A closer look at evaluating the Bit-Flip Attack against deep neural networks

Kevin Hector, Pierre-Alain Moellic, Mathieu Dumont, Jean-Max Dutertre

A Closer Look at Evaluating the Bit-Flip Attack Against Deep Neural Networks

Kevin Hector*†, Pierre-Alain Moëllic*†, Mathieu Dumont*†, Jean-Max Dutertre‡
*CEA Tech, Centre CMP, Equipe Commune CEA Tech - Mines Saint-Etienne, F-13541 Gardanne, France
†Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France
{*kevin.hector,pierre-alain.moellic, mathieu.dumont*}*@cea.fr*
‡Mines Saint-Etienne, CEA, Leti, Centre CMP, F-13541 Gardanne, France
dutertre@emse.fr

***Abstract*—Deep neural network models are massively deployed on a wide variety of hardware platforms. This results in the appearance of new attack vectors that significantly extend the standard attack surface, extensively studied by the adversarial machine learning community. One of the first attack that aims at drastically dropping the performance of a model by targeting its parameters stored in memory, is the *Bit-Flip Attack* (BFA). In this work, we point out several evaluation challenges related to the BFA. First, the lack of an *adversary's budget* in the standard threat model is problematic, especially when dealing with physical attacks. Moreover, since the BFA presents critical variability, we discuss the influence of some training parameters and the importance of the model architecture. This work is the first to present the impact of the BFA against fully-connected architectures that present different behaviors compared to convolutional neural networks. These results highlight the importance of defining robust and sound evaluation methodologies to properly evaluate the dangers of parameter-based attacks as well as measure the real level of robustness offered by a defense.**

***Index Terms*—Deep learning, Security, Fault Injection, Adversarial Attack, Robustness Evaluation**

I. Introduction

An important trend in deep neural networks is their deployment on hardware platforms. Such deployment raises major security concerns. It has increased attack surface, which was traditionally represented with algorithmic attack such as adversarial examples [2]. Some recent works have shown that model parameters (parameter-based attacks) or inference instructions [3] [4] are worrying attack vectors against the model integrity.

A milestone attack proposed by Rakin *et al*. [1], called Bit-Flip Attack (hereafter BFA), targets the parameters (also called *weights*) of a neural network stored in memory and bit-flips some bits of them in order to decrease model performance.

Following [1], several works proposed additional experiments and analysis [5] and defenses [6], [7]. Most of the works related to the BFA are based on simulations but refer to practical means to perform fault injection since there exists a rich state-of-the-art on this field, mainly in the context of cryptographic modules [8] and recently with a growing interest for embedded machine learning models [4], [9]. In [10], the BFA has been practically demonstrated using a single-sided RowHammer method against models stored in DRAM.

Even if parameter-based attacks are still in its infancy, a parallel has to be drawn with input-based attacks. An impressive number of adversarial example attacks and defenses have been proposed and a significant part raises very critical evaluation issues (pointed out in reference works [11]–[16]) that alter the confidence on the level of robustness a model can claimed. Being an essential concern, and regarding the rapid evolution of modern deep neural network models, parameters-based attacks also need further analysis and sound evaluation methodologies. In this context, our contributions are as follow:

- We question the relevance of the criteria used to measure the BFA since [1], that consists in reaching a random-guess level, because it misrepresents the evaluation of the attack, especially in the context of fault injection.
- Previous experiments show problematic high variance. We observe that the BFA can be dependent on training parameters, that should be taken into account for evaluation, and highlight the importance of the model architecture with the first results for fully-connected networks.
- Experimentally, for models that do not have the same properties than typical convolutional models, we show that the standard BFA can be significantly non-optimal compared to a very simple variation and therefore can lead to a false sense of robustness.

For reproducibility purpose, setups, codes of our experiments as well as complementary and detailed results are available on https://gitlab.emse.fr/securityml/closerlook-bfa.

II. Preliminaries and notations

A. Models and datasets

Following works addressing parameter-based attacks, we use CIFAR-10 composed of colored images (32x32) and MNIST composed of black and white digits (28x28).

As in [1] and [6], we apply the BFA on two popular convolutional neural network architectures (hereafter, **CNN**): **ResNet-20** [17] and **VGG-11** [18]. We add two custom models: (1) a fully-connected model (multi-layer perceptron, hereafter **MLP**) composed of four fully-connected layers with 512, 256, 128, 10 neurons and ReLU activation functions; (2)

978-1-6654-7355-2/22/$31.00 ©2022 IEEE

TABLE I: Mean (std dev) of bit-flips over 5 attacks to reach 11, 25, 50 and 75% of accuracy, for 5 models (training seeds).

