Mind the Scaling Factors: Resilience Analysis of Quantized Adversarially Robust CNNs

Nael Fasfous*, Lukas Frickenstein†, Michael Neumeier*, Manoj Rohit Vemparala†,
Alexander Frickenstein†, Emanuele Valpreda‡, Maurizio Martina‡, Walter Stechele*
*Department of Electrical and Computer Engineering, Technical University of Munich, Germany
†Autonomous Driving, BMW Group, Germany
‡Department of Electronics and Telecommunications, Politecnico di Torino, Italy

***Abstract*—As more deep learning algorithms enter safety-critical application domains, the importance of analyzing their resilience against hardware faults cannot be overstated. Most existing works focus on bit-flips in memory, fewer focus on compute errors, and almost none study the effect of hardware faults on adversarially trained convolutional neural networks (CNNs). In this work, we show that adversarially trained CNNs are more susceptible to failure due to hardware errors when compared to vanilla-trained models. We identify large differences in the quantization scaling factors of the CNNs which are resilient to hardware faults and those which are not. As adversarially trained CNNs learn robustness against input attack perturbations, their internal weight and activation distributions open a backdoor for injecting large magnitude hardware faults. We propose a simple weight decay remedy for adversarially trained models to maintain adversarial robustness and hardware resilience in the same CNN. We improve the fault resilience of an adversarially trained ResNet56 by 25% for large-scale bit-flip benchmarks on activation data while gaining slightly improved accuracy and adversarial robustness.**

I. Introduction

Deep learning algorithms are entering new, safety-critical application domains as research into their robustness and interpretability advances rapidly. Specifically, convolutional neural networks (CNNs) are dominant in computer vision, and are prime candidates for complex applications in autonomous driving and robotics. Next to adversarial robustness and interpretability, resilience against hardware (HW) errors must also be guaranteed before placing these algorithms in safety-critical settings. Understanding the failure cases for logic transient errors on datatype, frequency, bit-position, and number of affected computation units is important in carefully introducing hardware redundancy in a reasonable, cost-effective manner. Moreover, the method by which the CNN was trained affects its behavior in the presence of hardware errors [1], [2]. Consequently, it is also important to study the influence of compression [3] or adversarial training techniques [4] on fault resilience. Existing works in this domain have several limitations. Some only focus on robustness against input adversarial attacks without considering fault resilience [4], [5], others focus on random errors in different parts of the hardware with little attention to CNN training [6]. Works using aged CNNs without frequent, intermediate batch normalization layers have an exaggerated error-amplification effect for bit-flips [7], while others using targeted bit-flips attacks (BFA) construct network-specific attacks which are extremely unlikely to happen at random [2], [8], [9]. We holistically investigate hardware fault resilience and adversarial robustness with large-scale resilience analysis on differently trained CNNs and identify clear relationships between training-time CNN statistics and their deployment-time effect on scaling factors and clipping limits. We show that the common denominator for all resilient CNNs is small inter-layer data distributions, which result in smaller scaling factors at deployment. This allows small scaling factors to naturally introduce resilience by attenuating the largest possible perturbation. The contributions of this work can be summarized as follows:

- Across ∼10M bit-flip experiments, we consider regularly trained, adversarially trained, batch-norm free, weight decayed and pruned CNNs. Our bit-flip module allows us to test a wide range of bit-flip patterns to analyze the effect of training/compression on hardware fault resilience.
- We perform an in-depth analysis on the layer-wise data distributions of the considered CNNs, by observing the differences in the scaling factors required for their quantization. We provide key insights by studying the effect of batch normalization and weight decay, and harness scaling factors for improved fault resilience.
- We identify weaknesses in adversarially trained CNNs, which open a backdoor for injecting faults of large magnitude. We propose a simple weight decay remedy to shrink the quantization scaling factors, which improves resilience against faults in activation pixels by 25% on FastAT ResNet56, while preserving natural accuracy and adversarial robustness.

