

RESEARCH

Open Access



Corrosion Hammer: a self-activated bit-flip attack to the processing-in-memory accelerator

Zihao Yang^{1,2}, Mengxin Zheng³, Shengyu Fan^{1,2}, Qian Lou³, Rui Hou^{1,2}, Dan Meng^{1,2} and Mingzhe Zhang^{4*}

Abstract

The Resistive Random-Access-Memory (ReRAM) crossbar-based Processing-In-Memory (PIM) accelerator shows great promise in accelerating neural networks (NNs). This technique boasts low energy consumption and exceptional performance in multiplication and accumulations (MAC) operations, making ReRAM-based PIM accelerators an ideal solution for intelligent computing in wearable and low-power mobile devices. However, security concerns related to PIM have not been adequately addressed. In this paper, we present a new attack framework called SolutionName for ReRAM-based PIM accelerators. SolutionName builds upon the Bit Flip Attack (BFA), a weight modification attack that manipulates the NN function by flipping specific bits in the deployed quantized NN model. Unlike previous BFA methods that require explicit memory fault injection techniques, such as Row Hammer, to modify sensitive bits in the victim NN, SolutionName implants the trojan during the hardware-software co-design phase and flips sensitive bits using read disturbance. Read disturbance is a common noise in ReRAM caused by normal read operations. This approach enables the trojan to be activated quietly during normal use, eliminating the need for explicit attacks. Furthermore, we explore the impact of inputs on the activation time of the trojan and propose a method to expedite activation using normal input. Our experimental results demonstrate that SolutionName achieves an extremely covert trojan implantation and activation method, with an adversarial attack success rate of 94.38%. Additionally, with a specially designed method, the trojan activation can be accelerated on average by 61.98x, providing controllable activation.

Keywords PIM, Bit-flip attack, ReRAM, Row Hammer

Introduction

The utilization of neural network (NN) applications has been widespread across various fields. However, these applications still face several challenges, such as high computational requirements and excessive power consumption. To address these issues, the Processing-In-Memory (PIM) accelerator has gained significant

attention. The PIM accelerator is an emerging type of computational device that utilizes Non-Volatile Memory (NVM), such as ReRAM, STT-RAM, and Nor-Flash Memory. PIM offers notable efficiencies for contemporary applications, particularly in Neural Networks (NN), which often require extensive matrix computations during inference and training. PIM's advantage lies in its ability to perform these computations efficiently, avoiding additional overhead associated with data exchange (Imani et al. 2019).

However, the security implications of PIM have not been given enough consideration. PIM combines data storage and computation in the same device, making it more susceptible to side-channel attacks and potentially leading to model extraction (Lin et al. 2021). Furthermore, PIM accelerators are commonly employed in

*Correspondence:

Mingzhe Zhang
smartzmz@gmail.com; mingzhe-zhang@outlook.com

¹ Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS, Beijing, China

² School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

³ University of Central Florida, Orlando, FL, USA

⁴ Ant Group, Beijing, China

edge computing devices, which makes them susceptible to injection attacks due to their operation in unmonitored, open environments. Attackers can exploit physical methods to alter the data and thereby compromise its functionality. Research has shown that modifying certain weights in a neural network can lead to manipulation of its functionality (Chen et al. 2018; Wang et al. 2023). Although the above analysis primarily addresses the security concerns related to PIM accelerators, there is currently no established threat model for these accelerators that has been well investigated and validated.

In this paper, we introduce a new threat model for PIM accelerators, which is based on the widely investigated Bit-Flip Attack (BFA) (Rakin et al. 2019). BFA is an injection attack that analyzes the structure of a NN to identify sensitive bits and tampers with the NN behavior by flipping these bits. The typical BFA method relies on the Row Hammer issue in DRAM to accomplish bit flipping (Kim et al. 2020). However, this requires attackers to access the target device and execute explicit attack actions. In contrast, we observe that it is possible to identify the sensitive bits during the development process of PIM and then adopts the read disturbance in NVM devices to flip the sensitive bits. Since the PIM accelerator utilizes the read operation of NVM to perform multiplication calculations, this method allows the attack to be carried out discreetly without arousing suspicion during the normal functioning of the accelerator, which also enhances the crypticity of the attack.

Based on our proposed threat model, we present an attack framework, named Corrosion Hammer. Corrosion Hammer identifies the sensitive bits by analyzing the network structure and the triggers. Since most NVM-based PIM accelerators use MLC mode to increase storage density, Corrosion Hammer incorporates a dedicated algorithm to further refine and calibrate the identified sensitive bits based on the mapping method of network weight data in NVM units. This algorithm helps determine the specific sensitive bits that can be targeted for the PIM attack. Consequently, when the NN is deployed on the PIM accelerator, these sensitive bits are susceptible to flipping caused by Read Disturbance, effectively executing the attack automatically. Moreover, as the attack takes place during normal PIM operation, it becomes challenging to estimate the attack completion time accurately. To expedite the completion of PIM attacks, Corrosion Hammer also includes a dedicated algorithm that generates specific input data for PIM accelerators, resulting in increased read operations on the sensitive bits and thereby reducing the attack activation time.

Using the ReRAM-based PIM accelerator as a case study, we conducted experimental evaluations on Corrosion Hammer using various networks and datasets.

The experiment results shows that Corrosion Hammer is capable of effectively attacking NNs with different depths in PIM. Besides, we also demonstrate that the mapping method and input dataset have minimal impact on the success rate of the attack. For instance, when applied to the ResNet-18 network and MNIST dataset, Corrosion Hammer achieved an impressive maximum attack success rate of 94.38%. Moreover, the unique input data generated by Corrosion Hammer can significantly enhance the attack success speed, improving it by up to 61.98× compared to regular input data.

The contributions of this paper can be summarized as follow:

- We propose a threat model for PIM accelerators. This model enables the insertion of a Trojan during the generation of PIM accelerators and activates it automatically during their regular operation.
- We present a novel algorithm for identifying sensitive bits in various weight mapping models within PIM accelerators based on NVM cells.
- We propose a specialized algorithm that generates specific inputs for the PIM accelerator, which helps to reduce the time needed to carry out an attack.
- We experimentally prove the effectiveness of our proposed attack framework in targeting PIM accelerators. The results show that our framework can consistently and successfully attack these accelerators with a high rate of success. Importantly, we observed that the success rate of the attack remains largely unaffected by variations in the network, data, and weight mapping method.

Background and motivation

PIM accelerator

The ReRAM-based PIM is a widely used analog accelerator. In this architecture, data is stored as the resistive state within each ReRAM cell. As shown in Fig. 1, the current (I_n) flowing out of the bit line (BL), is the summation of currents passing through all the ReRAM cells connected to the bit line in the array. Each sub-current is determined by the product of voltage and conductance, resulting in the total current I being represented as the dot product of two vectors (V_1, V_2, V_3) and (W_1, W_2, W_3). In this way, the ReRAM array owns the capability to efficiently perform matrix–vector multiplications (MVMs) in the analog domain. Within the array, ReRAM cells connect each word line (WL) to the corresponding bit line. The weight matrix is initially mapped onto the ReRAM array, with the values of the weights directly corresponding to the conductance of the ReRAM cells. During MVM computation, the input

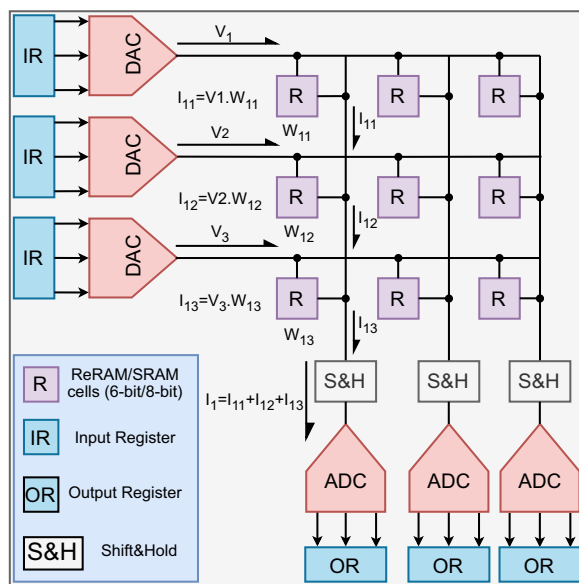


Fig. 1 Architecture organization of PIM accelerator

vector is applied to the word lines as voltages, and the partial sum results are generated in the bit lines based on Kirchhoff's circuit law (Gabelli et al. 2006). These partial sum results in each bit line are then converted into digital values by ADCs, with the final result being generated by the shifter and adder (S&A).

