Model Compression Hardens Deep Neural Networks: A New Perspective to Prevent Adversarial Attacks

Qi Liu and Wujie Wen, *Member, IEEE*

***Abstract*— Deep neural networks (DNNs) have been demonstrating phenomenal success in many real-world applications. However, recent works show that DNN's decision can be easily misguided by adversarial examples–the input with imperceptible perturbations crafted by an ill-disposed adversary, causing the ever-increasing security concerns for DNN-based systems. Unfortunately, current defense techniques face the following issues: 1) they are usually unable to mitigate all types of attacks, given that diversified attacks, which may occur in practical scenarios, have different natures and 2) most of them are subject to considerable implementation cost such as complete retraining. This prompts an urgent need of developing a comprehensive defense framework with low deployment costs. In this work, we reveal that "defensive decision boundary" and "small gradient" are two critical conditions to ease the effectiveness of adversarial examples with different properties. We propose to wisely use "hash compression" to reconstruct a low-cost "defensive hash classifier" to form the first line of our defense. We then propose a set of retraining-free "gradient inhibition" (GI) methods to extremely suppress and randomize the gradient used to craft adversarial examples. Finally, we develop a comprehensive defense framework by orchestrating "defensive hash classifier" and "GI." We evaluate our defense across traditional white-box, strong adaptive white-box, and black-box settings. Extensive studies show that our solution can enormously decrease the attack success rate of various adversarial attacks on the diverse dataset.**

***Index Terms*— Adversarial defense, adversarial examples, deep neural network (DNN), model compression.**

I. Introduction

DEEP neural networks (DNNs) have been achieving tremendous success across many exciting real-world applications, such as image classification, speech recognition, and self-driving cars [1], [2]. However, recently, the security of DNN has emerged as a major concern with the proliferation of DNN-based applications. Extensive studies show that the function of a DNN can be easily deceived by adversarial examples—a type of slightly polluted inputs [3]–[6]. The injected adversarial perturbations are almost imperceptible to human eyes but can mislead the decision of a target DNN model with very high confidence, i.e., misinterpreting the "Stop" sign as a "Speed Limit" or other sign in a self-driving car, thereby causing potential disastrous consequences [7].

Manuscript received September 27, 2020; revised March 22, 2021; accepted May 26, 2021. This work was supported by the National Science Foundation under Award CNS-2011260, Award CCF-2011236, and Award CCF-2006748. *(Corresponding author: Wujie Wen.)*

The authors are with the Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, PA 18015 USA (e-mail: qil219@lehigh.edu; wuw219@lehigh.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3089128.

Digital Object Identifier 10.1109/TNNLS.2021.3089128

To address this problem, researchers have proposed a series of defense techniques, which mainly include the following four categories: adversarial training [4], [8], gradient masking [9], input transformation [10], [11], and adversarial example detection [12]. However, the research on the defense is still lagging behind and facing the following significant challenges.

1) Inefficiency of a single defense solution due to different natures of existing adversarial examples. For example, the fast gradient sign method (FGSM)-based [4] adversarial examples could cross DNN's decision boundary easily because of the large input perturbations, while stronger examples based on Carlini & Wagner (C&W) method [13], which offers $\sim$100% attack success rate, are more close to decision boundary with much smaller perturbations. This leads to the fact that a single defense solution cannot work well when facing various attacks, e.g., model-specific methods such as "defensive distillation" [9] can mitigate FGSM attacks but fails in C&W attacks.
2) High implementation cost because of costly retraining, data preprocessing, and so on. For example, projected gradient descent (PGD)-based adversarial training can resist the L_∞ adversarial attacks [8] by doubling the training cost due to the augmented training dataset with adversarial patterns. Defensive distillation [9] also requires additional training on the soft labels from the first-round training. Note that training modern DNNs from scratch is very expensive as it involves long processing time over multiple GPU clusters [14], e.g., a few weeks or months. Apparently, these limitations make them difficult to protect DNNs, especially considering the varying strategies of attackers in practical scenarios. As DNN model size continues growing and state-of-the-art attacks become more diversified and stronger, exploring a comprehensive defense framework with low cost has become a necessity.

In this work, we attempt to fill this research gap by systematically attacking two fundamental entities that impact the effectiveness of any type of adversarial examples: Factor 1) "decision boundary"—the surface of trained DNN model's input space where adversarial examples need to cross for causing incorrect output predictions and Factor 2) "adversarial gradient"—the gradient with respect to inputs directly used to craft adversarial examples. Existing research has shown how each individual factor impacts the success rate of several selected adversarial attacks [8], [9], [15], and here, we advocate that the reason for different adversarial examples can exhibit different nature stems from the relative importance of both factors, e.g., ("Factor 2" dominated—"direct gradient") FGSM, and ("Factor 1" dominated—"indirect gradient") C&W attack with much smaller perturbations according to iterative and search process of better objective functions [13]. Therefore, a critical observation in this article is that DNN models can be comprehensively hardened if and only if the above two factors can be well handled simultaneously. Inspired by this observation, we propose to construct the model compression-based "defensive decision boundary" first, followed by a set of retraining-free "gradient inhibition" (GI) methods to further suppress the adversarial gradient and, at the same time, to obfuscate the iterative search procedure of fine-grained attacks (e.g., C&W). We summarize our key contributions as follows.

1) We rearchitect "hash compression" technique to harden the pretrained DNN model with defensive decision boundary, namely "defensive hash classifier," in order to provide the first line of defense. Besides, we propose a robustness estimator to guide the search of optimized compression rates (CRs) of selected layers for the "defensive hash classifier." The robustness estimator is built upon the synthetic contribution of both adversarial gradient and L_2 distance of logits (before the softmax function). The key advantages of such defensive classifier include: 1) quick installation by applying dedicated hash compression to a few selected fully connected layers of trained DNNs and 2) marginal retraining cost (i.e., three epochs) for accuracy recovery by using "transfer learning." This is an early attempt that explores how weight compression—an indispensable technique originally aiming to ease the memory/storage overhead during DNN hardware implementation—can be redesigned for enhancing the robustness of DNN models.
2) We develop a set of retraining-free "GI" methods complementary to the defensive hash classifier to effectively eliminate the remaining impact of adversarial gradients with marginal accuracy reduction. Unlike existing gradient masking methods, "GI" directly manipulates the trained weights without involving any retraining and reduces the impact of gradients to the minimum. Besides, one of the proposed methods also misleads the procedure of searching optimal adversarial perturbations, which is the key to find stronger adversarial examples, i.e., C&W attacks. This can translate into significant improvement in defense efficiency.
3) We orchestrate the "defensive hash classifier" and "GI" for a holistic solution set to defend against attacks under two settings: white box and black box. For white-box setting, experimental results show that our solution set can reduce the attack success rate from 93.45% to 14.39% (from 91.27% to 22.98% and from 87.18% to 34.63%) on average for the MNIST (CIFAR10 and ImageNet-subset) datasets across eight mainstream attacks with very marginal accuracy loss. For black-box setting, our solution can lower the success rate of state-of-the-art zeroth-order optimization (ZOO) attacks by ~50% (untarget) and ~70% (target) for both MNIST and CIFAR10, respectively.

We believe that this work provides a radically different perspective, e.g., model compression and retraining-free DNN hardening techniques, for developing a comprehensive defense strategy to protect DNNs against adversarial attacks, with the guarantee of low cost and high accuracy.

II. Background

A. Deep Neural Network

DNN can be simplified as $F(\cdot) : x \rightarrow y$, with the input $x \in \mathbb{R}^n$, output $y \in \mathbb{R}^m$, parameters θ (such as weights and bias), and DNN model function F. The randomly initialized θ will be iteratively updated until the convergence by minimizing the loss function, i.e., cross-entropy loss function, in the training phase. The softmax function is widely adopted in the output layer [16]–[19], to produce an m-dimensional probability vector that indicates the confidence of each class with respect to an input x. We define $z = Z(x)$ as the logits (multiply–accumulate features before softmax) and y as the DNN output (after softmax activation): $F(x) = \text{softmax}(Z(x)) = y$.

B. Adversarial Attacks

Adversarial examples are crafted by adding small and imperceptible perturbations into normal data, in order to mislead the DNN classification. We summarize several mainstream white-box attacks (1–7) and black-box attacks (8 and 9) as follows.

