Improving Robustness Against Stealthy Weight Bit-Flip Attacks by Output Code Matching

Ozan Özdenizci[1,2] and Robert Legenstein[1]
[1] Institute of Theoretical Computer Science, Graz University of Technology, Graz, Austria
[2] TU Graz - SAL Dependable Embedded Systems Lab, Silicon Austria Labs, Graz, Austria
{ozan.ozdenizci,robert.legenstein}@igi.tugraz.at

Abstract

Deep neural networks (DNNs) have been shown to be vulnerable against adversarial weight bit-flip attacks through hardware-induced fault-injection methods on the memory systems where network parameters are stored. Recent attacks pose the further concerning threat of finding minimal targeted and stealthy weight bit-flips that preserve expected behavior for untargeted test samples. This renders the attack undetectable from a DNN operation perspective. We propose a DNN defense mechanism to improve robustness in such realistic stealthy weight bit-flip attack scenarios. Our output code matching networks use an output coding scheme where the usual one-hot encoding of classes is replaced by partially overlapping bit strings. We show that this encoding significantly reduces attack stealthiness. Importantly, our approach is compatible with existing defenses and DNN architectures. It can be efficiently implemented on pre-trained models by simply re-defining the output classification layer and finetuning. Experimental benchmark evaluations show that output code matching is superior to existing regularized weight quantization based defenses, and an effective defense against stealthy weight bit-flip attacks.

1. Introduction

While deep neural networks (DNNs) are becoming ubiquitous in artificial intelligence applications, they also have been proven to be highly vulnerable to a variety of malicious attack paradigms. One of the most widely studied aspect is the adversarial input attack, where hardly-perceptible and intentionally crafted input perturbations can lead to confident incorrect decisions for DNNs [13, 33]. A recently emerged category of attacks exposes the parameter space vulnerability of DNNs by negatively influencing the inference process at the deployment stage. It has been shown that information stored in the form of bits on dynamic random-access memory (DRAM) chips can be simply manipulated by flipping any bit precisely as desired via fault-injection techniques (e.g., row-hammer attacks [19]). As the weight parameters of widely deployed DNNs are generally stored on the DRAM due to their high memory demand, such hardware-induced attacks open malicious pathways to jeopardize DNN predictions by changing vulnerable parameters [7, 17, 22, 41].

There has been growing interest in developing *adversarial weight bit-flip attack* algorithms to identify vulnerable quantized DNN bits in simulations (cf. Section 2.1), in order to provide practical guidance for fault-injection attacks towards reaching malicious goals against expected DNN behavior. As physical bit-flipping may become time consuming and lead to abnormal background processes [14, 36], constraining the number of malicious bit-flips for efficient attacks is essential for the adversary. Going forward, recently proposed algorithms also consider finding minimal bits for *targeted* and *stealthy* weight bit-flip attacks, i.e., having a targeted negative impact on an attacked source (a single input sample [3] or samples belonging to a class [26]) while having almost no change in performance for the remaining test samples. From a DNN operation perspective, such a scenario is far more concerning as it becomes impossible to suspect any unusual activity if the network shows expected behavior for untargeted test samples.

To date, relatively little guidance is available for how to improve network robustness against adversarial weight bit-flip attacks (cf. Section 2.2). Our goal in this study is to improve robustness from a DNN architecture perspective, which would also be naturally compatible to potential hardware-driven solutions against fault-injection attacks. We particularly focus on more realistic, targeted attack scenarios, where the existence of the attack also can not be easily detected via the usual DNN behavior, i.e., targeted bit-flip attack algorithms with stealthiness [3, 26]. We approach this problem using an alternative output coding scheme for multi-class classification with DNNs, in comparison to the usual one-hot encoded output representations. The proposed output code matching networks predict class-

specific partially overlapping bit strings, which constitutes an effective defense against stealthy weight bit-flip attacks. Contributions of this study are summarized as follows:

- We present for the first time a DNN defense mechanism, *output code matching*, to improve robustness against stealthy weight bit-flip attacks in various targeted settings. Our approach is compatible with any DNN backbone by re-defining the output classification layer and finetuning pre-trained model weights.
- The proposed output code matching networks outperform state-of-the-art defenses and scale to large DNN architectures. Our ImageNet experiments show that targeted stealthy attacks on a ResNet-50 require up to $20\times$ and $5\times$ more bits to be attacked on our models with respect to vanilla networks, and networks trained with the state-of-the-art defense [16], respectively.
- We empirically demonstrate that the proposed framework is also applicable to networks trained with existing defenses, such as DNNs trained with regularized weight quantization (i.e., piecewise clustering [16]).

2. Related Work

2.1. Adversarial Weight Bit-Flip Attacks

Fault-injection attacks on memory systems [4, 19] pave the way towards manipulating DNN weights at their storage site. Several proposed hardware solutions against these attacks are shown to be insufficient defense mechanisms to date [8, 11, 14]. Hence it becomes important to investigate the sensitivity of DNN parameters that can lead to a malfunction by only a few bits of information change. Earliest works exposed failure modes of state-of-the-art DNNs when only a few weights [22] or activation functions [7] are altered, rendering the models useless by making random predictions. Stealthy attacks, initially posed for DNNs by [22,47], aim to make the models misbehave only for particular inputs (e.g., a single sample, or samples from a class) and retain behavior on other inputs as expected. While these earlier attacks were simulated for floating-point precision DNNs, later studies extended this problem also to quantized DNNs with compact weight representations [17, 41].

One of the first powerful algorithms to efficiently search for vulnerable bits in quantized DNNs is the untargeted bit-flip attack (BFA) that converts a DNN into a random output generator with a few bit-flips [24]. BFA was later extended in a trojan attack scheme [25] which impacts the expected behavior only for inputs with an embedded pattern that triggers the flipped bit trojans. Such trojan attacks require modification of the inputs to make the attack stealthy. Recently, targeted *stealthy bit-flip attacks* on quantized DNNs that do not require input sample modification (hence being realistic, however concerning) posed a novel threat. Specifically [26] proposed a class-to-class targeted BFA (T-BFA) in a stealthy setting, and [3] introduced a single-sample targeted attack with limited bit-flips (TA-LBF) while being stealthy for other samples. We detail these two attacks in Section 3.2 in the scope of our work.