Bit-flip attack

Bit-Flip Attack is a hardware-based threat model wherein attackers manipulate the bit values within a system's DRAM from one position to another, that is, changing a bit from 0 to 1 or vice versa, this is named "**flipping**". As shown in Fig. 2, the BFA is notably executed through techniques such as the Row Hammer Attack (Kim et al. 2014). The Row Hammer Attack involves repeatedly reading data from a specific row within a memory module at a rate that causes electromagnetic interference in the adjacent rows. This interference can lead to bits in those neighboring rows inadvertently flipping, resulting in corrupted data. Deep Neural Networks (DNNs) are particularly vulnerable to bit-flip attacks. These networks rely heavily on the precise configuration of weights. A single-bit alteration can lead to significant performance degradation (Dong et al. 2023), incorrect decision-making, or the network behaving in unintended ways, posing a severe security risk, especially in sensitive applications (Rakin et al. 2019; Al Ghanim et al. 2023; Lou et al. 2023; Zheng et al. 2023). Modern error correction methods in primary memories, as discussed in Mutlu and Kim (2019), remain insufficient to counter Row Hammer attacks. Such attacks have been proven to effectively alter the weights of a neural network (Rakin et al. 2020; Zheng et al. 2023).

While the Row Hammer-style Bit-Flip attacks have evolved, spawning various offshoots that continue to threaten system security, they inherently necessitate that attackers directly interact with the DRAM to induce the

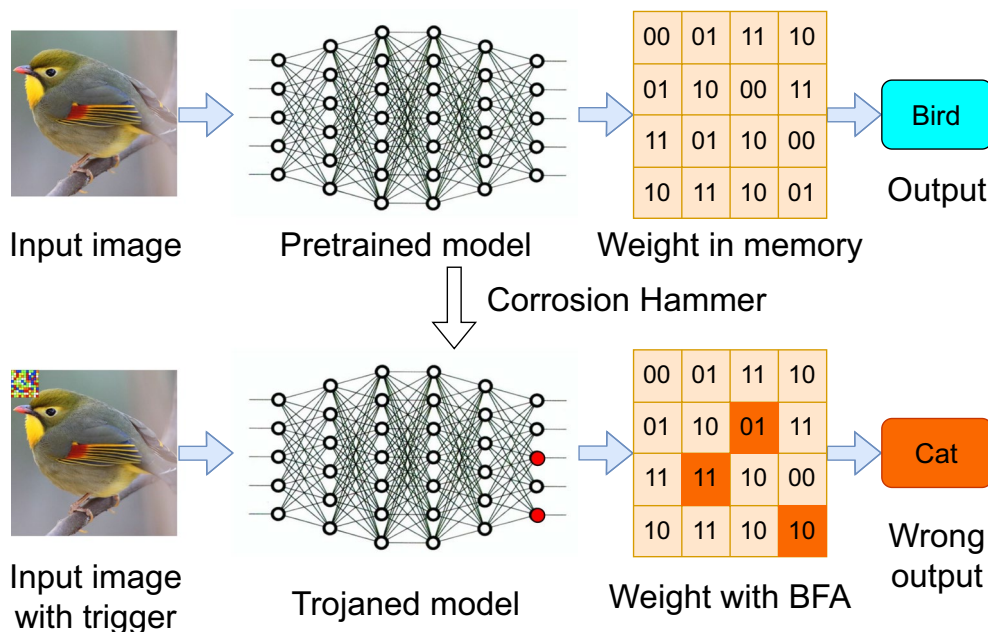


Fig. 2 Overview of Bit Flip Attack

requisite bit-flips (Razavi et al. 2016), representing a significant hurdle in the attack's execution. In contrast, our Corrosion Hammer integrates a trojan at the design stage during the hardware-software co-design process. It then capitalizes on the read disturbance—a common form of noise in ReRAM resulting from standard read operations—to flip sensitive bits. This approach allows the trojan to be surreptitiously activated through regular usage, without the need for a conspicuous attack.

Opportunity of constructing threat model

As introduced in Section 2.1, ReRAM-based PIM employs ReRAM cells for both data storage and in-situ computation. The multiplication computation carried out by these cells relies on Ohm's law, where a minute current is applied through the ReRAM cell, similar to the read operation when the cell serves as memory. However, in cases of dense computations, the current utilized for computation can induce a gradual alteration in the resistance value of the ReRAM cell, leading to an increase in calculation error. This alteration in resistance value, caused by a small current, is commonly referred to as *read disturbance*. The presence of *read disturbance* during the computation process also opens up opportunities for exploiting PIM through attacks. As introduced in Section 2.2, certain bits in the NN are sensitive, and altering these bits can impact the classification results produced by the NN. When the NN is operating in PIM mode, intensive computations can cause changes in the resistance of the cells that store these sensitive bits, leading to their alteration. Moreover, the speed at which success can be attained can be enhanced by generating precise inputs and consistently utilizing cells that store confidential information for calculations. This alteration occurs automatically during the computation process, allowing the attack to be carried out without explicit intervention. Based on the above principles, we propose an attack framework named Corrosion Hammer. In the following sections, we will provide a comprehensive introduction to the construction process of Corrosion Hammer.

Read disturbance of ReRAM

In this section, we briefly introduce the model used in this work for read disturbance estimation. For ease of understanding, the used notions are as shown in Table 1.

A ReRAM-based storage array is usually composed of 1T1R ReRAM storage cells. The typical structure of 1T1R ReRAM cell is as shown in the left portion of Fig. 3, which comprises two TiN electrodes (Top Electrode and Bottom Electrode) and one oxygen scavenging layer sandwiched between them (Raghavan et al. 2013). The oxygen scavenging layer is usually composed of a high-K material (i.e., HfO_2), which is found to be

Table 1 Symbols and notions used in this work

Symbol	Definition
N_{def}	Total defect density
t	The duration time at CVS stress
τ_T	The characteristic time for defect density generation
n	The number of voltage pulses
PW	The pulse width
$I_{disturb}$	The current disturbed by multiple read pulses
γ	The g-dependent local field enhancement factor
Q_{inj}	Total injection charge
I_{cell}	Input current
V_{cell}	Voltage drop on the ReRAM cell
R_{cell}	The resistance of the cell
α, β, A, E, m	Fitting constants

mainly amorphous after deposition (Govoreanu et al. 2011). During the read process, the biased bit line (V_{RBL}) enables the current flow through the cell, which is the same as the write process. Although the current used in the writing process is stronger, the state of the cell is still probably changed due to the accumulation of frequent reading current. This issue is named as the *read disturbance*.

Since the *read disturbance* has an impact on the stability of ReRAM-based memory, several models have been proposed to quantitatively describe this issue (Chen et al. 2014; Yang et al. 2020; Shim et al. 2020). In this work, we use the model proposed in Ref. (Yang et al. 2020) to estimate the read disturbance for the PIM accelerator. In ReRAM-based devices, the output current is used for resistance detection. Therefore, the *read disturbance* issue can also be described with the disturbed current. As shown in the right portion of Fig. 3, Eq. 1 defines the defect density created during *constant voltage stress* (CVS). The sequential read pulses (the voltage of each pulse is higher than CVS) can be considered as one discrete CVS stress in a row, which can also be expressed as Eq. 2. Therefore, the shift of the *high resistance state* (HRS) current after n pulses can be estimated as Eq. 3. With the consideration of fail criteria of HRS shift (δ), the defect generation density corresponds to the total injection charge (Q_{inj}) as shown in Eq. 4 and Eq. 5. The $\log(V_{cell})$ - $\log(\text{read counts})$ relation is derived as Eq. 6.

In this work, the read disturbance for the PIM accelerator is modeled with the above model and the parameters as shown in Table 2. Note that, the effectiveness of our proposed Corrosion Hammer does not depend on the specific model and parameters of the read disturbance model.