1) White-Box Attacks:

a) Fast gradient sign method: FGSM [4] is L_∞-norm nontarget attack and intends to add the largest perturbations (ϵ) into each pixel of an image, by following the direction of the gradient of loss function L with respect to input x ($\nabla_x L(x, y)$): $x^* = x + \epsilon \cdot \text{sign}(\nabla_x L(x, y))$.

b) Basic iterative method (BIM): BIM [20] is an iterative FGSM attack to add small perturbations α in each iteration until the largest perturbations level reaches ϵ or achieves a successful attack.

c) Momentum iterative method (MIM): Dong *et al.* [21] proposed to integrate the momentum method to accelerate and optimize iterative FGSM's gradient descent process, so as to further boost the effectiveness of adversarial examples.

d) Jacobian-based saliency map approach (JSMA): JSMA [22] is the L_0-norm attack against a specific class (targeted attack). Adversarial examples are created by modifying only few most significant pixels of the input image. As the saliency map can be generated either from the gradient of logits Z before softmax function or gradient of DNN output F after softmax function, JSMA attacks can be further categorized into two versions: JSMA-Z [23] and JSMA-F.

e) C&W method: Carlini *et al.* [13] proposed three types of attacks based on L_0, L_∞, and L_2 norms. Such examples are crafted by solving the following two-term objective function:

$$\min \ \|\delta\|_p + c \cdot f(x+\delta) \quad \text{s.t. } x+\delta \in [0,1]^n \tag{1}$$

where c is a critical parameter decided by the binary search using the gradient descent method to minimize both terms of (1), p is the norm, and f can be further expressed as follows:

$$f(x^*) = \max(\max\{Z(x^*)_i \quad i \neq t\} - Z(x^*)_t, -\kappa) \tag{2}$$

with κ the confidence level of incorrect classification ($i \neq t$).

f) Elastic-net attacks to DNNs (EAD): Elastic-net attack [24] further improves the C&W method by adding L_1 and L_2 regularization terms simultaneously to the optimization function. The new objective function can be expressed as: $\min_x \ \beta \cdot \|x - x^*\|_1 + \|x - x^*\|_2 + c \cdot f(x^*)$, where c and β are the regularization parameters.

g) Backward pass differentiable approximation (BPDA): Athalye *et al.* [25] introduced the BPDA, to circumvent the majority of state-of-the-art defenses, based on the assumption that the adversary has the full knowledge of defense method on top of the DNN model (a strong white-box attack setting). The key idea of BPDA is to find a differentiable approximation function $g(x)$ to replace the nondifferentiable function $f_i(x)$ designed by defense [where $g(x) \approx f_i(x)$] in layer i of a DNN model so that attackers can still generate adversarial examples on the backward pass.

2) Black-Box Attacks:

a) Black-box attack based on a substitute model: Papernot *et al.* [7] proposed a more practical adversarial attack method in the black-box setting. Adversary needs to train a substitute model for generating adversarial examples. As the decision boundary modeled by the substitute model is similar to the targeted model, adversarial examples generated in a substitute model (using white-box attack techniques listed above) exhibit high transferability and can mislead the targeted model.

b) Restricted black-box attack-based ZOO: To overcome the low transferability issue of black-box attack (especially on target attack) based on a substitution model, Chen *et al.* [26] proposed a stronger black-box attack based on ZOO. It directly approximates the gradient of a targeted DNN model by using the symmetric difference quotient. Note that the ZOO attack does not need to query targeted DNN several times to obtain the initial dataset for training substitute model. As a result, we introduced a more restricted black-box set—no query.

C. Hash Compression

DNN compression techniques, such as network pruning [27], [28], network quantization [28], and HashNet [29], [30], can effectively reduce the number of needed weights through pruning or weight sharing. "Hash compression"-based HashNet shares weights using the hash function to reduce storage overhead significantly. The basic idea is to randomly group original weights into hash bucket through the hash function so that the weights assigned to the same hash bucket can share the similar value. Then, the real weights in the hash bucket along with the corresponding hash index will be actually stored in hardware, costing much less memory than that of storing original weights. The tradeoff between DNN accuracy and memory overhead is realized by tuning the parameter–hash CR (defined as the ratio between the number of real weights and virtual weights in this article). To maximize the CR, hash compression is applied to all DNN layers, and however, a reasonable CR cannot be very high, i.e., $1/64$, due to the accuracy drop [29].

D. Related Defense Techniques

1) Adversarial Training: Adversarial training attempts to harden the DNN model by incorporating both adversarial examples and normal data during the training process [4], [8], [31]. However, defense efficiency is limited since it relies on the knowledge of assumed adversarial examples, which may be quite different from realistic attacks in terms of perturbation strength ϵ (i.e., FGSM) or attack types (i.e., FGSM versus variant FGSM [15], [21]). Madry *et al.* [8] also proposed a new adversarial training method using the PGD attack (BIM attack with random starts) as the strongest attack in the first-order adversary, in order to enhance the robustness of the DNN model. However, PGD-based adversarial training is very costly. Recent works mainly focus on accelerating the PGD-based adversarial training [32], [33]. For example, Shafahi *et al.* [32] proposed a free adversarial training algorithm to accelerate the computation by reusing the backward pass calculation.

2) Gradient Masking: The gradient masking method suppresses the adversarial gradient of output with respect to inputs as much as possible so that adversaries are unable to leverage such gradient to craft adversarial examples [9], [34]. For example, "defensive distillation" [9] is a classic gradient masking method. The basic idea is to first train a DNN model with smoothed labels generated by a trained teacher model with a modified softmax function, instead of hard labels (0 or 1). Then, the last layer of the distilled model after training will be replaced by a "harder" softmax function. By paying extra training on the teacher model, it can well resist some of "direct-gradient-based" attacks such as JSMA-F. However, studies proved that it cannot mitigate JSMA-Z or C&W [13], [23] attacks.

3) Model Compression and Adversarial Robustness: There exist a few recent studies that explore the interplay between model compression and adversarial robustness [35], [36]. For example, Gui *et al.* [35] proposed to concurrently integrate weight pruning, factorization, and quantization into the adversarial training framework with the goal of enabling model compression and preserving the adversarial robustness simultaneously, so as to tackle the dilemma of adversarial training—the prominent model accuracy drop versus the improvement of adversarial robustness. Ye *et al.* [36] proposed an ADMM-based weight pruning to reduce the larger network capacity caused by adversarial training without hurting the model accuracy and adversarial robustness. Therefore, previous works only utilize model compression to cope with

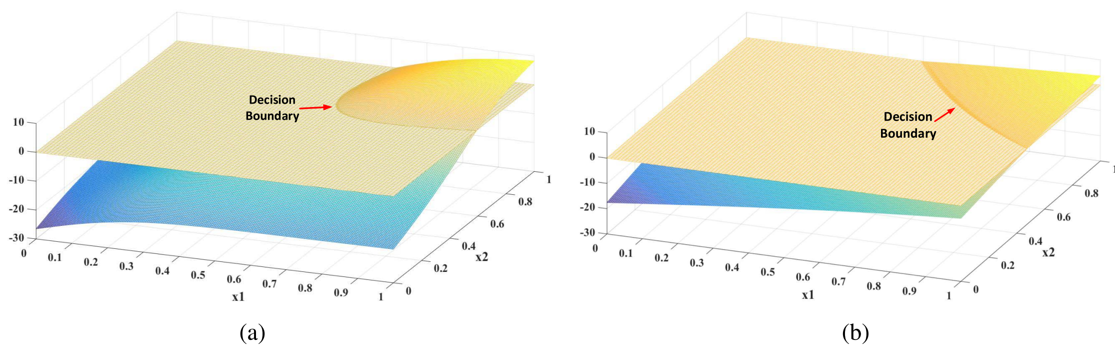


Fig. 1. Decision boundary illustration of an example neural network ("AND" function) before (a)/after (original model) and (b) hash compression (HashNet).

the compromised performance or larger model size incurred by adversarial training. There exists no work to explore whether it is possible to directly rearchitecture compression techniques in the context of security to significantly enhance model's adversarial robustness at both low cost and marginal accuracy loss. This work represents a very early attempt to redesign the hash compression technique, which is originally dedicated for storage overhead reduction, as a defense solution against a variety of adversarial attacks, by formulating the compression-enabled defensive decision boundary.

III. Factor 1: Compression-Based Defensive Decision Boundary

"Decision boundary" and "adversarial gradient" are two fundamental factors impacting the effectiveness of generated adversarial examples. In this section, we first present the concept of our advocated "defensive decision boundary" as well as the opportunity of hash compression for achieving such boundary. Then, we develop a robustness estimator to mathematically guide the design of compression-based "defensive decision boundary," followed by its implementation details.