2.2. Robustness Against Weight Bit-Flip Attacks

Earlier explorations of adversarial weight bit-flip attacks proposed weight quantization as a general defense mechanism [17, 41]. Subsequently, a limited amount of studies explored novel defenses for quantized DNNs by aiming to increase the necessary number of bit-flips for an attack such that the physical fault-injection process becomes possibly unrealistic. Harnessing adversarial examples during training have been the most effective defense against adversarial input attacks [23, 45], which was also translated to the domain of weight bit-flip attacks as a potential defense (i.e., random [31] and adversarial [32] bit-flip training). However these methods were found ineffective for large DNNs with millions of bits [16, 24]. From the perspective of adversarial input robustness and generalization, [40, 48] proposed training DNNs with adversarial weight perturbations (in line with the formal analyses on norm-bounded weight perturbations [35, 38]) which was similarly found to be an insufficient defense against recent weight bit-flip attacks [26].

An earlier defense proposed a weight reconstruction approach [21] with a benign accuracy trade-off. Differently from quantized DNNs, [27] adopted a binary DNN to increase the number of required bit-flips for attacks, at a higher cost of reduced benign accuracy for large models. [26] proposed that simply increasing model capacity can also provide a reasonable defense. Along this line, *piecewise clustering* of quantized weights [16] is so far shown to be the state-of-the-art defense for quantized DNNs, as also experimentally evaluated in the most recent attack studies [3, 26]. Piecewise clustering is a relaxation of the inherent defense mechanism of binarized networks for quantized DNNs, which exploits a regularization term to enforce the quantized weights to have a bi-modal distribution.

2.3. Output Coding in Deep Neural Networks

Alternative output representations for multiclass classification problems were studied in the context of ensemble models. The idea was initially implemented by encoding class labels with error-correcting output codes [2, 9], where the ensemble consists of multiple base classifiers that are assigned to binary sub-problems to infer independent code bits, -1 or 1. These models generally use Hadamard matrix type label-to-code encoding, which are known to be optimal error-correcting output codes to pool base classifiers for minimizing empirical probability of error [43]. Another approach decomposes the classification problem into a set

of simpler multiclass sub-problems, referred as n-ary output coding [10, 49], where sparse output codes can be used, e.g., bit strings with -1, 0, 1.

These methods were recently explored for deep ensemble learning [1, 46] and shown to be successful on small-scale datasets. DNN ensembles showed better success when parameters were not shared across the models solving the binary sub-problems, i.e., independent base DNNs (or with partially shared encoder layers). Recently [37] used error-correcting output codes to improve robustness of a DNN ensemble against adversarial inputs. In parallel [30] used this approach for standard DNNs with output coding which are regularized during training to disentangle the feature encoder across sub-tasks, i.e., ensemble diversity, to improve adversarial input robustness. Nevertheless, these methods were later shown to be ineffective against adversarial inputs crafted via adaptive attacks [34, 44]. To date, existing work on novel DNN output coding mechanisms was neither successfully applied to ImageNet-scale classification tasks, nor considered for adversarial weight bit-flip attacks.

3. Improving Robustness Against Stealthy Weight Bit-Flip Attacks

3.1. Quantized Deep Neural Networks

Weight quantization in DNNs refers to the process of representing the dense and convolutional layer weights with reduced precision, in order to meet memory constraints for deployment and implement efficient integer-arithmetic operations [18]. We focus on the layer-wise uniform weight quantization scheme in accordance with [3, 26]. For a DNN with original floating-point weights $\mathbf{W}_{\mathrm{f}}^{l} \in \mathbb{R}^{d_l}$ at layer l, where $l \in \{1, \dots, L\}$ and L being the output dense layer index, Q-bit quantization corresponds to symmetrically discretizing the weights to 2^{Q-1} levels such that the quantized weights $\mathbf{W}^{l}$ can be represented by Q-bits via:

$$\mathbf{W}^{l} = \mathrm{round}(\mathbf{W}_{\mathrm{f}}^{l} \,/\, \delta^{l}) \cdot \delta^{l}\,, \tag{1}$$

$$\delta^{l} = \max(|\,\mathbf{W}_{\mathrm{f}}^{l}\,|) \,/\, (2^{Q-1} - 1)\,, \tag{2}$$

where δ^{l} is the step-size of the layer-wise weight quantizer. Weights can then be stored in Q-bits signed-integer format by representing $\mathbf{W}^{l}\,/\delta^{l}$ in two's complement form in the memory as $\mathbf{B}^{l}$ (i.e., $\mathbf{b} = [b_{Q-1}; b_{Q-2}; \dots; b_0] \in \{0,1\}^{Q}$ for an individual weight w in $\mathbf{W}^{l}$), and independently storing the list of layer-wise δ^{l} constants. For any w in $\mathbf{W}^{l}$, $\mathbf{b}$ can be converted to the quantized weight by:

$$w = \Big(-2^{Q-1} \cdot b_{Q-1} + \sum_{i=0}^{Q-2} 2^{i} \cdot b_i \Big) \cdot \delta^{l}, \tag{3}$$

We consider the networks to be also trained in quantized form. During training we use the straight-through estimator [5] for backpropagation of the rounding in Eq. (1).

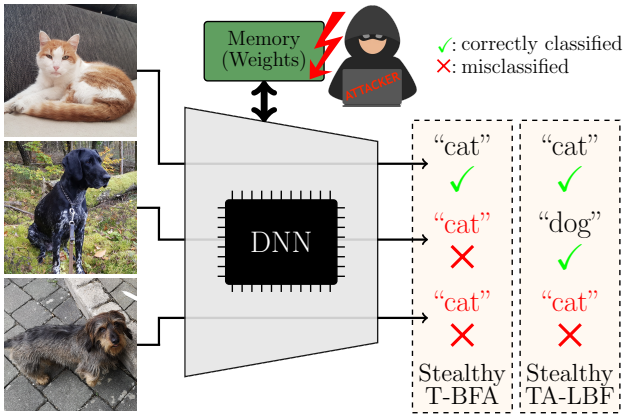


Figure 1. Inference illustrations under stealthy T-BFA attack [26], which targets a whole class (here: "dog"→"cat") and TA-LBF attack [3], which targets single examples (here: last dog→"cat").

