2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC) | 979-8-3315-0769-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/CCWC62904.2025.10903977

RA-BNN: Constructing a Robust & Accurate Binary Neural Network Using a Novel Network Growth Mechanism to Defend Against BFA

Adnan Siraj Rakin
Binghamton University
arakin@binghamton.edu

Li Yang
University of North Carolina
lyang50@uncc.edu

Jingtao Li
SONY AI
jingtao.li@sony.com

Fan Yao
University of Central Florida
fan.yao@ucf.edu

Chaitali Chakrabarti
Arizona State University
chaitali@asu.edu

Yu Cao
University of Minnesota
Twin Cities
yucao@umn.edu

Jae-sun Seo
Cornell Tech
js3528@cornell.edu

Deliang Fan
Arizona State University
dfan@asu.edu

***Abstract*—Adversarial bit-flip attack (BFA), a type of powerful adversarial weight attack demonstrated in real computer systems has shown enormous success in compromising Deep Neural Network (DNN) performance with a minimal amount of model parameter perturbation through rowhammer-based computer main memory bit-flip. For the first time in this work, we demonstrate to defeat adversarial bit-flip attacks by developing a Robust and Accurate Binary Neural Network (*RA-BNN*) that adopts a complete BNN (i.e., weights and activations are both in binary). Prior works have demonstrated that binary or clustered weights could intrinsically improve a network's robustness against BFA, while in this work, we further reveal that binary activation could improve such robustness even better. However, with both aggressive binary weight and activation representations, the complete BNN suffers from poor clean (i.e., no attack) inference accuracy. To counter this, we propose an efficient two-stage complete BNN growing method for constructing simultaneously robust and accurate BNN, named *RA-Growth*. It selectively grows the channel size of each BNN layer based on trainable channel-wise binary mask learning with a Gumbel-Sigmoid function. The wider binary network (i.e., RA-BNN) has dual benefits: it can recover clean inference accuracy and significantly higher resistance against BFA. Our evaluation of the CIFAR-10 dataset shows that the proposed *RA-BNN* can improve the resistance to BFA by up to *100* $\times$. On ImageNet, with a sufficiently large (e.g., 5,000) number of bit-flips, the baseline BNN accuracy drops to *4.3 %* from *51.9 %*, while our *RA-BNN* accuracy only drops to *37.1 %* from *60.9 %*, making it the best defense performance.**

Corresponding author email: arakin@binghamton.edu. Code Available at: https://github.com/adnansirajrakin/IEEECCWC2025.

979-8-3315-0769-5/25/$31.00©2025 IEEE

I. Introduction

Nowadays, Deep Neural Networks (DNNs) have been deployed in many safety-critical applications [1]. The security of DNN can be compromised using adversarial input examples [2], where the adversary maliciously crafts and adds input noise to fool a DNN model. Recently, the vulnerability of model parameter (e.g., weight) [3], [4] perturbation has raised another dimension of security concern on the robustness of DNN model.

The adversarial weight attack is defined as an attacker perturbing the target DNN model parameters to achieve malicious goals. Such perturbation of model parameters is practically feasible nowadays because of the development of advanced computer hardware fault injection techniques, such as rowhammer attack [5] and under-voltage attack [6]. Moreover, the development of side-channel attacks [7]–[9] could easily leak the complete DNN model information during inference (e.g., model architecture, weights, and gradients). Such a threat allows an attacker to exploit a DNN inference machine (e.g., GPU, FPGA, mobile device) under an almost white-box (i.e., the attacker knows everything about a target DNN model) threat model. Inspired by the potential threats of fault injection and side-channel attacks, several adversarial DNN model parameter attack algorithms [3], [4], [10], [11] have been developed

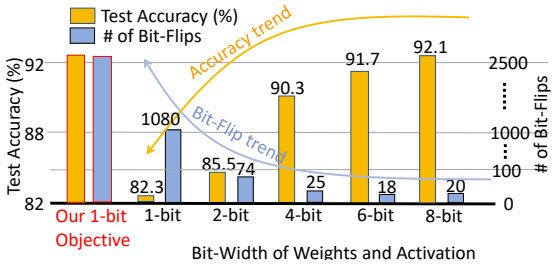

Fig. 1: *The orange curve shows the accuracy trend with Bit-Width of weights and activations of Resnet-20 on Cifar10. The blue curve shows the trend for # of bit-flips required to complete malfunction. On the* left*, the red label depicts the objective of this work which is to simultaneously improve robustness and clean accuracy.*

to study the model behavior under malicious weight perturbation.

Prior study [3] has shown that DNN with weight parameters stored in floating point IEEE format in computer main memory could be easily hijacked through single bit flip in the exponential bit, while the quantized weights are very robust against such bit-flip noise. However, our later works [4], [10], [12] have demonstrated a methodology, called adversarial bit-flip attack (BFA) as one kind of adversarial weight attacks, to be highly successful in identifying a small portion of vulnerable weight bits for noise-resilient quantized weight DNN stored in computer main memory (e.g., DRAM), through gradient ranking based searching algorithm. After locating this tiny amount (e.g., tens out of millions) of memory bits in a large DNN model, the attacker can precisely flip them through a well-studied memory fault injection technique, i.e., rowhammer [4], to hijack DNN functionality (e.g., degrading accuracy to a random guess level). In our prior work [4], we have demonstrated in the real computer system that the bit-flip attack could successfully identify vulnerable weight bits and destroy the functionality of a running DNN model in minutes through precise flipping techniques .