Acc goal (%)	ResNet-20					VGG-11				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
11 [1]	20.6 (**5.08**)	18.8 (**8.28**)	21.4 (**4.49**)	27.4 (**11.22**)	7.0 (**2.09**)	72 (**36.01**)	55.4 (**20.92**)	81.6 (**25.5**)	42.6 (**9.02**)	113.2 (**51.92**)
25	8.8 (1.83)	8 (0.63)	9.6 (1.02)	12.4 (1.02)	3.6 (0.49)	14.2 (4.21)	13.8 (2.71)	16.0 (2.19)	13.2 (1.47)	19.6 (3.77)
50	6.2 (1.72)	4.4 (0.49)	5.4 (0.8)	6.6 (0.8)	2.4 (0.49)	6.8 (0.75)	7 (1.1)	8.4 (1.36)	6.6 (0.8)	9(1.1)
75	3.4 (0.8)	2.2 (0.4)	3 (0.63)	3.6 (0.49)	1.6 (0.49)	3 (0)	3.2 (0.4)	3.4 (0.49)	3.2 (0.4)	4 (0)

a variant of MLP with an additional convolutional layer as the first layer (32 filters of size 3x3) that we refer as **C-CNN**.

Similarly to previous works, our models are trained with 8 bits quantization aware-training since crushing a full-precision model is as easy as attacking the most significant bit of the exponent part of a single weight [1] (value explosion). The accuracy of the models are presented in our public repository as well as the detailed training parameters for each experiment.

III. Bit-Flips Attack

A. Original threat model

The BFA [1] identifies and flips the most sensitive bits of the parameters of a model M_W in order to drastically decrease its accuracy. From [1], the associated threat model, that we discuss in section IV is as follows:

Adversary's knowledge: The BFA is a white-box attack, the attacker needs a perfect knowledge of M_W to compute gradients of the loss according to the weights $\nabla_w \mathcal{L}$.

Adversary's goal: As presented in [1] and widely reused in other works, the goal is to decrease the accuracy of M below the random-guess level ($\approx 1/C$ where C is the number of label). Accuracy used in [1] is 0.11 for CIFAR-10.

Adversary's budget: Interestingly, the maximum number of bit-flips allowed is hardly ever mentioned, i.e. the adversary is able to perform as many faults as needed to reach the random-guess objective.

B. Attack principle

The BFA starts with a Progressive Bit Search method (PBS) [1] that identifies the most sensitive bits, followed by the flipping of the bits previously identified. The two methods are performed iteratively until reaching the adversary's goal.

The PBS alternates, for each iteration, an in-layer and cross-layer search. First, the in-layer search selects the best bit in the layer l by ranking the gradients of the bit b w.r.t. the loss: $\nabla_b \mathcal{L}$. After the most sensitive bit of each layer is found, each bit is flipped (and then restored) to measure the performance loss after this (and only this) bit-flip. After processing all the layers, the one with the maximum loss is selected and the corresponding bit-flip is – this time – permanently performed. The bit-flip is realized along the gradient ascendant w.r.t. the loss $\mathcal{L}$ as defined in [1] with Eq. 1:

$$m = b \oplus \big(sign(\nabla_b \mathcal{L})/2 + 0.5\big), \hat{b} = b \oplus m \qquad (1)$$

With $\hat{b}$, the bit after bit-flip. Interestingly, BFA follows the principle of most adversarial example crafting methods by relying on the direction that may increase (for untargeted attack) or decrease (for targeted attack) the loss. Thus, Equation 1 can be seen as a variant of the Fast Gradient Sign Method (FGSM) [19]. However, this gradient heuristic is not said to be the most efficient and, very recently, authors from [20] show that a Taylor's expansion-based heuristic ($|w.\nabla_w \mathcal{L}|$) is more efficient than the gradient ($|\nabla_w \mathcal{L}|$) for sparse networks (i.e., models compressed by a high pruning rate).

For all of our experiments, the attack dataset of the BFA is a random sampling of 256 images from the train set and each architecture is trained five times (different seeds for the weight initialization) and each trained model is attacked five times (i.e., we perform a total of 25 BFA for each architecture).