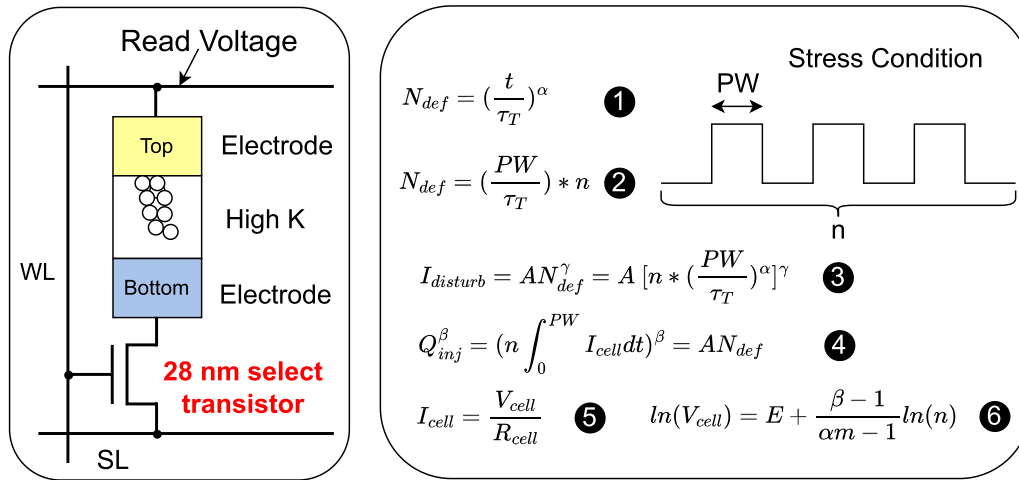


Fig. 3 The ReRAM cell structure and the quantitative model for read disturbance issue

Corrosion Hammer

Threat model

As shown in Fig. 4, the manufacturing process of commercial PIM accelerators typically involves several stages. Initially, hardware designers complete the hardware design of the PIM accelerator and proceed to chip manufacturing based on software requirements. Simultaneously, software designers train the NN and perform tasks like pruning and fine-tuning. Once the PIM chip is manufactured, the weights of the NN must be deployed to the ReRAM array of the PIM. However, due to variations in the manufacturing process Park et al. (2013), there is no guarantee that the deployed NN in the PIM will achieve the expected accuracy. Consequently, further optimization of the deployed NN is necessary to meet the design requirements. Currently, there is a lack of reliable

dedicated EDA tools, which means that the same company typically performs the collaborative design and optimization work between software and hardware. This allows for the construction of attacks specifically targeting PIM. In this paper, we assume that software designers can simultaneously locate sensitive bits and implant Trojans while training the network. Additionally, during the sensitive bit localization, further filtering can be performed based on the mapping method of the NN in the ReRAM cell. Moreover, designers can generate specialized input data based on the hardware parameters of the PIM and relevant information of the NN to reduce the time required for a successful attack. This enables the automatic triggering of a Trojan during the normal use of the PIM after it has been delivered. These assumptions align with the actual manufacturing process of PIM, making this type of attack both feasible and covert.

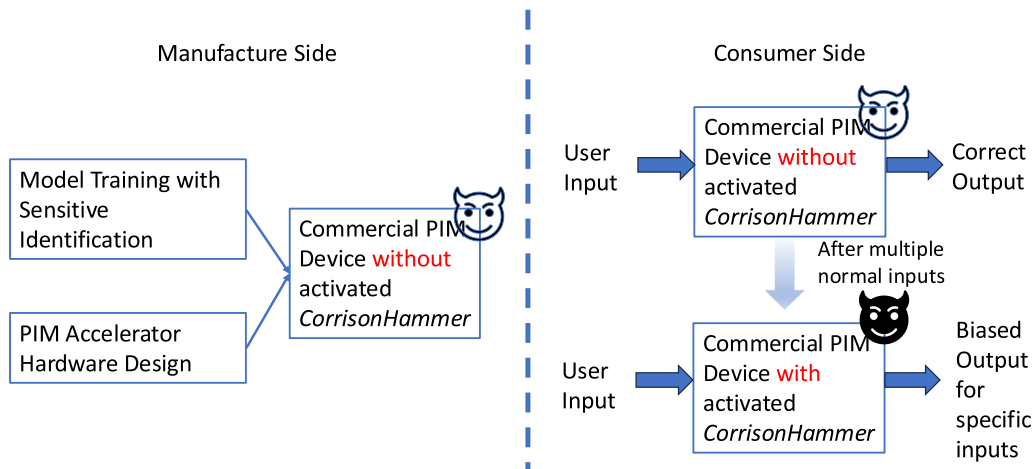


Fig. 4 The threat model of CorrisonHammer

Sensitive bit identification

The initial step in our process is to identify the sensitive bits (Rakin et al. 2020). Our objective is to modify a limited number of bits within the fully connected layer to increase the effectiveness and covert nature of our attack. Therefore, it is crucial to accurately identify the neurons that are closely related to the target class. Once these neurons have been pinpointed, we can calculate the necessary modifications needed to achieve our goal of misclassification. Additionally, it is important to develop a trigger that is not reliant on the dataset, ensuring that any image containing the trigger will be misclassified into the predetermined category.

Algorithm 1 Significant neuron identification

Input: pre-trained DNN model $\mathbf{A}$, data tensor $\mathbf{X}$, label tensor $\mathbf{Y}$, batch size B , test batch size B_t , learning rate r , target class t_c , number ω_b

Output: the index of the top ω_b neurons attached to target class.

- 1: Initialize the prediction network;
- 2: Input data tensors into the prediction network;
- 3: take all the weight connected to the K^{th} output neuron as G_K . (1);
- 4: using Neural Gradient Ranking (NGR) method, which could be expressed as (2);
- 5: returns the indexes j of ω_b number of gradients $g_{K,j}$ with highest absolute value;

As shown in Alg. 1, we consider a DNN model denoted as A designed for a classification task with M output categories or classes, where $K \in 1, 2, \dots, M$ represents the index corresponding to the target attack class. Assuming A 's final layer is a fully connected classifier with M output neurons and N input neurons, its weight matrix is denoted as $\hat{W} \in \mathbb{R}^{M \times N}$.

To compute the back-propagation gradients, we provide a set of sample data x and their corresponding labels t . The resulting accumulated gradients are then described as:

$$\hat{G} = \frac{\partial \mathcal{L}}{\partial \hat{W}} = \begin{matrix} OUT_1 \\ \vdots \\ OUT_K \\ \vdots \\ OUT_M \end{matrix} \begin{pmatrix} IN_1 & IN_2 & IN_3 & \dots & IN_N \\ g_{1,1} & g_{1,2} & g_{1,3} & \dots & g_{1,N} \\ \dots & \dots & \dots & \dots & \dots \\ g_{K,1} & g_{K,2} & g_{K,3} & \dots & g_{K,N} \\ \dots & \dots & \dots & \dots & \dots \\ g_{M,1} & g_{M,2} & g_{M,3} & \dots & g_{M,N} \end{pmatrix} \quad (1)$$

The loss function of model A is denoted by L . To target the specific misclassification category indexed by K , we identify the weights connected to the K -th output neuron as G_K . Then, we utilize the Neural Gradient Ranking

(NGR) method to ascertain the neuron exerting the most significant influence on the targeted K -th output neuron:

$$Top_{\omega_b} [|g_{K,1}, g_{K,2}, g_{K,3}, \dots, g_{K,N}|]; \quad \omega_b < N; \quad (2)$$

Let j be the indexes the above function returns for ω_b gradients $g_{K,j}$ with the highest absolute value. These indexes correspond to the weights connected to the output neuron of the last layer, the K -th neuron.

After the sensitive bits identification, the next step is to design a data-independent trigger. The trigger is an image with size $m \times m \times 3$, which is attached to specific locations on input images. For each image with the attached

trigger, our objective is for the neural network to recognize it as a predefined class.

Let us consider the output of the identified white-box neurons in the preceding step as $g(x; \hat{\theta})$, where $g(\cdot; \cdot)$ represents the inference function of model A , and $\hat{\theta}$ signifies the parameters of model A excluding the last layer (i.e., $\hat{\theta} \cap \hat{W} = \emptyset$). To facilitate trigger generation, an artificial target value $t_a = \beta \cdot I^{1 \times \omega_b}$ is formulated, with the constant β set to 100 in this study. Consequently, the trigger generation process can be expressed mathematically as Equation 3.

$$\min_{\hat{x}} |g(\hat{x}; \hat{\theta}) - t_a|^2 \quad (3)$$

The aforementioned minimization optimization is carried out through back-propagation, with $\hat{\theta}$ held constant. The trigger pattern $\hat{x} \in \mathbb{R}^{m \times m \times 3}$ is defined, and it is zero-padded to achieve the appropriate dimensions for input to model A . The $\hat{x}$ generated by the optimization process compels the neurons identified in the previous step to activate at a high magnitude (i.e., β). The specific algorithm process is detailed in Alg. 2.

Algorithm 2 Trigger generation

Input: pre-trained DNN model $\mathbf{A}$, test data tensor $\mathbf{X}_t$, test label tensor $\mathbf{Y}_t$, trigger location $start, end$, learning rate r

Output: trigger pattern $\hat{x}$.

- 1: Initialize trigger to zero padding and set the correct shape as the input of model $\mathbf{A}$;
 - 2: Set the loss function to the loss between the model's predicted triggered image result and the target class;
 - 3: Set the pixel of the trigger to modifiable.
 - 4: Adjust trigger with learning rate r . Find the optimal trigger according to the rules of (3);
 - 5: returns the final trigger and test it's performance;
-

Mapping restriction

When implementing NN in PIM, there are multiple mapping methods used to convert weight values to resistance values. The choice of mapping method can impact the identification of sensitive bits. Therefore, it is important to take the data mapping method into account when determining the sensitive bits. In this paper, we analyze two distinct mapping methods, referred to as *Single-bit method* and *2-bit Gray Code method*.

