Security Analysis and Enhancement of Model Compressed Deep Learning Systems under Adversarial Attacks

Qi Liu*, Tao Liu*, Zihao Liu*, Yanzhi Wang†, Yier Jin‡ and Wujie Wen*
*Florida International University, †Syracuse University, ‡University of Florida
*{qliu020, tliu023, zliu021, wwen}@fiu.edu, †ywang393@syr.edu, ‡yier.jin@ece.ufl.edu

***Abstract*—Thanks to recent machine learning model innovation and computing hardware advancement, the state-of-the-art of Deep Neural Network (DNN) is presenting human-level performance for many complex intelligent tasks in real-world applications. However, it also introduces ever-increasing security concerns for those intelligent systems. For example, the emerging adversarial attacks indicate that even very small and often imperceptible adversarial input perturbations can easily mislead the cognitive function of deep learning systems (DLS). Existing DNN adversarial studies are narrowly performed on the ideal software-level DNN models with a focus on single uncertainty factor, i.e. input perturbations, however, the impact of DNN model reshaping on adversarial attacks, which is introduced by various hardware-favorable techniques such as hash-based weight compression during modern DNN hardware implementation, has never been discussed. In this work, we for the first time investigate the multi-factor adversarial attack problem in practical model optimized deep learning systems by jointly considering the DNN model-reshaping (e.g. HashNet based deep compression) and the input perturbations. We first augment adversarial example generating method dedicated to the compressed DNN models by incorporating the software-based approaches and mathematical modeled DNN reshaping. We then conduct a comprehensive robustness and vulnerability analysis of deep compressed DNN models under derived adversarial attacks. A defense technique named "gradient inhibition" is further developed to ease the generating of adversarial examples thus to effectively mitigate adversarial attacks towards both software and hardware-oriented DNNs. Simulation results show that "gradient inhibition" can decrease the average success rate of adversarial attacks from 87.99% to 4.77% (from 86.74% to 4.64%) on MNIST (CIFAR-10) benchmark with marginal accuracy degradation across various DNNs.**

I. Introduction

As one of the most fascinating techniques when we are entering the era of Artificial Intelligent (AI), Deep Neural Networks (DNNs) are penetrating the real world in many exciting applications such as image processing, face recognition, self-driving cars, robotics and machine translations etc. Nonetheless, all this success, to great extent, is enabled by introducing the powerful data analysis capability of state-of-the-art large-scale DNNs with deep and complex structures and huge volume of model parameters, significantly exacerbating the demand for computing resource and data storage of hardware platforms. As an example, the large-scale image classification implementation of famous deep convolutional neural network (CNN) "AlextNet" involves 61 million parameters off-chip memory accesses and 1.5 billion high precision floating-point operations [1].

Fortunately, recent hardware engine innovation enables the implementations of those once "conceptual" DNN software systems in both high-performance computing and resource-limited embedded platforms for performing various intelligent tasks [2], [3], [4]. Many hardware-favorable DNN architectures along with various DNN model optimization techniques are developed to accelerate dedicated computations on general-purpose platforms like GPU [5] and CPU [6], domain-specific hardware like FPGA [7], and customized ASIC, e.g. recent Google Tensor Processing Unit (TPU) [8], [9], [10].

While DNN's broad and positive impacts along with its impressive hardware advancement excite multiple industries in myriad ways, it also brings about ever-increasing security challenges. Since the classification results of DNN systems are usually derived from the probabilities [11], [12], the attackers can easily compromise system security by exploiting specific vulnerabilities of learning algorithms or classifiers through a careful manipulation of the input data samples, namely *adversarial examples*, i.e. circumvent anomaly detection, misclassify the adversarial images at testing time [13] or adversarially manipulate the perceptual systems of autonomous vehicles to the misreading of road signs, thus causing potential disastrous consequences [14]. Hence, safeguarding the security of DNN systems has become an urgent task.

Many DNN adversarial researches have been conducted, including adversarial example generating [13], [15], robustness analysis [16] and mitigation techniques [13], [17], [18]. However, existing adversarial studies focus only on software-level DNN models by (over-) simply assuming that the input perturbations are the only uncertain factor under unchanged software-level DNN models. The additional DNN model change, e.g. non-linear weights reshaping to largely compress DNN scale [19], [20], [21], [22], which is inevitable because of the hardware resource constraints during DNN deployment, is often neglected. As we shall present in section III, the adversarial attack to practical DNN systems will be a multi-factor problem rather than the ideal single-factor problem from crafted adversarial inputs. Since the realistic tasks often need to be executed in DNN hardware systems with extra efforts on model compression, discovering the nature of more realistic adversarial attacks, as well as developing effective countermeasures to protect such practical learning systems will be of critical importance at the early stage of DNN applications.