A. Hash Compression for Defensive Decision Boundary

1) Defensive Decision Boundary: The "defensive decision boundary" in this work is assumed to convey the following natures.

1) *Condition 1:* The differences of logits Z among different classes under this boundary should be small, as long as a correct class prediction can be still made after softmax. This indeed softens the decision (boundary) and can increase the difficulty of generating adversarial examples relevant to logits, such as C&W method in (2). This assumption is confirmed by recent studies [37], [38] where the decision surface is highly correlated with logits Z, i.e., $D(x) = Z_i(x) - Z_j(x)$ $(i \neq j)$, and the authors proved that a wide and flat plateau of decision surface indicates better model robustness to adversarial attacks.
2) *Condition 2:* Statistically, the distance between input and the boundary should be large, so as to increase the needed perturbations of adversarial example generating. Note that greatly satisfying both conditions 1 and 2 is nontrivial due to accuracy constraint. Therefore, the fundamental goal is to create the boundary that seems to be somewhat "softened" in decision-making but can still maintain high prediction accuracy.

2) Conceptual View of Decision Boundary After Hash Compression: Our observation is that properly applying hash compression can achieve the aforementioned defensive decision boundary for DNN models. The reason is that random weight sharing of hash compression can cause the original DNN classification function to miss some precision to flatten the surface of the decision boundary. To test our hypothesis, we study the decision boundary change incurred by hash compression using a simple neural network for performing "AND" function. We train the original model and its hash compressed version (CR 0.2) using the cross-entropy loss for 100 epochs. The noninteger inputs are rounded up to nearest integer, i.e., if $0.5 \leq x_1 \leq 1$ and $0.5 \leq x_2 \leq 1$, then $F_{\text{AND}} = 1$ else $F_{\text{AND}} = 0$. As expected, the decision boundary can be approximated as the intersecting line of two decision surfaces: $D(x_1, x_2) = Z_1(x_1, x_2) - Z_2(x_1, x_2)$ and $D(x_1, x_2) = 0$. Fig. 1 shows the decision boundary of original model and HashNet, where the x- and y-axes are the inputs x_1 and x_2, respectively, and the z-axis represents the value range of decision surface $Z_1(x_1, x_2) - Z_2(x_1, x_2)$. Note that $Z_1(x_1, x_2) > Z_2(x_1, x_2)$ indicates that the input belongs to class $F_{\text{AND}} = 1$; otherwise, it belongs to $F_{\text{AND}} = 0$. As shown in Fig. 1, the plotted decision boundary of hash compressed model is more flattened than the original model. Given the fact that adversarial examples can be more easily found from the sharp part (larger curvature) of the decision boundary [37], a flattened decision boundary of the hash compressed model could potentially better protect DNN models from adversarial attacks (see our validation in Section V). On the other hand, its accuracy is degraded from 98.88% to 96.42% at a CR 0.2. This is because the boundary of original DNN model without compression [see Fig. 1(a)] is closer to the ideal one offering the best accuracy—a polyline with the right angle.

B. Robustness Estimator

Hash compression offers the possibility to realize our desirable "defensive decision boundary." Conceptually, the more softened the boundary is, the higher chance that DNN model can avoid being attacked by adversarial examples. Therefore, it is imperative to develop a quantitative measurement to evaluate the quality of defensive decision boundary. However, directly adopting attack success rate or defense efficiency as the metric is neither cost-effective nor applicable due to the variety of adversarial examples and unknown attack strategies. Instead, we propose a mathematical robustness estimator, which does not require any knowledge of adversarial attacks, to quickly estimate the quality of defensive decision boundary. Note that model compression can degrade DNN accuracy. In addition, we also incorporate "adversarial gradient" into robustness estimator as it is another factor to impact the effectiveness of adversarial examples. As we shall show in Section V, our robustness estimator can efficiently guide the search of the key parameter–CR, to maximize the capability of hash compression for hardening DNN model with marginal accuracy loss.

1) Modeling Defensive Decision Boundary: Directly plotting decision surface to find an optimized defensive CR for HashNet is impractical. In a multidimensional classification task, the decision surface of any two classes can be represented: $D(x) = Z_i(x) - Z_j(x)$ $(i \neq j)$. The larger the difference of $Z_i(x)$ and $Z_j(x)$ is, the steeper $D(x)$ will be. The extreme weight sharing may lead to the highest similarity between $Z_i(x)$ and $Z_j(x)$. Take the aforementioned neural network for "AND" function as an example again, and we can observe that the decision surface (blue hyperplane) of HashNet is flatter than that of the original model in Fig. 1. In an extreme case, if all weights are compressed to the same value (the highest CR), the decision surface will become $D(x_1, x_2) = 0$, given that $Z_1(x) \equiv Z_2(x)$. Apparently, it is not a reasonable classifier but achieves the most softened decision boundary. Therefore, softening the boundary of a DNN model can prevent the input from fitting too tight (confident) toward a certain class. Inspired by this observation, we model the softness of DNN model as the mutual distance of the logits $Z_j (j \in 1 \ldots m)$ (before softmax activation), where the largest Z_j is the true class. We choose the variance of logits (i.e., $\phi(x) = (1/m)\sum_{j=1}^{m}(Z_j(x) - (1/m)\sum_{i=1}^{m} Z_i(x))^2$) as the first term of our robustness estimator. A smaller $\phi(x)$ indicates a smoother decision boundary.

2) Modeling Adversarial Gradient: Generally, there are three types of gradients: $(\partial L(x)/\partial x)$, $(\partial F(x)/\partial x)$, and $(\partial Z(x)/\partial x)$, which correspond to the crafting of adversarial examples based on FGSM (BIM), JSMA-F, and JSMA-Z methods, respectively. Specifically, $(\partial L(x)/\partial x) = (\partial L/\partial F)(\partial F/\partial Z)(\partial Z/\partial x)$ and $(\partial F(x)/\partial x) = (\partial F/\partial Z)(\partial Z/\partial x)$. Note that other attacks such as C&W are also indirectly impacted by those gradients. To reduce the effectiveness of all created examples, one possible approach is to just minimize the common factor of the three gradients—$(\partial Z(x)/\partial x)$. This motivates us to incorporate the mean of absolute value of $((\partial Z(x))/\partial x)$ across all input dimensions of an image for all classes (i.e., $g(x) = (1/mn)\sum_{j=1}^{m}\sum_{i=1}^{n} |((\partial Z_j(x))/\partial x_i)|$) into the robustness estimator.

Finally, robustness estimator can be expressed as follows:

$$\mathcal{R}(x) = \alpha \cdot g(x) + (1-\alpha) \cdot \phi(x) \tag{3}$$

where α indicates the relative importance of adversarial gradient and decision boundary ($\alpha \in [0, 1]$). A smaller $\mathcal{R}$ indicates that the model can exhibit stronger resistance to adversarial attacks if the accuracy can be guaranteed.

Algorithm 1 Search Process for CRs

```
Data:
$Z$; // DNN model
$x_{\text{train}}, y_{\text{train}}, x_{\text{test}}, y_{\text{test}}$; // training and testing dataset
$Acc$; // normal testing accuracy
$T$; // threshold of accuracy
Result:
$\mathcal{R}(g, \phi)$; // robustness table
// get number of neurons
$N_1, M_1 \leftarrow$ number of presynaptic, postsynaptic neurons in layer 1
$N_2, M_2 \leftarrow$ number of presynaptic, postsynaptic neurons in layer 2
// initialize compression rate for 2-layer hash classifiers
$CR_1 \leftarrow (N_1 \cdot M_1)^{-\frac{1}{2}}$, $CR_2 \leftarrow \frac{1}{2}$
// initialize searching sets
$\mathcal{R}, \{g\}, \{\phi\}, \{\text{accuracy}_h\} \leftarrow \emptyset$
// setup boundary of compression rate for searching
while $CR_1 > (N_1 \cdot M_1)^{-1}$ do
    while $CR_2 > (N_2 \cdot M_2)^{-1}$ do
        $Z_h \leftarrow$ hashTraining($x_{\text{train}}, y_{\text{train}}, Z, CR_1, CR_2$)
        $\{\text{accuracy}_h\} \leftarrow$ hashTesting($x_{\text{test}}, y_{\text{test}}, Z_h$)
        $\{g\} \leftarrow \{g\} \cup g(x_{\text{test}}, Z_h)$
        $\{\phi\} \leftarrow \{\phi\} \cup \phi(x_{\text{test}}, Z_h)$
        $CR_2 = \frac{CR_2}{2}$
    $CR_1 = \frac{CR_1}{2}$
$\{g\}, \{\phi\} \leftarrow$ normalize($\{g\}, \{\phi\}$) : $[min, max] \rightarrow [0, 1]$
foreach $a^i \in \{accuracy_h\}$ do
    $A^i_{\text{degradation}} \leftarrow Acc - a^i$
    if $A^i_{degradation} < T$ then
        // robustness estimation based on 3
        $\mathcal{R} \leftarrow \mathcal{R} \cup (\alpha \cdot \{g\}^i + (1-\alpha) \cdot \{\phi\}^i)$
    else
        $\mathcal{R} \leftarrow \mathcal{R} \cup 1$
```

C. Implementing Defensive Hash Classifier

Developing "hash compression" for the purpose of hardening DNN models should satisfy three constraints: 1) soften the boundary as much as possible; 2) degrade the accuracy very marginally; and 3) make the implementation cost low. To meet these requirements, we rearchitect the compression mechanism as follows.