II. Related Work

A. Hardware Fault Resilience Analysis

He et al. [6] analyze the effect of logic transient errors using abstracted, high-level hardware models. The authors emphasize the importance of investigating faults on control and compute components compared to the limited analysis on memory-based bit-flips in existing works. No conclusions are drawn on the training scheme, compression, and adversarial robustness of the neural networks. Rakin et al. [8] introduce a progressive search technique to find optimal bit-flip attacks (BFA) that break CNNs. In a CNN with 93M-bits of weights, the authors can find 13 precise bit-flips which completely break the network. However, the probability of such an event happening at random

978-3-9819263-6-1/DATE22/©2022 EDAA

is infinitesimal. We refer to such non-random, specific cases as *targeted* bit-flips. He et al. [2] perform resilience investigations on differently trained CNNs while employing such targeted BFA. Several conclusions are drawn based on empirical results without further analysis to explain the underlying cause of the observations. Moreover, hardware designers cannot benefit from BFA analysis, as these are tightly optimized attacks for one considered CNN. To add hardware redundancy in an effective manner, large-scale resilience analysis covers more general error cases and can aid in making design decisions that benefit all CNN workloads. Lastly, BFA-based investigations only apply to bit-flips on a CNN's weights. In practice, logic transient errors may happen in any part of the logic, including input pixels, partial sums, or output activations [6].

B. Fault Resilient Training and Adversarial Robustness

Hoang et al. [7] propose to improve error resilience of CNNs by clipping activations. The investigations are limited to memory-based bit-flips on weights and only aged CNN architectures are tested, which have no batch normalization after each convolutional layer. Errors in such CNNs are typically exaggerated compared to modern CNNs, as batch normalization naturally reduces the activation distribution and scaling factors (as shown in Fig. 2). In an adversarial attack scenario, input noise is propagated and amplified through the layers causing a misclassification. Liao et al. [10] propose to mitigate the amplification error by using a denoiser to reduce input perturbations. Lin et al. [5] apply Lipschitz regularization to limit the error amplification in quantized CNNs. Both works focus on mitigating attacks injected at the input, but do not consider inter and intra-layer faults (depicted in Fig. 3). Zahid et al. [1] introduce a fault-injection layer at training time. The work focuses on a class of permanent errors and does not consider adversarial attacks. A defense method against targeted DRAM bit-flip attacks is proposed by Li et al. [9], where weights are preprocessed to limit their change of value. The method is limited to weight-based, memory-only, targeted BFA and does not consider input-based adversarial attacks.

III. Methodology

A. Problem Formulation: Quantization and Bit-flips

Without loss of generality, a single weight value w multiplied by an activation pixel a produces a partial result in the convolution (CONV) operation of the weight tensor W^l and input activation tensor A^{l-1} in layer l of an L-layer CNN. At training time, A^{l-1} and W^l $\forall l \in L$ are represented by high-precision floating-point (FP) values to maintain smooth training and fine adjustments through backpropagation. During inference, the values are quantized to reduce their memory footprint and arithmetic computation complexity on embedded HW. 8-bit signed integer (`INT8`) representation is one of the most common numerical representation formats for lean deployment on resource constrained devices. A floating-point operand x_f of the convolution operation (either w or a) is quantized to $\mathtt{x_q}$ as shown in Eq. (1).

$$\mathtt{x_q} = clip(round(x_f/v), c) \tag{1}$$

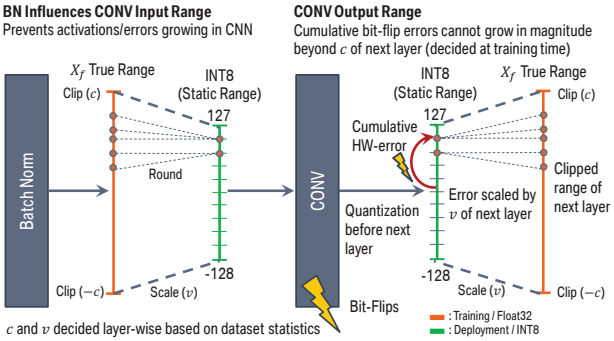


Fig. 1. Batch-norm limits activation range at *training time*, effectively lowering v and c of the subsequent CONV layer at *deployment time* (on HW). Errors in CONV can at most grow in magnitude to the defined clip c of the next layer.

The scaling factor v projects the quantized range of `INT8` $[-128, 127]$ onto the real range of values which $x_f \in X_f$ can take with respect to the $clip$ operator. Note that X_f is either W^l or A^{l-1}. The $round$ operation pushes the smooth values of X_f into the limited 256 integer values of `INT8`. The $clip$ operator cuts-off values of the X_f range beyond $[-c, c]$, maintaining symmetric linear quantization, even in cases where layers such as ReLU leave only positive activations, and weights use only a small portion of the negative number scale. The clipping limit c and scaling factor v are decided before deployment in a process called *calibration*. By observing the statistics of weight and activation distributions of a layer, calibration sets c and v, such that the range of values that appear in a certain layer can be covered by the `INT8` static range [11]. Therefore, c and v are directly influenced by the dataset, the weight values of the CNN (e.g., learned through vanilla or adversarial training, regularized or not) and its structure (e.g., existence of batch-norm layers). The described quantization of X_f to `INT8` is visualized in Fig. 1.