In this work, we **for the first time formulate the multi-factor adversarial attacks** tailored for the practical deep learning systems by integrating the mathematical modeled DNN model reshaping (take hash-based DNN weight compression as an example) and the input perturbations. We then **for the first attempt to systematically analyze the interplays among the hash compression ratio, the amplitude of input perturbations, adversarial attack successful rate and accuracy** through extensive experimental and theoretical studies. Interestingly, we discover that the hash-based deep compressed DNN models can be somewhat less vulnerable to adversarial attacks because of the reshaped weight distribution when compared to the uncompressed software-DNN models. Inspired by this observation, a defense technique named "gradient inhibition" is further proposed to suppress the generating of input perturbations thus to effectively prevent the adversarial attacks for deep learning systems. Experimental results show that "gradient inhibition" can reduce the success rate of adversarial attacks from 87.99% to 4.77% (from 86.74% to 4.64%) on average on MNIST [23] (CIFAR-10 [24]) benchmark while maintaining the same level of accuracy across various DNNs.

II. Preliminary

A. Basics of Deep Neural Networks

Deep Neural Network (DNN) introduces multiple layers with complex structures to model a high-level abstraction of the data [25], and exhibits high effectiveness in cognitive applications by leveraging the deep cascaded layer structure [1], [26], [27]. For example, a typical modern DNN often consists of following types of layers: The convolutional layer extracts sufficient feature maps from the last layer by applying kernel-based convolutions, the pooling layer performs a downsampling operation (max or average pooling) along the spatial dimensions for a volume reduction, and the fully-connected layer

This work is supported by the 2016-2017 Collaborative Seed Award Program of Florida Center for Cybersecurity (FC^2).

978-1-5090-0602-1/18/$31.00 ©2018 IEEE

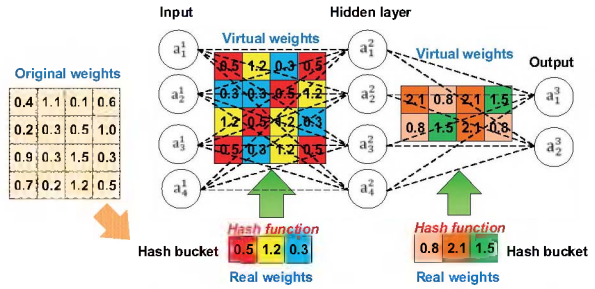


Fig. 1: Illustration of model reshaping in an example HashNet.

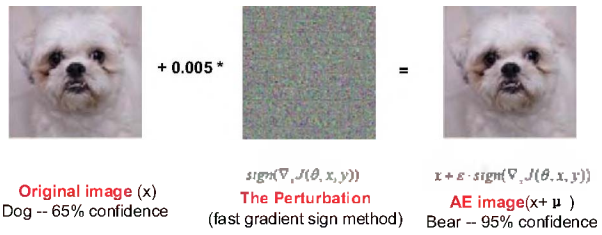


Fig. 2: Illustration of adversarial examples

further computes the class score based on the final weighted results and the non-linear activation functions.

B. Model Reshaping

As modern DNNs become more powerful with an ever-increasing model size, i.e. 60M to even 10B parameters to represent the weight connections [28], [29], [22], reducing their storage and computational costs becomes critical to meet the requirement of practical applications in hardware-oriented DNNs with limited resources, i.e. ASIC or FPGA. Therefore, removing the redundancy of DNN models has become a "must-have" step in deep learning system design [22].

Many studies are preformed to reshape the DNN models towards affordable hardware implementations, including network pruning [21], [22], HashNet [19], [20], etc. Those solutions can effectively compress the weights through some non-linear transformations. Take the HashNet adopted in this work as an example, a hash function is selected to randomly group connection weights into hash buckets. All connections within the same hash bucket share a single parameter value. Therefore, the needed memory to store the weights can be significantly reduced. Figure 1 demonstrates the idea of an example HashNet for achieving significant storage reduction with limited accuracy loss. The original weights are converted to real weights by a random hash procedure. The real weights together with the hash index which are physically stored in the DNN hardware only cost $\sim 20\%$ memory space compared to that of original (virtual) weights.

C. Adversarial Examples

Adversarial examples are maliciously crafted inputs dedicated to mislead the DNN classification by introducing small input perturbations. The generating of adversarial examples can be modeled as an optimization problem:

$$\arg\min_{\delta_X} \parallel \delta_X \parallel \ \ s.t.\ F(X+\delta_X) = Y^* \tag{1}$$

Here F represents the function of target DNN model, and is usually determined by the detailed DNN configurations such as the architecture and the weight. Y^* is the distorted output which is different from the correct output Y. $X^* = X + \delta_X$ denotes the adversarial example perturbed by δ_X. Hence the question becomes how to solve the optimization problem to find the minimized δ_X. The common approach to derive adversarial examples is to extract adversarial perturbations δ_X from the gradient information, since the gradient is a good measurement for the output response difference with respect to variations introduced in each dimension of an input vector. Hence, there are two gradient-based methods to generate adversarial examples from software DNN models: *Fast Gradient Sign Method* (FGSM) [13] and *Jacobian-based Saliency Map Approach* (JSMA) [15]. The former adds a small perturbation in the direction of the sign of the gradient of the loss function with respect to the input of the DNN to all input dimensions, while the latter only distorts the most significant input features based on the salience map extracted from gradient of model function w.r.t. inputs–Jacobian matrix.

