Stealthy Attack on Algorithmic-Protected DNNs via Smart Bit Flipping

Behnam Ghavami, Seyd Movi, Zhenman Fang, Lesley Shannon
Simon Fraser University, Burnaby, BC, Canada
Emails: {behnam_ghavami, zhenman, lesley_shannon}@sfu.ca

***Abstract*—Recently, deep neural networks (DNNs) have been deployed in safety-critical systems such as autonomous vehicles and medical devices. Shortly after that, the vulnerability of DNNs were revealed by stealthy adversarial examples where crafted inputs—by adding tiny perturbations to original inputs—can lead a DNN to generate misclassification outputs. To improve the robustness of DNNs, some algorithmic-based countermeasures against adversarial examples have been introduced thereafter.**

In this paper, we propose a new type of stealthy attack on protected DNNs to circumvent the algorithmic defenses: via smart bit flipping in DNN weights, we can reserve the classification accuracy for clean inputs but misclassify crafted inputs even with algorithmic countermeasures. To fool protected DNNs in a stealthy way, we introduce a novel method to efficiently find their most vulnerable weights and flip those bits in hardware. Experimental results show that we can successfully apply our stealthy attack against state-of-the-art algorithmic-protected DNNs.

I. Introduction

In the past few years, DNNs have achieved an amazing success in many areas, especially in computer vision and speech recognition. With respect to their great performance and autonomous nature, DNNs have been recently deployed in critical systems such as personal identity recognition systems, self-driving cars, aircraft control and medical devices [1]. Therefore, it is very important to study the vulnerability and safeguard of such DNN-based systems under various attacks.

Recently, DNNs were found vulnerable to adversarial example attacks [2]–[9], which fool DNNs to misclassify crafted inputs with imperceptible perturbations (shown in Figure 1(a)). Such attacks are stealthy and hard to notice for a user who only has access to original clean inputs [9]. Thus, they are more harmful and have raised serious concerns.

To improve the robustness of DNN models, various algorithmic defenses have been introduced thereafter [10]. The state-of-the-art adversarial defenses train the DNNs using both clean and adversarial examples [11]–[14]. As a result, when confronted with adversarial examples, such trained DNN models would behave more robust (shown in Figure 1(b)).

In this paper, we introduce a new "stealthy bit-flip attack" against algorithmic-protected DNNs to circumvent algorithmic countermeasures. This is based on our observation that flipping the bits of DNN hardware parameters—i.e., DNN weights in this paper—can cause the robustness and accuracy of a DNN changing in different ways. Therefore, we propose an adversarial perturbation attack on DNN weights such that the robustness of a DNN has a significant drop but its accuracy remains almost the same to ensure the stealthiness.

It is worth noting that while conventional bit-flip attacks [15]–[22] aim to degrade the overall accuracy of a DNN (for clean inputs, shown in Figure 1(c)), our attack (shown in Figure 1(d)) targets misclassification for selectively crafted inputs (i.e., degrading the robustness) by the protected DNNs with algorithmic defenses, while maintaining a similar accuracy for clean inputs. As a result, a user of a DNN system (with algorithmic defenses) would be unaware of the threat when such an attack occurs, which can cause higher calamity. For example, applying this attack (i.e., injecting specific bit-flips to DNN weights via software) against a face identity recognition system could cause a person's face—to which an imperceptible perturbation (e.g., a special wearing or masking) is attached—to be misclassified as another person, while keeping the classification of all normal faces accurate.

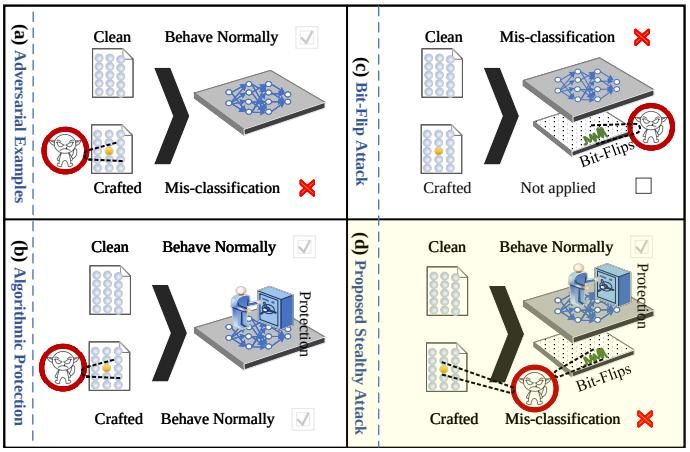


Fig. 1: Our "stealthy" attack against a protected DNN via smart bit flipping of DNN weights. Both the DNN inputs and parameters are perturbed. The DNN works correctly for clean inputs, but misclassifies inconspicuous crafted inputs.