Single-bit restriction In the ReRAM-based PIM accelerators, Read disturbance has unidirectionality. The data stored in PIM is encoded based on the resistance value of each cell, and under continuous high voltage, the drift of cell resistance value is unidirectional. For single-bit data, we typically can change all data from 0 to 1 or from 1 to 0. This process can be described by the formula:

$$c_{i,j} = \min|f_{single-bit}(a_{i,j}; b_{i,j})|; \quad (4)$$

$$f_{single-bit} = \begin{cases} (a_{i,j})_m |_{0 \rightarrow 1}; & \text{Eq. (6) and (7) are satisfied} \\ (a_{i,j})_m; & \text{Otherwise} \end{cases} \quad (5)$$

$$(a_{i,j})_m = 0 \quad (6)$$

$$(b_{i,j})_m - (a_{i,j})_m |_{0 \rightarrow 1} < (b_{i,j})_m - (a_{i,j})_m; \quad (7)$$

In the above formula, $a_{i,j}$ and $b_{i,j}$ come from the original tensor and target tensor, respectively. f is a constraint function. And $c_{i,j}$ is the value which is closest to $b_{i,j}$ and derived from $a_{i,j}$.

2-bit gray code restriction In edge computing devices, data is commonly stored in 2-bit units. Under continuous high voltage, modifications are generally performed following the Gray code order. Therefore, while deploying the Trojan, we need to modify the quantized weights of the fully connected layer in the order of the 2-bit Gray Code to achieve the target weight values. If an exact match is not

achievable, we aim to minimize the discrepancy between the two sets of weights.

$$c_{i,j} = \min|f_{2-bits}(a_{i,j}; b_{i,j})|; \quad (8)$$

$$f_{2-bit} = \begin{cases} (a_{i,j})_m^* & (10) \text{ is satisfied} \\ (a_{i,j})_m & \text{otherwise} \end{cases} \quad (9)$$

$$(b_{i,j})_m - (a_{i,j})_m^* < (b_{i,j})_m - (a_{i,j})_m \quad (10)$$

In the above formula, $(a_{i,j})_m^*$ represents the possible modification value of the m -th bit of $a_{i,j}$ in the order of the Gray code. Our aim is to minimize $(b_{i,j})_m - (a_{i,j})_m^*$.

Specific input generation

Upon completion of the trojan generation, we proceed to extract the output from the layer directly preceding the fully connected layer for each image. This particular output provides us with valuable insights into the wear rate of the fully connected layer. By determining the image with the highest output value, we establish it as the reference image. The pixel values of this image are then designated as trainable parameters, which we subsequently modified to maximize the output of the preceding layer. This iterative process acts as the activation mechanism for the introduced trojan, with the ultimate goal of achieving optimal activation of the trojan.

Our main objective is to manipulate the weight data in the fully connected layer. This layer performs computations with the output data from the previous layer, and when the magnitude of the output data increases, the cells storing the input of the fully connected layer experience higher voltages. Thus, to accelerate read disturbance, it is crucial to maximize the magnitude of each output data from the preceding layer, which results in larger voltages in the storage cells of the fully connected layer.

In a given dataset, the sizes of the output results from the layer immediately preceding the fully connected layer vary as different images undergo computations within the neural network. In our model, data is stored in 8-bit units, so the output values range from 0 to 255. We establish a threshold, denoted as k , which considers the output data x as sufficiently large when x is greater than or equal to k . We then calculate the proportion of instances in which the output results from the layer preceding the fully connected layer satisfy the condition x greater than or equal to k for each image after the neural network computations.

Algorithm 3 Mask generation

Input: pre-trained DNN model $\mathbf{A}$, test data tensor $\mathbf{X}_t$, threshold k .

Output: solid color mask $\hat{x}_m$ which can accelerate the bit flipping.

- 1: Feed all images from the test set into the neural network, and capture the output from the layer immediately preceding the fully connected layer. Store output in list **intermediate_outputs**;
 - 2: For x in **intermediate_outputs** : count the number of x greater than k , denoted as $flag$, and store them in list $flags$;
 - 3: Create a blank image that matches the DNN input format. Set the pixels of the three channels of the image to be equal.
 - 4: Traverse the pixel of the image from 0 to 1 in steps of 0.01. Calculate the corresponding flag value for each iteration;
 - 5: Find the image $\hat{x}_m$ with the highest flag, and record the corresponding pixel.
-

In the context of a test set, we select the image with the highest proportion that meets the specified threshold as the baseline image. We make subsequent training adjustments based on this baseline image to ensure that the newly generated images during training result in larger input values into the fully connected layer. This strategic modification aims to expedite the occurrence of resistive state drifting. The whole algorithm for specific input generation is shown in Alg. 3.

Noise control

For the previous design, the impact of noise generated during the inference on the results is not considered. However, in practice, the read disturbance issue could happen in any ReRAM cell that fulfills enough reads

Table 2 Parameters for read disturbance estimation Yang et al. (2020)

Symbol	Value	Symbol	Value
t	$2\mu s$	τ_T	200ns
PW	20ns	γ	0.013
V_{cell}	0.144V	V_{RBL}	0.2V
α	0.09	β	1

and incurs noise to the effective process of Corrosion Hammer. Fortunately, the distribution of such noise can be predicted according to the distribution of frequently used weight values in fully connected layers. Based on this observation, we can incorporate noise correction into our method.

Taking the example of 8-bit quantized weights with 2-bit encoding, each weight is represented by four cells. As shown in Fig. 1, the voltage applied to the four cells representing the same weight is equal. Therefore, when stimulating a specific 2-bit cell, the remaining three cells in the same row will also be affected. Consequently, the noise will be prominently distributed among these three cells.

After understanding the principle of noise distribution, we can make appropriate adjustments to our attack method. Considering that when stimulating sensitive bits, bits in the same row will also be subjected to similar levels of stimulation, we incorporate this constraint when calculating the network weights after the attack. This process can be described as Alg. 4.

Algorithm 4 Noise correction

Input: pre-trained DNN model $\mathbf{A}$, The FC-layer weights $\mathbf{W}$ of model $\mathbf{A}$, sensitive bits $X = \{x_1, x_2, x_3, \dots, x_n\}$.

Output: trojaned FC-layer weight $\mathbf{W}_t$,

- 1: Set $\mathbf{W}$ to be trainable and only modify the sensitive bits;
- 2: Add modification constraints to each bit. Split the 8-bit data into four 2-bit cells, ensuring that each modification follows the Gray code direction for all four cells of the weight;
- 3: Set the loss function as the average of the prediction loss on normal images and the prediction loss on triggered images;
- 4: Feed the model with normal images and their correct prediction labels, as well as triggered images and their misleading prediction labels, for training;
- 5: Get the trojaned FC-layer weight $\mathbf{W}_t$.

Evaluation**Methodology**

To test our proposed Corrosion Hammer, we implement it on a system with NVIDIA A100 GPGPU and evaluate it with various datasets, which includes CIFAR-10 (Abouelnaga et al. 2016), MNIST (LeCun et al. 2010), Fashion-MNIST (Xiao et al. 2017) and SHVN (Netzer et al. 2011). We conduct the experiment based on ResNet-18 (He et al. 2016), ResNet-50 (Koonce and Koonce 2021) and MobileNetV2 (Sandler et al. 2018) respectively.

The target PIM architecture are as shown in Fig. 1, while the read disturbance of ReRAM is estimated with the model introduced in Section 3 with the parameters as shown in Table 2. For 28nm technique node, the fail criterion is set to be 3% of HRS current shift, and the read count for disturbance is 1×10^{18} . The effectiveness of our attacks is quantified using the Attack Success Rate (ASR). ASR measures the proportion of test samples that are accurately classified into a specific target class by the Trojaned DNN model. This accuracy is achieved by incorporating a designated trigger. Furthermore, we evaluate the acceleration ratios

Table 3 Average ASR (%) in different dataset and mapping-method

	CIFAR-10	MNIST	Fashion-MNIST	SVHN
ResNet-18				
Single-bit encode	93.269	94.375	88.32	91.68
2-bit Gray Code	89.91	92.46	81.56	92.04
ResNet-50				
Single-bit encode	91.485	93.285	89.72	93.15
2-bit Gray Code	87.06	85.46	84.36	88.36
MobileNetV2				
Single-bit encode	92.16	93.25	89.42	91.15
2-bit Gray Code	88.32	91.56	84.75	90.62

obtained by masks positioned at different locations and sizes within the image. The Acceleration Ratio, in this context, represents the ratio of the wearout rate of the circuit inferred from the image to the average wearout rate of all images in the test set. In our evaluations, we selected five distinct regions and three different size ratios.