Figure 2 shows a conceptual view of *FGSM* based adversary on a representative DNN model–"AlexNet" with perturbation parameter $\epsilon = 0.005$. The image originally correctly classified as "Dog" by the "AlexNet" (65% confidence) is now misclassified as "Bear" with a much higher confidence (95%) due to the slightly polluted input. However, such an adversarial example is so close to the original image that the differences are indistinguishable to human eyes.

III. Attack Design

To analyze the vulnerabilities of practical deep learning systems under adversarial attacks, we first present the threat model, followed by an attack methodology developed for conducting adversarial attacks over the hash-based deep compressed DNN models.

A. Threat Model

In this work, we adopt a white-box adversarial attack model. We assume that the attacker has full access to all target compressed /non-compressed DNNs, training and testing dataset. The objective of adversarial attack is to mislead the classification of an original class to a different target, i.e. original $\neq$ target. To conduct the attack, the attacker first acquires the DNN model information such as weights, cost function, hash compression, gradient with normal input etc. Then the imperceptible perturbations are calculated through derived adversarial crafting algorithms and injected into normal inputs to generate adversarial examples. Finally, the adversarial examples will be sent to compressed/non-compressed DNN models, fooling the deep learning systems with adversarial classification results.

B. Adversarial Attack Design

To exert effective adversarial attacks to practical deep learning systems, our first step is to extend the single-factor adversarial examples generating algorithm to the multi-factor version based on the augmentation of software-model oriented FGSM and JSMA approaches by taking mathematical characterized hash-based deep compression into consideration. Then a synthesized attack methodology is presented as our basis for security analysis and robustness evaluation.

1) Multi-factor Adversarial Example Generating

To better illustrate how the adversary generating will be altered by the input perturbation and model reshaping in deep learning systems, the adversarial attack is again modeled as an optimization problem:

$$\arg\min_{\delta_{X_H}} \parallel \delta_{X_H} \parallel \ \ s.t.\ F_H(X+\delta_{X_H}) = Y^* \tag{2}$$

where F_H represents the hardware-oriented hashed DNN model derived from its software version (or uncompressed DNN model) F with marginal accuracy reduction. Y^* is the distorted output which is different from the correct output Y. Apparently, the minimum input perturbations of F_H (δ_{X_H}) will be less likely to be equal to that of ideal software DNN model, i.e. F (δ_X), even for the same adversarial target Y^*: $min\delta_{X_H} \neq min\delta_X$ because of the model reshaping ($F_H \neq F$). If we define W and $W_H = \varphi(W)$ as the weight matrix of DNN model F and F_H, the activation output will be $W(X+\delta_X)$ and $\varphi(W)(X+\delta_{X_H})$, respectively, where $\varphi(W)$ denotes a hardware-oriented weight transformation–hashing in HashNet. Since the hardware-oriented model reshaping should always minimize the accuracy loss, the corresponding results after activation

f should be $f(WX) \approx f(\varphi(W)X)$. However, the DNN output perturbations will be changed from $f(W\delta_X)$ to $f(\varphi(W)\delta_{X_H})$ accordingly. Even for the same adversarial example ($\delta_{X_H} = \delta_X$), the responses from the two models will be quite different. Different from the single uncertainty factor assumption, i.e. input perturbations, adopted in the software DNN models, the compressed version of adversarial attacks will be more complicated and become a multi-factor problem due to the additional weight transformations.

As the foundation for hardware-oriented adversarial example generating, we first mathematically model the deep compressed DNN model–HashNet. In HashNet, the derived classification output a_i^{l+1} for neuron i in layer $l+1$ and the gradient (δ_j^l) of loss function $\mathcal{L}$ over activation j in layer l can be presented as:

$$a_i^{l+1} = f\left(\sum_j^{n^l} w^l_{h^l(i,j)}\xi^l(i,j)a_j^l\right) \tag{3}$$

$$\delta_j^l = \left(\sum_{i=1}^{n^{l+1}} w^l_{h^l(i,j)}\xi^l(i,j)\delta_j^{l+1}\right) f'(z_j^l) \tag{4}$$

where $h^l(i,j)$ is the hash function associated with the weights in layer l and $\xi^l(i,j) : \mathbb{N} \rightarrow \pm 1$ is the second hash function independent of h for sign function to remove the bias of hashed inner-products caused by collisions [30]. $f'(\cdot)$ represents the first derivative of activation function $f(\cdot)$, and z_j^l is the result before activation function. The weight transformation function will be modeled as $\varphi(W) = W_h \bigodot \xi$ by introducing the two hash functions h and ξ. Here $\bigodot$ denotes the elementary multiplication and the compression rate can be set by tuning ξ. Accordingly, augmented from the FGSM, we can derive the Hardware-oriented Fast Gradient Sign Method (HFGSM) dedicated to HashNet as:

$$X^{AE} = X + \epsilon sign(\nabla_X J(X)) \tag{5}$$

where, the gradient can be calculated as:

$$\begin{cases} \nabla_X J(X) = \sum_{i=1}^{n^{l=2}} w^{l=1}_{h^{l=1}(i,j)}\xi^{l=1}(i,j)\delta_j^{l=2} \\ \delta_j^l = \left(\sum_{i=1}^{n^{l+1}} w^l_{h^l(i,j)}\xi^l(i,j)\delta_j^{l+1}\right) f'(z_j^l) \end{cases} \tag{6}$$

where ϵ is the amplitude coefficient of perturbations, $\nabla_X J(X)$ is the gradient of loss function J w.r.t. input X.

Similarly, the Hardware-oriented Jacobian-based Saliency Map Approach (HJSMA) for HashNet can be further developed with the same weight transformation but forward derivative gradient that can be obtained from the result of output layer. Thus an "adversarial saliency map" that indicates *the correlation between inputs and outputs* can be calculated from the gradient $\nabla_{x_i} F(x)$:

$$S(X,t)[i] = \begin{cases} 0 \text{ if } \nabla_{X_i} F_t(X) < 0 \text{ or } \sum_{o \neq t} \nabla_{X_i} F_o(X) > 0 \\ \nabla_{X_i} F_t(X) \left|\sum_{o \neq t} \nabla_{X_i} F_o(X)\right| \text{ otherwise} \end{cases} \tag{7}$$

where each element of saliency map $S(X,t)[i]$ for a false target class t is obtained based on the rule of rejecting input components with negative target derivative or an overall positive derivative on other classes o, otherwise accepting input components based on synthetic results of positive target derivative and all the other forward derivative components together. Therefore, only the input features corresponding to large values of $S(X,t)[i]$ in saliency map can be identified for adding adversarial perturbations, thus to efficiently mislead the classification result to a certain target.

2) Attack Methodology

To facilitate comprehensive adversarial attacks for the deep compressed DNN model, we develop a synthesized attack methodology by integrating the derived HFGSM and HJSMA approaches. As

Algorithm 1: Adversarial Attack Methodology

```
// O is the inference on target DNN model
// R is the random selected inputs for a round of attack
// ε is the amplitude coefficient of perturbation in HFGSM
// i is the number of perturbation elements in HJSMA
1  foreach x⃗ ∈ R do
       // get the original input X and inference result Y
2      D(X, Y) ← {(x⃗, O(x⃗))|x⃗∈R}
       // calculate the gradient s.t. input X
3      ∇_X J ← Equation 6
       // generating perturbation
4      δ_{X_H} ← HFGSM(∇_X J) or HJSMA(∇_X J)
5      X* = X + δ_{X_H}
       // perform inference using adversary as input
6      Y* ← O(X*)
7      if Y = Y* then
           // the adversarial attack is not success
8          if ε or i < predefined upper-bound then
9              increase ε in HFGSM (Equation 6) or
10             increase i in HJSMA (Equation 7)
11             GOTO: line 4
12     else
13         adversarial success counter += 1
```

Algorithm 1 shows, an upper-bound of the perturbation amplitude coefficient ϵ in HFGSM (see Eq. 6) or the number of perturbation elements i in HJSMA (see Eq. 7) will be predefined to guarantee that the crafted adversarial perturbations can be maintained at an imperceptible level, which is more desirable in practical attacks. A randomly selected original input-output pair (X, Y) will be recorded and compared with the adversarial input-output pair (X^*, Y^*). The adversarial example generating process will be terminated once a successful adversarial attack happens, i.e. $Y \neq Y^*$, otherwise ϵ or i will be increased until reaching the respective upper-bound. The success rate of adversarial attacks will be adopted as a measurement in our following security analysis.

IV. Security Analysis

We conduct the multi-factor adversarial attacks on the following tailored DNN model (i.e. 784-C64-C128-F512-10) applied with HashNet model reshaping by following the proposed attack methodology. A full MNIST database is adopted as our benchmark for a comprehensive analysis of attacking effectiveness in deep compressed/non-compressed deep learning systems.

A. Effectiveness of Multi-factor Adversarial Attacks

We first designed several hash compressed DNN models–HashNets with different compression rates (from $\frac{1}{8}$ to $\frac{1}{64}$) based on the aforementioned uncompressed DNN model. To make a fair adversarial attack analysis, our HashNets minimize the testing accuracy degradation (with normal input data without adversarial perturbations) introduced by weight compression. As shown in Fig. 3, the testing accuracy on HashNets is only slightly decreased as the compression rate increases (i.e. 99.25% at rate $\frac{1}{8}$ v.s. 99.13% at $\frac{1}{64}$) but still very close to the uncompressed model (99.29%).