1) Compressed Layer Selection: We only identify a few layers to compress to reduce the accuracy drop and implementation cost. The selected layers should contain enough number of parameters for compression purpose. In our practice, we find that compressing two fully connected layers that are close to the output can better balance the accuracy and the smoothness of decision boundary, given their importance to final decision-making. This is quite different from the goal of original hash compression: compressing all layers to achieve the highest entire CR.

2) CR Selection: For selected layers, we attempt to compress them aggressively with extremely high CRs so that the decision boundary can become softened. To select optimized CR configuration for multiple layers, we adopt robustness estimator to guide and accelerate the search process. Note that the accuracy constraint is used to terminate the search. As we shall show later, our adopted CR is much higher than that of normal compression, e.g., 1/8192 versus 1/64.

3) Dedicated Low-Cost Training Policy: We borrow the concept of transfer learning to accelerate the learning process.

1) *Step 1:* Replace selected layers of a pretrained DNN model, e.g., the last two fully connected layers, with untrained HashNet–hash bucket and index table with a corresponding CR for each selected layer.
2) *Step 2:* Initialize compressed weights of replaced layers and retrain the entire model for a few training epochs to recover the accuracy.

The hash compressed model can be quickly converged with very low training cost (i.e., 2–3 epochs here) since we only manipulate the last two fully connected layers and leave all the other layers untouched (e.g., convolutional layers for feature extraction).

To summarize, the key idea is to quickly design and install a more robustness DNN classifier for any pretrained DNN model (instead of its original classifier). We name the new classifier as "defensive hash classifier" and the corresponding DNN model as "defensive hash model." To determine the defensive CRs at selected layers for defensive hash classifier, we further propose a detailed search algorithm based on robustness estimator. As shown in Algorithm 1, for each CR configuration (CR$_1$ and CR$_2$), we perform steps 1 and 2 of dedicated low-cost training policy. Once the training is finished, we characterize the testing accuracy and the values of the two terms in robustness estimator at (CR$_1$ and CR$_2$). Then, we scale CR$_1$ (CR$_2$) by two and repeat the above process until reaching the limit of defined CRs. To balance the importance of gradient and decision boundary (i.e., their importance is only controlled by α), we normalize $g(x)$ and $\phi(x)$ to [0, 1].

IV. Factor II: GI

In this section, we propose free-retraining "GI" to address the remaining gradient problem, i.e., suppress and obfuscate the gradient simultaneously. This method, as the second line of defense, can assist the defensive hash classifier to effectively mitigate adversarial attacks.

A. Theoretical Analysis of GI

The "GI" method should satisfy the following conditions: 1) low implementation cost without extra training; 2) high defense efficiency against both direct-gradient- and indirect-gradient-based attacks; and 3) marginal accuracy reduction. To achieve these goals, we investigate the basic idea of defense distillation and propose a set of "GI" methods.

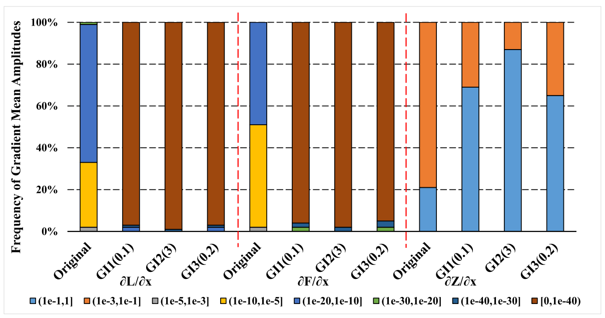


Fig. 2. Impact of different GIs on the amplitude of three types of the adversarial gradient. The values in brackets are inhibition coefficient ϵ for different GIs. The data are collected using all 10 000 images from the MNIST test data.

In defensive distillation [9], "teacher model," which generates the soft labels to train the distilled model, needs to apply a modified softmax function with a high temperature T, i.e., $T = 100$, at the training stage. The modified softmax function can be expressed as follows:

$$F_i(x,T) = \frac{e^{Z_i/T}}{\sum_j^m e^{Z_j/T}} \tag{4}$$

where m is the number of classes. In the training phase, the distilled model uses the same T with the teacher model, whereas in the testing stage, the distilled model needs to reset a smaller T, i.e., $T = 1$, to saturate the softmax. This leads to significantly reduced adversarial gradient $(\partial F/\partial x)$ so that it can address the JSMA-F attack efficiently. Since the goal of defensive distillation is to saturate softmax to reduce gradient, our observation is that we do not necessarily change the expression of original softmax and spend two-step training cost to achieve this goal. If we assume that DNN is trained with the original softmax F $(T = 1)$ in (4) and cross-entropy loss function $(L(x) = -\sum_i^m t_i \log(F_i)$, where t is the hard label). $(\partial L/\partial x)$ can be expressed as follows:

$$\frac{\partial L}{\partial x_i} = \sum_j^m \sum_k^m (F_k - t_k)\frac{\partial Z_j}{\partial x_i}. \tag{5}$$

$(F_k - t_k)$ should be a dominant factor for gradient reduction. Ideally, if every output of softmax function F_k is infinitely approaching t_k, i.e., 100% confidence on the ground truth class and 0% confidence on others), $(\partial L/\partial x_i)$ would be 0. In other words, directly saturating the original softmax can also vanish the gradient $(\partial L/\partial x)$. Similarly, $(\partial F/\partial x)$ can be presented as follows:

$$\frac{\partial F_k}{\partial x_i} = \sum_j^m \frac{\partial F_k}{\partial Z_j}\frac{\partial Z_j}{\partial x_i} = \sum_j^m F_k(\mathbb{1}_{j=k} - F_j)\frac{\partial Z_j}{\partial x_i} \tag{6}$$

where $\mathbb{1}$ is the indicator function and $F_k(\mathbb{1}_{k=j} - F_j)$ is the key element to reduce the gradient $(\partial F/\partial x)$. The saturated softmax results can make this element close to 0, thereby again suppressing the gradient as much as possible.

Therefore, pushing the original softmax to its saturation region can directly ease the adversarial gradient—$(\partial L/\partial x)$ and $(\partial F/\partial x)$. Since softmax is a monotonically increasing function of logits Z, if we can enlarge the absolute value of Z

through manipulating the weights (i.e., adding small changes in the direction of weight), the softmax can reach the saturation region with marginal accuracy reduction.

B. GI as Defense

According to the above analysis, we design three types of weight transformation to enlarge the weight w, which can be depicted as follows:

$$\phi_1(w) = w + \epsilon \cdot \text{sign}(w) \tag{7}$$

$$\phi_2(w) = (1+\epsilon) \cdot w \tag{8}$$

$$\phi_3(w) = w + \epsilon \cdot \text{uniform}(0,1) \cdot \text{sign}(w) \tag{9}$$

where ϵ is the inhibition coefficient. Specifically, $\phi_1(w)$ aims at adding the perturbation constant ϵ to weights along the direction of each weight. $\phi_2(w)$ further introduces the perturbation proportional to the magnitude of each weight (amplitude is controlled by ϵ) on top of $\phi_1(w)$. $\phi_3(w)$ injects random perturbation that follows the uniform distribution into the weight. We name the three weight transformations—$\phi_1(w)$, $\phi_2(w)$, and $\phi_3(w)$ as "GI" methods—GI$_1$, GI$_2$, and GI$_3$, respectively. Besides vanishing the adversarial gradients for direct gradient-based attacks, we expect that $\phi_3(w)$ (GI$_3$) with random distortion can also mislead the iterative search and optimization process of other types of attacks, such as C&W attacks, so as to improve the defense efficiency. If we simply assume that $Z(x) = wx$, then the input transformation defense techniques, such as "feature squeezing," can be modeled as $Z(x) = wg(x)$, where $g(x)$ is the input preprocessing such as bit depth reduction or spatial filtering. Similarly, our method can be modeled as $Z(x) = \phi(w)x$, where $\phi(w)$ is a set of GI methods. Note that the deployment of GIs occurs at the testing stage, so it only impacts the crafting process of adversarial examples rather than the training process. Moreover, GIs can be directly applied to the weights of pretrained DNN model without involving any training process. To achieve the best defense efficiency but lowest accuracy reduction of original DNN model, GIs can be quickly implemented in the last few fully connected layers close to the output, due to the moderate number of weights and the importance to decision-making.