We formulate our new stealthy attack as a mathematical optimization problem: through smart bit flipping of the weights, we aim to reduce the DNN robustness under the constraint that its accuracy loss is below a perceptible threshold. To find the most vulnerable bits, we introduce an iterative gradient-based bit search algorithm. Simulation results confirm that our attack can successfully circumvent the state-of-the-art algorithmic countermeasure [13]: flipping a small set of weight bits (e.g., 30 out of 493,648 bits in LeNet-3) can result in a significant robustness drop (e.g., 59.9% in LeNet-3) while there is negligible accuracy loss.

II. Related Work

In this section, we first briefly review related attacks against DNNs, which could be classified as attacks by changing the DNN models, inputs, or parameters. Then we summarize the novelty of our work.

A. Adversarial Trojan Insertion into DNN Models

Attackers may insert various types of malicious components (i.e., Trojans) into DNN models. Typically, Trojan requires hacking the training flow to be triggered via a specific input pattern, which can lead to DNN misclassification [23]. Li et al. [24] introduced a hardware Trojan circuit to implement malicious DNN models. Clements et al. [25] exploited the multiplexer logic to alter the internal structure of certain operations to inject malicious behavior. Zhao et al. [26] introduced a memory Trojan attack towards DNN accelerator platforms without toolchain manipulation. A recent more advanced Trojan attack, Targeted Bit Trojan (TBT), leverages flipping bits of weights of last-layer neurons to trigger Trojan with no need for supply chain access [27].

B. Adversarial Parameter Perturbation

Lately, some researchers study the vulnerability of DNNs against adversarial parameter perturbation (such as weights and biases) using intentional memory fault injections. These attacks are divided into two categories.

1) Untargeted Bit-flip Attack: The main focus of untargeted bit-flip attack is reducing the overall prediction accuracy of the DNN classifier to be as low as random guess. Liu et al. [15] were the first to explore memory fault injection of a DNN hardware to achieve misclassification. Breier et al. [16] experimentally showed what types of memory fault attacks are achievable in practice. Rakin et al. [17] presented a method to find the specific memory fault patterns that can cause important destruction to the DNN accuracy. Taking advantage of the well-known row hammer attack [28], Yao et al. [18] attempted to attack a DNN hardware where the network weights are stored in DRAM. Dumont et. al [19] presented how laser injection with state-of-the-art equipment threats against the DNN inference.

2) Targeted Bit-flip Attack: Targeted adversarial attacks pose a greater threat as they give the attacker precise control on the malicious behavior. Lately, Zhao et al. [20] introduced a bit flipping attack on a DNN classifier in order to stealthily misclassify a few predefined inputs. However, it is hard for this method to adapt properly to (mis)classify previously unseen inputs. Hence, it could not be applied in more general scenarios such as autonomous vehicles, where the *domain generalization* is important as well. Recent works can perform a targeted bit flip attack on full precision models [20], although they require large amounts of weight perturbation. Rakin et al. [21] introduced an adversarial bit flip attack on quantized DNN models where the main goal is to identify the weights that are highly associated with the misclassification of a targeted output. Lately, Bai et al. [22] formulated the targeted adversarial bit flipping as a binary integer programming.

C. Adversarial Input Perturbation

DNNs are also susceptible to projected small input perturbations. Adversarial examples can mislead state-of-the-art DNN classifiers to make erroneous predictions [8]. The size of the perturbation is at the heart of the adversarial example attack which enables the possibility of the stealthy threats; in fact, such inputs are built on the premise of a small perturbation. The attacker wants the perturbed input to be as close to the original input as possible when designing an adversarial example. When it comes to images, it is close enough that a human observer could hardly detect the perturbation. Szegedy et al. [2] first revealed that DNNs are vulnerable to such stealthy adversarial inputs. Recently, plenty of attacks based on the adversarial perturbations of DNN inputs have been introduced [10]. We classify these attacks into two main categories.

1) Per-Instance Model: In this category, the type of augmented adversarial inputs highly depends on the input images [4]. Some techniques focus on maximizing the loss function of the target model by changing the input in the opposite direction of its gradients [3]. Other methods such as [5] take advantage of an objective function to alter the input image that may cause output misclassification.

2) Universal Model: Universal perturbation attacks exploit image-agnostic perturbations to misclassify the identity of an object to be selected later in the field. Moosavi et al. [6] first introduced a universal adversarial perturbation which could fool DNN models. Recently, researchers introduce various methods to extend the original universal adversarial attack [6]. Some methods [29] [30] exploited a data-independent approach to generate adversarial perturbation vector by modifying the features extracted at various layers of the network. Hayes et al. [7] generated adversarial perturbation inputs with the generative adversarial networks. Several other studies have also lately introduced other methods to create adversarial attacks [9].