3.2. Stealthy Weight Bit-Flip Attacks

Stealthy attacks aim at the misclassification of some set of samples while preserving expected behavior for all others. The attacker is assumed to have physical access to the stored binary representation of weights $\mathbf{B}^{l} \in \{0,1\}^{d_l \times Q}$, as well as knowledge of the quantization step-sizes δ^{l} and the DNN architecture such that quantized weights $\mathbf{W}^{l}$ can be calculated. To facilitate stealthiness the attacker is also assumed to have an auxiliary set $\mathcal{D}_{\mathrm{aux}} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n_{\mathrm{aux}}}$ of test samples and their correct labels, and the goal is to identify the vulnerable stored bits to be flipped, i.e., obtaining $\hat{\mathbf{B}}$. Figure 1 illustrates the state-of-the-art attacks in our scope, namely stealthy T-BFA [26] and TA-LBF [26].

Stealthy T-BFA [26]: The aim is the misclassification of all samples belonging to a source class s as a target class t, while test examples outside the attacked source class are not impacted. Using a set of auxiliary samples $\mathcal{D}_{\mathrm{aux}} = \{(x^{(i)}, y^{(i)}) \,|\, y^{(i)} \in \{1, \dots, C\} \setminus \{s\}\}_{i=1}^{n_{\mathrm{aux}}}$ and a set of source class samples $\mathcal{D}_{\mathrm{src}} = \{(x^{(i)}, s)\}_{i=1}^{n_{\mathrm{src}}}$, stealthy T-BFA aims to solve the following objective:

$$\min_{\hat{\mathbf{B}}} \; \mathbb{E}_{\mathcal{D}_{\mathrm{src}}} \big[\, \mathcal{L}(f(x; \hat{\mathbf{B}}); t) \big] + \mathbb{E}_{\mathcal{D}_{\mathrm{aux}}} \big[\, \mathcal{L}(f(x; \hat{\mathbf{B}}); y) \big], \tag{4}$$

with $f(x; \hat{\mathbf{B}})$ being the quantized DNN inference output and $\mathcal{L}$ the training loss function. In practice, another constraint on this objective is also that the Hamming distance between the pre- and post-attack binary tensors $d_{\mathrm{H}}(\mathbf{B}, \hat{\mathbf{B}})$ is at most equal to the allowed number of bit-flips. The objective (4) is approximated in [24, 26] using a heuristic progressive inter- and intra-layer bit search algorithm based on ranking the gradient of the loss function. The adversary can target vulnerable bit-flips in any network layer.

Stealthy TA-LBF [3]: The aim of this attack is to find minimal bit-flips specifically in the final layer that lead to

the misclassification of a single sample x as a target class t while not changing the decisions for remaining samples by exploiting $\mathcal{D}_{\text{aux}} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n_{\text{aux}}}$. Note that different from T-BFA, $\mathcal{D}_{\text{aux}}$ can contain samples with any class label including s. This renders TA-LBF to have a different real-world applicability (e.g., sneaking past a facial recognition system). The overall objective of TA-LBF is as follows:

$$\min_{\hat{\mathbf{B}}^L} \mathcal{L}_{\text{eff}} + \gamma \, \mathbb{E}_{\mathcal{D}_{\text{aux}}} \left[\mathcal{L}(f(x; \hat{\mathbf{B}}); y) \right], \; d_{\text{H}}(\mathbf{B}, \hat{\mathbf{B}}) \leq k, \quad (5)$$

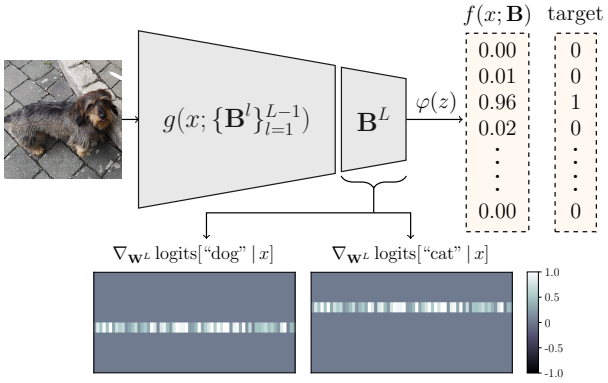
where $\mathcal{L}$ is the training loss, k is the number of maximum bit-flips and γ is the trade-off parameter between stealthiness and the effectiveness loss $\mathcal{L}_{\text{eff}}$, which aims to maximize the marginal difference between the target class and source class logits for input x (see Section 3 of the Supplementary for details). [3] approaches this optimization objective as a binary integer programming problem with cardinality constraints, solved via l_p-box ADMM [39]. Parameters γ and k are chosen from a set of values using greedy search.

3.3. Output Code Matching with Bit Strings

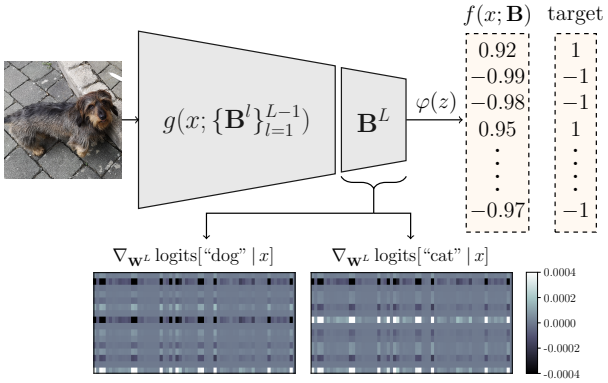
Protecting the output layer weights was recently suggested as a promising hardware-based defense [26]. We revisit conventional DNNs to realize this from an architectural aspect. Our defense goal is to increase the required number of bit-flips for stealthy attacks, while enforcing a larger pre- and post-attack accuracy gap to break stealthiness, such that a truly stealthy attack via physical fault-injection becomes practically unrealistic. To date, no effective defense tailored against stealthy weight bit-flip attacks existed.

Notation: We define the quantized DNN output $f(x; \mathbf{B})$ as a composition of $g(x; \{\mathbf{B}^l\}_{l=1}^{L-1})$, the output dense layer with quantized weights $\mathbf{W}^L$, and output activation $\varphi(.)$. We will use $\mathbf{B}^L$ and $\mathbf{W}^L$ alternatively considering Eq. (3). In standard DNNs $\varphi(z)$ denotes a softmax activation, z denotes the logits (class scores), and softmax assigns the highest probability to the largest logit. For simplicity of illustrations, we will consider a stealthy attacker targeting $\mathbf{B}^L$ [3].