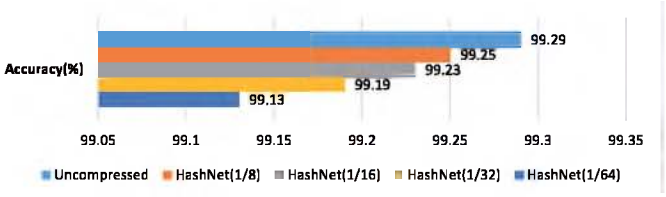


Fig. 3: Testing accuracy without adversarial perturbations.

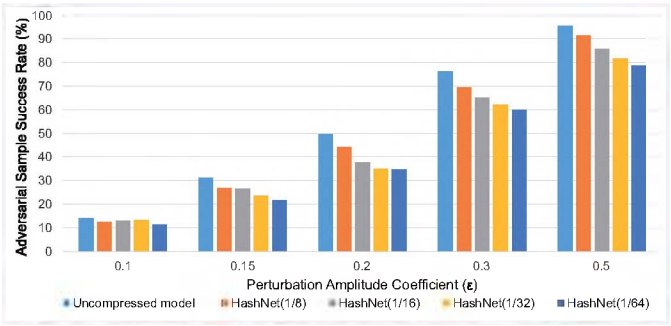


Fig. 4: Success rate of multi-factor adversarial attacks with HFGSM approach.

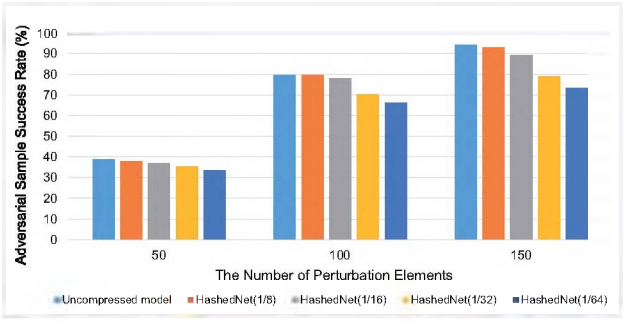


Fig. 5: Success rate of multi-factor adversarial attacks with HJSMA approach.

Fig. 4 shows the success rate of multi-factor adversarial attacks implemented with HFGSM method at various compression rates (i.e. HashNet($\frac{1}{8}$) $\rightarrow$ HashNet($\frac{1}{64}$) over different perturbation amplitude coefficients. For comparison purpose, the results of the uncompressed DNN model–the common basis of different HashNets, under the original single-factor based FGSM attacks are also presented. As expected, the attack success rates of both uncompressed DNN model and various compressed models are increased monotonically along with the growing perturbation amplitude coefficient, i.e. $\epsilon = 0.1 \rightarrow 0.5$. This is because the attacking capability of crafted adversarial examples can be significantly enhanced by larger input perturbations (see $\epsilon = 0.5$) for all DNN models regardless of the model reshaping. However, for each individual ϵ, the attack success rates of any HashNet models are always lower than that of uncompressed model. Moreover, the higher the compression rate is, the lower the attack success rate will be at each ϵ. We also conduct the same set of experiments under HJSMA based adversarial attacks. Again, our results in Fig. 5 demonstrate the similar trend at different combinations of compression rate and the number of perturbation elements, i.e. the attack success rates are decreased when increasing compression rate on HashNet at each selected number of perturbation elements. Surprisingly, *these results indicate that the hash compressed DNN model, which have significantly reduced number of model parameters for affordable hardware implementation (see Fig. 1), exhibits better resistance to adversarial attacks than that of its uncompressed or less compressed version. This is in contrast to the empirical intuition that the more compressed DNN models should be more susceptible to the input perturbations.*

Since the compressed DNN models maintain the similar level of the stability (or testing accuracy in Fig. 3) as that of uncompressed model, a reasonable explanation for the attack success rate reduction is that the destructiveness of crafted adversaries may be alleviated in HashNets when compare with those generated in uncompressed DNN model. *That is being said, the effectiveness of multi-factor adversary attacks depends on the perturbation amplitude coefficient ϵ in HFGSM (or the number of perturbation elements i in HJSMA) and the compression rate,* as we shall discuss in the following section.

B. Theoretical Analysis of Adversarial Attacks on Hashed DNNs

To validate our hypothesis and deeply understand the relationship between adversary and model reshaping, we characterize the two critical components for adversary example generating in compressed models: weight and gradient amplitude under various compression rates. Fig. 6 compares the distributions of weights for uncompressed and two compressed DNNs–HashNet ($\frac{1}{16}$) and HashNet ($\frac{1}{64}$). As Fig. 6 shows, the model with a higher compression rate yields a larger range of weights (i.e. $\sim 2\times$ and $\sim 4\times$ in HashNet ($\frac{1}{16}$) and HashNet ($\frac{1}{64}$) w.r.t. uncompressed model). Given significantly decreased number of unique weights (or increased compression rate) introduced by hash-based weight sharing mechanism, the weight distribution in compressed DNN model shall be much broader since such model has to re-balance the activations through enlarged weights during training to achieve an accuracy close to that of uncompressed model. However, such weight transformation can directly impact the gradient, thus the strength of generated adversaries.