A series of defense works [13], [14] have attempted to mitigate the potent threat of BFA. Among them, the binary weight neural network from our prior work [13] has proven to be the most successful one. However, [13] has not explored the impact of binary activation. For a binary weight neural network, with a sufficiently large number of attack iterations (e.g., hundreds of bit-flips, rather than tens), the attacker can still successfully degrade its accuracy to a random guess level for specific models. Moreover, due to aggressively compressing the floating-point weights (i.e., 32 bits or more) into binary (1 bit), Binary Neural Network (BNN) inevitably sacrifices its clean model accuracy by *10-30 %*, which is widely discussed in many prior works [15]–[17].

As a motivation of this work, we tested the *accuracy and robustness trade-off* of a ResNet-20 [18] network with different bit-width of quantized weights and activation using CIFAR-10 dataset [19], following the same bit-flip attack methods in prior works [4], [10]. As displayed in Fig. 1, a lower bit-width network increases network robustness against BFA, especially for BNN (1 bit) – a trend similar to prior work [13]. Besides, BNN has additional computation and memory benefits, which makes it a great candidate to reduce neural network computation complexity [20]. As a result, prior works [16] investigate different ways of training a *complete BNN* (i.e., both weight and activation are binary). However, a general conclusion is that complete BNN suffers from heavy inference accuracy loss. In Fig. 1, we observe a similar trend where decreasing the bit-width of a model affects the inference accuracy negatively. Such observations present *a challenging optimization problem where a lower bit-width network has improved robustness but at the cost of lower accuracy*.

To mitigate this, recent works have proposed different methods to improve the accuracy of a BNN [15], [21]–[24]. Among them, increasing the network width [24] is a proven strategy to improve BNN accuracy, with increasing model size. On the other side, larger width is demonstrated to improve the resilience against adversarial weight [10] and input [25] noise. Hence, in this work, we propose to leverage this dual property of a wider neural network: first, to recover the accuracy of a binary neural network and, second, to defend against BFA. However, increasing the width of a model formally known as *model growing* is a multidimensional optimization problem and requires a more sophisticated design, especially for a complete BNN.

Our goal here is to develop a complete binary neural network (BNN) to improve its robustness (i.e., resistance to bit-flip attack (BFA)), while not sacrificing the clean accuracy (see Fig. 1). To achieve this, we propose *Robust & Accurate Binary Neural Network* (*RA-BNN*), a novel defense scheme against BFA, comprising two key components. First, inspired by our prior work [13], we take advantage of BNN's capability to provide improved resistance against BFA by completely binarizing both activations and weights of every DNN layer. Second, to further boost robustness and address the clean model accuracy loss, we develop an efficient network growing method designed for BNN, *RA-Growth*, which grows the

output channels of every BNN layer based on trained channel masks. We summarize our contributions as:

- *First,* our prior conference work [13] is the first to investigate and analyze the effect of binary weights in defending against BFA. In this work, after further analyzing the impact of binary activation, our proposed *RA-BNN*, utilizes binary neural network's (BNN) intrinsic robustness improvement against BFA to binarize both weight and activation of a DNN model as an effective defense method. Note that, unlike most prior BNN related works, RA-BNN binarizes the weights of *every layer*, including the first and last layer. We refer this as *complete BNN*. All activations are binarized except the input into the first (i.e., input image) and last layer (i.e., final classification layer).
- *Second,* to compensate clean model accuracy loss of a complete BNN model, we propose a novel binary model growing scheme, *RA-Growth*, intending to construct a simultaneously robust and accurate binary neural network. Our proposed RA-Growth adopts a trainable mask-based growing method to gradually and selectively grow the output channels of BNN layers. Since the network growing inevitably requires immense computing complexity, to improve training efficiency, our RA-Growth follows a two-stage training mechanism with an early stop mechanism during network growth. The first stage is *to jointly train binary weights and channel-wise masks through Gumbel-Sigmoid function at the early iterations of training.* As the growth of binary channels converges, it goes to the second stage, where the channel growing is stopped, and only the binary weights will be trained based on the network structure trained in stage-1 to minimize accuracy loss further.
- *Third,* relying on our proposed binary model growing scheme, the RA-BNN achieves the best defense performance ever reported in the literature, with significantly improved robustness against BFA. Meanwhile, it recovers the clean accuracy of a binary model close to the 8-bit counterpart with a smaller model size.
- We perform extensive experiments on CIFAR-10, CIFAR-100 and ImageNet datasets on popular DNN architectures (e.g., ResNet, VGG), where our RA-BNN all achieves the *best defense performance to date.* The evaluation on CIFAR-10 shows that the RA-BNN can improve the resistance to BFA by more than *29-104* $\times$. On ImageNet, for the first time, RA-BNN can completely defeat BFA (i.e., *5,000* bit-flips only degrade the accuracy to around *37%*). In comparison, the accuracy of baseline BNN without growing degrades to *4.3 %* with *5,000* bit-flips.

