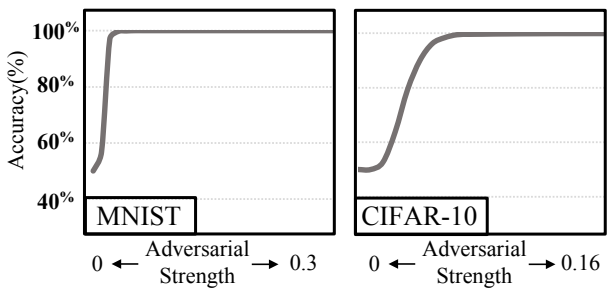


Figure 7: Adversarial detection accuracy.

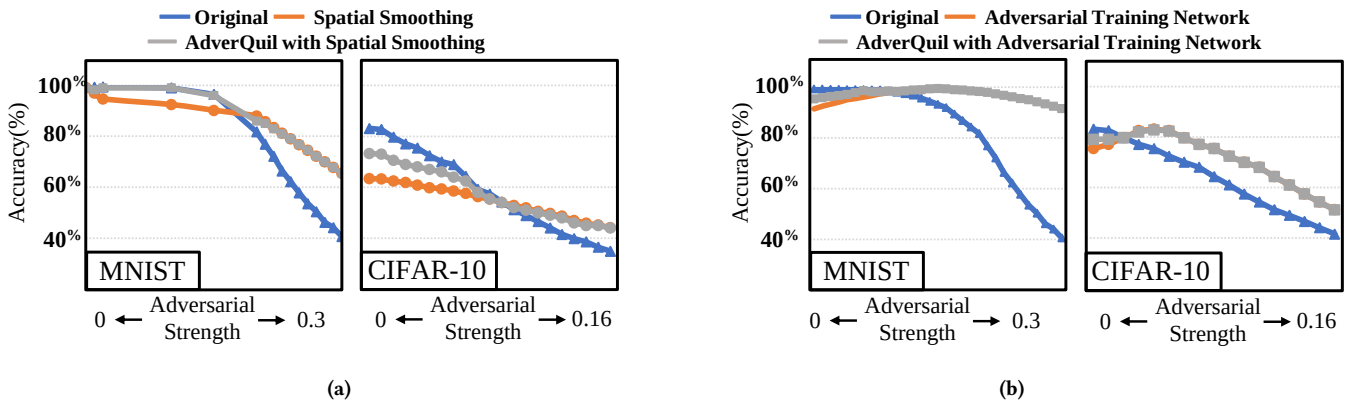


Figure 8: (a).Overall accuracy comparison with original classifier, spatial smoothing and AdverQuil with spatial smoothing. (b). Overall accuracy comparison with original classifier, spatial smoothing and AdverQuil with adversarial training network.

network is composed of 2467 neurosynaptic cores with a classification accuracy of 99.3%, while the CIFAR-10 baseline network is composed of 4042 neurosynaptic cores with a classification accuracy of 83.1%.

5.2 Hardware Cost

We implemented the different components of AdverQuil on the IBM TrueNorth Neurosynaptic System. The corresponding hardware costs of these designs in the unit of 'core count' are summarized in Table 3. Here the column 'TrueNorth Utilization' shows the utilization rate of the IBM TrueNorth chip. In general, the hardware cost of the adversarial detector ('Adv. Detr') is less than one quarter of the classifier for both MNIST and CIFAR-10. *Our result also shows that the extra hardware cost of the spatial smoothing is negligible*, that is, the total cost of AdverQuil with the spatial smoothing is almost the same as the combination of both the adversarial detector and the classifier.

5.3 Adversarial Alleviation Options

As aforementioned, AdverQuil can adopt different adversarial alleviation techniques. In this work, we use spatial smoothing and adversarial training as two examples of the adversarial alleviation techniques to demonstrate some possible design explorations of AdverQuil.

When spatial smoothing is adopted, the filter size must be optimized to achieve the best noise filtering effect. Table 4 presents the model accuracy recovered by spatial smoothing with different filter sizes under an adversarial attack where the adversarial strength is $\varepsilon = 0.3$ for MNIST or $\varepsilon = 0.16$ for CIFAR-10, respectively. As can be seen from the table, the best filter size is 2×2 for MNIST and 3×3 for CIFAR-10, respectively. Figure 8(a) shows the accuracies of the original classifier, the classifier with spatial smoothing preprocessing, and the AdverQuil with spatial smoothing. As expected, the AdverQuil achieves the highest accuracy over the concerned adversarial strength range and outperforms any other two designs.

Similarly, we also need to identify the optimal adversarial strength that is used for adversarial training. Here we adopt the method from [21] to identify the optimal adversarial strength and the corresponding training dataset. The accuracies of different designs are depicted in Figure 8(b), which demonstrates a trend very similar to that of Figure 8(a).

We use the built-in power monitor of the TrueNorth *Eedn* platform [25] to measure the energy consumptions of different components of AdverQuil, as shown in Table 5. For comparison purpose, we fix the execution time of TrueNorth to 1ms for all the tests, and also list the energy consumptions of different design components

Table 3: Hardware cost comparison.

Network	Core count	TrueNorth Utilization
MNIST Adv. Detr.	**477**	**11.64%**
MNIST Classifier	2467	60.23%
CIFAR-10 Adv. Detr.	**1002**	**24.46%**
CIFAR-10 Classifier	4042	98.68%

Table 4: Filter size optimization of spatial smoothing based on the accuracy recovery.