Without loss of generality, we use the output layer with softmax activation to roughly explain the underlying principle. The final activation of output layer can be calculated through the following *Softmax* function:

$$F(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{n} e^{z_k}} \tag{8}$$

where the input z_j of *Softmax* function can be expressed as:

$$z_j = \sum_{i=1}^{n} w_{ji} x_i \tag{9}$$

Note that we omit the bias because it can be included in weight by adding an additional connection with weight as the bias and a constant input 1. Since the *Softmax* function increases monotonically as the input z_j grows, **the enlarged weights in highly compressed models can possibly augment the desired activations but suppress the others, thus a possible stronger confidence for the final decision.**

If we use FGSM based adversarial example generating algorithm as an example, the cross-entropy loss function and its gradient w.r.t. input can be obtained as:

$$J(x_i, t_j) = -\sum_{j=1}^{n} t_j \log F(z_j) \tag{10}$$

$$\nabla_{x_i} J(x_i, t_j) = \sum_{j=1}^{n} w_{ji}(F(z_j) - t_j) \tag{11}$$

where x_i is the i^{th} input and t_j is the target for j^{th} class. Consider the $F(z_j)$ is an exponential function of w_{ji}, the absolute gradient amplitude will be dominated by term $(F(z_j) - t_j)$. **With the enlarged weights in compressed models, the activation $F(z_j)$ may be closer to t_j, thus a possible reduced gradient and perturbation amplitude, meaning alleviated adversarial severity.** Fig. 7 shows the distributions of absolute value of mean gradient over uncompressed and compression DNNs with different compression rates. The

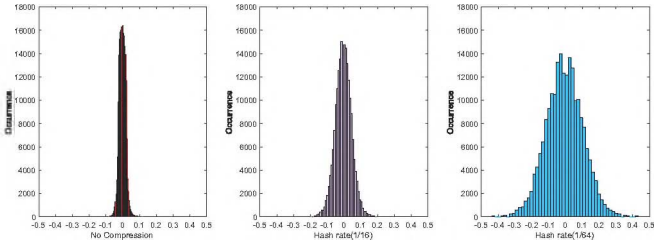


Fig. 6: The weight distributions for uncompressed DNN and two HashNets.

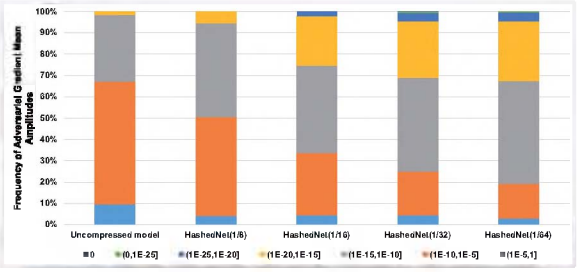


Fig. 7: Absolute gradient amplitude with different compression rate.

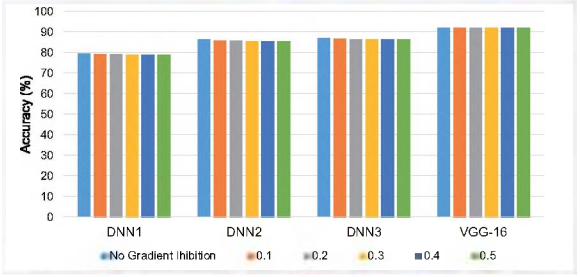


Fig. 8: Inference accuracy of CIFAR-10 with Gradient Inhibition

proportion of large gradients ($10^{-10} \sim 1$) is reduced from $\sim$ 68% (uncompressed model) to $\sim$ 19% (HashNet($\frac{1}{64}$)) as compression rate grows, while that of small gradients ($10^{-25} \sim 10^{-15}$) is increased from $\sim$ 1% to $\sim$ 32%, which is in excellent agreement with our theoretical analysis and validates the degraded attack capability of compressed DNNs compared with the uncompressed version.

V. Mitigation Approach

In our security analysis, we show that the magnitude of weights in DNNs becomes a new factor that can significantly impact the the severity of adversarial attacks. Hash-based weight compression enlarges the magnitude of weights, thus to prevent the generating of stronger adversarial examples. However, its effectiveness is very limited, e.g. $\leq$ 30% success rate reduction at any perturbation amplitude coefficient in Fig. 4, because the weight enlargement, introduced by non-linear weight transformation, can only be guaranteed at a certain probability (see Fig.6). Inspired by this observation, a novel mitigation technique named *Gradient Inhibition* is further proposed to effectively mitigate the adversarial attacks.

A. Gradient Inhibition method

Our proposed *Gradient Inhibition* intends to control the weights linearly with enlarged magnitude guarantee for each weight:

$$w = w + \tau * sign(w) \tag{12}$$

where τ is the inhibition coefficient. Different levels of weight enlargement can be achieved by a fine-grained control parameter τ for both positive and negative weights, thus to minimize the gradient needed for adversarial perturbations generating and effectively mitigate or even eliminate the threats of adversarial attacks for DNNs.