Gradient Reduction Effectiveness: We now observe gradient reduction effectiveness of our three types of GIs on MNIST dataset. To fairly observe the efficiency of gradient reduction for three GIs, we adjust ϵ for each GI to make sure that the average perturbation added to the weight for three GIs is similar, i.e., $\epsilon = 0.1$ for $\phi_1(w)$ (GI$_1$), $\epsilon = 3$ for $\phi_2(w)$ (GI$_2$), and $\epsilon = 0.2$ for $\phi_3(w)$ (GI$_3$), for the average perturbation 0.1. Fig. 2 shows the statistics of gradient amplitude for the original DNN model and model with different GIs. As shown in Fig. 2, all GIs reduce the average absolute value of gradients $(\partial L/\partial x)$ and $(\partial F/\partial x)$ remarkably. The largest proportion of $(\partial L/\partial x)$ of the original DNN is ~66% at $(1e^{-20}, 1\text{e}^{-10}]$, while that of DNN with GI_1, GI_2, and GI_3 are ~97%, ~99%, and ~97%, at $[0, 1e^{-40})$, respectively. This can be translated into $\sim 10^{30}$ gradient reduction for $(\partial L/\partial x)$. We also observe the similar gradient reduction for $(\partial F/\partial x)$. Among the three GIs, GI_2 achieves the best result, e.g., ~99% at $[0, 1\text{e}^{-40})$. However, we also find that all GIs cannot reduce but increase the gradient $(\partial Z/\partial x)$ slightly when compared with the original model. For example, the proportion of gradient at $(1e-1, 1]$ is increased from ~21% on original DNN to ~69%, ~87%, and ~65% on DNN with GI$_1$–GI$_3$, respectively. Such minor increase is almost negligible when compared with the far more significant reduction of $(\partial L/\partial x)$ and $(\partial F/\partial x)$. Therefore, GIs show great potentials to mitigate adversarial attacks.

V. Evaluation

A. Experimental Setup

1) Benchmarks and Model Configurations: We choose MNIST, CIFAR-10, and ImageNet-subset as our evaluation benchmarks. For MNIST and CIFAR-10, we use a representative seven-layer DNN model (i.e., Conv64(3 × 3)-Conv128(3 × 3)-MaxPooling-Conv128(3 × 3)-Conv64 (3 × 3)-MaxPooling-FC256-FC256-FC10) and popular DNN architecture–VGG-19 [16], with their accuracy reaching 99.50% and 90.09%, respectively. Adversarial training, which relies on enhanced decision boundary for defense, has been proved to be difficult at ImageNet scale to ensure high defense efficiency and marginal accuracy degradation simultaneously [20], [25], [39], e.g., 1.5%–3.9% defense efficiency. Similarly, our defensive hash classifier also has the same issue because of the limited adjustable space of decision boundary for all classes with the requirement of marginal accuracy degradation in a super multidimensional classification task. Therefore, to better test our method on a large dataset with affordable simulation cost, we choose a subset of Imagenet, e.g., with first 50 classes. This assumption is also reasonable considering that in practice, the dataset of a common classification task (e.g., road sign) may not be as complex as a complete Imagenet with 1000 classes. We use popular Inception-V3 [18] on the ImageNet-subset with 50 classes to achieve competitive top-1 accuracy 80.95% (78% for ImageNet with 1000 classes).

2) Defense Deployment: In the defensive hash classifier deployment stage, we replace the last two fully connected layers with defensive hash classifiers with the same structure. Then, we set $\alpha = 0.5$ and $T = 1\%$ to search optimal defensive CR in algorithm on the MNIST dataset. As expected, a higher CR could produce a smaller $\mathcal{R}$ that indicates the stronger resistance to adversarial attacks but incur higher accuracy degradation. As a result, we choose CR combination—$(1/16 \cdot (N_1 \cdot M_1)^{-1/2}, 2/N_2)$—to offer the optimal balance between defense efficiency and accuracy in general. To avoid the time cost from the search process, we directly use this empirical result on VGG-19 (CIFAR-10) and Inception-V3 (ImageNet). Note that we add a fully connected hashed layer between the output layer and the last-second layer with 512 units for InceptionV3 since it only has one fully connected layer. Then, we retrain the defensive hash model for three epochs. In the GI deployment stage, we apply our GI$_1$–GI$_3$ methods in all the last three layers on three types of benchmarks. As shown in Table I, the proposed defense techniques do not compromise the original classification accuracy, e.g., only marginal degradation (<1% on MNIST and CIFAR-10, ~1–2% on ImageNet). We use "TensorFlow" as our deployment

TABLE I

ACCURACY (%) OF CLEAN EXAMPLES FOR ORIGINAL DNN, DEFENSIVE HASH CLASSIFIER ONLY, GI ONLY, AND COMBINED SOLUTION ON MNIST, CIFAR-10, AND IMAGENET (SUBSET)

	Original DNN (baseline)	$hash$	GI_1	GI_2	GI_3	$hash + GI_1$	$hash + GI_2$	$hash + GI_3$
MNIST	99.5	99.25	99.47	**99.5**	99.44	99.21	99.26	99.25
CIFAR-10	90.09	89.35	89.95	**90.05**	89.83	89.27	89.82	89.2
ImageNet	80.95	**80.11**	79.30	79.51	79.27	79.18	79.45	79.02

framework and conduct all the simulations on NVIDIA GTX 1080 GPUs.

3) Adversarial Examples Crafting: We craft various types of adversarial examples under two different settings: eight under the white-box setting and two under the black-box setting, for conducting comprehensive evaluations.

a) White-box setting: For low-cost untargeted attacks such as FGSM and BIM, we use all test sets from each benchmark to craft adversarial examples. For high-cost targeted attacks including two JSMAs (JSMA-Z and JSMA-F), C&W, and EAD, we select the first 100 correctly predicted images from the test dataset of each benchmark. For MNIST and CIFAR-10, we set all other classes ($t \neq$ Label) as the attack target (900 adversarial samples) for each method. For ImageNet, we use next class as the target ($t =$ Label $+ 1$ mod #classes) for C&W and EAD (JSMA is too costly for complex DNN models and datasets). We directly adopt the implementations of FGSM, BIM, MIM, JSMA, C&W, and EAD from the Cleverhans library [40]. For strong white-box adaptive attack—BPDA, the basic idea is to find a reasonable gradient by leveraging a function replacement (approximation) to fix obfuscated gradient incurred by our defense, given the attacker knows the defense method. In our case, we attempt to generate a "normal-size" gradient to evaluate our defense. By assuming that the attacker knows that the effectiveness of GI (or close to zero gradient) lies in the saturation of softmax, we replace the softmax with sigmoid function, to generate the BPDA adversarial examples. This is because using sigmoid activation not only preserves the testing accuracy but also outputs meaningful gradients for the adversary. Considering the simulation cost, we use 1000 images from the test data of MNIST and CIFAR-10 and 100 test images of ImageNet to craft adaptive adversarial examples (all untargeted attacks). Note that since the defense principle of "defensive decision boundary" could be similar to that of the BPDA-unbroken "adversarial training," we follow the same method in [25] to evaluate our "defensive decision boundary" and the combined solutions.

b) Black-box setting: For substitute model-based black-box attack [13], we use a sample DNN architecture as the substitute model for MNIST and CIFAR-10, i.e., FC1000-FC1000-FC10 for MNIST and Conv32(5×5)-Conv32(5×5)-FC512-FC10 for CIFAR-10. We use the first 150 images from the test dataset and run five Jacobian augmentation epochs to train the substitute model (codes are provided from Cleverhans library [40]). We implement FGSM and C&W attack on the substitute model for untargeted or targeted transfer attacks in the black-box model. For the ZOO black-box attack, we run the ZOO-ADAM method with untargeted attack and targeted attack provided in [26] to generate adversarial examples (1000 examples for untargeted attack and 9×100 examples for targeted attack).