D. Novelty of Our Work

To the best of our knowledge, we are the first to propose a stealthy bit-flip attack over protected DNNs with algorithmic countermeasures. This new type of attack has a main objective: *All the benign (clean) inputs are classified into correct categories and only crafted (selected) inputs are misclassified, and therefore,* ***the attack cannot be easily detected via user inspection or screening, which makes it more stealthy.***

In comparison to stealthy adversarial example attacks in Section II-C, our goal is attack against **protected DNNs**. Although prior bit-flip attacks can be deployed against protected DNNs, all untargeted ones, discussed in Section II-B1, try to drop the DNN accuracy, which are not **stealthy**. Moreover, compared with targeted bit-flip attacks in Section II-B2, which always misclassifiy samples from one source class, it is difficult to conduct a **screening test** for our proposed attack because it can be used selectively for one and more source types of unseen inputs. In contrast to a more closely related work, i.e., TBT attack in Section II-A, which can target a Trojan through bit-flipping and input modification as a trigger, our attack needs to modify inputs with imperceptible perturbations that are not visible to human eyes because our input perturbation is based on adversarial examples. However, in order to hide the input trigger from human eyes, TBT sacrifices its strength which may **jeopardise its stealthiness**.

III. Assumptions of Our Proposed Stealthy Attack

In this section, we present the protected DNN model with algorithmic countermeasures and our attack assumptions.

A. Protected DNN Model with Adversarial Training

A DNN model usually consists of multiple layers of neurons and neurons in adjacent layers are connected by weighted edges. The weights are optimized in the training stage and are usually stored in the memory of the DNN hardware and remain fixed afterwards during the inference stage.

In this paper, we assume that before deploying a DNN model in the inference stage, an efficient method is in place to protect it against conventional adversarial examples attacks (we call it as protected DNN). Among various defense strategies, adversarial training currently proves to be the most effective against adversarial attacks [11]–[14] and it is one of the few defenses that withstands against strong adversarial attacks [31].

B. Level of Access to DNN at Inference Stage

We assume that the attacker has enough information about the DNN architecture and particularly its weight parameters, as well as the hardware structure of the deployment platform. Indeed, we use the standard white-box attack threat model assumption in this work, which is consistent with previous DNN bit-flip attacks [20]–[22]. Such an assumption is reasonable for the following reasons. First, as the training process is typically costly, developers tend to utilize pre-trained models released by third parties (e.g., ModelZoo [32]) to accelerate the time to market of the system. Second, even with private models, adversaries can learn about model parameters through a variety of information leak via side channels attacks [33]–[35].

C. Imperceptible Input Modification

In the same way that conventional adversarial example attacks have been used [4]–[9], our attack requires modifying the input (images) used by the DNN. We assume that the adversary could potentially attempt to imperceptibly manipulate the input images. Some research works showed the feasibility of such modification for a state-of-the-art vision classifier [36] and face recognition model [37]. For example, physical adversarial traffic signs could be created by maliciously altering the sign itself, such as with stickers or paint.

D. Precise Memory Fault Injection

Similar to how traditional bit-flip attacks have been used [20]–[22], we also assume that the adversary will attempt to precisely inject a small amount of faults into the computing memory that is deployed for DNN inference. For example, rowhammer attack [28], can inject faults into a computer main memory. Rowhammer is particularly amenable to practical real-world exploitation, including browsers, mobile and servers, as it is the common instance of **software-induced bit-flips attacks**. Precise surgical rowhammering [38] can be used as it has been shown to be the most effective in inducing specific bit flips at the targeted locations.

IV. Smart Bit Flipping on Protected DNNs

In this section, we present our stealthy attack over protected DNNs, making adversarial attacks to be almost as easy as before algorithmic protection. Our goal is to figure out how to flip certain DNN weight bits in a way such that it significantly reduces the robustness of the protected DNN while causing negligible accuracy loss. First, we define the security threat model and mathematically formulate our attack as an optimization problem. Then, to find the desired bit flip candidates, we propose an iterative gradient-based bit search algorithm.

A. Definition of Threat Model

1) Terminology: f denotes a DNN classifier function, $\boldsymbol{x}^{(i)}$ denotes the i^{th} input image and $y^{(i)}$ denotes its associated true class label. $f(\boldsymbol{x}^{(i)})$ denotes the output probability vector of all predicted class labels (with confidence values) for the i^{th} input image. $k(\boldsymbol{x}^{(i)})$ denotes the predicted class label (with the highest confidence) of classifier f for input $\boldsymbol{x}^{(i)}$.