II. Background and Related Work

A. Adversarial Weight Attack

Adversarial input example attack [25], [26] has been the primary focus of DNN security analysis. However, the recent advancement of adversarial weight attack [3], [4], [10], [12] has also exposed serious vulnerability issues, where the adversary maliciously perturbs the weight parameters to compromise the performance of DNN. Among them, our prior proposed adversarial Bit-Flip Attack (BFA) [4], [10], is one of the most powerful attacks, which is demonstrated in real computer system to inject malicious fault into a DNN model thorough rowhammer to flip an extremely small amount of memory bits of the weight parameters stored in main memory (i.e., DRAM). Such rowhammer based BFA on the computer main memory was proven to be successful even with the traditional error correction and detection method, like ECC, etc. [4], [27]. Following the accuracy degradation attack through BFA, a more recent variant of our BFA can perform a targeted attack or Trojan attack [11], [12], where the adversary can fool the DNN to predict inputs to a specific target class. To dive deep into the adversarial bit-flip attack, next, we introduce the threat model, attack methodology, and existing defense measures. To counter the efficacy of BFA, we have explored weight binarization [13], piece-wise weight clustering [13], weight reconstruction [14], and model size increasing [11] in our prior works. Among them, the binarization of the weight parameters [13] shows significant improvement in limiting the efficacy of BFA, but sacrificing clean accuracy a lot.

B. Threat Model

Prior works [3], [4], [10] of malicious weight attacks have concluded that attacking a model with quantized weights (e.g., 8-bit or lower) is more difficult than attacking a full-precision weight model, since single bit-flip in the exponential bit of IEEE format of floating point representation could destroy the system [3]. While, the weight quantization could restrict the range of weight values that an attacker might manipulate. Besides, DNN weight quantization is one of the most popular techniques to compress large-scale network size. Thus, in

this work, we will follow the same BFA threat model [4], [10], where BFA is applied to a more robust quantized DNN model. Meanwhile, the attacker can access model parameters, architecture, and gradient information. Such a white-box threat model is practical because of the advancement of side-channel model information leakage attacks [7]–[9]. The attacker has access to a sample test batch input ($\boldsymbol{x}$) with the correct label($\boldsymbol{t}$). However, the attacker has no access to training information (e.g., training data, learning rate, algorithm).

Finally, in a different line of work, training channel-specific masks to explore the architecture space have been investigated previously in both model pruning [28], [29], and model growing [30] domain. Recently, such pruning schemes have become popular in related Network Architecture Search (NAS) methods [31] as well. However, prior model growing methods ignore the challenges of training a complete binary neural network. Our work is the first to optimize such a growing scheme in a complete BNN and attempts to defend against adversarial attacks with assistance from model growing.

III. Defense Rationale

The defense rationale of *RA-BNN* is inspired by three critical observations outlined here.

First, the experiment in Table I shows that complete BNN models are intrinsically more resilient to BFA. Unlike prior works [13], a complete binary model (with both binary weights and activations) requires *∼12* × more number of bit-flips than binary weight only model to achieve similar accuracy degradation after attack.

Observation 1. *A complete binary neural network (i.e., both binary weights and activations) significantly improves the robustness against bit-flip attack (see Table I).*

TABLE I: ***Effect of Binarization.*** *ResNet-20 performance on CIFAR-10 dataset. We report the # of bit-flips required to degrade the DNN accuracy to 10.0 % as the measure of robustness [10].*

Model Type	Clean Acc. (%)	Acc. After Attack (%)	# of Bit-Flips
8-bit weigh [10]	92.7	10.0	28
Binary weight [13]	89.01	10.99	89
Binary weight + activation (ours)	*82.33*	*10.0*	*1080 (12 ×)*

Second, apart from binarization, our prior works [4], [13] have also concluded that increasing the network width (i.e., input and output channel multiplier) of a DNN helps improve its resistance against BFA. For example, in *Table II*, we observe an 8-Bit model with larger channel width (e.g., 2/4) requires an increasing amount of bit-flips to deteriorate its functionality.

Observation 2. *A DNN model's resistance to BFA can be enhanced by increasing the input and output channel width.*

TABLE II: ***Effect of Network Width.*** *To model the effect of channel multiplier against BFA, we increase the channel multiplier by 1/2/4 × of an 8-Bit ResNet-20 model and report the # of bit-flips required to compromise the inference accuracy to ∼10 %.*

Channel Multiplier	×1	×2	×4
bit-flip #	28	30	36

Finally, as summarized in *Table I*, while the complete BNN may provide superior robustness, it comes at the cost of reducing the clean accuracy by 10.0 % (from 92.7 % to 82.33 %) even on a small dataset with ten classes. To counter this, [24] has discussed a potential method of recovering the accuracy of a binary model by increasing the input and output channel width (i.e., channel multiplier) of each layer. To summarize the effect of channel multiplier in Fig. 2, we vary the BNN channel multiplier as 1/2/3 and report the clean accuracy. As the channel multiplier increases from 1 to 3, it is possible to recover the accuracy of a complete BNN model close to an 8-Bit model. However, naively applying a uniform channel multiplier to all layers causes quadratic model size and computing complexity increment.

Observation 3. *Utilizing the channel multiplier with a large factor (e.g., 3) can largely compensate BNN accuracy degradation.*

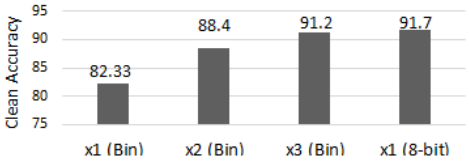

Fig. 2: *Accuracy of three binary models with channel multiplier 1/2/3 which will yield a quadratic model size of 1/4/9 × and an 8-bit model (x1).*

Design objective. In summary, inspired by observation-1, we aim to construct an intrinsically robust BNN model with both binary weight and activation. Second, inspired by observation-2 and -3,, we propose to leverage the two beneficial properties of a wide neural network: i) it supplements the intrinsic robustness improvement of a complete BNN model,

and ii) it recovers a major portion of the clean accuracy loss from binarization. Therefore, our objective is to construct a simultaneously robust and accurate BNN through channel-wise *trainable network widening (i.e., growing)*, instead of naively applying a uniform channel multiplier across all layers. The technical ***challenge*** is how to minimize the model size increment through optimizing fine-grained channel multipliers for each BNN layer.