We implement a runtime reconfigurable bit-flip module which can change the value of any position in the 8-bit representation, for weights and activations, and for any subset of multipliers in a standard spatial DNN accelerator [12]. Flipping the n-th bit of an operand at the input of any affected multiplier translates to a 2^n absolute change in magnitude within the static `INT8` range $[-128, 127]$. However, it is more important to analyze the precise severity of a 2^n flip with respect to the values of the projected *real range* of X_f, i.e. after applying scaling factors v. With this understanding of quantization and bit-flips, some general insights can be made:

Quantization clips the largest possible perturbation when projecting a larger, dynamic representation, such as FP32. As the clip limits c and scaling factors v are decided based on statistics *before* deployment on HW, a single or multiple bit-flips on HW cannot perturb the network beyond c of the next layer (Fig. 1). This is an inherent improvement in bit-flip resilience over float/dynamic numerical representations.

Resilience analysis on aged CNNs (LeNet, VGG and AlexNet) *without* batch normalization after every CONV layer cannot be extended to modern CNNs. The lack of batch-norm layers on aged CNNs aggravates the effect of bit-flips, as their activation

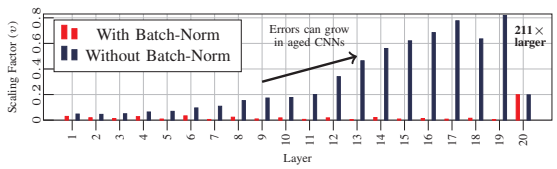


Fig. 2. Layer-wise scaling factors v of ResNet20 CNNs trained on CIFAR-10, with and without batch-norm. Works investigating bit-flips on aged CNNs (without batch-norm after every layer) cannot be extended to modern CNNs.

and weight distributions are much larger than modern CNNs. Consequently, `INT8` variants of aged CNNs will have large scaling factors v to accommodate the activations that appear in the CONV layers, resulting in a much larger true magnitude error for any bit-flip. We show the scaling factors v of ResNet20 with and without batch-norm in Fig. 2 to visualize this problem. Errors in aged CNNs can also propagate and get amplified, as the scaling factors grow in deeper layers.

Adversarially trained CNNs need to be robust against input perturbations which may not follow the statistical distribution of the original training dataset. This affects the statistical distribution of the learned weights in adversarially robust CNNs compared to vanilla trained ones, thereby influencing the scaling factors v during hardware deployment, even when calibrating on non-adversarial, natural data.

B. Adversarially Trained CNNs

An increasingly important aspect of CNN deployment in safety-critical scenarios is robustness against adversarial attacks in the form of input perturbations. Adversarial attacks against a neural network can be formulated as an optimization problem of finding the minimal perturbation δ for an input image I that changes the prediction of the neural network $\mathcal{N}$. These present a different threat model when compared to HW-faults, as shown in Fig. 3. During adversarial training, we introduce perturbed examples to maximize the loss $\mathcal{L}$ with respect to the label Y, within a reasonable perturbation budget ϵ as shown in Eq. (2).

$$\min_{W} \mathop{\mathbb{E}}_{(I,Y)\sim\mathcal{D}} \left[\max_{|\delta|\leq\epsilon} \mathcal{L}\left(\mathcal{N}(I+\delta, W), Y\right) \right] \tag{2}$$

A set of randomly sampled images from dataset $\mathcal{D}$ are chosen, where the expected loss $\mathbb{E}$ on the random samples is minimized through an adversarial training scheme, such as fast adversarial training (FastAT) [4]. A commonly used attack to introduce imperceptible adversarial perturbations is the fast gradient sign method (FGSM) [13]. The advantage of FGSM is that generating an adversarial example is faster than with other attack methods, such as projected gradient descent (PGD) [14]. FGSM in combination with random initialization is particularly effective to incorporate into the training loop. For the final evaluation of adversarial robustness, we apply an unseen PGD attack to the CNN.

C. Error Model and Benchmark Phases

Bit-flips at the compute level fall under logic transient errors [6], and capture a broader range of error patterns compared to memory-based faults. A memory-based fault on a weight parameter w implies all computations using w are affected.