B. Results and Analysis

1) Baselines and Evaluation Metrics: We evaluate three different combinations of the proposed defensive hash classifier and GI, i.e., hash($\mathrm{CR}_1, \mathrm{CR}_2$) + $\mathrm{GI}_i(\epsilon)$, on each dataset with detailed parameters shown in Table II. The baselines selected for attack success rate comparisons are the original DNN without any defense and DNN with defensive hash classifier only—hash($\mathrm{CR}_1, \mathrm{CR}_2$) and with GIs only—$\mathrm{GI}_i(\epsilon)$. Note the latter two baselines use the same parameter configuration as their respective "combo" solution. Meanwhile, we use DeepFool [41] to compute minimal attack distance (distance between inputs and decision boundary) to measure the robustness of DNN further and validate our defensive hash model for Condition 2 (see Section III-A).

2) Robustness: As shown in Table II, all the combined solutions significantly improve the DNN's robustness (enlarge distance between inputs and decision boundary) when compared with baselines. For example, the robustness of original model is 0.0863, while that of hash + GI_{1-3} is 0.1259, 0.1269, and 0.1275 on MNIST data. Besides, we can observe that adding GIs to defensive hash model (robustness of the defensive hash model is 0.1253) does not change the distance between inputs and decision boundary.

3) Defense Under White-Box Attacks: In order to analyze the impact of the three different defense setup: hash only—hash, GI only—GI_i, and combined solutions—hash + GI_i, $i = 1, 2, 3$. As shown in Table II, for the state-of-the-art direct gradient-based attack—MIM, its attack success rate is reduced significantly from 99.1% (original) to 3.02% (hash + GI_1), 1.15% (hash + GI_2), and 3.2% (hash + GI_3) on MNIST, which well preserves the capability of their respective GI (3.35%, 1.02%, and 3.28%) when mitigating such an attack. For other direct gradient-based attacks, e.g., FGSM, BIM, and JSMA-F (except for JSMA-Z), we can observe similar results. Note that all GI only solutions are not good at mitigating JSMA-Z attack because they are unable to reduce the gradient ($\partial Z/\partial x$) (see Fig. 2). However, defensive hash classifiers can address it for GIs effectively, i.e., 25.33% (hash+GI_1), 28.44% (hash+GI_2), and 24.44% (hash + GI_3) versus 90.22% (original) on MNIST. For indirect gradient-based attacks, i.e., C&W and more advanced EAD, GI_1 or GI_2 only defense becomes completely helpless, while both defensive hash classifier and GI_3 can mitigate them, i.e., 56.67% (78.44%) and 24.33%/(54.56%) for C&W (EAD) on hash only and GI_3 only defense, respectively, on MNIST. This also indicates that defensive decision boundary and random distortion could effectively mislead the iterative search and optimization process of these attacks. We

TABLE II

SUCCESS RATE OF WHITE-BOX ADVERSARIAL ATTACKS FOR ORIGINAL DNN, DEFENSIVE HASH CLASSIFIER ONLY, GI ONLY, AND COMBINED SOLUTION (INCLUDING THE ROBUSTNESS OF DNN MODEL MEASURED BY DEEPFOOL) ON MNIST, CIFAR-10, AND IMAGENET

Dataset	Model	Robustness (DeepFool)	Attack Success Rate(%)								
			FGSM	BIM	MIM	JSMA-F	JSMA-Z	C&W	EAD	BPDA	AVG
MNIST	original DNN	0.0863	72.8	97.82	99.1	95.44	90.22	100	100	92.2	93.45
	hash(1/4096,1/128)	0.1253	62.66	87.32	86.23	60.89	32.22	56.67	78.44	25.6	61.25
	GI_1(0.1)	0.0857	3.28	3.14	3.35	0.33	89.11	100	100	83.5	56.55
	GI_2(3)	0.0868	**0.45**	**0.63**	**1.02**	**0.22**	90.56	100	100	89.5	47.8
	GI_3(0.2)	0.0872	3.45	3.24	3.28	0.44	89.44	24.33	54.56	83.7	32.8
	hash(1/4096,1/128)+GI_1(0.1)	0.1259	3.25	3.31	3.02	0.89	25.33	82.22	79.67	22.8	27.56
	hash(1/4096,1/128)+GI_2(3)	0.1269	0.91	0.95	1.15	0.44	28.44	75.33	76.56	25.2	26.12
	hash(1/4096,1/128)+GI_3(0.2)	0.1275	3.21	3.4	3.2	0.78	**24.44**	**5.44**	**52.56**	**22.1**	**14.39**
CIFAR-10	original DNN	1.73×10^{-3}	79.05	95.12	96.5	84.33	86.44	100	100	88.7	91.27
	hash(1/65536,1/2048)	2.49×10^{-3}	75.58	88.9	88.21	75.22	43.67	85.56	84.22	45.6	73.37
	GI_1(0.3)	1.72×10^{-3}	3.41	2.58	2.06	1.33	86.78	100	100	74.5	46.33
	GI_2(5)	1.69×10^{-3}	**1.42**	**1.35**	**1.32**	**0.56**	85.56	100	100	82.6	45.59
	GI_3(0.6)	1.74×10^{-3}	3.42	2.48	2.15	1.11	82.89	33.56	65.89	73.8	33.16
	hash(1/65536,1/2048)+GI_1(0.3)	2.51×10^{-3}	5.45	5.69	5.59	1.33	43.67	85.22	87.89	42.5	34.66
	hash(1/65536,1/2048)+GI_2(5)	2.49×10^{-3}	2.87	2.91	2.88	0.67	44.56	84.44	85.33	46.3	33.75
	hash(1/65536,1/2048)+GI_3(0.6)	2.52×10^{-3}	5.17	6.89	5.93	1.22	**36.11**	**23.67**	**62.67**	**42.2**	**22.98**
ImageNet	original DNN	4.23×10^{-4}	64.52	75.72	90.82	-	-	100	100	92	87.18
	hash(1/8192,1/256)	6.49×10^{-4}	45.28	46.54	75.62	-	-	68	85	56	62.74
	GI_1(0.4)	4.52×10^{-4}	16.12	18.8	17.37	-	-	100	100	87	56.55
	GI_2(6)	4.58×10^{-4}	12.76	15.68	16.48	-	-	100	100	84	54.82
	GI_3(0.8)	4.39×10^{-4}	14.44	17.24	16.94	-	-	42	78	85	42.27
	hash(1/8192,1/256)+GI_1(0.4)	6.55×10^{-4}	12.72	16.34	15.66	-	-	67	87	62	43.45
	hash(1/8192,1/256)+GI_2(6)	6.54×10^{-4}	13.56	15.72	15.53	-	-	69	87	59	43.3
	hash(1/8192,1/256)+GI_3(0.8)	6.67×10^{-4}	**12.47**	**15.43**	**14.91**	-	-	**38**	**75**	**52**	**34.63**

TABLE III

SUCCESS RATE OF BLACK-BOX ATTACK FOR ORIGINAL DNN, DEFENSIVE HASH CLASSIFIER ONLY, GI ONLY, AND COMBINED SOLUTION ON MNIST AND CIFAR-10

Dataset	Model	Attack Success Rate(%)			
		Untargeted		Targeted	
		sub-FGSM	ZOO	sub-CW	ZOO
MNIST	original DNN	62.7	100	32.22	83.33
	hash	62.1	100	28.89	82.67
	GI_3	62.9	61.6	27.78	14.44
	hash+GI_3	62.5	**45.8**	28.11	**13.89**
CIFAR-10	original DNN	52.5	100	14.44	92.22
	hash	52.1	100	12.22	91.89
	GI_3	50.4	52.5	13.11	30.11
	hash+GI_3	51.8	**50.1**	10	**26.67**

also found that the combined solution—hash + GI_3—can further lower the attack success rate, i.e., C&W 5.44% and EAD 52.56%. For the adaptive attack, BPDA, as expected, all GIs are incapable of mitigating it on MNIST [attack success rate: 83.5% (GI_1), 89.5% (GI_2), and 83.7% (GI_3)], given that GIs are still based on gradient obfuscations though no retraining is required. In contrast, our defensive hash classifier (as well as the combined solution) can significantly reduce the attack success rate of BPDA to 22.8% (hash + GI_1), 25.2% (hash + GI_2), and 22.1% (hash + GI_3), respectively. This is because our defensive hash classifier can reconstruct defensive decision boundary [8], instead of generating obfuscated gradients, thus, it is more resistant to BPDA. Therefore, hash+GI_3 delivers the best defense effectiveness to all types of adversarial attacks, e.g., reduce the average attack success rate from 93.45% to 14.39% on MNIST. We also observe the similar results on CIFAR-10 and ImageNet—the average attack success rate is dropped sharply from original 91.27% (87.18%) to 22.98% (34.63%) by hash + GI_3 on CIFAR-10 (ImageNet).