IV. Proposed RA-BNN

In this section, we will introduce our proposed RA-BNN. To summarize, RA-BNN improves robustness and accuracy simultaneously through the following two methods. *First*, we *binarize* the weights of **all DNN layers**, including the first and last layers. Further, to improve the robustness, we also binarize the inputs going into each layer (i.e., activation), except the first and last one. *Second*, to recover the clean accuracy and further boost the robustness, we develop *RA-Growth*, a fast mask-based method of growing a complete BNN with a trainable channel width of each layer.

As shown in *Fig. 3*, *RA-Growth* is a two-stage training scheme. In stage-1 (*growing stage*), the model grows gradually and selectively from the initial baseline model to a larger model in a channel-wise manner. It is achieved by learning the binary mask associated with each weight channel (growing when switching from '0' to'1' at the first time) to enable training-time growth (i.e., network growing simultaneously with weight parameter training) for training cost reduction. As the network growth becomes stable after a few initial iterations, the expansion will stop and enter stage-2. In stage-2, the new model obtained from stage-1 will be re-trained to minimize the cross-entropy loss. After completing both the growing and re-training stages, we expect to get a wide complete BNN with simultaneously improved robustness and clean accuracy.

Finally, another essential optimization in our growing method dedicated to binary neural networks is leveraging a differentiable Gumbel-Sigmoid approach to learn binary masks. Typically model growing methods (e.g., [30]) generate the binary mask by learning full precision mask following non-differential Bernoulli sampling leading to multiplication between binary weights and full precision masks in the forward path of a BNN model. In contrast, our method only uses binary multiplication with both operands (i.e., weight and mask) in binary format, reducing computing complexity. Such optimization could lead to efficient binary multiplication between binary weight and mask in the forward path. Next, we will describe each component of constructing *RA-BNN*.

A. Binarization

The first step is to construct a complete BNN with binary weights and activations. Training a neural network with binary weights and activations presents the challenge of approximating the gradient of a non-linear sign function [17]. To compute the gradient efficiently and improve the binarization performance, in this work, we use training aware binary approximation function [15] instead of the direct sign function:

$$f(\boldsymbol{z}) = \begin{cases} k \cdot (-sign(\boldsymbol{z})\frac{t_s^2 \boldsymbol{z}^2}{2} + \sqrt{2}\boldsymbol{z}), & \text{if } |\boldsymbol{z}| < \frac{\sqrt{2}}{t_s}. \\ k \cdot sign(\boldsymbol{z}), & \text{otherwise.} \end{cases} \tag{1}$$

$$t_s = 10^{-2+\frac{3i}{T}}; k = max(\frac{1}{t_s}, 1), \tag{2}$$

where i is the current training iteration and T is the total iterations. At the early training stage, $\frac{i}{T}$ is low, and the above function is continuous. As the training progresses, the function gradually becomes a sign function whose gradient can be computed as:

$$f'(\boldsymbol{z}) = \frac{\delta f(\boldsymbol{z})}{\delta \boldsymbol{z}} = max(k \cdot \sqrt{2} t_s - |t_s^2 \boldsymbol{z}|, 0) \tag{3}$$

For weight binarization, z represents full-precision weights ($\boldsymbol{z} = \boldsymbol{w}_{fp}$), and for activation binarization, z represents full-precision activation input going into the next convolution layer ($\boldsymbol{z} = a_{fp}$). In our RA-BNN, we binarize the weights of every layer including the first and last layer. However, for activation, we binarize every input going into the convolution layer, except the first layer input.

B. RA-Growth

As shown in Fig. 3, our RA-Growth consists of two training stages: *Stage-1: Growing*; *Stage-2: Re-Training*. The objective of the growing stage is to learn to grow the output channel size of each layer during the initial iterations of training. As the network architecture becomes stable, the growing stage will stop and generate a new model architecture for stage-2 to re-train the weight parameters.

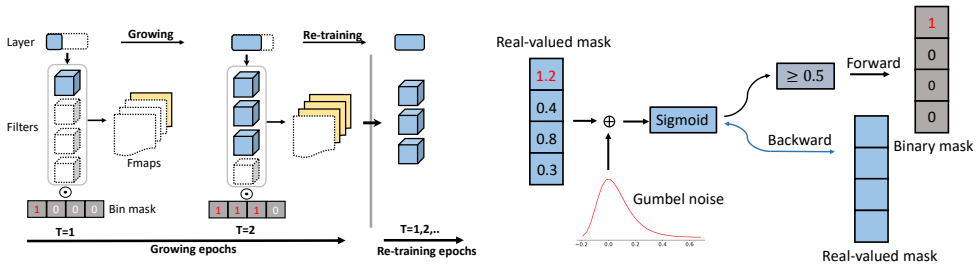

Fig. 3: ***Left:*** *Overview of the two-stage RA-Growth. i) Growing Stage & ii) Re-training Stage.*