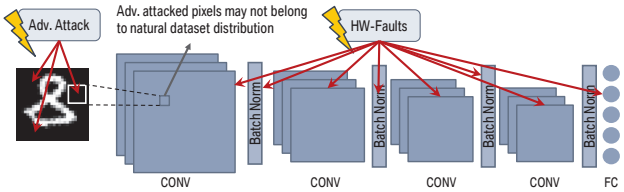


Fig. 3. Adversarial attacks apply input perturbations to cause incorrect classifications. Training for such attacks implies training for pixel value distributions outside of the natural dataset. Differently, HW-faults can occur at any point within the CNN, and are not limited to the input of the network.

With logic transient errors, we can replicate that error *as well as* every other case, where a subset of w's computations are affected, providing finer granularity in error injection control. We perform large-scale resilience analysis by exploiting the flexibility of our run-time reconfigurable bit-flip injection hardware module. Large-scale statistical fault injection is an established approach to analyzing errors in logic [6]. However, it is often infeasible due to slow RTL simulations. We circumvent the need for RTL simulations by directly implementing the bit-flip module on NVDLA [12] and injecting the desired bit-flip patterns on the running hardware. We develop benchmarks with well-defined bit-flip patterns, allowing us to better understand the effect of bit-flip position, frequency of occurrence, affected datatype, and affected percentage of multipliers.

We define our benchmark in steps, where each successive step changes one aspect of the error model, i.e., bit-flip pattern. The bit-flip pattern is maintained and the accelerator performs inference of an entire test set of input images. Once the test set is exhausted, the next step begins with a new bit-flip pattern and the test set is passed once more. The benchmark steps are shown as a nested-loop in Alg. 1.

First, the frequency f of bit-flip occurrence is set. The frequency indicates the rate of bit-flip injection per computation, i.e., if f is set to 0.1, a bit-flip is introduced at every 10-th computation of the affected hardware component. Next, we set the affected datatype t, as in activations A or weights W. Third, we loop over the bit-flip position b, indicating the severity in magnitude change for the value of the input operand of the affected computation. Finally, we vary the number of affected multipliers m as a percentage of the accelerator's total multiply-accumulate (MAC) units. Fig. 4 visualizes these bit-flip characteristic parameters. At the core of the nested-loop in Alg. 1, we program the characteristics into the bit-flip module and allow the accelerator to perform inference over the *entire* test set. We define system failures as those cases when the prediction with hardware errors disagrees with that of the *same* CNN without any bit-flips. Therefore, failures are not counted based on the accuracy of the model or the true label of the input image. This definition aligns with existing work [6], and is fair when comparing different networks, as their underlying baseline accuracy is orthogonal to their resilience against hardware errors.

IV. Experiments

We perform experiments on the CIFAR-10 dataset, using 50K images for training and 10K test images for evaluation.

Algorithm 1 Large-scale Resilience Analysis Benchmark

```
P = EvaluateTestSet()                          ▷ Get Predictions w/o Bit-Flips
for f in F do                                  ▷ Frequency of Bit-Flip Occurance
    for t in T do                              ▷ Datatype (Weight, Activation)
        for b in B do                          ▷ Bit Position in INT8 Value
            for m in M do                      ▷ Affected MAC Units
                ProgramBitFlipModule(f,t,b,m)
                P̂ = EvaluateTestSet()
                FailRate = Count(P ≠ P̂)/TestSetSize
```

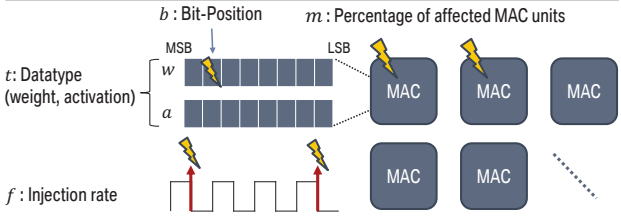


Fig. 4. Parameters to determine bit-flip characteristics of the benchmark.

The test set also serves as the hardware fault test set in Alg. 1. ResNet20 and ResNet56 represent shallow and deep baseline models for the CIFAR-10 dataset. If not otherwise mentioned, all hyper-parameters specifying the task-related training were adopted from ResNet's base implementation [15]. Pruned variants are obtained by re-implementing the reinforcement-learning-based pruning agent proposed in AMC [3]. We investigate the fault resilience of CNNs with $50\% - 60\%$ fewer operations remaining compared to their unpruned variants. For defensive training against adversarial attacks, we use the popular FastAT [4] approach and the training hyper-parameters described in the paper. To evaluate adversarial robustness, we apply a strong unseen PGD [14] adversarial attack on all considered CNNs with 20 iterations and a perturbation budget ϵ=2. We use the entropy-based calibrator of TensorRT to find the optimal v and c for each layer of the full-precision CNNs, before `INT8` execution. All CNNs are calibrated on the same dataset, i.e., the same images are passed to compute v, c of each layer of each CNN, before deployment on hardware.