4) *Defense Under Black-Box Attacks:* As shown in Table III, we observe that our hash, GI_3, and hash + GI_3 exhibit limited efficiency against the substitute model-based black-box attacks (i.e., untargeted sub-FGSM attack and targeted sub-CW attack). For example, the success rate of sub-FGSM attack on all models is ~62% on MNIST. However, the transferability of substitute model-based black-box attack is relatively bad (especially targeted sub-CW attack) compared with the state-of-the-art ZOO attack (note that our experimental results are consistent with [26]). For example, sub-CW attack achieves a 32.22% (14.44%) success rate, whereas ZOO attack achieves 83.33% (92.22%), on original DNN for MNIST (CIFAR-10). Thereby, the threat of substitute model-based black-box attack is not that significant for DNN in practice, at least for targeted attack with more substantial threats. We believe that the state-of-the-art ZOO black-box attack is a reasonable measurement standard for evaluating the robustness of the defensive model against black-box attacks. In Table III, we found that our GI_3 achieves significant mitigation efficiency to both untargeted and targeted ZOO attacks for both datasets. For example, the success rates of untargeted and targeted ZOO attacks are reduced from 100% (original DNN) to 61.6% (GI_3) and 83.33% (original DNN) to 14.44% (GI_3) on MNIST, respectively. ZOO attack is to directly approximate gradient of the targeted model, but the approximated gradient is still small (near 0) on GI_3. This explains why our GI_3 can mitigate the ZOO-based black-box attack significantly. Because of the impact of GI_3, our combined solution (hash + GI_3) can also mitigate ZOO attack on MNIST [success rate of untargeted (targeted) ZOO attack is 45.8% (13.89%)]. We can observe similar results on CIFAR-10. Our combined solution (hash + GI_3) reduce the success rate of untargeted (targeted) ZOO attack from 100% (92.22%) to 50.1% (26.67%).

5) *Comparison With PGD-Based Adversarial Training:* Since "PGD-based adversarial training" is also to harden the DNN model but can only resist L_∞-bounded white-box attacks [8], [42], we focus on evaluating the attack success

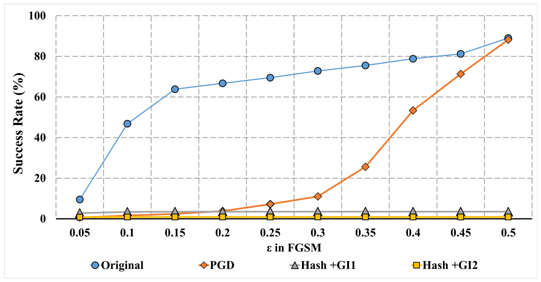


Fig. 3. Success rate of FGSM attack with various values of ϵ on PGD-based adversarial training versus combined defense method.

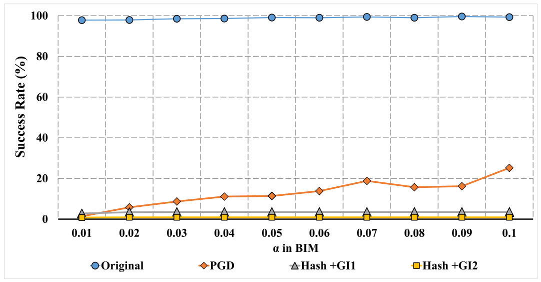


Fig. 4. Success rate of BIM attack with various values of α on PGD-based adversarial training versus combined defense method.

rate of FGSM and BIM attacks. Since the goal of the two state-of-the-art adversarial training methods [32], [33] is to reduce the computational cost of PGD-based adversarial training, rather than enhancing the defense efficiency, we only compare our method with the original PGD-based adversarial training. The success rate variance of FGSM attack with varied ϵ in the original model and the proposed defensive combined models is shown in Fig. 3 (we do not show the result of hash + GI_3, because it is very similar with hash + GI_1 in resisting the FGSM and BIM attacks). PGD slightly surpasses hash + GI_1 and hash + GI_2, at $\epsilon \leq 0.2$ and $\epsilon = 0.05$, respectively. However, when ϵ changes from 0.2 to 0.5 (0.1 to 0.5), our hash + GI_1 (hash + GI_2) outperforms PGD model remarkably. In fact, our methods are barely related to the ability of FGSM adversary, while PGD is highly dependent on the ϵ of PGD internal adversary. If ϵ of exotic adversary is larger than that of PGD internal adversary ($\epsilon > 0.3$), it can easily lead to degraded mitigating effectiveness. We also notice a similar trend in BIM attack. As shown in Fig. 4, different values of α barely impact the mitigating effect of our methods to BIM attack, while PGD is very sensitive to the value of α in exotic (practical) BIM adversary. As α of exotic BIM increases, the resistance efficiency of PGD can get worse. In practice, we cannot guarantee that the adversarial settings of the exotic adversary are similar to that of adversarial training. For the computational cost, adversarial training-based defense first needs to generate a sufficient number of adversarial examples as extra training inputs for each training epoch. Then, the model needs to be trained from the scratch (e.g., 100 epochs for training VGG19) to obtain a defensive model with combined adversarial and clean training inputs, incurring expensive computational cost. While the latest adversarial training (see [32]) can reduce the cost of crafting adversarial inputs, it still involves an entire training process. In contrast, our method requires neither crafting expensive adversarial examples as the extra training data nor training from scratch (e.g., just three retraining epochs for fun-tuning), to quickly install hash defensive classifier (GI is training-free). Therefore, compared with adversarial training, our solution is more lightweighted.

TABLE IV
ATTACK SUCCESS RATE OF ADVERSARIAL EXAMPLES FOR OUR HASH + GI_3 AND TWO INPUT TRANSFORMATION-BASED DEFENSE ON MNIST AND CIFAR-10

Dataset	Defense	Attack Success Rate (%)			
		FGSM	JSMA-Z	C&W	BPDA
MNIST	bit-depth (1 bit)	9.1	33.51	12.67	88.2
	median smooth (2*2)	40.97	30.24	49.11	83.8
	hash+GI_3	**3.21**	**24.44**	**5.44**	**22.1**
CIFAR10	bit-depth (4 bit)	76.29	80.8	50.22	89.7
	median smooth (2*2)	50.34	60.12	53.56	86.1
	hash+GI_3	**5.17**	**36.11**	**23.67**	**42.2**

TABLE V
ACCURACY (%) OF CLEAN EXAMPLES FOR ORIGINAL DNN AND OUR COMBINED DEFENSE SOLUTION ON RESNET-34, RESNET-50, AND RESNET-101 ON CIFAR-10

Accuracy(%)	Model		
	ResNet-34	ResNet-50	ResNet-101
original DNN	90.96	92.26	92.85
hash+GI_3	90.02	91.7	92.32

TABLE VI
ATTACK SUCCESS RATE (%) OF ADVERSARIAL EXAMPLES FOR DNNS (RESNET-34/RESNET-50/RESNET-101) WITHOUT AND WITH OUR COMBINED DEFENSE SOLUTION ON CIFAR-10

Attack Success Rate (%)	Model					
	ResNet-34		ResNet-50		ResNet-101	
	ori.	hash+GI_3	ori.	hash+GI_3	ori.	hash+GI_3
FGSM	84.8	6.3	80.4	7	68.1	7.4
BIM	92.5	3.4	92.9	3.1	84.8	3.7
MIM	92.6	3	93.9	4.7	85	4.5
C&W	100	25	100	24	100	24
EAD	100	70	100	74	100	76
BPDA	92	27	93	26	90	25
AVG	93.65	22.45	93.36	23.13	87.98	23.43

6) Comparison With Input Transformation: We compare the defense effectiveness of our hash + GI_3 with that of two representative input transformation-based defense: squeezing input's bit depth and median smoothing [10] under four typical attacks crafted from MNIST and CIFAR-10. As shown in Table IV, our hash + GI_3 achieves significantly lower attack success rates than the two input transformation-based defense on all attacks and datasets. Furthermore, two input transformations cannot mitigate BPDA, e.g., the attack success rate of BPDA is more than 80% on MNIST and CIFAR-10, because the input transformation-based defense is mainly based on gradient obfuscation and can be easily broken by BPDA [25].