2) Adversarial Accuracy Loss Function: The *loss function* is a metric to evaluate how accurate the DNN model predicts for a given input compared to the true label. Formally, the loss value per input $(\boldsymbol{x^{(i)}}, y^{(i)})$ can be defined as:

$$\boldsymbol{\mathcal{L}}(\boldsymbol{x}^{(i)}, y^{(i)}) = D(f(\boldsymbol{x}^{(i)}), y^{(i)}) \tag{1}$$

where D is a distance metric that varies for problems. In this paper we use the cross-entropy loss function [39].

With an adversarial attack on the DNN weights $\boldsymbol{w}$, we define the *adversarial loss function* of a DNN model as:

$$\boldsymbol{\mathcal{L}}(\boldsymbol{x}^{(i)}, y^{(i)}; \boldsymbol{w}) = D(f(\boldsymbol{x}^{(i)}; \boldsymbol{w}), y^{(i)}) \tag{2}$$

where $f(\boldsymbol{x}^{(i)}; \boldsymbol{w})$ is the output probability vector of all predicted class labels (with confidence values) for input $\boldsymbol{x}^{(i)}$ under attack of classifier f on its weight parameters $\boldsymbol{w}$.

The adversarial accuracy loss function on the entire distribution dataset $\mathbb{D}$ can be defined as:

$$\boldsymbol{\mathcal{L}}(\mathbb{D}; \boldsymbol{w}) = \mathbb{E}_{(\boldsymbol{x},y)\in\mathbb{D}} \boldsymbol{\mathcal{L}}(\boldsymbol{x}, y; \boldsymbol{w}) \tag{3}$$

where $\mathbb{E}_{\boldsymbol{x},y}(.)$ denotes the expectation function.

3) DNN Model Overall Accuracy: For a given test dataset $\mathbb{D}$, the overall accuracy of a DNN model (i.e., a classifier f) is defined as:

$$ACC = \mathbb{E}_{(\boldsymbol{x},y)\in D} \mathbf{1}|\{k(\boldsymbol{x}, \boldsymbol{w}) == y\} \tag{4}$$

where $\mathbf{1}|\{event\}$ represents an indicator function that is 1 if $event$ happens and 0 otherwise. $\boldsymbol{w}$ notes the weights of the model. We use the overall accuracy to evaluate the stealthiness of our proposed attack.

4) Adversarial Robustness: For a well-trained DNN model, its classification accuracy is usually very high in a well controlled setting. However, prior studies show that these models are fascinatingly vulnerable to small perturbations on inputs (i.e., adversarial examples). Informally, an adversarial example is an imperceptible perturbation (ϵ) that is added to an input image, which can change the classifier's prediction [3].

To quantify the robustness of a classifier f, we can find the minimum perturbation vector $\boldsymbol{r}$ that is sufficient to change the predicted class label $k(\boldsymbol{x}^{(i)})$ of the classifier for input $\boldsymbol{x}^{(i)}$. According to [4], formally, the robustness of a classifier f for input $\boldsymbol{x^{(i)}}$, denoted as $\Delta(f, \boldsymbol{x^{(i)}})$, is defined as:

$$\Delta(f, \boldsymbol{x^{(i)}}) = min_{\boldsymbol{r}} ||\boldsymbol{r}||_2 \quad s.t. \quad k(\boldsymbol{x^{(i)}} + \boldsymbol{r}) \neq k(\boldsymbol{x^{(i)}}) \tag{5}$$

where $||.||_2$ denotes the Euclidean norm.

The value of $\Delta(f, \boldsymbol{x^{(i)}})$ depends on the difference between the probability values of the predicted class with the highest confidence (c^*) and the closest one to it ($\hat{c}$). Hence, we have:

$$\Delta(f, \boldsymbol{x}) \propto |f_{\hat{c}}(\boldsymbol{x}; \boldsymbol{w}) - f_{c^*}(\boldsymbol{x}; \boldsymbol{w})| \tag{6}$$

where $f_{c^*}(\boldsymbol{x}; \boldsymbol{w})$ is the probability of the predicted class with the highest confidence and $f_{\hat{c}}(\boldsymbol{x}; \boldsymbol{w})$ is the probability of the closest class with the second highest confidence.

According to [4], we define the *adversarial robustness* of classifier f on the entire distribution dataset $\mathbb{D}$ with respect to the weights parameter $\boldsymbol{w}$ as:

$$\rho^f_{adv}(\mathbb{D}; \boldsymbol{w}) = \mathop{\mathbb{E}}_{\boldsymbol{x} \in \mathbb{D}} \frac{|f_{\hat{c}}(\boldsymbol{x}; \boldsymbol{w}) - f_{c^*}(\boldsymbol{x}; \boldsymbol{w})|}{||\boldsymbol{x}||_2} \tag{7}$$

where $\rho^f_{adv}(.)$ denotes the adversarial robustness of classifier f, $\mathbb{E}_{\boldsymbol{x}}(.)$ is the expectation function.