We synthesize a 64 MAC unit variant of the NVDLA accelerator on the Xilinx ZCU102 board. The bit-flip module is written in Verilog and wraps around the MAC units without adding any delays to any critical paths of the accelerator design. The sets in Alg. 1 are $\mathcal{F} = \{0.1,\ 0.02,\ 0.01,\ 0.005,\ 0.002\}$, $\mathcal{T} = \{A, W\}$, $\mathcal{B} = \{5,\ 6,\ 7\}$, and $\mathcal{M} = \{25\%,\ 50\%,\ 100\%\}$, where the benchmark loops over the elements in the order they are presented here. Bit-position $b = 7 \in \mathcal{B}$ indicates a flip in the sign-bit of `INT8`. The sets $\mathcal{F}$, $\mathcal{T}$, $\mathcal{B}$, and $\mathcal{M}$ were chosen after an ablation study on the considered networks. The ranges for each bit-flip characteristic adequately represent weak-to-strong influence on CNN fault rate for the purpose of our analysis.

A. Large Scale Resilience Analysis

The results of the benchmark detailed in Alg. 1 are shown in Fig. 5. The following observations can be made:

Activation sensitivity. Flipping bits of input activations A is more likely to cause failures compared to flipping weight bits in W at any bit-flip position, on any number of multipliers and any frequency of bit-flip injection. Many memory-based and targeted bit-flip works only flip the weights of the CNN, without investigating input activations [7], [9], which are persistently more vulnerable in all our tested CNNs, and all bit-flip patterns of the benchmark.

Sign-bit sensitivity. An expected (common) observation is the high impact of the sign-bit in deciding the probability of failure. However, it is interesting to note the *degree* of its importance; in almost all cases, flipping the sign-bit in 25% of the multipliers is more potent than flipping the 6-th bit on 100% of the multipliers, at any given frequency, for both weights W and activations A. Flips on the 5-th bit (or lower, based on observations not shown for brevity) are almost negligible at low injection rates, even on 100% of the MAC units.

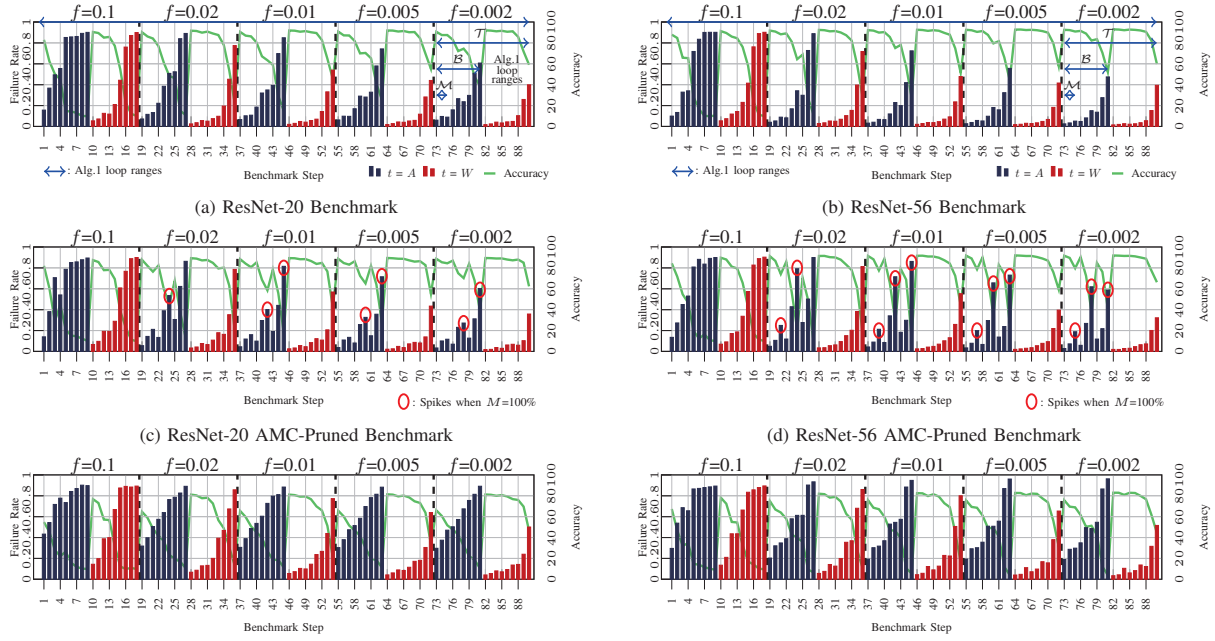
Adversarially robust CNNs are vulnerable to hardware errors. There is a clear degradation in fault resilience for adversarially robust CNNs, particularly for activation based bit-flips. We address this observation more closely in the next section. Pruned CNNs exhibit resilience properties close to their unpruned counterparts. This is justified as their scaling factors v are similar to the original (vanilla) unpruned network. However, spikes of high failure rates (marked in Fig. 5) occur when $m = 100\%$, indicating that injecting many perturbations in a CNN with fewer computations (due to pruning), leads to slightly weaker fault resilience.