a) Stage-1: Growing: In order to gradually grow the network by increasing the output channel size, we use the channel-wise binary mask as an indicator (i.e., on/off). Considering a convolution layer with input channel size c_{in}, output channel size c_{out}, filter size $k \times k$ and output filter $\boldsymbol{w}^j \in \mathbf{R}^{\{k \times k \times c_{out}\}}$; then the j^{th} ($j \in \{1, 2, ..c_{out}\}$) output feature from a convolution operation becomes:

$$h^j_{out} = \textbf{Conv}(h_{in}, \boldsymbol{w}^j_b \odot \boldsymbol{m}^j_b) \tag{4}$$

where $\boldsymbol{w}^j_b \in [-1, 1]$ is a binary weight ($\boldsymbol{w}^j_b = f(\boldsymbol{w}^j_{fp})$), and $\boldsymbol{m}^j_b \in [0, 1]$ is a binary mask. When the binary mask is set to 0, the entire j^{th} output filter is detached from the model. As both weight and mask are in binary format, such element-wise multiplication can be efficiently computed using a binary multiplication operation (i.e. XNOR), instead of floating-point multiplication. This is the reason we need to guarantee both the weight and mask are in binary format in the forward path. The growing stage starts with a baseline model (i.e., $\times 1$ channel width) and each channel is associated with a trainable mask. During training, it will create a new output channel (i.e., growing) when the mask switches from 0 to 1 for the first time. We illustrate an example of this growing procedure in fig. 3. We can mathematically formalize the growing stage optimization objective as:

$$\min_{\boldsymbol{w}_b, \boldsymbol{m}_b} \mathcal{L}_E(g(\boldsymbol{w}_b \odot \boldsymbol{m}_b; \boldsymbol{x}_t), \boldsymbol{y}_t) \tag{5}$$

where g(·) is the complete BNN inference function. However, the discrete states (i.e. non-differential) of the binary masks $\boldsymbol{m}_b$ make their optimization using gradient descent a challenging problem.

b) Training Binary Mask.*:* To generate the binary trainable mask, we avoid training a learnable real-valued mask ($\boldsymbol{m}_{fp}$) followed by a hard threshold function (i.e., sign function). Because such a hard threshold function is not differentiable, the general solution is to approximate the gradients by skipping the threshold function during back-propagation and update the real-value masks directly. Our approach is different in that we eliminate the gradient estimation step and make **whole mask learning procedure differentiable**, and thus compatible with the existing gradient-based back-propagation training process.

First, we relax the hard threshold function to a continuous logistic function:

$$\sigma(\boldsymbol{m}_{fp}) = \frac{1}{1 + \exp(-\beta \boldsymbol{m}_{fp})}, \tag{6}$$

where β is a constant scaling factor. Note that the logistic function becomes closer to the hard thresholding function for higher β values.

Then, to learn the binary mask, we leverage the Gumbel-Sigmoid trick, inspired by Gumbel-Softmax [28], [29], [32] that performs a differential sampling to approximate a categorical random variable. Since we can view sigmoid as a special two-class case of Softmax, we define $p(\cdot)$ using the Gumbel-Sigmoid trick as:

$$p(\boldsymbol{m}_{fp}) = \frac{\exp((\log \pi_0 + g_0)/T)}{\exp((\log \pi_0 + g_0)/T) + \exp((g_1)/T)}, \tag{7}$$

where π_0 represents $\sigma(\boldsymbol{m}_{fp})$. g_0 and g_1 are samples from Gumbel distribution. The temperature T is a hyper-parameter to adjust the range of input values, where choosing a larger value could avoid gradient vanishing during back-propagation. Note that the output of $p(\boldsymbol{m}_{fp})$ becomes closer to a Bernoulli sample as T is closer to 0. We can further simplify Eq. (7) as:

$$p(\boldsymbol{m}_{fp}) = \frac{1}{1 + \exp(-(\log \pi_0 + g_0 - g_1)/T)} \tag{8}$$

Benefiting from the differentiable property of Eq. (6) and Eq. (8), the real-value mask $\boldsymbol{m}_{fp}$ can be embedded with existing gradient based back-propagation training without gradient approximation. During training, most values in the distribution of $p(\boldsymbol{m}_{fp})$ will move towards

either 0 or 1. To represent $p(\boldsymbol{m}_{fp})$ as binary format, we use a hard threshold (i.e., 0.5) during forward-propagation of training, which has no influence on the real-value mask to be updated for back-propagation as shown in Fig. 3. Finally, the optimization aim in Eq. (5) can be reformulated as:

$$\min_{\boldsymbol{w}_{fp},\boldsymbol{m}_{fp}} \mathcal{L}_E(g(f(\boldsymbol{w}_{fp}) \odot p(\boldsymbol{m}_{fp}); \boldsymbol{x}_t), \boldsymbol{y}_t) \tag{9}$$

c) Stage-2: Re-training: After stage-1, a new grown BNN structure (with channel-wise mask $\boldsymbol{m}_b$ for each layer) is achieved. Then, in stage-2, we construct the new BNN model $\boldsymbol{w}^*_{fp}$ using the weight channels with mask values as 1 in $\boldsymbol{m}_b$ and discard the rest. In stage-2 training, the newly constructed BNN's weight parameters will be trained involving no masks. The re-training optimization objective can be formulated as:

$$\min_{\boldsymbol{w}^*_{fp}} \mathcal{L}_E(g(f(\boldsymbol{w}^*_{fp})); \boldsymbol{x}_t), \boldsymbol{y}_t) \tag{10}$$

After completing the re-training stage, we expect to obtain a complete BNN with simultaneously improved robustness and clean model accuracy.