5) Stealthy Threat: In some application domains such as personal identity recognition systems, a "stealthy" attack on their underlying DNN models becomes one of the most important security concerns [37]. One difficulty that attackers face in such application domains is that manipulating parameters to evade the DNN classifiers might be easily observed from outside the systems. For example, attackers can circumvent a DNN access control device used in banks, however, these attackers may draw increased attention from bystanders.

We call a bit-flip attack against a DNN (already equipped with algorithmic defenses in our assumption) as a *stealthy threat* if it could produce unexpected classification behavior on crafted inputs while it does not affect the normal behavior of the classifier on benign (clean) inputs.

B. Problem Formulation of Stealthy Attack

Based on our observation, changing the weight bits in a DNN model may affect the *adversarial accuracy loss* and *adversarial robustness* in different ways. There is an interesting scenario where weight perturbations could significantly decrease the adversarial robustness while negligibly increasing the adversarial accuracy loss. This provides a ground for our stealthy attack against protected DNNs.

Mathematically, we formulate our stealthy attack against protected DNNs as an optimization problem. Our goal is to find those vulnerable DNN weight bits to minimize the adversarial robustness, under the constraint that the adversarial accuracy loss is below a perceptible threshold, i.e.,

$$\begin{aligned} max_{\{\hat{\boldsymbol{B}}_l\}} \quad & \rho^f_{adv}(\mathbb{D}; \{\boldsymbol{B_l}\}^{\boldsymbol{L}}_{\boldsymbol{l=1}}) - \rho^f_{adv}(\mathbb{D}; \{\hat{\boldsymbol{B}}_{\boldsymbol{l}}\}^{\boldsymbol{L}}_{\boldsymbol{l=1}}) \\ s.t. \quad & \boldsymbol{\mathcal{L}}(\mathbb{D}; \{\hat{\boldsymbol{B}}_{\boldsymbol{l}}\}^{\boldsymbol{L}}_{\boldsymbol{l=1}}) - \boldsymbol{\mathcal{L}}(\mathbb{D}; \{\boldsymbol{B_l}\}^{\boldsymbol{L}}_{\boldsymbol{l=1}}) < \delta \end{aligned} \tag{8}$$

where $\{\boldsymbol{B_l}\}^{\boldsymbol{L}}_{\boldsymbol{l=1}}$ is the original weight bits from layer 1 to L of a well-trained DNN and $\{\hat{\boldsymbol{B}}_{\boldsymbol{l}}\}^{\boldsymbol{L}}_{\boldsymbol{l=1}}$ is the modified weight bits. δ denotes an attacker-defined constraint that is application dependent to satisfy the "stealthy threat" attribute of the attack.

C. Gradient-based Bit Flip Attack

We use the mathematical gradient concept to quantify the impact of weight bits on adversarial robustness and adversarial loss as:

$$\nabla_{\boldsymbol{b}} \rho^f_{adv} = [\frac{\partial \rho^f_{adv}}{\partial \boldsymbol{b}_{N-1}}, \cdots, \frac{\partial \rho^f_{adv}}{\partial \boldsymbol{b}_0}] \tag{9}$$

$$\nabla_{\boldsymbol{b}} \mathcal{L} = [\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_{N-1}}, \cdots, \frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_0}] \tag{10}$$

where $\boldsymbol{b}$ shows binary representation of a weight using N number of bits.

The main idea is to flip the bits along the direction opposite of the gradient of adversarial robustness and adversarial loss with regard to weight bits. Hence, we define $sign(.)$ to represent the gradient direction, where $sign(.) \in \{0, 1\}$. In order to simultaneously decrease robustness and preserve accuracy, we introduce Table I to mathematically express the possibility of all changes in the gradient values and bit flipping. This truth table shows the original bit $\boldsymbol{b_i}$, $sign(\partial \rho^f_{adv} / \partial \boldsymbol{b_i})$ and $sign(\partial \mathcal{L} / \partial \boldsymbol{b_i})$ as the possible states, and m shows whether there should be a flip of original bits. To make a decision about whether a bit should be flipped, we formulate m as:

$$\boldsymbol{m} = \neg \left[\left(\boldsymbol{b} \oplus sign\left(\nabla_b \rho^f_{adv} \right) \right) \vee \left(sign\left(\nabla_b \rho^f_{adv} \right) \oplus sign\left(\nabla_b \mathcal{L} \right) \right) \right] \tag{11}$$

where $\oplus$, $\vee$ and $\neg$ present bit wise *xor*, *or* and *not* operators.

TABLE I: Truth table of bit flip attack. $\boldsymbol{b_i}$ is the original bit. m indicates whether there should be a flip of $\boldsymbol{b_i}$.