Deep CNNs with batch-normalization are resilient. Deeper CNNs (56-layers) have improved fault resilience over their shallow (20-layers) counterparts for vanilla, pruned, and adversarially robust variants. The errors introduced in the early layers of the network do not *grow* with the depth of CNN. This can be credited in part to the batch-norm layers which take place after every CONV layer, regulating the maximum possible perturbation that can pass to the next layer, (1) due to calibration-time statistics (helps in lowering scaling factors v of the layers) and (2) run-time normalization. He et al. [2] show benefits of batch-norm against *targeted* (search-based) bit-flip attacks. We further show the benefits of batch-norm more generally against any HW-based faults (non-targeted). The two ResNet20 variants presented in Fig. 2 (Vanilla and No Batch-Norm) are evaluated in Tab. I. The overall mean failure rate is *doubled* in the variant without normalization, due to its high scaling factors which amplify errors in the CNN.

The results in Fig. 5 can shed light on parsimonious hardware-error resilience options. For example, the designer may apply a redundancy method on the computations against the sign-bit or allocate resilient memory holding activation bits (e.g. 8T-SRAM). More conservatively, the designer may apply that redundancy to only a subset of multipliers, e.g. 50% of the MAC array, further saving resources and area-on-chip. Such design decisions can be made based on large-scale resilience experiments, and would not be possible based on targeted bit-flip attacks [2], [8], [9].

B. In-depth Analysis of Adversarially Trained CNNs

To better understand the observation of reduced fault resilience of adversarially trained CNNs, we refer back to our problem definition in Sec. III-A. Quantization to a constrained



(a) ResNet-20 Benchmark

(b) ResNet-56 Benchmark

(c) ResNet-20 AMC-Pruned Benchmark

(d) ResNet-56 AMC-Pruned Benchmark

(e) ResNet-20 FastAT Benchmark

(f) ResNet-56 FastAT Benchmark

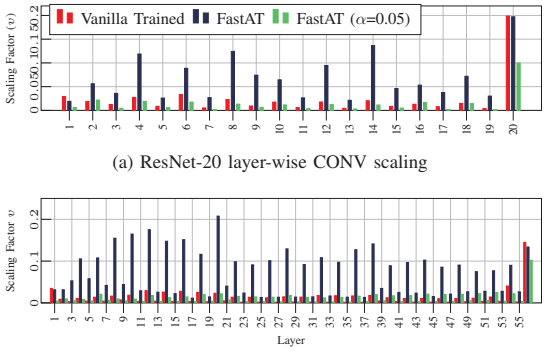
Fig. 5. Bit-flip experiments following Alg. 1 on vanilla, pruned and adversarially trained ResNet20 and ResNet56. Each bar represents the failure rate of a particular bit-flip setting $\{f, t, b, m\}$ tested over 10K test images. Each sub-figure comprises 900K bit-flip experiments.

numerical representation (INT8 or similar), implies that few discrete values must represent a wider range of dynamic real-values. The true range covered by the INT8 representation depends on the scaling factor v and clipping limit c. Since adversarially trained and vanilla CNNs are structurally identical, the first point of investigation is the data distributions of these CNNs. In Fig. 6, we show the scaling factors v for each CONV layer of the CNN after calibration, obtained through the entropy-based calibrator on the *same* calibration dataset. A clear difference can be observed, where adversarially trained (FastAT) layers can have up to $\sim 7\times$ higher scaling factors compared to the respective vanilla trained CNN.