Results

Success rate Table 3 presents the ASR within different datasets and encoding methods. When considering the ResNet-18, for *single-bit encoding* method, the ASR varies from 88.32% to 94.375%, which reaches the highest ASR within the MNIST dataset. On the other hand, for the *2-bit Gray Code* method, the ASR ranges from 81.56% to 92.46%.

Similarly, the ResNet-50 exhibits the highest average ASR when considering the MNIST dataset, while showcasing the lowest average ASR for the Fashion MNIST dataset. However, it is worth noting that under the 2-bit Gray Code encoding method, the average ASRs for all datasets are lower compared to the single-bit encoding method. Furthermore, the mean ASR for CIFAR-10 and MNIST is lower for ResNet-50 in comparison to the results for ResNet-18. On the other hand, the ASR for Fashion-MNIST and SVHN is higher within the

Table 4 Acceleration ratio of solid color masks of different sizes placed in different positions

Size	Position 0	Position 1	Position 2	Position 3	Position 4
100%	61.983	61.983	61.983	61.983	61.983
75%	9.380	6.070	11.771	8.644	9.932
50%	7.909	4.598	8.828	8.461	6.989
25%	6.437	2.943	6.070	5.518	3.862
0	12.875	12.875	12.875	12.875	12.875

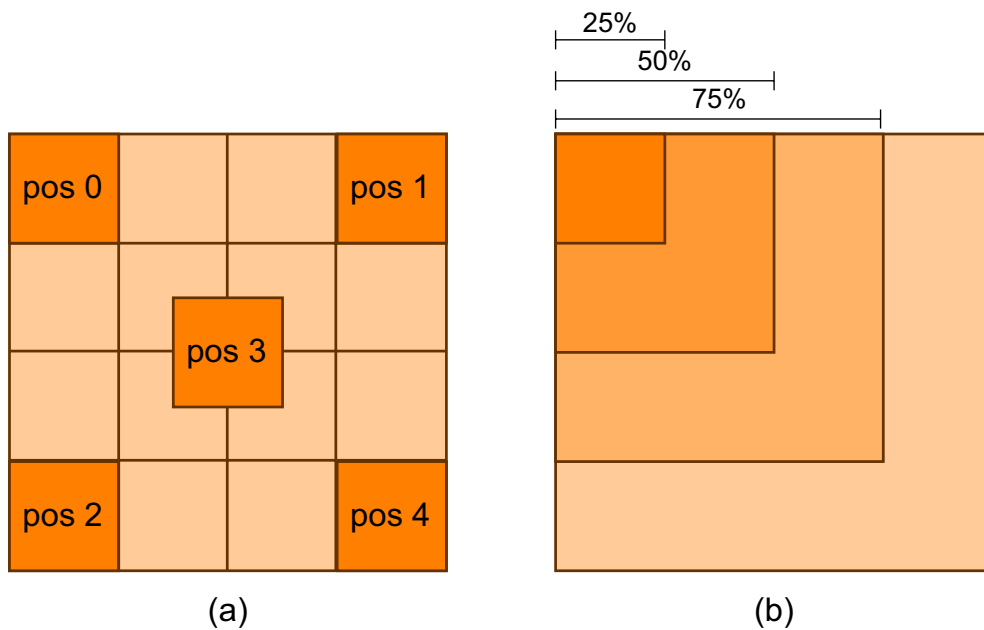


Fig. 5 The schematic diagram of the position and size of the mask

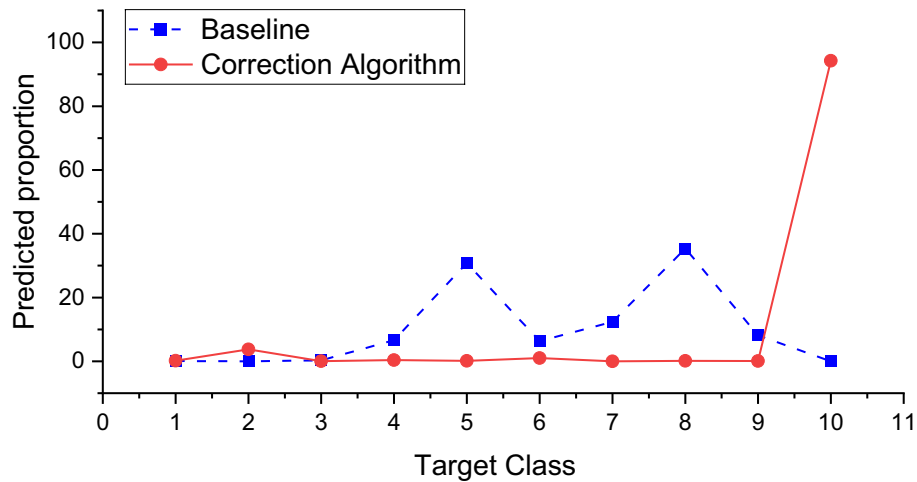


Fig. 6 The effectiveness comparison of Corrosion Hammer w/w.o. bit-flipping direction restriction(target class is *class 10*). The encoding method is 2-bit Gray code. The dataset used for training and testing is CIFAR-10

single-bit encoding method, but lower within the 2-bit gray code encoding method.

Attack acceleration We experimented to investigate the impact of various positions and sizes of the mask on the acceleration ratio. The findings from this experiment are presented in Table 4, while Fig. 5 illustrates a schematic diagram depicting the different positions and sizes of the mask.

We have chosen five distinct locations, as illustrated in Fig. 5a. In the case of various mask sizes, position 0 corresponds to aligning the upper-left vertex of the

mask with the upper-left vertex of the entire image. Likewise, position 1, position 2, position 3, and position 4 correspond to aligning the top right vertex, bottom left vertex, center point, and bottom right vertex, respectively.

We conducted measurements on the acceleration ratios of five mask sizes, categorized based on the ratio of the mask's edge length to the entire image's edge length, as depicted in Fig. 5b. The results for the 100% and 0% masks remain consistent across different positions. However, when the mask size is non-zero, there

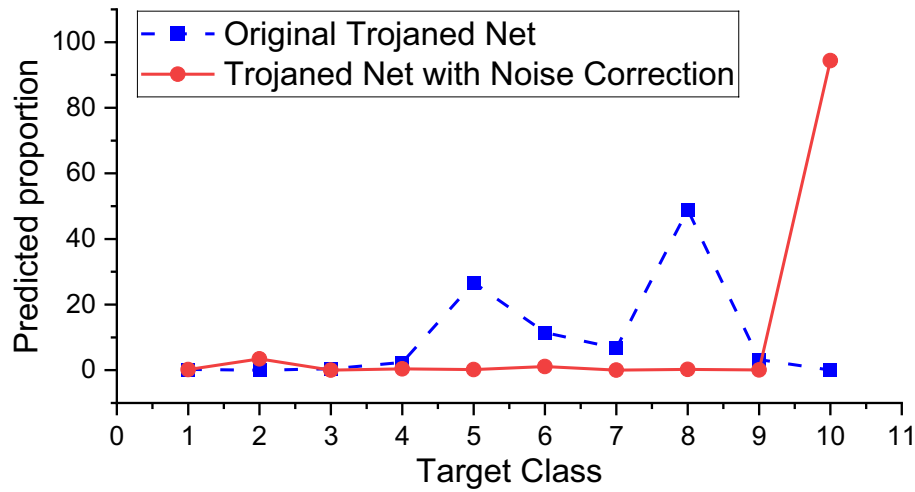


Fig. 7 Effectiveness comparison for the Corrosion Hammer w/w.o. noise correction mechanism. Encoding method is 2-bit Gray code. The dataset for training and testing is CIFAR-10. Target class is 10 and noise rate=5%

is a direct correlation between size and acceleration ratio. Larger mask sizes yield higher acceleration ratios. Furthermore, the acceleration ratio varies at different positions for the same mask size. Specifically, position 1 exhibits the lowest acceleration ratio among all sizes, while position 2 consistently demonstrates the highest acceleration ratio, except for the 25% mask size. Lastly, the acceleration ratio for the 0% mask size falls between the results for the 75% and 100% mask sizes.

Baseline comparison for bit-flip direction restriction Compared to prior BFA attacks, Corrosion Hammer restricts the bit-flipping direction according to the actual encoding scheme. To evaluate the effectiveness of this approach, we conducted baseline without the consideration for the direction of bit flips. During the inferences, the baseline design directly ignores the bit flips in the opposite directions. The comparison of the effectiveness between our method and the baseline is as shown in Fig. 6. According to Fig. 6, the noise significantly affects the effectiveness of Corrosion Hammer, which fails to guide the input to the target classification, and the result distribution has no obvious regularity. On the other hand, 94.24% of the inference results within Corrosion Hammer are the biased to the target classification. Note that, the evaluations for other target classification also show similar results.