TABLE III: ***Evaluation of VGG-Small model****. Each model activation and weights are binarized except for the Binary (W) (i.e., binary weights only). We report the gain compared to the binary (W) model. RA-BNN improves the post-attack accuracy (PA) by 74-100 ×.*

Model Bit Width (W+A)	*Model Size (Mb)*	*Clean acc. (%)*	*Post-Attack acc. (%)*	*# of Bit Flips*
Un-Targeted Attack				
8-bit	37.38	93.47	10.8	34
4-bit	18.64	90.76	10.93	26
Binary(W)	4.66	92.30	10.86	48
Binary(W+A) (***ours***)	4.66	88.75(↓ 3.55)	10.99	1969 (× 41)
RA-BNN (***ours***)	***26.16***	***91.75***	***80.39***	***5000 (× 104)***
Targeted Attack				
8-bit	37.38	93.47	10.75	28
4-bit	18.64	90.76	10.53	13
Binary (W)	4.66	92.30	10.90	67
Binary(W+A) (***ours***)	4.66	88.75	10.99	3956 (× 59)
RA-BNN (***ours***)	***26.16***	***91.75***	***70.47***	***5000 (× 74)***

V. Experimental Details

A. Dataset and Architecture

We evaluate RA-BNN on three popular vision datasets: CIFAR-10 [19], CIFAR-100 [19] and ImageNet [35]. We follow the same DNN architecture, training configuration, and hyper-parameters as rotated binary neural network (RBNN) [15]. We also follow the same weight binarization method as RBNN including applying the rotation on the weight tensors before binarization.

TABLE IV: ***Evaluation of ResNet-18 model****. RA-BNN improves the post attack accuracy (PA) by 28.93 % against un-targeted BFA in comparison to Binary(W).*

Model Bit Width	*Model Size (Mb)*	*Clean acc. (%)*	*Post-Attack acc. (%)*	*# of Bit Flips*
Un-Targeted Attack				
8-bit	89.44	93.74	10.01	17
4-bit	44.72	93.13	10.87	30
Binary (W)	11.18	93.70	10.97	157
Binary(W+A) (***ours***)	11.18	91.10	10.98	666
RA-BNN (***ours***)	***31.85 (× 2.84)***	***92.92***	***39.90 (↑28.93)***	***5000***
Targeted Attack				
8-bit	89.44	93.74	10.71	20
4-bit	44.72	93.13	10.21	21
Binary (W)	11.18	93.70	10.99	145
Binary(W+A) (***ours***)	11.18	91.10	10.95	493
RA-BNN (***ours***)	***31.85 (× 2.84)***	***92.92***	***10.99***	***4230(×29)***

TABLE V: ***Evaluation on CIFAR-100 and ImageNet dataset.*** *We show RA-BNN post attack accuracy improves by 39 % on CIFAR-100 and 33 % on ImageNet compared to a complete binary (W+A) model.*

Model Bit Width	*Model Size (Mb)*	*Clean acc. (%)*	*Post-Attack acc. (%)*	*# of Bit Flips*
ImageNet				
Baseline (8-bit)	93.60	69.10	0.11	13
Binary(W+A) (***ours***)	11.70	51.90	4.33	5000
RA-BNN (***ours***)	***73.09(×6)***	***60.90(↑9)***	***37.10 (↑ 32.77)***	***5000***
CIFAR-100				
Baseline (8-bit)	90.55	75.19	1.0	23
Binary(W+A) (***ours***)	11.32	66.14	15.47	5000
RA-BNN (***ours***)	***39.53(×3.5)***	***72.29(↑6)***	***54.22(↑38.75)***	***5000***

B. Attack and Defense Hyper-Parameters.

In general, Un-targeted BFA [10] degrades the DNN accuracy close to a random guess level (1/no. of class)×100 (e.g., 10 % for CIFAR-10, 1 % for CIFAR-100 and 0.1 % for ImageNet). We also perform N-to-1 targeted [11] attack where the attacker classifies all the inputs into a specific target class, which again will degrade the overall test accuracy to a random guess level. We run the attack three rounds and report the best round in terms of the number of bit-flips required to achieve the desired goal.

Defense success definition. We assume the maximum # of bits the attacker can flip is 5,000, based on practical prior experiments. [36] shows, to get around strong bit-flip protection schemes (e.g., SGX [37]), requires around 95 hours to flip 91 bits using double-sided row-hammer attack. At this rate, to flip 5,000 bits would cost around

TABLE VI: ***Comparison to state-of-the-art binary ResNet-18 models on Image-Net.*** *It shows RA-BNN achieves 33 % higher post-attack accuracy (PA) in comparison to existing binary neural networks. The accuracy range represents two corner cases of with and without binarizing the first and last layer weights.*

Categories	*Methods*	*Model Size (Mb)*	Clean Acc. (%)	*Post-Attack Acc. (Bit-Flips)*
8-bit	Baseline	93.6	69.1	0.1 (13)
Prior BNN works	[16], [33], [34]	11.7	42.7-57.1	$\sim$ 4.3 (5000)
State-of-the-art BNNs	[15], [23]		51.9-59.9	
RA-BNN	***Ours***	***73.09***	***60.9-62.9***	$\sim$ ***37.1*** (5000)

TABLE VII: *Comparison to other competing defense methods on CIFAR-10 dataset evaluated attacking a ResNet-20 model.*

Models (Model Size Comparison ×)	*Clean Acc.(%)*	*Post-Attack acc.(%)*	*Bit-Flips #*
Baseline ResNet-20 [10] (8×)	91.71	10.90	20
Piece-wise Clustering [13] (8×)	90.02	10.09	42
Binary weight [13] (1×)	89.01	10.99	89
Model Capacity × 16 [11], [13] (16×)	93.7	10.00	49
Weight Reconstruction [14] (8×)	88.79	10.00	79
***RA-BNN (proposed* (7×))**	***90.18***	***10.00***	***1150***

seven months, which is practically impossible. Thus, if RA-BNN could tolerate 5,000 bit flips, we can safely claim to defeat BFA.