Modifying DNN Output Representations: Our output code matching (OCM) framework is motivated by the idea that for any bit-flip in $\mathbf{B}^L$ (hence a change in $\mathbf{W}^L$) to be *non-stealthy*, ideally all class scores should change their values for any input. Conventional one-hot encoding of DNN outputs combined with a softmax activation counteracts to this motivation and benefits stealthy attacks. Figure 2a illustrates an example for standard DNNs with one-hot output encoding. For a correct decision with a high pre-softmax logit and probability estimate $p(\text{“dog”} \,|\, x)$=0.96, finding minimal bit-flips for parameters from row c of $\mathbf{W}^L$ can simply increase the “cat” score without interfering much with other predictions. Similarly an attacker can target row d of $\mathbf{W}^L$ to reduce only the “dog” score of any input



(a) Standard one-hot output encoding.



(b) Proposed output code matching (OCM) framework.

Figure 2. Illustration of one-hot class encoding vs OCM. Gradients of the output probability scores with respect to $\mathbf{W}^L$ are visualized from a ResNet-20 trained on CIFAR-10 (#columns: input dimensionality of final layer = 64, #rows: (a) C=10, (b) N=16).

(see Fig. 2a bottom). This independency structure creates a wide search space for vulnerable bits to stealthy attacks.

Figure 2b illustrates our approach to this problem. For each class $y \in \{1, \ldots, C\}$, we define a bit string $\mathbf{S}_y \in \{-1, 1\}^N$ of length N. The goal of the network is to predict this bit-string instead of the usual one-hot encoded target vector (i.e., output dimensionality becomes N). Accordingly, we replace the usual softmax output layer of the network by a layer of N neurons with tanh activation functions. During inference, ideally the output $\varphi(z)$ becomes equivalent to $\mathbf{S}_y$ for a sample from class y. In order to reduce attack *stealthiness*, we use output codes that are partially overlapping across classes. As illustrated at the bottom part of Figure 2b, in this case an attacker has to target multiple rows of $\mathbf{B}^L$ for bit-flips (hence changing $\mathbf{W}^L$) to influence the score of one class towards misclassification, which however will also lead to changes in the scores for other classes due to the use of overlapping codes. As a result, the effectiveness of our defense comes from using overlapping output codes at test time, and thus increasing uncertainty across several classes in the face of adversity.

Bit String Code Design: We design output codes of length N using Hadamard matrices with optimal row separation, constructed via Sylvester's method for matrices of order 2^k [29, 43] (i.e., the overlap between any given pair of class codes is $N/2$, see Section 1.3 of the Supplementary). For C-class problems where C is not a power of 2, we randomly select C rows of length N out of the N rows of the Hadamard matrix. We denote a DNN trained via output code matching with a length-N bit string code by OCM_N.

Optimization Objective: We train OCM networks under the objective of minimizing the l_1-norm of the distance between the network output and the target bit strings:

$$\mathcal{L}_{\text{OCM}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}}\big[\,|f(x;\mathbf{B}) - \mathbf{S}_y|\,\big], \qquad (6)$$

which is equivalent to the training objectives previously used in the context of error-correcting output codes [37, 46].

Decision Criterion: We perform the classification based on how much the predicted output is positively correlated with the predefined codes, i.e., $\arg\max_y[\mathbf{S}_y \cdot f(x;\mathbf{B})]$, which is analogous to a minimum Hamming distance decoding principle. Accordingly, we define the class scores for OCM as the dot product between the class code and network output. One can also normalize these correlations to obtain per-class probability estimates in the form of $p(y|x) = \max(\mathbf{S}_y \cdot f(x;\mathbf{B}), 0) \;/\; \sum_c \max(\mathbf{S}_c \cdot f(x;\mathbf{B}), 0)$.

4. Experiments

4.1. Experiment Design

Datasets & Models: We performed benchmark experiments with CIFAR-10/100 [20] and ImageNet [28]. We used ResNet-20 [15] models for CIFAR-10, consistently with previous work [3, 16]. We used WideResNet [42] models with depth 28 for CIFAR-100, and ResNet-50 [15] models for ImageNet. For all networks, layer-wise uniform weight quantization was used as described in Section 3.1.

Stealthy Attack Configurations: We evaluated robustness of our models with the two state-of-the-art attacks, namely T-BFA [26] to examine *stealthy source class attacks*, and TA-LBF [3] to examine *stealthy single sample attacks*. We performed white-box attacks on the defended models until successful, anticipating an adversary with full knowledge of our defense (i.e., bit string codes and loss function) [6].

Stealthy T-BFA: The attacker requires a set of source class samples $\mathcal{D}_{\text{src}}$ and auxiliary samples from other classes $\mathcal{D}_{\text{aux}}$. Size of the sets $\mathcal{D}_{\text{src}}$ and $\mathcal{D}_{\text{aux}}$ were both determined as in [26]. For CIFAR-100 and ImageNet, we only considered the first 50 classes as a target or source class in the attacks. In total, we performed 500 experiments for each of these datasets which differed in the source and target class, the random choice of the auxiliary set $\mathcal{D}_{\text{aux}}$, and the samples drawn from the source class for $\mathcal{D}_{\text{src}}$. For CIFAR-10, we performed T-BFA similarly using all 10 classes as the source or target in a total of 100 experiments. Further details on attack settings and compute budgets of these experiments are provided in Sections 1.4 and 2 of the Supplementary.

Stealthy TA-LBF: Proposed OCM defense trivially renders the original TA-LBF attack in [3] to be non-applicable. This is the case since $\mathcal{L}_{\text{eff}}$ in Eq. (5) measures the effectiveness based on individual logits of the output softmax, which does not exist in the OCM network. To allow comparisons we adjusted the TA-LBF optimization objective to consider in $\mathcal{L}_{\text{eff}}$ instead the average of the binary cross-entropies across the output units (see Section 3 of the Supplementary for attack details). Our experiments with this adjusted TA-LBF revealed same evaluation results for ResNet-20 models on CIFAR-10 as reported in [3]. However, the larger networks used for CIFAR-100 and ImageNet in our study were not considered in [3], and we could not obtain successful attacks for them. Hence we only report TA-LBF evaluations on CIFAR-10. We used 1000 single sample attacks in total [3], where each one of the 10 classes is the target class for 100 different source images belonging to any other class. An auxiliary set of size 64 was used, and a similar parameter search as in [3] was performed for k and γ in Eq. (5).