Evaluation for noise correction To evaluate the noise correction method in Corrosion Hammer, we conduct two sets of comparative experiments and report the results in Fig. 7. The first set of experiments did not consider the impact of noise and used the attack algorithm under ideal conditions, assuming that we could accurately target the sensitive bits. On contrary,

the second set of experiments accounted for the impact of noise and applied noise correction according to the method described in Algorithm 4. Theoretically, due to the presence of inherent noise, the first attack method cannot achieve the desired attack effect. This is also evident from the experimental results in Fig. 7, where the results of the ideal attack method are relatively scattered. In contrast, after incorporating noise correction, the experimental results aligned with our expectations, with the majority 94.38% of the results pointing to our designated target class.

Discussion

Generality of Corrosion Hammer The Corrosion Hammer is constructed based on the *Bit-Flip Attack* of the NNs and the *Read Disturbance* issue of the non-volatile memory devices. Therefore, the proposed Corrosion Hammer contains the general capability in the following perspectives: (1) *Different neural networks* the *Bit-Flip Attack* focuses on tampering the function of NNs by flipping the sensitive bits in the FC-layer. As the FC-layer is contained in various NNs, it is possible to attack any NN deployed on the PIM accelerator which includes the FC-layer. Our experiment result shows that, the structure and depth of NN has minor impact on the success rate of the Corrosion Hammer attack. (2) *Different PIMs based on various devices* the *Read Disturbance* issue is widely existed in various memory devices, such as ReRAM (Yang et al. 2020), PCM (Li et al. 2016), STT-RAM (Pan et al. 2018) and Nor-Flash (Zhang et al. 2021). Therefore, the PIM accelerators based on these devices are probably affected by the attack method similar to the Corrosion Hammer. Our proposed Corrosion Hammer framework

contains dedicated algorithm to identify the sensitive bits with the consideration of different device characteristics and encode methods, which allows to migrate the attack framework to the PIMs based on various memory devices.

Dependence on encoding rather than layout The Corrosion Hammer methodology primarily relies on the *weight encoding and cell-mapping format* (e.g., single-bit or 2-bit Gray code), rather than on proprietary layout or routing details. In other words, the attack algorithms operate at the logical quantization-and-mapping level: they identify sensitive logical bits in quantized weights and exploit read-disturb phenomena of the corresponding memory cells. Consequently, while tighter organizational separation of hardware and software toolflows (e.g., strict EDA partitioning or per-chip randomized physical mapping) can raise the bar for practical exploitation, such measures do not fully eliminate the threat unless the encoding/mapping rules themselves are concealed or randomized per device.

Potential defence strategies For the defense of Corrosion Hammer, exploration can be conducted in the following aspects: (1) *Micro-architecture approach* some micro-architectural techniques may help to protect the NNs on the PIM accelerators, such as the schemes within selective refresh the involved ReRAM cells (Shin et al. 2022) or optimize the cell structure to minimize the Read Disturbance (Shim et al. 2020). However, such modifications also increase the access latency and decrease the performance of PIM accelerators. (2) *Software approach* several software-based approaches have been proposed, which have the potential to defense the BFA. Since Corrosion Hammer is based on BFA, such works can also be used to defense the Corrosion Hammer. (3) *System approach* As we discussed in Section 2.3, the root cause of the Corrosion Hammer is the high coupling of the software and hardware in PIM architecture. Therefore, at the system level, the key of defending Corrosion Hammer lies in the standardization of PIM system development, which allows the PIM software and hardware to be implemented by different companies, thus avoiding the attackers from implementing Corrosion Hammer through software-hardware co-design. Such decoupling development method requires: (1) a standard library with sufficient Application Portable Interfaces (APIs) supported by the PIM hardware, (2) a third-party compiler for the PIM software developer, and (3) The download and debug tools for the PIM software developer. Although such decoupling development method probably have negative impact on the performance and the energy efficiency of PIM, they may raise the difficulty of implementing similar attack as the Corrosion Hammer.

Beyond high-level design concepts, several measurable countermeasures can mitigate Corrosion Hammer-like behaviors with bounded cost. First, *selective refresh and row remapping* mechanisms inspired by CROW (Hassan et al. 2019) can periodically reprogram 0.1% of frequently accessed ReRAM cells every 10^6 reads, reducing disturbance-induced errors by over 90% with 1–2% performance and energy overhead. Second, *in-situ read-disturb detection* has been experimentally validated in recent CIM prototypes (Yaldagard et al. 2023), which detect over 95% of disturbance events with less than 1% false positives, incurring only 0.2% area, 0.5–0.8% dynamic power, and 0.4% latency overhead. Third, *algorithmic retraining with hardware-noise modeling* (Cherupally et al. 2022) improves network robustness against injected or hardware-induced bit errors by 9–18% in accuracy with less than 2% energy overhead. Together, these circuit- and algorithm-level techniques provide concrete, quantifiable mitigation paths that limit attack success while maintaining overall system efficiency.

Specific defense case In this section, we briefly introduce a selective refresh-based defense case for Corrosion Hammer. During PIM operation, the PIM controller randomly selects N rows within the PIM at fixed time intervals (several seconds to tens of seconds) and performs refresh operations using weak current. The weak-current-based erase-write mechanism can effectively mitigate the negative impact on PIM unit lifespan, but introduces increased write latency (Li et al. 2013). However, since PIM chips do not operate continuously, performing refresh operations during chip idle periods does not affect inference latency. After completing the refresh operation, the system marks the refreshed rows and prioritizes unrefreshed rows in subsequent refresh cycles. When all rows have been refreshed, the system resets the refresh flags to initiate a new round of selective refresh cycles. Through this mechanism, the probability of successful Corrosion Hammer attacks can be significantly reduced. It should be noted that this defense method introduces additional refresh operations, thereby increasing energy consumption, which may adversely affect the operational duration of battery-powered PIM chips.

Extension to non-vision models Although our evaluation primarily focuses on vision networks such as CNNs, the proposed Corrosion Hammer framework is not limited to visual domains. Non-vision models—including keyword spotting networks in speech recognition, multilayer perceptrons (MLPs) in small large-language-model (LLM) heads, and other modality-specific accelerators—share similar fully connected structures and weight quantization schemes that are equally vulnerable

to bit-flip perturbations. Recent studies have shown that bit-flip attacks can degrade the performance of language and audio models to near-random accuracy with only a few critical weight flips (e.g., SBFA(Chen et al. 2024) and DeepHammer (Yao et al. 2020) variants). Furthermore, keyword-spotting networks often deploy compact, quantized inference models on edge PIM accelerators, where the read-disturbance effect can occur more frequently due to continuous low-voltage access patterns. These properties make such models potential victims of self-activated bit-flip attacks similar to Corrosion Hammer.

Impact on energy and latency Although Corrosion Hammer introduces additional read operations through its accelerated input mechanism, our analysis indicates that the energy and latency overheads are negligible compared with normal PIM inference workloads. The activation acceleration technique intentionally increases the access frequency to the cells storing sensitive bits, which leads to slightly higher dynamic current along the bitlines and ADC conversion paths. As only less than 0.1% of memory cells are repeatedly accessed during the accelerated activation phase — a ratio consistent with prior bit-flip attack studies (Rakin et al. 2019, 2020; Shin et al. 2022) — the overall dynamic power increase is marginal (1–3%), primarily due to localized read-current amplification along a few bitlines. This estimation aligns with device-level measurements of ReRAM read-disturbance energy fluctuation reported in Yang et al. (2020); Shin et al. (2022). Similarly, the latency impact is minimal: the additional read pulses only extend the total inference time by less than 0.5%, which is below the observable threshold for most edge-level accelerators. Therefore, the Corrosion Hammer maintains stealth not only in its logical activation pattern but also in its power-performance footprint, ensuring that no abnormal timing or energy signature is generated during the attack.

Future work In future work, we plan to extend the Corrosion Hammer framework from simulation to real hardware validation on actual Processing-In-Memory (PIM) chips fabricated with different non-volatile memory (NVM) technologies. Specifically, we will conduct experiments on commercially available or prototype PIM platforms based on multiple physical media, including Resistive RAM (ReRAM), Phase-Change Memory (PCM), Spin-Transfer Torque RAM (STT-RAM), and NOR-Flash. These heterogeneous devices exhibit distinct read-disturbance behaviors, resistance drift mechanisms, and endurance limits, which can significantly influence both the activation latency and stability of the self-triggered bit-flip attack. By characterizing the real disturbance effects and comparing the device-level variability across materials, we aim to validate the feasibility and generality of Corrosion

Hammer under practical circuit and environmental conditions. This hardware-based evaluation will also provide critical data for developing more effective defense mechanisms and robustness-enhanced PIM architectures. In addition, we will validate these estimations through SPICE-level circuit simulations and on-chip power measurements to quantify the exact energy-latency trade-offs under different NVM technologies.