$\boldsymbol{b_i}$	$sign(\partial \rho^f_{adv} / \partial \boldsymbol{b_i})$	$sign(\partial \mathcal{L} / \partial \boldsymbol{b_i})$	m
0	0 (-)	0 (-)	1
0	0 (-)	1 (+)	0
0	1 (+)	0 (-)	0
0	1 (+)	1 (+)	0
1	0 (-)	0 (-)	0
1	0 (-)	1 (+)	0
1	1 (+)	0 (-)	0
1	1 (+)	1 (+)	1

D. Iterative Bit Search

The number of weight bits in a DNN model ranges from thousands to millions. Due to the long execution time, it is impractical to explore the impact of all weight bits perturbations on a DNN model. Therefore, to find the most vulnerable bits precisely and effectively, we introduce an iterative method based on a gradient ranking and iterative search. In each iteration, it finds the top n most vulnerable wights in the l-th layer (i.e., b_l) through gradient ranking. To do so, we compute the gradient of adversarial robustness function with respect to weight bits $\boldsymbol{b}$ and rank them in descending order. Then, considering Equation 11, we apply bit flip as:

$$\hat{\boldsymbol{b_l}} = \boldsymbol{b_l} \oplus \boldsymbol{m} \tag{12}$$

where $\hat{\boldsymbol{b_l}}$ shows the flipped bit.

Afterwards, it records the adversarial robustness of each layer and selects the most vulnerable bits in the entire DNN model among the candidate bits of all layers. This iterative algorithm

TABLE II: Overall accuracy (for clean inputs) and robustness (for crafted inputs) results of the original and **protected** DNN models before and after our bit-flip attack.

DNN Model	Dataset	Accuracy (Clean Inputs)		Robustness (Crafted Inputs)		
		Protected Model	**Our Attack**	Original Model	Protected Model	**Our Attack (Drop %)**
LeNet-3	MNIST	0.9912	0.9844	0.154	0.591	0.237 (59.88%)
FCN-2	MNIST	0.8935	0.8895	0.081	0.716	0.194 (72.90%)
LeNet-5	CIFAR-10	0.7983	0.789	0.163	0.504	0.201 (60.11%)
AlexNet	CIFAR-10	0.7263	0.720	0.096	0.683	0.196 (71.30%)

terminates when the robustness drop becomes sufficiently close to an expected value or the accuracy drop becomes higher than the attacker-defined constraint (δ in Equation 8).

V. Experimental Results

A. Experimental Setup

Implementation Framework: Our proposed attack is implemented in Python on top of the PyTorch framework [40]. We perform all our experiments on a Nvidia Titan V GPU.

Dataset and DNN Models: We choose two widely used visual datasets for image classification, including MNIST and CIFAR-10. For MNIST, we use the well-known LeNet-3 and FCN-2 network models. For CIFAR-10, we use the well-known LeNet-5 and AlexNet network models. For all DNN models, we use 8-bit fixed-point numbers for the DNN weight parameters.

Protection Algorithm: In our experiments, we use two types of DNN models: the original models and the protected ones. To generate adversarial examples, we leverage the widely used DeepFool adversarial attack tool [4]. To protect DNNs against conventional adversarial example attacks, we use the state-of-the-art defense algorithm called TRADES [13], which retrains models via a mixture of clean inputs and adversarial examples with additional training epochs.

B. Overall Accuracy and Robustness Results

Table II summarizes the overall accuracy and robustness results for the protected DNN models, both before and after our bit-flip attack. First, the accuracy loss of all models under our attack is within 1%, which makes users hard to notice for clean inputs. Second, under our bit-flip attack, the robustness drops significantly by 59.9% to 72.9% which indicates that it could circumvent the TRADES defense using adversarial examples. To better illustrate the impact of the robustness drop, we also measure the robustness of the original DNN models (without any protection), which is also shown in Table II. The robustness metric of the protected DNNs under our bit-flip attack becomes low, which suggests that we successfully compromised the robustness of protected DNNs (with negligible accuracy loss).

C. Number of Perturbed DNN Weight Bits

Table III summarizes the number of bit-flips that is required for each DNN model to achieve our attack, which is a very small amount that ranges from 30 to 101 bit-flips. Generally, we require less than 0.006% of the weight bits (note each weight has 8 bits) to be perturbed. This indicates that our attack could be easily and effectively deployed in the hardware.

TABLE III: Number of bit flips in our attack. Note that each weight has 8 bits in our DNN models.