IV. Adversary's goal and budget

We derive the setup from [6] and adapt the learning rate scheduler (exponential scheduler) for better convergence and keep shorter training (40 epochs) to limit over-fitting issues.

A first evaluation outcome is that the assessment of BFA efficiency only on the total number of bit-flips needed to reach a random-guess level is not an appropriate criterion. Indeed, after a fast decrease of the accuracy, the remaining efforts to reach the random-guess objective gather most of the bit-flips with the highest variance. We highlight the variability of the attack against the training initialization by reporting in Table I the performance of five models. For each model, we present the average number and the standard deviation of the bit-flips necessary to decrease the accuracy below 11%, 25%, 50% and 75%. The average is computed over five attacks. When the adversary's goal is set to 25%, 50% and 75%, the standard deviation is limited, contrary to the random-guess objective (11%). This important variability of the attack makes its evaluation more complex.

From an adversary point of view, we can question the necessity to performed so many faults to simply go from 25% to 11% while the model is no longer *reliable*. The random-guess objective is a radical threat model in which an adversary targets the integrity as well as the availability of the model (i.e., the model is *useless*) without any assumption on his real capacity. Indeed, typical threat models (for example, as defined in the majority of adversarial examples works) also consider an *adversary's budget* that we claim to be an essential factor when dealing with fault injection attacks from RowHammer to laser beam injection.

Outcome: Practically, most of the fault injection attacks rely on a limited number of faults. Fixing an adversary budget or, at least, using several gradual objectives is compulsory to properly evaluate the BFA and reduce variance issues.

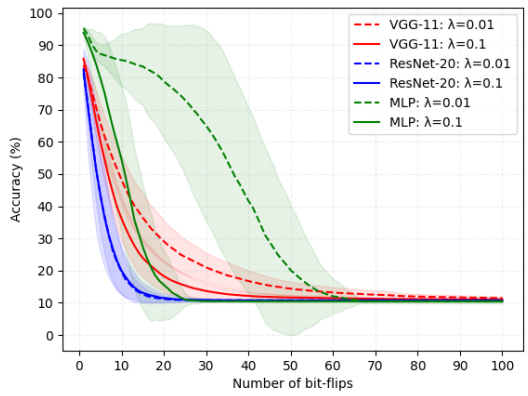


Fig. 1: BFA performance with different learning rates (VGG-11 (CIFAR-10), ResNet-20 (CIFAR-10), MLP (MNIST)).

V. Impact of the learning rate

Because of the very nature of BFA, the evaluation and analysis of the natural robustness of models with respect to their training is an essential step. If some training parameters influence the BFA, it becomes compulsory to take these factors into account when evaluating a defense, especially if the benefit offered is at the same level as the model's variability when trained with different parameters. Moreover, it may help to promote *good practice* to design and develop more secure models. Here, we focus on the impact of the learning rate.

A. Setups

Experiments are conducted with VGG-11, ResNet-20 on CIFAR-10 and MLP on MNIST with two learning rates, $\lambda = 0.1$ and 0.01. Except learning rate and epochs[1], all training parameters are the same as in Section IV.

B. Experiments

We evaluate the impact of the learning rate λ by using two initial values: 0.1 and 0.01. The scheduler is the same as before (exponential scheduler - 0.95 - and a weight decay of 3.10^{-4}). The weights are initialized using a normal distribution.

Fig. 1 shows very opposite impact of λ according to the architecture. We observe no influence of λ on ResNet-20 whatever the adversary's objective. For VGG-11, the lowest λ provides more robustness when the objective is to drop the accuracy below 40% (a difference of almost 20 bit-flips is needed for 20%). The most important influence is measured for MLP with a very significant difference of the number of bit-flips to reach the random-guess level (about 45 bit-flips).

We led further experiments for the MLP by analyzing the distribution of the bit-flips and the gradients across the layers. Table II shows how much the learning rate affects the bit-flips distribution. For $\lambda = 0.01$, all the bit-flips are focused on the last layer. The distribution is more balanced for $\lambda = 0.1$. These observations are confirmed by the gradients ($|\nabla_W \mathcal{L}|$) distribution for $\lambda = 0.1$, with the highest gradients spread on the second, third and last layers (Fig. 2).

[1]Setups are detailed in the public repository

TABLE II: Learning rate influence (λ): Bit-flips distribution and contribution per layer (MLP) for a random-guess goal.