Another advantage of *Gradient Inhibition* method is its low implementation cost applicable to both software or hardware-oriented compressed DNN models. *Gradient Inhibition* can be applied at any layer after the training process. Our practice is to deploy this method at the layers close to the output layer (i.e. the last fully connected layer) for higher attack rate reduction but lowest accuracy loss due to the usually moderate number of weights and strongest impacts on decision making.

B. Evaluation of Gradient Inhibition

1) Experiment Setup

Various HashNets and MNIST benchmark [23], which are used in section IV, are adopted in our experiment to evaluate efficiency of *Gradient Inhibition*. Additionally, the CIFAR-10 database [24] is selected as a new benchmark in our evaluation, including 60K 32×32 color images in 10 classes, 50K for training and 10K for testing. As shown in Table. I, four representative DNN models with different architectures, including state-of-art VGG-16 [26], are chosen to verify the feasibility and scalability of Gradient Inhibition across various types of DNN models. We assume the adversarial examples are generated through the FGSM and HFGSM for uncompressed and compressed models, respectively.

Candidate Models	DNN1	DNN2	DNN3	VGG-16
Relu Convolutional	4 layers	6 layers	9 layers	13 layers
Relu Fully Connected	2 layers	2 layers	2 layers	3 layers
Max Pooling	2 layers	3 layers	3 layers	5 layers

TABLE I: Architectures of selected neural network candidates.

2) Inference Accuracy

An effective mitigate technique against adversarial attacks should not impact the functionality of the DNN models integrated with mitigate techniques. Before we evaluate the effectiveness, we first verify the inference accuracy changes introduced by *Gradient Inhibition*. As shown in 8, the inference accuracy on CIFAR-10 database for each DNN model implemented with *Gradient Inhibition* is always at the same level as that of its corresponding model without such technique at different inhibition coefficients. We also find the similar accuracy trend in Hash compressed DNNs with different compression rates for the MNIST dataset. Note that the adopted inhibition coefficient $\tau = 0.1 \rightarrow 0.5$ can introduce flexible weight adjustments, i.e. $\pm 0.1 \rightarrow \pm 0.5$, with very minor accuracy change.

3) Gradient Inhibition Efficiency

Fig. 9 shows the statistics of suppressed gradients across various inhibition coefficients for an uncompressed DNN model testing the MNIST dataset. As shown in Fig. 9, even with a very small adjustment on original weights, i.e. inhibition coefficient τ = 0.01, the gradient amplitude can be much lower than the one generated on HashNet($\frac{1}{64}$) in Fig. 7, which is the best case in compressed DNN models. Note that in HashNet($\frac{1}{64}$), the range of weights has been enlarged from ± 0.1 to ± 0.4 (see Fig. 6), which is far exceed that of ± 0.01 in Gradient Inhibition. Therefore, our proposed method can significantly suppress the gradients with much lighter weight transformation. Moreover, as shown in Fig. 9, most of gradients are approaching to "0" along with the increased inhibition coefficient, indicating the possible elimination of adversarial perturbations, thus to prevent the adversarial attacks remarkably.

4) Mitigation Measures

Adversarial attacks are conducted by following the proposed attack methodology, on both DNN and compressed HashNet models

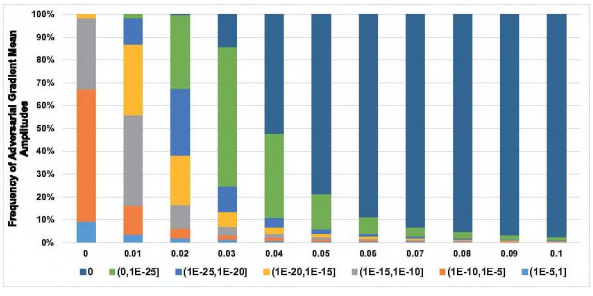
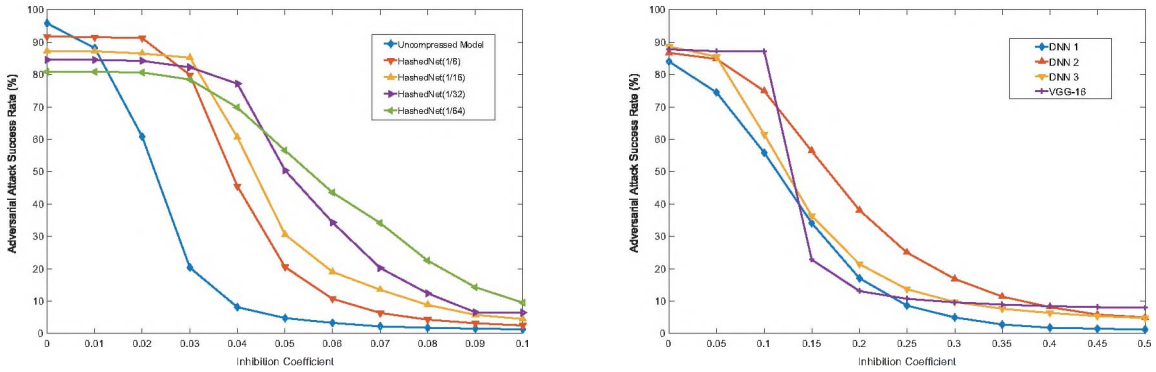