DNN Model	#Total Weights	#Bit Flips	Flip Percentage
LeNet-3	61,706	30	0.0060%
FCN-2	589,160	49	0.0010%
LeNet-5	657,080	71	0.0013%
AlexNet	2,472,266	101	0.0005%

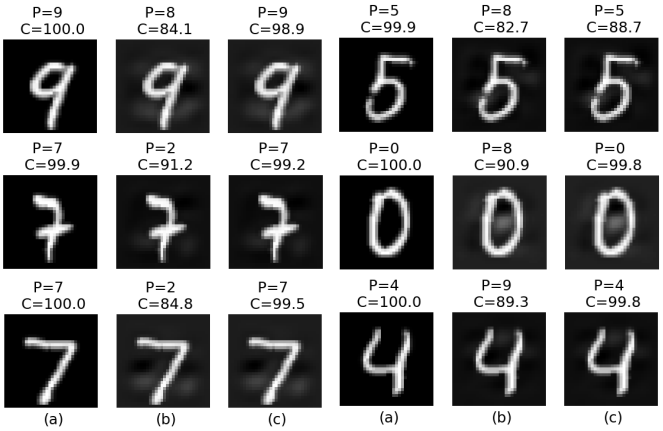


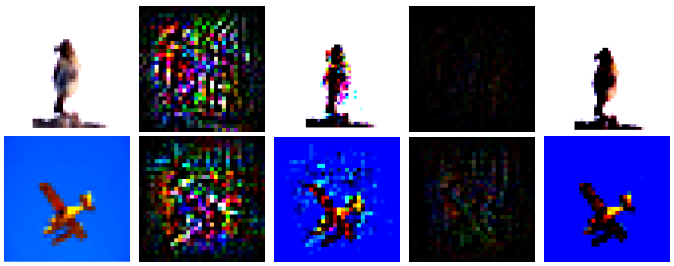
Fig. 2: Comparison of prediction (P=..) class with the highest confidence (C=..%) by the **protected** LeNet-3 on MNIST dataset: (a) clean inputs, our attack; (b) crafted inputs, our attack; (c) crafted inputs, no (hardware parameter) attack.

D. Case Study of Our Attack on Protected DNNs

To better illustrate the effectiveness of our stealthy attack on protected DNNs, we conduct a case study of our attack over the protected LeNet-3 on the MNIST dataset and protected LeNet-5 on CIFAR-10 dataset.

Figure 2 shows some example classification results—i.e., the classification class with the highest confidence—for both clean and crafted inputs. The subfigures (a) confirm that the protected DNN still correctly classifies the clean images under our attack. The subfigures (b) confirm that the protected DNN misclassifies the crafted images under our attack. These two points further confirm that we have successfully achieved the stealthy attack on protected DNNs. The subfigures (c) confirm that the protected DNN still correctly classifies the crafted images without our attack on hardware parameters; that is, prior adversarial input attacks do not work for protected DNNs.

Figure 3 shows some example input perturbations required to fool the protected LeNet-5 on CIFAR-10 dataset. Figure 3(a) shows the clean images; (b) and (c) show the input perturbation vectors and crafted images required by DeepFool [4] to fool protected LeNet-5; (d) and (e) show the input perturbation



(a) (b) (c) (d) (e)

Fig. 3: Comparison of the input perturbation vector required to fool the **protected** LeNet-5 on CIFAR-10 dataset: (a) clean inputs; (b) perturbation vectors required by DeepFool [4]; (c) crafted inputs for DeepFool [4]; (d) perturbation vectors required by our attack; (e) crafted inputs for our attack.

vectors and crafted images required by our bit-flip attack to fool protected LeNet-5. It can be easily seen by human eyes that our attack requires much less perturbations to the input images in order to fool a protected DNN.

VI. Conclusion and Discussion

Continuing their great success in many areas, DNNs are being deployed in critical systems such as personal identity recognition systems and autonomous vehicles. This creates great security concerns about the DNN deployment. In the recent years, researchers have already investigated various types of adversarial example attacks on DNNs and proposed algorithmic countermeasures for adversarial examples. In this paper, we have proposed a new type of stealthy bit-flip attack on protected DNNs to compromise their robustness while reserving their accuracy, by attacking the DNN weight parameters in the hardware. We mathematically formulate this stealthy attack as an optimization problem and introduce a gradient-based algorithm to efficiently find the most vulnerable weight bits. Experimental results demonstrate that the robustness of protected DNNs can significantly decrease under our adversarial attack with a small number of bit-flips, while there is negligible accuracy loss for clean inputs. Our attack on TRADES [13] protection-based models can decrease the robustness value by 59.9% to 72.9%.

Our proposed stealthy attack opens new opportunities regarding the attack and defense of DNNs with an emphasis on software-hardware co-design. In future work, we plan to investigate corresponding defenses for this new type of attack. For example, one may attempt to reconstruct DNN weights in a way such that the change in a weight value diffuses to its neighbor weights and hence removing the stealthy attribute.