Layer	$\lambda = 0.01$		$\lambda = 0.1$	
	bit-flips (%)	Damage (%)	bit-flips (%)	Damage (%)
Dense 1	0	0	0	0
Dense 2	0	0	**56**	**66.4**
Dense 3	0	0	20	12.53
Dense 4	**100**	**100**	24	21.07

Outcome: Depending on the architecture of the model, the training parameters may have a strong influence on the impact of the BFA and, therefore, should be carefully set and reported when evaluating the model's robustness. Analysis of the weights and the gradients distribution are efficient tools to better understand the model's behavior and explain potential variability of the attack.

VI. A Focus on Multilayer Perceptrons

MLP and CNN do not share the same behavior when facing BFA. We go deeper in the analysis by comparing both models and experimenting with a mixing architecture.

A. Gradients distribution

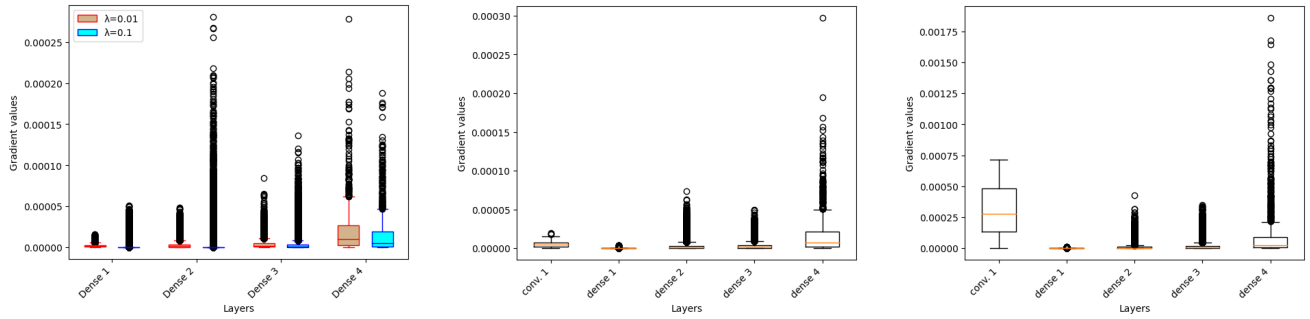
VGG-11 and ResNet-20 concentrate highest gradients in the first layers. Because parameters are shared in a CNN, as mentioned in [1] or [6], the error induced by bit flips performed on first convolution layers are accumulated and propagated throughout network (as for adversarial examples). This phenomenon explains the efficiency of BFA on such deep networks. As observed in Fig 2a, the gradients distribution is significantly different for MLP with most of highest gradients at the model's end.

Another important consequence is related to the gradient back-propagation: a change (consequently to a bit-flip) of the value of the parameter at a layer l directly alters the value of the gradients of the previous layers. This back-propagation phenomenon does not occur (or is limited) for bit-flips targeting the first layers which is not the case for the MLP since the highest gradients are located at the end of the network.

B. Attack limitation

The previous observations raise an open question about the way BFA selects the most appropriate bits. PBS works well when all the highest gradients are concentrated on the earliest convolutional layers. In other cases, the PBS is not able to evaluate if a combination of bit-flips (that may benefit from error propagation) is more efficient than a single bit-flip yet associated to the highest gradient at the end of the model.

To illustrate that potential limitation, we simply add a convolutional layer at the beginning of the MLP model. The resulting model, called C-CNN, has a gradient distribution that is very close to the original MLP (Fig. 2b and 2a respectively). Coherently, BFA has a relatively close performance against



(a) MLP ($\lambda = 0.01$ and $\lambda = 0.1$) (b) C-CNN ($\lambda = 0.01$) (c) C-CNN ($\lambda = 0.01$), after 10 bit-flips

Fig. 2: Gradients distribution for MLP (Left) with two learning rates and C-CNN before the first (middle) and after the 10^{th} (right) bit-flip. All the 10 bit-flips target the last fully-connected layer (*dense 4*).

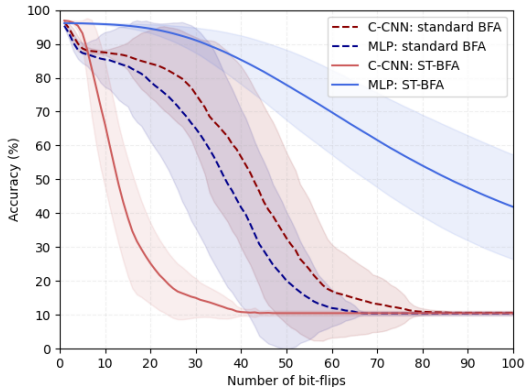


Fig. 3: BFA results (C-CNN and MLP)

both models (Fig. 3, dotted lines) because the bit-flips (after PBS) will be almost exclusively concentrate on the last layer.