Evaluation Metrics: We quantify robustness under stealthy attacks based on the following metrics in accordance with previous studies: (1) clean accuracy on the test set, (2) attack success rate (ASR), (3) post-attack test accuracy (PA-ACC), (4) total number of bit-flips needed for the attack. For T-BFA, ASR (%) is calculated as the proportion of successfully misclassified source class samples among the held-out source class test set samples that were not used by the attacker. We report ASR for TA-LBF (on CIFAR-10) as the proportion of successful misclassifications among 1000 attacks. For T-BFA attacks, PA-ACC (%) is calculated on the test set except the samples belonging to the attacked source class and the auxiliary samples. For TA-LBF, PA-ACC is calculated on the test set only except the single attacked source sample and the auxiliary samples. A stealthy attack aims for a high PA-ACC and ASR, while requiring as few bit-flips as possible [3, 26]. Our aim is to train models that will ideally increase the number of bit-flips needed for the attacks, as well as increase the gap between the clean test accuracy and PA-ACC to break stealthiness.

Implementations: We used stochastic gradient descent with momentum for parameter optimization in all models. For CIFAR-10/100 experiments, OCM networks were trained end-to-end from scratch for 160 epochs with a batch size of 128. For ImageNet, we optimized OCM networks with a batch size of 256 by finetuning vanilla models (which were trained for 100 epochs) for 60 epochs, starting from random initialization of the final dense layer with the new output dimensionality. Further details on

Table 1. Evaluations of 8-bit and 4-bit quantized ResNet-20 models under stealthy weight bit-flip attacks for CIFAR-10. Test set clean accuracy, ASR and PA-ACC percentages (%) are presented alongside # bit-flips needed for the attack. Stealthy T-BFA attacks [26] are run until all source class set examples used by the attacker are misclassified, and all stealthy T-BFA evaluation metrics are averaged across 100 targeted attack experiments. Stealthy TA-LBF attacks [3] are performed for 1000 single sample attacks, where each one of the 10 classes is the target class for 100 different source images that belong to any other class.

			Vanilla	Piecewise Clustering [16]	**Ours**		
					OCM_{16}	**OCM_{32}**	**OCM_{64}**
ResNet-20 (8-bit)	Clean Acc. on **CIFAR-10**		**92.25**	91.11	90.67	90.72	90.26
	Stealthy T-BFA [26]	ASR (↘)	**99.10**	99.46	99.48	99.56	99.58
		PA-ACC (↘)	84.38 (3.39)	76.78 (7.45)	53.22 (21.5)	50.01 (18.2)	**46.39 (16.7)**
		# bit-flips (↗)	27.91 (8.70)	74.93 (26.7)	95.65 (32.4)	127.88 (54.0)	**281.75 (115.6)**
	Stealthy TA-LBF [3]	ASR (↘)	100.00	100.00	97.60	98.20	**72.40**
		PA-ACC (↘)	88.06 (2.55)	87.64 (2.09)	86.45 (3.31)	86.07 (3.26)	**84.08 (3.18)**
		# bit-flips (↗)	5.42 (0.91)	18.14 (7.05)	31.12 (10.3)	47.52 (13.7)	**73.65 (15.67)**
ResNet-20 (4-bit)	Clean Acc. on **CIFAR-10**		**91.87**	90.72	89.97	89.83	89.29
	Stealthy T-BFA [26]	ASR (↘)	**99.33**	99.57	99.44	99.40	99.65
		PA-ACC (↘)	80.70 (9.39)	74.21 (12.5)	53.73 (20.5)	51.34 (20.2)	**45.27 (18.5)**
		# bit-flips (↗)	27.91 (10.2)	71.96 (28.6)	97.11 (38.2)	138.23 (39.8)	**278.48 (110.9)**
	Stealthy TA-LBF [3]	ASR (↘)	100.00	100.00	96.90	98.20	**87.20**
		PA-ACC (↘)	87.88 (2.36)	87.50 (2.06)	85.24 (3.42)	83.41 (3.35)	**83.27 (3.08)**
		# bit-flips (↗)	5.41 (1.20)	16.75 (6.20)	22.91 (8.59)	34.81 (7.63)	**66.20 (13.93)**

the experimental settings are provided in Section 1 of the Supplementary. Our implementations are available at: https://github.com/IGITUGraz/OutputCodeMatching.

4.2. Evaluating Robustness to Stealthy Attacks

Experiments on CIFAR-10: We evaluate our approach in comparison to the *piecewise clustering* defense, which is widely studied as the state-of-the-art defense method [3,26]. We trained models with the quantized weight regularization term proposed in [16] using varying regularization strengths λ and reported best models (e.g., $\lambda = 0.001$ for CIFAR-10 as in [16]). Table 1 depicts attack evaluations on quantized ResNet-20 models (2.16M bits in total with 8-bit quantization) trained on CIFAR-10. Results show that stealthy T-BFA requires up to 10× more (281.75 vs. 27.91) and 3.7× more (281.75 vs. 74.93) bits to be attacked on OCM models (8-bit) with respect to undefended vanilla networks and networks trained with piecewise clustering. Our method trades off less than 1% decrease in clean accuracy with respect to piecewise clustering, and breaks the stealthiness of attacks significantly better with up to 30% more decrease in PA-ACC (as low as 46.39% with OCM_{64}) for stealthy T-BFA.

Against TA-LBF our method also increases the number of necessary bit-flips significantly (up to 14× and 4× with respect to vanilla models and piecewise clustering defense), while yielding state-of-the-art resistance for ResNet-20 (see [3]). By using longer output codes, e.g., OCM_{64}, ASR for TA-LBF attacks significantly reduced. For both attacks, similar results were also observed with ResNet-20 models at 4-bit quantization (see bottom half of Tab. 1).