VI. Results

A. Defense evaluation on CIFAR-10

We evaluate our defense on CIFAR-10 dataset using a small VGG (in *Table III*) & a ResNet-18 (in *Table IV*) architecture. In both tables, RA-BNN achieves $\sim$ *20-70 %* gain in accuracy after attack compared to a binary (W) (i.e., only binary weights) network.

First, in *Table III*, a baseline 8-bit model requires just *34* bit-flips to degrade the accuracy from *93.47 %* to *10.8 %*. Next, we apply complete binary weights to increase the number of bit-flip requirements to *48*. This gain in resisting BFA could be exponentially increased (i.e., *41* ×) by applying both binary weights and activations. However, a complete binary (W+A) network suffers from *3.55 %* clean accuracy degradation. The proposed RA-BNN thus comes with two benefits: i) it recovers clean accuracy & ii) it improves resistance to BFA. For example, RA-BNN recovers the clean accuracy back to *91.75 %*. In addition, we observe that *104* × additional bit-flips could not decrease the accuracy of the target model to a random guess level. We achieve a significant gain in accuracy and resistance to BFA (both targeted/un-targeted) at the expense of *5.61* × model size compared to a binary network. In conclusion, using a similar model size as a 6-bit model, we can completely defend a bit-flip attack using our binary model, whereas a 6-bit model is extremely vulnerable to BFA.

The results in *Table IV* show that the attack cannot break our defense on *ResNet-18* architecture as well. For un-targeted attack, we show that even after *5,000* flips the test accuracy still holds at *39.90 %*, while it requires only *17-30* flips to cause malfunction in the baseline models (e.g., 4-bit/8-bit). Thanks to RA-BNN, on top of the robustness gain, our BNN model also achieves a *2.8* × compression rate compared to the 8-bit model. Similar to un-targeted attack, we observe a robustness gain (i.e., × *29*) for targeted attack as well.

B. Defense evaluation on CIFAR-100 and ImageNet.

Our proposed *RA-BNN* improves the resistance to BFA on large-scale datasets (e.g., CIFAR-100 and ImageNet) as well. As presented in *Table V*, the baseline 8-bit models require less than *23* bit-flips to degrade the inference accuracy close to a random guess level (i.e., 0.1 %). While binary model improves the resistance to BFA, it is still possible to significantly reduce (e.g., *4.3 %*) the inference accuracy with a sufficiently large (e.g., *5,000*) number of bit-flips. In addition, a complete binary model (i.e., weights+activation of every layer) suffers from *19 %* accuracy loss on ImageNet. But our proposed RA-BNN outperforms both 8-bit and binary baseline, as even after *5000* bit flips, the accuracy only degrades to *37.1 %* while maintaining a *60.9 %* clean accuracy on ImageNet. Similarly, on CIFAR-100, *RA-BNN* improves the clean accuracy by *6 %* & post attack accuracy by *39 %* compared to a binary model with a similar model size as a 4-bit model.

C. Comparison to state-of-the-art (SOTA).

We summarize our *RA-BNN* defense performance compared to other SOTA BNN in *Table VI*. We divide the

existing BNN works into two classes based on accuracy range: i) prior BNN works & ii) most recent SOTA BNN training methods. Our method achieves on-par accuracy with the other state-of-the-art BNN methods. For example, we achieve an impressive *60.9 % Top-1* clean accuracy on ImageNet dataset. To evaluate the existing BNN models against BFA, we do not find any complete binary neural network open-sourced from prior works. Hence, we took a representative (i.e., [15]) one and trained a complete binary version from scratch. After attacking this model, we observed our defense achieves *33 %* higher post-attack accuracy than a complete SOTA binary model. In addition, our RA-Growth is compatible with any existing techniques (e.g., [15], [23]) to improve BNN accuracy and robustness. It is also compatible with recently proposed DNN architectures [22], [38] designed for binary neural networks. Hence, *RA-Growth* is not necessarily a competing method for these existing BNN techniques. As a trade-off, RA-BNN would cost higher memory (e.g., 4-6 $\times$) than a binary neural network, while still generating a smaller model size than an 8-bit. Such a trade-off between privacy/robustness and memory/computation is present across all the defenses [25], [26] in the adversarial domain.

Apart from existing BNN methods, we also present a comparative study with state-of-the-art BFA defenses in *Table VII*. Again *RA-BNN* outperforms all existing defenses with improved accuracy and robustness. Compared to the best defenses against BFA in the literature, *RA-BNN* enhances the robustness (i.e., number of bit-flips) by $\sim$*15-28* $\times$. At the same time, the model size overhead is less than all the existing defenses except the binary weight-only case. However, our binary activation and weight would make the computation at the inference much more efficient through hardware optimizations [39]. In addition, RA-BNN hardly adds any computational cost. Since prior works [40] have shown that 1-bit weight and activation can reduce the computation significantly ($\sim$ 0.1x), compensating for the increased network width. In summary, proposed RA-BNN presents a novel way of adopting binary neural network as a defense against BFA at the same time improving accuracy with little/no computation overhead.