Both models seem similar but what happen if we simply constraint the attack to target weights belonging to first layer? This attack, noted ST-BFA (Spatially-Targeted BFA), provides a surprising result (red and blue lines in Fig. 3). For C-CNN, the ST-BFA is far more efficient than the BFA: there is a difference of more than 50 bit-flips to reach a 20% goal in favor of the ST-BFA. Figure 2c shows the gradients distribution after 10 bit-flips (that target exclusively the last layer) and demonstrates how much the bit-flips alter the gradients of the previous layers: the gradients of the convolutional first layer significantly increase during the attack (see the gradient level of *conv1* in Fig. 2b and Fig. 2c).

Evaluating the C-CNN model with the standard BFA leads to a false robustness level since a stronger attack can be performed by only selecting one layer rather than the whole model. On the contrary, for MLP, the ST-BFA has an opposite effect: the difference of bit-flips to reach 50% of accuracy is closed to 40 bit-flips in favour of the standard BFA.

Outcome: The results obtained on standard CNN cannot be generalized to other architectures such as MLP because of strong differences on the way gradients are distributed throughout the model. The BFA relies on a complex mixing of forward error accumulation and backward propagation on the gradients. That results in situations where the BFA is significantly non-optimal compared to a localised application of the same attack. Therefore, to avoid evaluations that lead to a false sense of robustness, standard and localised-attacks should be carefully evaluated.

VII. Conclusion

An important need is robust evaluation methodologies to properly assess the real impact of parameter-based attacks and the level of robustness offered by defense schemes. We show that the standard threat model suffers from the lack of an adversary's bugdet which is an important factor in a fault injection context. Moreover, BFA also suffers from high variability when trying to reach a random-guess objective. This variability is also observed with training parameters as well as regarding the model's architecture. Therefore, the analysis of the weights and the gradients distribution appear as useful tools to better understand the mechanism of the BFA or to detect *special cases* that could lead to a false sense of security. Finally, thanks to first experiments on pure fully-connected networks – that present very different behaviors than classical CNN – we show that the standard BFA could be significantly sub-optimal, which highlights the need of careful, complete evaluations.

Acknowledgement

This work benefited from the French Jean Zay supercomputer with the AI dynamic access program. This collaborative research is supported by (CEA-Leti) the European project ECSEL InSecTT (www.insectt.eu, InSecTT: ECSEL Joint Undertaking JU under grant agreement No 876038) and by the French National Research Agency (ANR) in the framework of the *Investissements d'avenir* program (ANR-10-AIRT-05, irtnanoelec); and (Mines Saint-Etienne) by the French program ANR PICTURE (AAPG2020).

References

[1] 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)*, October 2019.

[2] 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.

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

[4] 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.

[5] Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. T-bfa: Targeted bit-flip adversarial weight attack. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[6] 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)*, June 2020.

[7] Sai Kiran Cherupally, Adnan Siraj Rakin, Shihui Yin, Mingoo Seok, Deliang Fan, and Jae-sun Seo. Leveraging noise and aggressive quantization of in-memory computing for robust dnn hardware against adversarial input and weight attacks. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 559–564, 2021.

[8] Michel Agoyan, Jean-Max Dutertre, Amir-Pasha Mirbaha, David Naccache, Anne-Lise Ribotta, and Assia Tria. How to flip a bit? In *2010 IEEE 16th International On-Line Testing Symposium*, pages 235–239, 2010.

[9] Mathieu Dumont, Pierre-Alain Moëllic, Raphael Viera, Jean-Max Dutertre, and Rémi Bernhard. An overview of laser injection against embedded neural network models. In *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, pages 616–621, 2021.

[10] 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.

[11] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[12] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[13] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017.

[14] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

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

[16] Roland Zimmermann, Wieland Brendel, Florian Tramer, and Nicholas Carlini. Increasing confidence in adversarial robustness evaluations.

[17] 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*.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[19] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[20] Kyungmi Lee and Anantha P Chandrakasan. Sparsebfa: Attacking sparse deep neural networks with the worst-case bit flips on coordinates. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4208–4212. IEEE, 2022.