Experiments on CIFAR-100: Table 2 presents our experiments on CIFAR-100 with 8-bit quantized WideResNet models. Our OCM_{128} and OCM_{256} models again significantly outperform vanilla and piecewise clustering defended ($\lambda = 0.001$) models by requiring ∼14× more (143.63 vs. 10.11) and ∼1.5× more (143.63 vs. 88.39) bits to be attacked on WRN-28-4 models (47M bits in total). With a 5% drop in clean accuracy, OCM_{256} model can decrease PA-ACC by 10% compared to piecewise clustering.

Increasing the model capacity was demonstrated as an alternative defense approach by [3, 26]. To investigate this we trained 2× wider vanilla WRN-28-8 models to test the viability of using larger models as a defense. Results show that using a WRN-28-4 with OCM is already a significantly better defense than using a vanilla WRN-28-8 with 187M bits. This observation confirms that simply increasing the bit search space (4× larger) without any explicit defense mechanism does not truly improve robustness to adversarial bit-flip attacks. We also investigated the impact of OCM for models with larger width by performing OCM_{128} on a WRN-28-8 (rightmost column in Table 2). While increasing the width of a vanilla network was only 1.6× effective

Table 2. Stealthy T-BFA [26] evaluations with WRN-28-4 and WRN-28-8 (×2 width) for CIFAR-100. Attacks are run until all source class set examples used by the attacker are misclassified. Test set clean accuracy, ASR and PA-ACC (%) are presented alongside # bit-flips needed to attack. All evaluation metrics are averaged across 500 targeted attack experiments.

		WRN-28-4 (8-bit)				WRN-28-8 (8-bit)	
		Vanilla	Piecewise Clustering [16]	**Ours**		Vanilla	**Ours**
				OCM$_{128}$	**OCM$_{256}$**		**OCM$_{128}$**
Clean Acc. on **CIFAR-100**		78.44	76.11	75.43	71.72	**80.22**	77.25
Stealthy T-BFA [26]	ASR (↘)	94.38	**91.16**	94.38	94.87	94.86	**94.38**
	PA-ACC (↘)	74.32 (1.98)	64.02 (10.9)	62.12 (12.3)	**53.77 (12.5)**	77.33 (1.92)	**59.88 (15.6)**
	# bit-flips (↗)	10.11 (4.60)	88.39 (71.6)	121.43 (65.7)	**143.63 (115.2)**	16.14 (4.89)	**323.11 (295.1)**

in terms of # bit-flips (with an increase from 10.11 to 16.14), for OCM$_{128}$ the increase in network width led to 2.7× more bit-flips required by the attacker, corresponding to >300 bit-flips for stealthy T-BFA. Increasing network width for OCM$_{128}$ also marginally contributed to enlarge the gap between clean accuracy and PA-ACC by 5%.

Experiments on ImageNet: Table 3 presents our ImageNet experiments with ResNet-50. We performed OCM by finetuning pre-trained vanilla models after re-initializing the final layer with the new length N output dimensionality. This saves a lot of computation time for our defense to be adapted to large-scale models. Our vanilla network evaluations confirm previous studies that stealthy and targeted misclassification can be enforced by flipping only ∼7 bits among the 204M bits stored on memory for an 8-bit quantized ResNet-50, while only having a 7.3% decrease in PA-ACC. Our OCM evaluations with ResNet-50 (8-bit) show that stealthy attacks require up to 20× more (145.05 vs. 7.69) and 3× more (145.05 vs. 48.65) bits to be attacked with respect to vanilla networks and networks trained with piecewise clustering (see Sec. 5 of the Supplementary for the T-BFA impact when the maximum allowed # bit-flips increase gradually). Our defense yields significantly higher clean accuracies (∼73%) for more robust models than existing defenses, and greatly impacts stealthiness by dropping PA-ACC to 50% for an 8-bit quantized ResNet-50.

For 4-bit quantized ResNet-50 models with OCM, a larger increase in # bit-flips was observed with respect to piecewise clustering (up to 5× more with OCM$_{1024}$: 143.82 vs. 28.60), with no significant differences in PA-ACC. In Tab. 3 we report piecewise clustering defense with two different λ values to depict the robustness trade-off gap. Our observations presented in Sec. 4 of the Supplementary conclude that reducing the piecewise clustering regularization strength to e.g., $\lambda = 0.0001$, or finetuning models with piecewise clustering as opposed to the end-to-end regularized training, leads to relatively higher clean accuracies however not showing any further robustness benefits.

4.3. Combining OCM with Piecewise Clustering

We tested the compatibility of piecewise clustering with the proposed OCM. Table 4 presents CIFAR-10 experiments with ResNet-20 using OCM$_{16}$ and piecewise clustering ($\lambda = 0.0005$). In comparison to OCM$_{16}$ alone, stealthy T-BFA requires 17 and 65 more bits to be flipped for 8-bit and 4-bit quantized ResNet-20 models, and lead to 5–6% less PA-ACC to impact stealthiness. For TA-LBF, PA-ACC decreases to its lowest around 80%, while again requiring more bits to be flipped. We conclude that higher robustness gains can be obtained when both defense methods are combined in an end-to-end regularized training pipeline.

4.4. Further Experiments

Ablation Study with Non-Hadamard Rand-OCM: We performed an ablation study on replacing the Hadamard-type output code bit strings with randomly generated binary codes (Rand-OCM), i.e., when the overlap between any given pair of codes is not necessarily $N/2$. To generate feasible random codes such that DNNs can still be successfully trained, we had to ensure that at least r indices of each length-N bit string were -1, where r was uniformly sampled between $N/4$ and $3N/4$ for each class. Remaining indices of the bit string were set to 1. On CIFAR-10 with ResNet-20, Rand-OCM$_{16}$ and Rand-OCM$_{64}$ yielded 90.0% and 90.5% test accuracies. Under TA-LBF these models required 24.5 and 69.9 bit-flips respectively, as opposed to the 31.1 and 73.6 with OCM$_{16}$ and OCM$_{64}$ (see Tab. 1). On ImageNet, the Rand-OCM$_{1024}$ model yielded 65.7% clean accuracy with 91.1 bit-flips required for stealthy T-BFA, as opposed to the 72.7% clean accuracy and 121.2 bit-flips required for OCM$_{1024}$ (see Tab. 3). Overall, we concluded that the proposed code design contributes to both benign accuracy and robustness to stealthy attacks as expected.