Regularization loss $\mathcal{L}_{\text{Reg}}$ is an auxiliary loss typically added to the cross-entropy loss $\mathcal{L}_{\text{CE}}$ to penalize weights with high magnitude during neural network training. As shown in Eq. 3, this loss is scaled with the weight decay (α) hyper-parameter, to strengthen/weaken its effect on the overall loss formulation $\mathcal{L}_{\text{total}}$ during backpropagation and weight update.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \alpha\mathcal{L}_{\text{Reg}} \tag{3}$$

Weight decay α was set equivalently for both vanilla and FastAT training ($\alpha = 0.0005$ and $\alpha = 0.0004$, respectively, based on original papers [4], [15]). For the same $\mathcal{L}_{\text{Reg}}$ and α settings, the FastAT CNN incorporated large inter-layer data distributions to achieve its high robustness against adversarial attacks. Following the third insight made in Sec. III-A, adversarial training introduces pixel values which do not fall under the distribution of the standard training dataset. This forces the CNN to learn them to achieve higher adversarial robustness, *stretching* its trainable parameter distributions (weights and batch-norm parameters). When performing calibration on natural, unattacked data, the scaling factors v *grow* accordingly (Fig. 6). **This opens a *backdoor* to inter- and intra-layer HW perturbations (bit-flips) during execution, which end up having *high true magnitude* as a consequence of the larger scaling factors (v) in adversarially robust CNNs.**



(a) ResNet-20 layer-wise CONV scaling

(b) ResNet-56 layer-wise CONV scaling

Fig. 6. CONV layer scaling factors for vanilla trained and adversarially robust variants of ResNet20 and ResNet56. High weight decay (α=0.05) brings the high scaling factors v of FastAT back to vanilla levels.

Strengthening the effect of $\mathcal{L}_{\text{Reg}}$ during training pushes the weights to a more constrained distribution, and correspondingly, the activations resulted by those weights. As an initial remedy, we propose increasing the weight decay during training, which naturally shrinks the weight distributions. In Fig. 6, we show the scaling factors of FastAT-trained ResNet20 and ResNet56 CNNs with high weight decay α=0.05, bringing

TABLE I
SUMMARY OF RESULTS ON SHALLOW (RESNET-20) AND DEEP (RESNET-56) CNNS AS VANILLA, PRUNED, AND ADVERSARIALLY TRAINED VARIANTS. PERCENTAGE IMPROVEMENT SHOWN FOR FASTAT $\alpha = 0.05$ OVER REGULAR FASTAT.

Model	Train/Config	Baseline (INT8) Acc. [%]	PGD-20 Atk. Acc.[%]	Mean Failure Rate (MFR) - Lower is better								
				Overall	$f = 0.005$	$f = 0.1$	$b = 5$	$b = 7$	$m = 25\%$	$m = 100\%$	$t = W$	$t = A$
ResNet20 CIFAR-10	Vanilla	92.03	1.04	0.29	0.21	0.53	0.09	0.55	0.19	0.37	0.19	0.39
	No BatchNorm	79.12	5.01	0.60	0.57	0.69	0.48	0.72	0.53	0.70	0.40	0.81
	60% Pruned	89.59	1.21	0.27	0.17	0.56	0.11	0.47	0.15	0.41	0.19	0.35
	FastAT [4]	81.58*	72.85	0.47	0.40	0.67	0.27	0.70	0.39	0.57	0.30	0.64
	FastAT α=0.05	77.72	70.36	0.40 (15%)	0.31 (23%)	0.65 (3%)	0.20 (26%)	0.62 (11%)	0.24 (38%)	0.55 (4%)	0.33 (-10%)	0.47 (27%)
ResNet56 CIFAR-10	Vanilla	92.94	4.53	0.22	0.13	0.49	0.06	0.46	0.13	0.32	0.17	0.28
	50% Pruned	92.04	2.66	0.28	0.19	0.54	0.10	0.47	0.15	0.44	0.19	0.37
	FastAT [4]	82.71*	72.72	0.43	0.35	0.66	0.21	0.69	0.31	0.54	0.30	0.56
	FastAT α=0.05	83.37	74.72	0.36 (16%)	0.25 (29%)	0.65 (2%)	0.17 (19%)	0.63 (9%)	0.25 (19%)	0.48 (11%)	0.31 (-3%)	0.42 (25%)

*: Accuracy degradation from vanilla-training is common in state-of-the-art adversarial training to achieve high adv. robustness (see accuracy after PGD attack)

them *back* to vanilla training levels.

C. Results and Discussion

In Tab. I, we summarize the results of Fig. 5, as well as the weight decayed variants of FastAT (α=0.05). As a coarse indicator of hardware fault resilience, we provide the mean failure rates (MFR) of each CNN for the entire benchmark in Alg. 1 (Overall). Additionally, to help in understanding the effect of individual characteristics of the bit-flip patterns, we fix one bit-flip characteristic (f, b, m, or t) and measure the mean failure rate over all steps varying the other bit-flip parameters.