Fig. 9: Absolute gradient amplitude of uncompressed DNN at various inhibition coefficients



(a) HashNet with model compression – MNIST

(b) DNNs – CIFAR-10

Fig. 10: Success rate of adversarial attacks with Gradient Inhibition mitigate technique.

with the Gradient Inhibition method. Fig. 10 (a) and (b) show the success rates of adversarial attacks under Gradient Inhibition over HashNets (for MNIST) and four DNNs (for CIFAR-10), respectively. As Fig. 10a shows, the average success rate of adversarial attacks (HashNets, perturbations crafted through HFGSM with $\epsilon = 0.5$) can be reduced from 87.99% to 4.77% by increasing the inhibition coefficient τ from 0 to 0.1. Specifically, the uncompressed model presents the best efficiency (95.81% $\rightarrow$ 1.24%) while all compressed HashNets exhibit some resistance to Gradient Inhibition and eventually reduce the adversarial success rate to less then 10% at all selected compression rates. Fig. 10b evaluate the efficiency of proposed Gradient Inhibition on DNNs with CIFAR-10 database. The average success rate is dropped from 86.74% to 4.64% across various DNNs, demonstrating effective mitigations for adversarial attacks.

VI. Conclusion

The emerging adversarial attacks leave the prevalent hardware accelerated Deep Neural Networks (DNNs) exposed to hackers. However, existing DNN security researches solely focus on the input perturbations but neglect the impacts of model-reshaping essential for DNN hardware deployment. In this work, the multi-factor adversarial attack problem is for the first time modeled and studied through extensive experimental and theoretical analysis. Based on the explorations of model-reshaping and adversarial examples generating, a novel mitigation technique – "Gradient Inhibition" is further proposed to effectively alleviate the severity of adversarial attacks for various DNNs. Our simulations demonstrate that "Gradient Inhibition" can significantly reduce the success rate of adversarial attacks while maintaining the desired inference accuracy without additional trainings. We hope that our results enable the community to examine the emerging security issues of hardware-oriented DNNs.

References

[1] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] https://cloudplatform.googleblog.com/2016/05/Google-supercharges-machine-learning-tasks-with-custom-chip.html.

[3] https://research.fb.com/category/facebook-ai-research-fair/.

[4] https://www.microsoft.com/en-us/research/research-area/artificial-intelligence/.

[5] D.C. Ciresan *et al.*, "Flexible, high performance convolutional neural networks for image classification," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1. Barcelona, Spain, 2011, p. 1237.

[6] V. Vanhoucke *et al.*, "Improving the speed of neural networks on cpus," in *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, vol. 1, 2011, p. 4.

[7] C. Farabet *et al.*, "Large-scale fpga-based convolutional networks," *Scaling up Machine Learning: Parallel and Distributed Approaches*, pp. 399–419, 2011.

[8] https://www.perspectiveapi.com/.

[9] N.P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," *arXiv preprint arXiv:1704.04760*, 2017.

[10] https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu.

[11] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, p. 452, 2015.

[12] M. Barreno *et al.*, "The security of machine learning," *Machine Learning*, vol. 81, no. 2, pp. 121–148, 2010.

[13] I.J. Goodfellow *et al.*, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[14] N. Papernot *et al.*, "Practical black-box attacks against deep learning systems using adversarial examples," *arXiv preprint arXiv:1602.02697*, 2016.

[15] N. Papernot *et al.*, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 372–387.

[16] A. Fawzi *et al.*, "Analysis of classifiers' robustness to adversarial perturbations," *arXiv preprint arXiv:1502.02590*, 2015.

[17] S. Gu *et al.*, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[18] N. Papernot *et al.*, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 582–597.

[19] W. Chen *et al.*, "Compressing neural networks with the hashing trick," in *International Conference on Machine Learning*, 2015, pp. 2285–2294.

[20] Z. Cao *et al.*, "Hashnet: Deep learning to hash by continuation," *arXiv preprint arXiv:1702.00758*, 2017.

[21] S. Han *et al.*, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[22] S. Han *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[23] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[24] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," 2009.

[25] G.E. Hinton *et al.*, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[26] K. Simonyan *et al.*, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[27] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[28] Y. Cheng *et al.*, "An exploration of parameter redundancy in deep networks with circulant projections," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2857–2865.

[29] S. Han *et al.*, "Eie: efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*. IEEE Press, 2016, pp. 243–254.

[30] K. Weinberger *et al.*, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1113–1120.