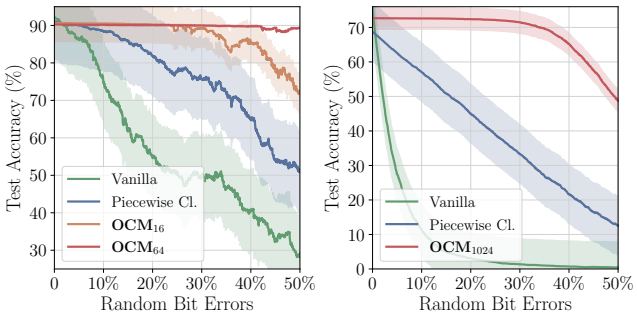
Random Bit Error Robustness: We addressed another realistic parameter vulnerability setting by investigating fault tolerance of these models under random bit errors that may occur due to, e.g, low-voltage operation of DNN accelera-

Table 3. Stealthy T-BFA [26] evaluations with 8-bit and 4-bit quantized ResNet-50 models on ImageNet. Attacks are run until all source class set examples used by the attacker are misclassified. Test set clean accuracy, ASR and PA-ACC percentages (%) are presented alongside # bit-flips needed to attack. All evaluation metrics are averaged across 500 targeted attack experiments.

			Vanilla	Piecewise Clustering [16]		**Ours**	
				$\lambda = 0.0001$	$\lambda = 0.0005$	**OCM**$_{1024}$	**OCM**$_{2048}$
ResNet-50 (8-bit)	Clean Acc. on **ImageNet**		**75.92**	74.64	68.73	72.71	73.25
	Stealthy T-BFA [26]	ASR (↘)	94.74	91.29	**89.32**	91.35	92.37
		PA-ACC (↘)	68.64 (9.25)	57.64 (11.4)	54.81 (10.2)	50.93 (10.9)	**50.63 (11.3)**
		# bit-flips (↗)	7.69 (3.88)	26.24 (13.8)	48.65 (17.0)	121.26 (297.3)	**145.05 (366.4)**
ResNet-50 (4-bit)	Clean Acc. on **ImageNet**		**72.56**	70.26	65.36	70.98	71.02
	Stealthy T-BFA [26]	ASR (↘)	91.81	92.57	90.49	**89.68**	90.20
		PA-ACC (↘)	70.50 (4.58)	65.87 (7.52)	**59.14 (7.75)**	59.87 (7.20)	59.86 (7.23)
		# bit-flips (↗)	8.97 (3.63)	14.88 (4.69)	28.60 (9.38)	**143.82 (299.29)**	122.06 (347.96)

Table 4. Evaluations on CIFAR-10 for ResNet-20 models trained both with OCM and piecewise clustering (PC). $\Delta_{\mathbf{OCM}_{16}}$ shows the difference of the metric mean with respect to **OCM**$_{16}$ models.

		ResNet-20 (8-bit)		ResNet-20 (4-bit)	
		OCM$_{16}$ +PC	$\Delta_{\mathbf{OCM}_{16}}$	**OCM**$_{16}$ +PC	$\Delta_{\mathbf{OCM}_{16}}$
	Clean	87.55	-3.12	87.17	-2.80
T-BFA	ASR	99.45	**-0.03**	99.47	**-0.11**
	PA-ACC	47.88	**-5.34**	47.66	**-6.07**
	# bit-flips	113.24	**+17.59**	162.76	**+65.65**
TA-LBF	ASR	100.00	+2.30	100.00	+3.10
	PA-ACC	81.39	**-5.06**	80.90	**-4.34**
	# bit-flips	37.23	**+6.11**	32.95	**+10.04**



(a) CIFAR-10 with ResNet-20 (b) ImageNet with ResNet-50

Figure 3. Impact of random bit errors at the output layer on test accuracy. Results are averaged across 5 and 3 repetitions for CIFAR-10 and ImageNet respectively (shading: ± 0.5 standard deviation).

tors with on-chip scratchpad memory where DNN weights may be stored [12, 31]. Here we introduce randomly occurring bit-flips only to the output layer $\mathbf{B}^L$ by sampling from a uniform distribution, as opposed to the previously considered adversarial scheme. Figure 3 presents the impact of random bit errors on CIFAR-10 and ImageNet test accuracies. Results clearly show superior resistance of our output representation mechanism to such bit-flips, as the faulty percentage of bits in $\mathbf{B}^L$ increases. Specifically for ResNet-50, OCM$_{1024}$ maintains the benign performance of 72.71% even if 30% of the 16.8M output layer bits are faulty.

5. Conclusion

We have presented OCM, a defense approach to improve DNN robustness against stealthy weight bit-flip attacks that exploit parameter vulnerabilities of DNNs. Our results show that the simple method of an alternative output coding scheme is a very effective defense against such attacks. Our results on various benchmark datasets and architectures show that OCM increases the number of necessary bit-flips by at least an order of magnitude over vanilla networks and by 1.5–5× compared to the best known previous defense. In addition, OCM significantly reduces stealthiness and typically reduces post-attack accuracy.

One advantage of OCM is that the output code length can be scaled. Our results indicate that longer output codes generally improve robustness to attacks and reduce stealthiness of the attack with only a small decrease in clean accuracy (compare e.g., OCM$_{16}$ to OCM$_{64}$ in Table 1). Another beneficial feature of OCM is that models trained in the standard manner can easily be adopted. For a pre-trained model, only the output layer has to be replaced followed by a finetuning phase, as we performed for our ImageNet experiments.

Acknowledgments: We thank Jakub Breier and Xiaolu Hou for the fruitful discussions and comments. This work has been supported by the "University SAL Labs" initiative of Silicon Austria Labs (SAL).