Furthermore, we also plan to experimentally extend our framework to these non-vision domains, evaluating the susceptibility of small LLM heads and audio inference tasks under different NVM technologies.

Related work

Read disturbance Read disturbance is a prevalent stability issue in various memory devices (i.e., DRAM (Kim et al. 2014), ReRAM (Shin et al. 2022), Flash (Cai et al. 2015), etc) based on advanced process technologies (i.e., 28nm and beyond). In specific, the read disturbance in DRAM manifests as frequent accessing to a row in the memory can affect the integrity of data in physically adjacent but unaccessed memory rows. RowHammer and RowPress stand as two typical instances of the read disturbance phenomenon for DRAM, wherein a DRAM row (known as the victim row) may suffer from bitflips. This occurs when an adjacent DRAM row, referred to as the aggressor row, is either 1) subjected to repeated activation (commonly termed as Rowhammer)(Kim et al. 2014) or 2) maintained in an open state for an extended duration (referred to as pressing) (Luo et al. 2023). The aforementioned attack methods are commonly employed to tamper with critical data in memory, thereby disrupting the normal operation of programs (Seaborn and Dullien 2015; Bhattacharya and Mukhopadhyay 2018), or to trigger other more deeply concealed attacks (i.e., Bitflip Attack (Rakin et al. 2019, 2020)). Although a series of prior works propose various software-based (Zhang et al. 2022; Bock et al. 2019) and hardware-based (Hassan et al. 2019; Loughlin et al. 2022) techniques to prevent the read disturbance issue, it remains feasible to pinpoint the rows that meet the desired vulnerability criteria (e.g., through templating) in order to enhance the possibility of the attack's success (Rakin et al. 2022; Cai et al. 2022).

Unlike DRAM, the read disturbance issue in NVM (i.e., ReRAM, STT-RAM, Flash) manifests as the frequent reading of data from a non-volatile memory cell potentially leading to the alteration of that data. This phenomenon arises because NVM cells typically store data through resistance values, utilizing weak currents to detect the resistance state of the cell and strong currents to modify it. Excessive read operations may inadvertently change the structural state of the memory cell

(Chen et al. 2014; Yang et al. 2020; Shim et al. 2020). In this paper, we take a ReRAM-based PIM accelerator as an example to explore how to leverage read disturbance to activate a pre-embedded bit-flip attack trojan during normal operations. Similar attacks can also be implemented in PIM accelerators based on other NVM materials.

Bit-Flip attack With the widespread application of various DNNs, an increasing number of bit-flip attack methods have been proposed. Bit-flip attack refers to the manipulation of a model's functionality by locating and flipping sensitive bits within the DNNs, which can be triggered with various methods (i.e., Row-hammer (Kim et al. 2014), clock glitching (Liu et al. 2020), Voltage Frequency Scaling (Gnad et al. 2017), etc). In general, the bit-flip attacks can be classified as the untargeted attack and the targeted attack. The untargeted attack implies that, following the flipping of bits within certain weight parameters, the DNN's predictions for all inputs will become skewed, potentially degrading to the level of random guessing in an ideal scenario (Rakin et al. 2019). The goal of untargeted attacks include misclassification rate increment (Liu et al. 2020), accuracy degradation (Jap et al. 2021), random accuracy (Yao et al. 2020), etc. In contrast to untargeted attacks, targeted attacks enable the attacker to fulfill more precise control over the attack. For non-targeted objects, the accuracy can remain unaffected, thereby rendering this bit-flip attack more covert. The common goal of targeted attack is increasing misclassification on targeted data, while rest accuracy unaffected (Liu et al. 2017). Moreover, the bit-flip attack can also be used as the trigger of other trojan of DNNs (Cai et al. 2022).

All of the aforementioned bit-flip attack methods require explicit modification of sensitive bits. In this paper, leveraging the read disturbance phenomenon in NVM devices, we propose a method to activate bit-flip attacks in PIM accelerators through normal read accesses. We also tailor the attack model by considering the data arrangement and bit-flip characteristics specific to PIM. Additionally, we introduce a technique to expedite the activation of bit-flip attacks using normal input data.

Conclusion

In this paper, we propose a novel attack framework for PIM, named Corrosion Hammer, which supports self-activation and can activate trojan without manually modifying the hardware and software. As the attack is fulfilled, the input image with the trigger will be misclassified into the specific class, without affecting the classification of the normal image. Furthermore, we also propose specific algorithm to generate special input

data for attack acceleration. The experiment results prove that the attack success rate can be up to 94.38%, while the generated input can accelerate the attack by 61.98 $\times$. To our best of knowledge, this is the first threat model for PIM accelerator. In our future work, we will further improve Corrosion Hammer and explore the potential defense techniques.

Acknowledgements

The authors would like to thank all contributors who assisted in technical discussions, experimental support, and language improvements.

Author contributions

Mingzhe Zhang conceived the original idea and supervised the study. Zihao Yang developed the methodology, performed the experiments, conducted the data analysis and wrote the initial manuscript draft. Mengxin Zheng provided guidance on the experimental code and made necessary modifications. Shengyu Fan performed language polishing and textual refinements of the manuscript. All authors reviewed and approved the final manuscript.

Funding

Not applicable.

Data availability

The datasets generated and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Received: 20 August 2025 Accepted: 3 November 2025
Published online: 23 December 2025

References

- Abouelnaga Y, Ali OS, Rady H, Moustafa M (2016) Cifar-10: Knn-based ensemble of classifiers. In: 2016 international conference on computational science and computational intelligence (CSCI), IEEE, pp 1192–1195
- Al Ghanim M, Santriari M, Lou Q, Solihin Y (2023) Trojbits: a hardware aware inference-time attack on transformer-based language models. In: ECAI 2023, IOS Press, pp 60–68
- Bhattacharya S, Mukhopadhyay D (2018) Advanced fault attacks in software: exploiting the rowhammer bug. In: Fault tolerant architectures for cryptography and hardware security, Springer, pp 111–135
- Bock C, Brasser F, Gens D, Liebchen C, Sadeghi A-R (2019) Rip-rh: preventing rowhammer-based inter-process attacks. In: Proceedings of the 2019 ACM Asia conference on computer and communications security, pp 561–572
- Cai Y, Luo Y, Ghose S, Mutlu O (2015) Read disturb errors in mlc nand flash memory: characterization, mitigation, and recovery. In: 2015 45th Annual IEEE/IFIP international conference on dependable systems and networks, IEEE, pp 438–449
- Cai K, Zhang Z, Yao F (2022) On the feasibility of training-time trojan attacks through hardware-based faults in memory. In: 2022 IEEE international symposium on hardware oriented security and trust (HOST), IEEE, pp 133–136
- Chen P-Y, Sharma Y, Zhang H, Yi J, Hsieh C-J (2018) Ead: elastic-net attacks to deep neural networks via adversarial examples. In: Proceedings of the AAAI conference on artificial intelligence, vol 32
- Chen C-Y, Shih H-C, Wu C-W, Lin C-H, Chiu P-F, Sheu S-S, Chen FT (2014) Ram defect modeling and failure analysis based on march test and a novel squeeze-search scheme. *IEEE Trans Comput* 64(1):180–190