The observations made in Sec. IV-A are supported by the MFR presented in Tab. I. For FastAT CNNs, we see a 62% and 95% degradation in overall MFR for ResNet20 and ResNet56, respectively, compared to their vanilla-trained variants. When increasing α, the FastAT CNNs improve by up to 16% in overall MFR. More specifically, the fault resilience against activation bit-flips t=A is improved by 27% and 25% for the high weight decay FastAT ResNet20 and ResNet56, compared to the regular FastAT implementation. Although we consider accuracy orthogonal to our fault resilience analysis (explained in Sec. III-C), it is interesting to discuss the trade-offs that can be achieved in fault resilience, adversarial robustness and natural accuracy. In general, adversarial training techniques in literature incur a degradation in natural accuracy when trying to learn adversarial attacks as well as their target classification task [4]. We notice that the smaller FastAT ResNet20 suffers a further drop of 3.8 p.p. in accuracy after applying high α. However, the larger ResNet56 has a slightly improved accuracy after weight decay compared to the regular FastAT implementation. Weight decay can be harsh, particularly on smaller CNNs, as more weights approach zero and lose their feature representation capability. ResNet56 has sufficient redundancy to compensate for this (and even benefits through regularization); however, the smaller ResNet20 loses some of its natural accuracy. Although we propose weight decay as an initial, simple remedy for the adversarial training and fault resilience problem, the analysis provided in this work identifies a larger challenge in bringing robustness of both domains (adversarial attacks and hardware faults) in the same CNN. It is also important to note that adversarially trained CNNs, even with the proposed high α, are still less fault resilient than vanilla CNNs. The reason being that weight decay indeed shrunk the CONV layers' scaling factors, but the batch-norm trainable parameters (*gamma*, *beta*, etc.) are not directly affected by weight decay, leaving their scaling factors large due to adversarial training.

V. CONCLUSION

We shed light on the importance of scaling factors for maintaining fault resilience of quantized CNNs. The importance of scaling factors was verified by performing large-scale resilience analysis experiments on regularly trained, adversarially trained, batch-norm free, weight decayed, pruned, deep and shallow CNNs. We identified that adversarial training resulted in CNNs learning distributions beyond the natural dataset, which led to larger scaling factors opening a backdoor for bit-flips with large true magnitude perturbations. As a simple, first remedy, we proposed applying weight decay to bring down the CONV scaling factors, improving the resilience of adversarially robust ResNet56 by 25% on activation faults. As weight decay does not constitute a final solution to all the weaknesses of adversarially trained CNNs, future training schemes must consider the internal data distributions and scaling factors of these networks.

REFERENCES

[1] U. Zahid, G. Gambardella, N. J. Fraser, *et al.*, "Fat: Training neural networks for reliable inference under hardware faults," in *ITC*, 2020.
[2] Z. He, A. S. Rakin, J. Li, *et al.*, "Defending and harnessing the bit-flip based adversarial weight attack," in *CVPR*, 2020.
[3] Y. He, J. Lin, Z. Liu, *et al.*, "AMC: AutoML for Model Compression and Acceleration on Mobile Devices," in *ECCV*, 2018.
[4] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *ICLR*, 2020.
[5] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," in *ICLR*, 2018.
[6] Y. He, P. Balaprakash, and Y. Li, "Fidelity: Efficient resilience analysis framework for deep learning accelerators," in *MICRO*, 2020.
[7] L.-H. Hoang, M. A. Hanif, and M. Shafique, "Ft-clipact: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation," in *DATE*, 2020.
[8] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *ICCV*, 2019.
[9] J. Li, A. S. Rakin, Y. Xiong, others, *et al.*, "Defending bit-flip attack through dnn weight reconstruction," in *DAC*, 2020.
[10] F. Liao, M. Liang, Y. Dong, *et al.*, "Defense against adversarial attacks using high-level representation guided denoiser," in *CVPR*, 2018.
[11] NVIDIA Corporation, *NVDLA TensorRT Documentation*. https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html.
[12] NVIDIA Corporation, *NVDLA Open Source Project - Primer*. http://nvdla.org/primer.html.
[13] I. j. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *ICLR*, 2015.
[14] A. Madry, A. Makelov, L. Schmidt, *et al.*, "Towards Deep Learning Models Resistant to Adversarial Attacks," in *ICLR*, 2018.
[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.