References

[1] Sara Atito Ali Ahmed, Cemre Zor, Muhammad Awais, Berrin Yanikoglu, and Josef Kittler. Deep convolutional neural network ensembles using ECOC. *IEEE Access*, 9:86083–86095, 2021. 3

[2] Erin L Allwein, Robert E Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000. 2

[3] Jiawang Bai, Baoyuan Wu, Yong Zhang, Yiming Li, Zhifeng Li, and Shu-Tao Xia. Targeted attack against deep neural networks via flipping limited weight bits. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 2, 3, 4, 5, 6

[4] Alessandro Barenghi, Luca Breveglieri, Israel Koren, and David Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076, 2012. 2

[5] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3

[6] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):984–996, 2013. 5

[7] Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. Practical fault attack on deep neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2204–2206, 2018. 1, 2

[8] Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. Exploiting correcting codes: On the effectiveness of ECC memory against rowhammer attacks. In *IEEE Symposium on Security and Privacy (SP)*, pages 55–71, 2019. 2

[9] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1994. 2

[10] Sergio Escalera, Oriol Pujol, and Petia Radeva. On the decoding process in ternary error-correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):120–134, 2008. 3

[11] Pietro Frigo, Emanuele Vannacc, Hasan Hassan, Victor Van Der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Trrespass: Exploiting the many sides of target row refresh. In *IEEE Symposium on Security and Privacy (SP)*, pages 747–762, 2020. 2

[12] Shrikanth Ganapathy, John Kalamatianos, Keith Kasprak, and Steven Raasch. On characterizing near-threshold SRAM failures in finFET technology. In *Proceedings of the 54th Annual Design Automation Conference*, pages 1–6, 2017. 8

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015. 1

[14] Daniel Gruss, Moritz Lipp, Michael Schwarz, Daniel Genkin, Jonas Juffinger, Sioli O'Connell, Wolfgang Schoechl, and Yuval Yarom. Another flip in the wall of rowhammer defenses. In *IEEE Symposium on Security and Privacy (SP)*, pages 245–261, 2018. 1, 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 5

[16] Zhezhi He, Adnan Siraj Rakin, Jingtao Li, Chaitali Chakrabarti, and Deliang Fan. Defending and harnessing the bit-flip based adversarial weight attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14095–14103, 2020. 2, 5, 6, 7, 8

[17] Sanghyun Hong, Pietro Frigo, Yiğitcan Kaya, Cristiano Giuffrida, and Tudor Dumitraș. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 497–514, 2019. 1, 2

[18] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018. 3

[19] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. *ACM SIGARCH Computer Architecture News*, 42(3):361–372, 2014. 1, 2

[20] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*, 2009. 5

[21] Jingtao Li, Adnan Siraj Rakin, Yan Xiong, Liangliang Chang, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. Defending bit-flip attack through DNN weight reconstruction. In *57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020. 2

[22] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. Fault injection attack on deep neural network. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 131–138, 2017. 1, 2

[23] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[24] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1211–1220, 2019. 2, 3

[25] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. TBT: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13198–13207, 2020. 2

[26] Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. T-BFA: Targeted bit-flip ad-

versarial weight attack. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[27] Adnan Siraj Rakin, Li Yang, Jingtao Li, Fan Yao, Chaitali Chakrabarti, Yu Cao, Jae-sun Seo, and Deliang Fan. RA-BNN: Constructing robust & accurate binary neural network to simultaneously defend adversarial bit-flip attack and improve accuracy. *arXiv preprint arXiv:2103.13813*, 2021. 2

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 5

[29] Jennifer Seberry and Mieko Yamada. Hadamard matrices, sequences, and block designs. *Contemporary Design Theory: A Collection of Surveys*, pages 431–560, 1992. 5

[30] Yang Song, Qiyu Kang, Wee Peng Tay, Y Song, Q Kang, and WP Tay. Error-correcting output codes with ensemble diversity for robust learning in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9722–9729, 2021. 3

[31] David Stutz, Nandhini Chandramoorthy, Matthias Hein, and Bernt Schiele. Bit error robustness for energy-efficient DNN accelerators. In *Fourth Conference on Machine Learning and Systems (MLSys)*, 2021. 2, 8

[32] David Stutz, Nandhini Chandramoorthy, Matthias Hein, and Bernt Schiele. Random and adversarial bit error robustness: Energy-efficient and secure DNN accelerators. *arXiv preprint arXiv:2104.08323*, 2021. 2

[33] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1

[34] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems*, volume 33, pages 1633–1645, 2020. 3

[35] Yu-Lin Tsai, Chia-Yi Hsu, Chia-Mu Yu, and Pin-Yu Chen. Formalizing generalization and robustness of neural networks to weight perturbations. *arXiv preprint arXiv:2103.02200*, 2021. 2

[36] Victor Van Der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clémentine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. Drammer: Deterministic rowhammer attacks on mobile platforms. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1675–1689, 2016. 1

[37] Gunjan Verma and Ananthram Swami. Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. *Advances in Neural Information Processing Systems*, 32:8646–8656, 2019. 3, 5

[38] Tsui-Wei Weng, Pu Zhao, Sijia Liu, Pin-Yu Chen, Xue Lin, and Luca Daniel. Towards certificated model robustness against weight perturbations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6356–6363, 2020. 2

[39] Baoyuan Wu and Bernard Ghanem. ℓ_p-Box ADMM: A versatile framework for integer programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(7):1695–1708, 2018. 4

[40] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[41] Fan Yao, Adnan Siraj Rakin, and Deliang Fan. Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1463–1480, 2020. 1, 2

[42] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016. 5

[43] Aijun Zhang, Zhi-Li Wu, Chun-Hung Li, and Kai-Tai Fang. On Hadamard-type output coding in multiclass learning. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 397–404, 2003. 2, 5

[44] Bowen Zhang, Benedetta Tondi, Xixiang Lv, and Mauro Barni. Challenging the adversarial robustness of dnns based on error-correcting output codes. *Security and Communication Networks*, 2020. 3

[45] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482, 2019. 2

[46] Hao Zhang, Joey Tianyi Zhou, Tianying Wang, Ivor W Tsang, and Rick Siow Mong Goh. Deep n-ary error correcting output codes. *arXiv preprint arXiv:2009.10465*, 2020. 3, 5

[47] Pu Zhao, Siyue Wang, Cheng Gongye, Yanzhi Wang, Yunsi Fei, and Xue Lin. Fault sneaking attack: A stealthy framework for misleading deep neural networks. In *56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019. 2

[48] Yaowei Zheng, Richong Zhang, and Yongyi Mao. Regularizing neural networks via adversarial model perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8156–8165, 2021. 2

[49] Joey Tianyi Zhou, Ivor W Tsang, Shen-Shyang Ho, and Klaus-Robert Müller. N-ary decomposition for multi-class classification. *Machine Learning*, 108(5):809–830, 2019. 3