VII. Conclusion

In this work, for the first time, we demonstrate our proposed *RA-BNN* could completely defeat BFA. We use BNN's intrinsic robustness to bit-flips and propose *RA-Growth* method to increase the channel size of a BNN to simultaneously recover clean data accuracy and further enhance the defense efficiency. We demonstrate through extensive experiments that low bit-width DNN models constructed by RA-BNN satisfy the conflicting requirements of high robustness and high clean accuracy. We successfully construct a complete BNN to defend against BFA with a significantly improved clean accuracy and memory budget.

VIII. Acknowledgement

This work is supported in part by the National Science Foundation under Grant No.2314591, No.2505326, No.2452573, No.2452657, No.2503906 and No.2505209.

References

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *ICLR*, 2015.

[3] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitraş, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," *arXiv:1906.01017*, 2019.

[4] F. Yao, A. S. Rakin, and D. Fan, "Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 1463–1480.

[5] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," in *ACM SIGARCH Computer Architecture News*. IEEE Press, 2014, pp. 361–372.

[6] D. R. Gnad, F. Oboril, and M. B. Tahoori, "Voltage drop-based fault attacks on fpgas using valid bitstreams," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–7.

[7] X. Hu, L. Liang, S. Li, L. Deng, P. Zuo, Y. Ji, X. Xie, Y. Ding, C. Liu, T. Sherwood *et al.*, "Deepsniffer: A dnn model extraction framework based on learning architectural hints," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 385–399.

[8] L. Batina, S. Bhasin, D. Jap, and S. Picek, "Csi nn: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 515–532.

[9] A. S. Rakin, M. H. I. Chowdhuryy, F. Yao, and D. Fan, "Deepsteal: Advanced model extractions leveraging efficient weight stealing in memories," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1157–1174.

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

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

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

[13] Z. He, A. S. Rakin, J. Li, C. Chakrabarti, and D. 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)*, June 2020.

[14] J. Li, A. S. Rakin, Y. Xiong, L. Chang, Z. He, D. Fan, and C. Chakrabarti, "Defending bit-flip attack through dnn weight reconstruction," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[15] M. Lin, R. Ji, Z. Xu, B. Zhang, Y. Wang, Y. Wu, F. Huang, and C.-W. Lin, "Rotated binary neural network," *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

[16] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 722–737.

[17] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[19] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," *http://www. cs. toronto. edu/kriz/cifar. html*, 2010.

[20] F. Conti, P. D. Schiavone, and L. Benini, "Xnor neural engine: A hardware accelerator ip for 21.6-fj/op binary neural network inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2940–2951, 2018.

[21] J. Bethge, C. Bartz, H. Yang, Y. Chen, and C. Meinel, "Meliusnet: An improved network architecture for binary neural networks," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1439–1448.

[22] Z. Liu, Z. Shen, M. Savvides, and K.-T. Cheng, "Reactnet: Towards precise binary neural network with generalized activation functions," in *European Conference on Computer Vision*. Springer, 2020, pp. 143–.

[23] H. Qin, R. Gong, X. Liu, M. Shen, Z. Wei, F. Yu, and J. Song, "Forward and backward information retention for accurate binary neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2250–2259.

[24] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "Wrpn: Wide reduced-precision networks," *arXiv preprint arXiv:1709.01134*, 2017.

[25] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[26] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[27] L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting correcting codes: On the effectiveness of ecc memory against rowhammer attacks," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 55–.

[28] M. Kang and B. Han, "Operation-aware soft channel pruning using differentiable masks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5122–5131.

[29] Y. Wang, X. Zhang, X. Hu, B. Zhang, and H. Su, "Dynamic network pruning with interpretable layerwise channel selection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6299–6306.

[30] X. Yuan, P. Savarese, and M. Maire, "Growing efficient deep networks by structured continuous sparsification," *arXiv preprint arXiv:2007.15353*, 2020.

[31] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, P. Vajda, and J. E. Gonzalez, "Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[32] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[33] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," *arXiv preprint arXiv:1711.11294*, 2017.

[34] A. Bulat and G. Tzimiropoulos, "Xnor-net++: Improved binary neural networks," *arXiv preprint arXiv:1909.13863*, 2019.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[36] D. Gruss, M. Lipp, M. Schwarz, D. Genkin, J. Juffinger, S. O'Connell, W. Schoechl, and Y. Yarom, "Another flip in the wall of rowhammer defenses," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 245–261.

[37] F. McKeen, I. Alexandrovich, I. Anati, D. Caspi, S. Johnson, R. Leslie-Hurd, and C. Rozas, "Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave," in *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*, 2016, pp. 1–9.

[38] J. Bethge, C. Bartz, H. Yang, Y. Chen, and C. Meinel, "Meliusnet: Can binary neural networks achieve mobilenet-level accuracy?" *arXiv preprint arXiv:2001.05936*, 2020.

[39] M. Rastegari, V. Ordonez, J. Redmon, and Ali, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[40] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "WRPN: Wide reduced-precision networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=B1ZvaaeAZ