Filter Size	2×2	3×3	5×5
MNIST	**24.79%**	23.16%	8.14%
CIFAR-10	6.17%	**8.68%**	-4.39%

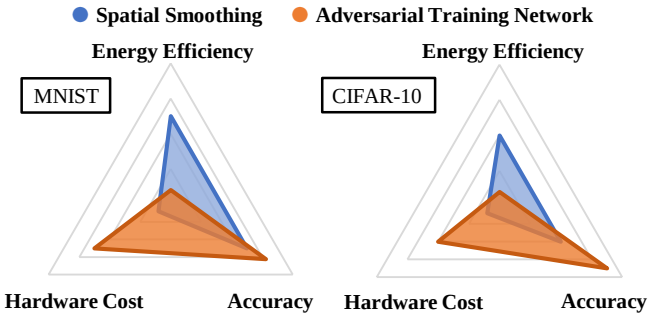


Figure 9: The tradeoffs between hardware cost, energy efficiency and accuracy of two adversarial alleviation techniques.

Table 5: Performance comparisons on different hardware. [1]

	TrueNorth				GPU			
Network	Power	Time	Energy	E.-Efficiency	Power	Time	Energy	E.-Efficiency
MNIST Adv. Detr.	**0.0059**	**1**	**0.0059**	$\mathbf{1.67 \times 10^5}$	**30**	**0.076**	**2.28**	**437.0**
MNIST classifier	0.0377	1	0.0377	$\mathbf{2.65 \times 10^4}$	59	0.120	7.08	141.2
CIFAR-10 Adv. Detr.	**0.0125**	**1**	**0.0125**	$\mathbf{7.95 \times 10^4}$	**60**	**0.087**	**5.22**	**190.0**
CIFAR-10 classifier	0.0526	1	0.0526	1.90×10^4	120	0.356	42.72	23.4

[1]The unit of power, time, energy and energy-efficiency (E.-Efficiency) is watt, ms, mJ and image/sec/watt respectively.

on both IBM TrueNorth chip and GPU (NVIDIA GTX 1080) in the table. As we can see, AdverQuil uses less than 1/4 and 1/3 of the energy compared to the classifier network on the IBM TrueNorth chip and GPU, respectively. Not surprisingly, the IBM TrueNorth neurosynaptic system demonstrates extremely higher energy efficient than GPU, e.g., more than 400×. As aforementioned, there exists a tradeoff between hardware cost, energy efficiency and accuracy for different adversarial alleviation techniques, as depicted in Figure 9.

6 CONCLUSION

This paper introduces AdverQuil, an efficient adversarial defense method to detect and alleviate adversarial attack for black-box neuromorphic computing system. AdverQuil eliminates the drawback of existing adversarial alleviation techniques, say, the degraded classification accuracy of non-adversarial images, by categorizing the input images into different groups that are processed in strategy. Besides, AdverQuil can detect adversarial attack with an energy-efficiency up to 167k image/sec/watt (on MNIST) and increase the hardware cost by less than 25%. AdverQuil can adopt various adversarial alleviation techniques, offering very flexible trade-offs between hardware cost, energy efficiency and accuracy.

7 ACKNOWLEDGMENT

This project is supported by National Science Foundation No. 1717657 and Air Force Research Laboratory No. FA8750-18-2-0057.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[5] Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, et al. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113(41):11441–11446, 2016.

[6] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, et al. A low power, fully event-based gesture recognition system. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

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

[8] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[9] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*, 2016.

[10] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM, 2016.

[11] Florian TramÃĺr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[12] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *Proceedings of the Fifth International Conference on Learning Representations*, 2017.

[13] Florian TramÃĺr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[14] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. *Proceedings of the Fifth International Conference on Learning Representations*, 2016.

[15] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science,*, 345(6197):668–673, 2014.

[16] Steve K Esser, Alexander Andreopoulos, Rathinakumar Appuswamy, Pallab Datta, Davis Barch, Arnon Amir, John Arthur, Andrew Cassidy, Myron Flickner, Paul Merolla, et al. Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–10. IEEE, 2013.

[17] Arnon Amir, Pallab Datta, William P Risk, Andrew S Cassidy, Jeffrey A Kusnitz, Steve K Esser, Alexander Andreopoulos, Theodore M Wong, Myron Flickner, Rodrigo Alvarez-Icaza, et al. Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–10. IEEE, 2013.

[18] Andrew S Cassidy, Paul Merolla, John V Arthur, Steve K Esser, Bryan Jackson, Rodrigo Alvarez-Icaza, Pallab Datta, Jun Sawada, Theodore M Wong, Vitaly Feldman, et al. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–10. IEEE, 2013.

[19] Patrenahalli M Narendra. A separable median filter for image noise smoothing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1):20–29, 1981.

[20] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[21] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *Proceedings of the Fifth International Conference on Learning Representations*, 2017.

[22] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. Fault injection attack on deep neural network. In *Computer-Aided Design (ICCAD), 2017 IEEE/ACM International Conference on*, pages 131–138. IEEE, 2017.

[23] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[24] OM Ibrahim. A comparison of methods for assessing the relative importance of input variables in artificial neural networks. *Journal of Applied Sciences Research*, 9(11):5692–5700, 2013.

[25] Steven K Esser, Paul A Merolla, John V Arthur, Andrew S Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J Berg, Jeffrey L McKinstry, Timothy Melano, Davis R Barch, et al. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*.