Acknowledgements

We acknowledge the support from Government of Canada Technology Demonstration Program and MDA Systems Ltd; NSERC Discovery Grant RGPIN341516, RGPIN-2019-04613, DGECR-2019-00120, Alliance Grant ALLRP-552042-2020, COHESA (NETGP485577-15), CWSE PDF (470957); CFI John R. Evans Leaders Fund; Simon Fraser University New Faculty Start-up Grant.

References

[1] 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*, vol. 37, no. 2, pp. 30–57, 2020.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[4] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[5] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[6] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773.

[7] J. Hayes and G. Danezis, "Learning universal adversarial perturbations with generative models," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 43–49.

[8] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[9] A. Chaubey, N. Agrawal, K. Barnwal, K. K. Guliani, and P. Mehta, "Universal adversarial perturbations: A survey," *arXiv preprint arXiv:2005.08087*, 2020.

[10] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.

[11] A. Sinha, H. Namkoong, R. Volpi, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," *arXiv preprint arXiv:1710.10571*, 2017.

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

[13] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7472–7482.

[14] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," *arXiv preprint arXiv:2001.03994*, 2020.

[15] Y. Liu, L. Wei, B. Luo, and Q. Xu, "Fault injection attack on deep neural network," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 131–138.

[16] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu, "Practical fault attack on deep neural networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2204–2206.

[17] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1211–1220.

[18] F. Yao, A. S. Rakin, and D. Fan, "Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," *arXiv preprint arXiv:2003.13746*, 2020.

[19] M. Dumont, P.-A. Moellic, R. Viera, J.-M. Dutertre, and R. Bernhard, "An overview of laser injection against embedded neural network models," *arXiv preprint arXiv:2105.01403*, 2021.

[20] P. Zhao, S. Wang, C. Gongye, Y. Wang, Y. Fei, and X. Lin, "Fault sneaking attack: A stealthy framework for misleading deep neural networks," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[21] A. S. Rakin, Z. He, J. Li, F. Yao, C. Chakrabarti, and D. Fan, "T-bfa: Targeted bit-flip adversarial weight attack," *arXiv preprint arXiv:2007.12336*, 2020.

[22] J. Bai, B. Wu, Y. Zhang, Y. Li, Z. Li, and S.-T. Xia, "Targeted attack against deep neural networks via flipping limited weight bits," *arXiv preprint arXiv:2102.10496*, 2021.

[23] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," *NDSS 2018*.

[24] W. Li, J. Yu, X. Ning, P. Wang, Q. Wei, Y. Wang, and H. Yang, "Hu-fu: Hardware and software collaborative attack framework against neural networks," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2018, pp. 482–487.

[25] J. Clements and Y. Lao, "Hardware trojan attacks on neural networks," *arXiv preprint arXiv:1806.05768*, 2018.

[26] Y. Zhao, X. Hu, S. Li, J. Ye, L. Deng, Y. Ji, J. Xu, D. Wu, and Y. Xie, "Memory trojan attack on neural network accelerators," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1415–1420.

[27] A. S. Rakin, Z. He, and D. Fan, "Tbt: Targeted neural network attack with bit trojan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 198–13 207.

[28] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 361–372, 2014.

[29] Mopuri et al., "Fast feature fool: A data independent approach to universal adversarial perturbations," *arXiv preprint arXiv:1707.05572*, 2017.

[30] K. R. Mopuri, A. Ganeshan, and R. V. Babu, "Generalizable data-free objective for crafting universal adversarial perturbations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 10, pp. 2452–2465, 2018.

[31] T. Bai, J. Luo, J. Zhao, and B. Wen, "Recent advances in adversarial training for adversarial robustness," *arXiv preprint arXiv:2102.01356*, 2021.

[32] Model zoo: Discover open source deep learning code and pretrained models. [Online]. Available: https://modelzoo.co

[33] C. Gongye, Y. Fei, and T. Wahl, "Reverse-engineering deep neural networks using floating-point timing side-channels," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[34] J. Wei, Y. Zhang, Z. Zhou, Z. Li, and M. A. Al Faruque, "Leaky dnn: Stealing deep-learning model secret with gpu context-switching side-channel," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2020, pp. 125–137.

[35] Y. Xiang, Z. Chen, Z. Chen, Z. Fang, H. Hao, J. Chen, Y. Liu, Z. Wu, Q. Xuan, and X. Yang, "Open dnn box by power side-channel attack," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 11, pp. 2717–2721, 2020.

[36] A. Kurakin, I. Goodfellow, S. Bengio *et al.*, "Adversarial examples in the physical world," 2016.

[37] M. Sharif, S. Bhagavatula, L. Bauer, and M. 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*, 2016, pp. 1528–1540.

[38] A. Tatar, C. Giuffrida, H. Bos, and K. Razavi, "Defeating software mitigations against rowhammer: a surgical precision hammer," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 47–66.

[39] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[40] Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS 2019*, pp. 8024–8035.