- Chen Y, Xu J, Li S, Fan D (2024) Sbfa: single sneaky bit-flip attack against large language models. *arXiv preprint arXiv:2409.21843*
- Cherupally SK, Meng J, Rakin AS, Yin S, Yeo I, Yu S, Fan D, Seo J-S (2022) Improving the accuracy and robustness of rram-based in-memory computing against rram hardware noise and adversarial attacks. *Semicond Sci Technol* 37(3):034001
- Dong J, Qiu H, Li Y, Zhang T, Li Y, Lai Z, Zhang C, Xia S-T (2023) One-bit flip is all you need: when bit-flip attack meets model training. In: *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, pp 4688–4698
- Gabelli J, Fève G, Berroir J-M, Plaçais B, Cavanna A, Etienne B, Jin Y, Glattli D (2006) Violation of kirchhoff's laws for a coherent rc circuit. *Science* 313(5786):499–502
- Gnad DR, Oboril F, Tahoori MB (2017) Voltage drop-based fault attacks on fpgas using valid bitstreams. In: *2017 27th international conference on field programmable logic and applications (FPL)*, IEEE, pp 1–7
- Govoreanu B, Kar GS, Chen Y, Paraschiv V, Kubicek S, Fantini A, Radu I, Goux L, Clima S, Degraeve R (2011) 10x 10nm 2 hf/hfo x crossbar resistive ram with excellent performance, reliability and low-energy operation. In: *2011 international electron devices meeting*, IEEE, pp 31–36
- Hassan H, Patel M, Kim JS, Yaglikci AG, Vijaykumar N, Ghiasi NM, Ghose S, Mutlu O (2019) Crow: a low-cost substrate for improving dram performance, energy efficiency, and reliability. In: *Proceedings of the 46th international symposium on computer architecture*, pp 129–142
- Hassan H, Patel M, Kim JS, Yaglikci AG, Vijaykumar N, Ghiasi NM, Ghose S, Mutlu O (2019) Crow: a low-cost substrate for improving dram performance, energy efficiency, and reliability. In: *2019 ACM/IEEE 46th annual international symposium on computer architecture (ISCA)*, pp 129–142
- He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 770–778
- Imani M, Gupta S, Kim Y, Rosing T (2019) Floatpim: In-memory acceleration of deep neural network training with high precision. In: *Proceedings of the 46th international symposium on computer architecture*, pp 802–815
- Jap D, Won Y-S, Bhasin S (2021) Fault injection attacks on softmax function in deep neural networks. In: *Proceedings of the 18th ACM international conference on computing frontiers*, pp 238–240
- Kim JS, Patel M, Yağlıkçı AG, Hassan H, Azizi R, Orosa L, Mutlu O (2020) Revisiting RowHammer: an experimental analysis of modern dram devices and mitigation techniques. In: *2020 ACM/IEEE 47th annual international symposium on computer architecture*, IEEE, pp 638–651
- Kim Y, Daly R, Kim J, Fallin C, Lee JH, Lee D, Wilkerson C, Lai K, Mutlu O (2014) Flipping bits in memory without accessing them: an experimental study of dram disturbance errors. *ACM SIGARCH Comput Archit News* 42(3):361–372
- Koonce B, Koonce B (2021) Resnet 50. *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*, pp 63–72
- LeCun Y, Cortes C, Burges C (2010) Mnist handwritten digit database. *ATT Labs* [Online]. Available: <http://yann.lecun.com/exdb/mnist>
- Li Q, Jiang L, Zhang Y, He Y, Xue CJ (2013) Compiler directed write-mode selection for high performance low power volatile pcm. In: *Proceedings of the 14th ACM SIGPLAN/SIGBED conference on languages, compilers and tools for embedded systems*, pp 101–110
- Lin N, Chen X, Xia C, Ye J, Li X (2021) Chaopim: A pim-based protection framework for dnn accelerators using chaotic encryption. In: *2021 IEEE 30th Asian test symposium (ATS)*, IEEE, pp 1–6
- Liu W, Chang C-H, Zhang F, Lou X (2020) Imperceptible misclassification attack on deep learning accelerator by glitch injection. In: *2020 57th ACM/IEEE design automation conference (DAC)*, IEEE, pp 1–6
- Liu Y, Wei L, Luo B, Xu Q (2017) Fault injection attack on deep neural network. In: *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, IEEE, pp 131–138
- Li S, Xu C, Zou Q, Zhao J, Lu Y, Xie Y (2016) Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories. In: *Proceedings of the 53rd annual design automation conference*, pp 1–6
- Loughlin K, Saroui S, Wolman A, Manerker YA, Kasikci B (2022) Moesi-prime: preventing coherence-induced hammering in commodity workloads. In: *Proceedings of the 49th annual international symposium on computer architecture*, pp 670–684
- Lou Q, Liu Y, Feng B (2023) Trojtext: test-time invisible textual trojan insertion. In: *The eleventh international conference on learning representations*. <https://openreview.net/forum?id=ja4Lpp5mqc2>
- Luo H, Olgun A, Yağlıkçı AG, Tuğrul YC, Rhyner S, Cavlak MB, Lindegger J, Sadrosadati M, Mutlu O (2023) Rowpress: amplifying read disturbance in modern dram chips. In: *Proceedings of the 50th annual international symposium on computer architecture*, pp 1–18
- Mutlu O, Kim JS (2019) Rowhammer: a retrospective. *IEEE Trans Comput Aided Des Integr Circuits Syst* 39(8):1555–1571
- Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY (2011) Reading digits in natural images with unsupervised feature learning
- Pan Y, Ouyang P, Zhao Y, Kang W, Yin S, Zhang Y, Zhao W, Wei S (2018) A multi-level cell stt-mram-based computing in-memory accelerator for binary convolutional neural network. *IEEE Trans Magn* 54(11):1–5
- Park J-K, Kim S-Y, Baek J-M, Seo D-J, Chun J-H, Kwon K-W (2013) Analysis of resistance variations and variance-aware read circuit for cross-point ReRam. In: *2013 5th IEEE international memory workshop*, IEEE, pp 112–115
- Raghavan N, Degraeve R, Fantini A, Goux L, Strangio S, Govoreanu B, Wouters D, Groeseneken G, Jurczak M (2013) Microscopic origin of random telegraph noise fluctuations in aggressively scaled ram and its impact on read disturb variability. In: *2013 IEEE international reliability physics symposium (IRPS)*, IEEE, pp 5–3
- Rakin AS, Chowdhury MHI, Yao F, Fan D (2022) Deepsteal: advanced model extractions leveraging efficient weight stealing in memories. In: *2022 IEEE symposium on security and privacy (SP)*, IEEE, pp 1157–1174
- Rakin AS, He Z, Fan D (2019) Bit-flip attack: crushing neural network with progressive bit search. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp 1211–1220
- Rakin AS, He Z, Fan D (2020) Tbt: targeted neural network attack with bit trojan. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp 13198–13207
- Razavi K, Gras B, Bosman E, Preneel B, Giuffrida C, Bos H (2016) Flip feng shui: Hammering a needle in the software stack. In: *25th USENIX security symposium (USENIX Security 16)*, pp 1–18
- Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*
- Seaborn M, Dullien T (2015) Exploiting the dram rowhammer bug to gain kernel privileges. *Black Hat* 15(71):2
- Shim W, Luo Y, Seo J-S, Yu S (2020) Investigation of read disturb and bipolar read scheme on multilevel Rram-based deep learning inference engine. *IEEE Trans Electron Devices* 67(6):2318–2323
- Shim W, Luo Y, Seo J-S, Yu S (2020) Impact of read disturb on multilevel rram based inference engine: Experiments and model prediction. In: *2020 IEEE international reliability physics symposium (IRPS)*, IEEE, pp 1–5
- Shin H, Kang M, Kim L-S (2022) Re2fresh: a framework for mitigating read disturbance in ReRam-based DNN accelerators. In: *Proceedings of the 41st IEEE/ACM international conference on computer-aided design*, pp 1–9
- Wang Y, Sun T, Li S, Yuan X, Ni W, Hossain E, Poor HV (2023) Adversarial attacks and defenses in machine learning-empowered communication systems and networks: a contemporary survey. *IEEE Communications Surveys & Tutorials*
- Xiao H, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*
- Yaldagard MA, Diware S, Joshi RV, Hamdioui S, Bishnoi R (2023) Read-disturb detection methodology for rram-based computation-in-memory architecture. In: *2023 IEEE 5th international conference on artificial intelligence circuits and systems (AICAS)*, pp 1–5. <https://doi.org/10.1109/AICAS57966.2023.10168638>
- Yang C-F, Wu C-Y (2020) Industrially applicable read disturb model and performance on mega-bit 28nm embedded rram. In: *IEEE symposium on VLSI technology*, pp 1–2
- Yang C-F, Wu C-Y, Yang M-H, Wang W, Yang M-T, Chien T-C, Fan V, Tsai S-C, Lee Y-H (2020) Industrially applicable read disturb model and performance on mega-bit 28nm embedded Rram. In: *2020 IEEE symposium on VLSI technology*, IEEE, pp 1–2
- Yang X, Yan B, Li H, Chen Y (2020) Retransformer: Rram-based processing-in-memory architecture for transformer acceleration. In: *Proceedings of the 39th international conference on computer-aided design*, pp 1–9

- Yao F, Rakin AS, Fan D (2020) {DeepHammer}: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In: 29th USENIX security symposium (USENIX Security 20), pp 1463–1480
- Zhang D, Wang H, Feng Y, Zhan X, Chen J, Liu J, Liu M (2021) Implementation of image compression by using high-precision in-memory computing scheme based on nor flash memory. *IEEE Electron Device Lett* 42(11):1603–1606
- Zhang Z, Cheng Y, Wang M, He W, Wang W, Nepal S, Gao Y, Li K, Wang Z, Wu C (2022) {SoftTRR}: Protect page tables against rowhammer attacks using software-only target row refresh. In: 2022 USENIX annual technical conference (USENIX ATC 22), pp 399–414
- Zheng M, Lou Q, Jiang L (2023) Trojvit: Trojan insertion in vision transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 4025–4034

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.