C. Discussion

In this section, we discuss the generalization of our combined defense methods on deeper models and compare our defensive hash compression with other compression techniques in the context of adversarial robustness.

1) Defense Against Adversarial Attacks on Deeper Networks (ResNets): We evaluate our combined solution across three different ResNet models: ResNet-34, 50, and 101 using CIFAR-10. Since ResNet only has one fully connect layer (output layer), we introduce an additional fully connected

TABLE VII

ATTACK SUCCESS RATE (%) OF ADVERSARIAL EXAMPLES OF ORIGINAL DNN AND PRUNING/QUANTIZED MODELS ON MNIST

Model	Attack Success Rate (%)		
	FGSM	JSMA-Z	C&W
original DNN	72.8	90.22	100
Pruning	72.9	90.18	100
Quantization (8 bits)	73.51	90.05	100
Quantization (3 bits)	74.84	89.2	100

hashed layer between the output layer and last-second layer with 512 neurons and apply the optimized CR combination $(1/16 \cdot (N_1 \cdot M_1)^{-1/2}, 2/N_2)$, i.e., (1/8192, 1/256), to two fully connected layers to train the defensive hash model in three epochs. Then, we apply GI_3 with $\epsilon = 0.2$ to the well-trained defensive hash model. Table V reports the accuracy of clean examples on models protected without (original) and with our combined defense solution. The results show that our hash + GI_3 incurs very marginal accuracy loss (e.g., $<1\%$) compared with original baseline accuracy on all three ResNet models. For defense effectiveness, as shown in Table VI, our hash + GI_3 can significantly drop the attack success rate on ResNet-34/50/101 under six white-box attacks, e.g., from the original 93.65%, 93.36%, 87.98% to 22.45%, 23.13%, and 23.43% on three ResNet models. This indicates that our defense can be well generalized to large DNN models.

2) Defense Effectiveness of Other Compression Techniques: We also explore whether other model compression techniques, e.g., pruning and quantization, could enhance the DNN model's adversarial robustness. As shown in Table VII, the standard pruning and quantization [28] (with two different bit widths—3 and 8) achieve almost no defense effectiveness against three basic adversarial attacks. We believe that this is because they usually can only slightly reshape and share model weights based on given value ranges, e.g., from "well-trained" or "half-trained" weights, to maintain the accuracy and thus are unable to substantially change the decision boundary. On the other hand, our hash compression performs a random weight sharing technique to randomly group untrained weights into the respective cluster and hence can reshape decision boundary significantly when CR is high.

VI. CONCLUSION

As DNNs are subject to ever-increasing adversarial input-based security concerns, this work investigates how to design a comprehensive defense framework to mitigate a wide range of adversarial attacks at low cost. We consider both defensive decision boundary and small gradient as two critical factors to address attacks with different natures. For the first aspect, we found that hash weight sharing, a DNN model compression technique dedicated to storage overhead reduction, can be wisely used to reconstruct a defensive decision boundary to prevent adversarial attacks. To satisfy the requirements of defensive decision boundary, marginal accuracy degradation, and low implementation cost, we redesign the original "HashNet" as the defensive hash classifier. For the second aspect, we propose a set of retraining-free "GI" methods to extremely suppress adversarial gradients while obfuscating the search and optimization process of strong attacks like C&W attacks. Finally, we orchestrate the defensive hash classifier and "GI" for a combined defense solution that can effectively resist most of the traditional white-box, strong adaptive white-box, and black-box attacks. We believe that our proposed solutions create a new paradigm of safeguarding DNNs from a radically different perspective by model compression with a focus on integrating defenses into compression and weight transformation of pretrained DNN models with low cost.

REFERENCES

[1] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[3] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–10.

[4] I. J. Goodfellow *et al.*, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–11.

[5] D. Yang *et al.*, "Realistic adversarial examples in 3D meshes," 2018, *arXiv:1810.05206*. [Online]. Available: http://arxiv.org/abs/1810.05206

[6] Z. Yang *et al.*, "Characterizing audio adversarial examples using temporal dependency," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–15.

[7] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 506–519.

[8] A. Madry *et al.*, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–28.

[9] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 582–597.

[10] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2018, pp. 1–15.

[11] Z. Liu *et al.*, "Feature distillation: DNN-oriented JPEG compression against adversarial examples," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 860–868.

[12] J. H. Metzen *et al.*, "On detecting adversarial perturbations," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–12.

[13] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.

[14] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer, "FireCaffe: Near-linear acceleration of deep neural network training on compute clusters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2592–2600.

[15] F. Tramèr *et al.*, "Ensemble adversarial training: Attacks and defenses," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–22.

[16] K. Simonyan *et al.*, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.

[17] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[20] A. Kurakin *et al.*, "Adversarial machine learning at scale," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1–17.

[21] Y. Dong *et al.*, "Boosting adversarial attacks with momentum," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9185–9193.

[22] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 372–387.

[23] N. Carlini and D. Wagner, "Defensive distillation is not robust to adversarial examples," 2016, *arXiv:1607.04311*. [Online]. Available: http://arxiv.org/abs/1607.04311

[24] P. Y. Chen *et al.*, "EAD: Elastic-net attacks to deep neural networks via adversarial examples," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–8.

[25] A. Athalye *et al.*, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 274–283.

[26] P. Y. Chen *et al.*, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 15–26.

[27] S. Han *et al.*, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[28] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*. [Online]. Available: http://arxiv.org/abs/1510.00149

[29] W. Chen, J. T. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 2285–2294.

[30] Z. Cao, M. Long, J. Wang, and P. S. Yu, "HashNet: Deep learning to hash by continuation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5609–5618.

[31] J. Wang *et al.*, "Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct./Nov. 2019, pp. 6629–6638.

[32] A. Shafahi *et al.*, "Adversarial training for free!," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 3358–3369.

[33] D. Zhang *et al.*, "You only propagate once: Painless adversarial training using maximal principle," in *Proc. 33rd Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–16.

[34] S. Gu *et al.*, "Towards deep neural network architectures robust to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1–9.

[35] S. Gui *et al.*, "Model compression with adversarial robustness: A unified optimization framework," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1285–1296.

[36] S. Ye *et al.*, "Adversarial robustness vs. Model compression, or both?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 111–120.

[37] F. Yu *et al.*, "Interpreting adversarial robustness: A view from decision surface in input space," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2019, pp. 1–15.

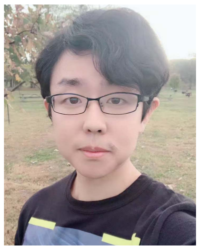
[38] D. Jakubovitz *et al.*, "Improving DNN robustness to adversarial attacks using jacobian regularization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 514–529.

[39] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," 2018, *arXiv:1803.06373*. [Online]. Available: http://arxiv.org/abs/1803.06373

[40] N. Papernot *et al.*, "Cleverhans v2. 0.0: An adversarial machine learning library," 2016, *arXiv:1610.00768*. [Online]. Available: http://arxiv.org/abs/1610.00768

[41] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016.

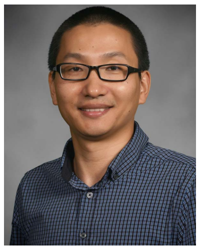
[42] Y. Sharma *et al.*, "Attacking the Madry defense model with L_1-based adversarial examples," 2017, *arXiv:1710.10733*. [Online]. Available: http://arxiv.org/abs/1710.10733



Qi Liu is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, PA, USA.

His works have been published widely on artificial intelligence (AI) and electronic design automation conferences, including Conference on Computer Vision and Pattern Recognition (CVPR), Design Automation Conference (DAC), International Conference on Computer Aided Design (ICCAD), Asia and South Pacific Design Automation Conference (ASP-DAC), and International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). His research focuses on secure, privacy preserving, and efficient deep learning.

Mr. Liu received the Best Paper Nomination from ASP-DAC 2018. He is also a Reviewer for journals, such as IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and *Neurocomputing*.



Wujie Wen (Member, IEEE) received the B.S. degree in electronic engineering from Beijing Jiaotong University, Beijing, China, in 2006, the M.S. degree in electronic engineering from Tsinghua University, Beijing, in 2010, and the Ph.D. degree from the University of Pittsburgh, Pittsburgh, PA, USA, in 2015.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, PA, USA. His research interests include reliable, secure, and energy-efficient deep learning, neuromorphic computing, and electronic design automation.