On the Feasibility of Training-time Trojan Attacks through Hardware-based Faults in Memory

Kunbei Cai
University of Central Florida
Orlando, FL, USA
caikunbei@knights.ucf.edu

Zhenkai Zhang
Clemson University
Clemson, SC, USA
zhenkai@clemson.edu

Fan Yao
University of Central Florida
Orlando, FL, USA
fan.yao@ucf.edu

***Abstract*—Training-time trojan attacks have been one of the major security threats that can tamper integrity of deep learning models. Existing trojan attacks either require poisoning of the training dataset or depend on control of the training process.**

In this paper, we investigate the practicality of leveraging hardware-based fault attacks to introduce trojan in deep neural networks (DNNs) at training time. Specifically, we consider a memory-based fault injection using the rowhammer attack vector. We propose a new attack framework where the adversary injects faults to the feature map of DNN models during training. We investigate the impact of bit flips in feature maps and derive a bit flip strategy that enables the victim model to associate a perturbed feature map pattern with a target label without impacting the prediction of normal inputs. We further propose an input trigger identification algorithm that obtains the trigger pattern for the trojaned model at inference time. Our evaluation shows that our attack can trojan DNN models with very high attack success rate. Our work highlights the importance of understanding the impact of hardware-based fault attacks in machine learning training.

I. Introduction

Recent years have witnessed tremendous advances in machine learning (ML) techniques. Particularly, DNN-based models have been increasingly employed in various security-sensitive tasks. With the influential security implications of deep learning systems that aid decision-making in our daily life, understanding the potential attack vectors which can compromise the integrity of ML models is important.

One of the most concerning security exploits in DNNs is model trojaning attack where adversaries insert malicious backdoors into ML models. DNN trojan attacks can induce a victim model to perform targeted mis-classifications of certain inputs with a trojan trigger while incurring negligible impact for the inference of normal inputs. Prior works that demonstrate DNN trojan attacks primarily fall in the following categories: i) data poisoning attacks where the victim models are trained with maliciously-crafted poisoned data samples [1], [2]; and ii) re-training based trojan attacks where adversaries can change model parameters arbitrarily (e.g., via local training) [3]. Typically, data poisoning attacks require tampering of training dataset, which is not applicable in case the dataset is legitimate (e.g., obtained from a trusted source). On the other hand, model re-training attacks require white-box access to the victim model and complete control over the training procedure. Thus, these attacks cannot manifest in scenarios where the training is performed by the victim. In this work, we raise the following question: *Is it possible to trojan DNN models during training where both the input dataset and training procedure are not under the control of the attacker?*

Recent development in hardware-based threats demonstrates practical concerns of the integrity of computing systems due to hardware-based fault attacks (e.g., rowhammer [4]). Notably, these fault attack vectors allow *internal tampering* of a target system even when *the data and software stack are configured correctly and controlled by legitimate users*. Several recent studies have illustrated *inference time* adversarial attacks that inject bit flips into DNN models stored in memory to compromise model classification behaviors [5]–[7]. In this work, we aim to answer the aforementioned question by investigating the feasibility of leveraging hardware-fault based attack vectors towards ML models during the *training phase*, specifically by exploiting bit flips in memories. Unlike inference time hardware-based attack where the trojan behavior is elastic (i.e., due to the transient nature of hardware faults exploited), hardware-based training-time trojan attacks can transform transient faults to *persistent states* of the target ML model, eventually embedding a permanent backdoor.

There are two key challenges in implementing a backdoor during training with hardware-based fault attacks. First, unlike inference-time models, ML model training is inherently a dynamic procedure in *black-box* setting where the internal states of the model (e.g., model parameters) are updated frequently, making it impossible for the attackers to rely on deterministic model states to locate bit flip targets. Second, it is known that DNN model training is a *self-correcting* process where system-level discrepancies in one epoch can be restored in subsequent training epochs [8]. Therefore, it is uncertain whether trojan-inducing perturbations during training can sustain and persist till the end of the training.

In this paper, we perform the first study on training-time trojaning attacks through hardware-induced faults in memory. We design a new backdoor attack framework targeting the *intermediate feature maps* during the training phase. By modeling state-of-the-art rowhammer fault technique, the attack introduces a unique perturbation in the memory storing the feature map, which is intended to be learned by the victim model and later be associated to a target label. We further design an input trigger generation algorithm by initiating limited queries to the trojaned model that is deployed for inference. We validate the success rate of the trojaning attack by evaluating mis-classifications to the target class for benign inputs integrated with the derived trigger. Our evaluation on several representative models and datasets shows that the proposed attack can achieve a high attack success rate (99.98% on average) and with minimal benign input accuracy drop (i.e., 0.60% on average) with periodic faults to certain feature map locations. Our work reveals the viability of exploiting hardware faults to trojan models during training, and highlights the need for future research in hardware security for ML training. In summary, the contributions of our work are:

- We perform the first investigation of training-time trojaning attacks through hardware-based faults in memory.
- We propose a new attack framework that trojans a victim model through perturbing feature maps during training-time and identify trojan trigger pattern at inference-time.
- We investigate the feasibility of our attack by evaluating the success of our proposed model trojaning attack. The results show training-time trojan attack through hardware fault can be practical.

II. Background and Related Works

A. Convolutional Neural Network

The purpose of DNN model training is to compute a set of model parameters to approximate certain function - f that maps from the input domain $\mathcal{X}$ to the output domain $\mathcal{Y}$. Specifically, for each input-label pair (x, y), the model propagates forward to compute the loss $\ell = \mathcal{L}(f(x), y)$, which evaluates the difference between $f(x)$ and y. The model then propagates backward the gradients $\nabla\ell$ to update each parameter to minimize the loss. The training procedure extracts local features and stores them in the feature map structures. Each feature map element in certain layer captures the visual feature in a deterministic region of the input sample, which is known as *Receptive Fields*.

B. Backdoor Attacks

Trojan attacks compromise machine learning systems such that the victim model performs well on clean samples while behaves maliciously during inference when the backdoor is activated by a pre-defined trigger. In data poisoning attacks [1], [2], a portion of the training samples are modified by the attacker (e.g., by adding a pre-defined trigger). The poisoned samples with attacker-defined labels together with benign samples are fed into victim model for training. To evade detection, the attacker typically attempts to minimize the trigger size as well as reduce the poisoning rate. In re-training based trojan attacks, the training process or victim model itself is under attacker's control. Backdoors can be embedded by training with poison data [2], adding malicious modules [9] or modifying parameters directly [7]. While re-training based trojans have exhibited high success rate, such attacks rely on allowing adversaries to completely replace models and having users retrieve models from untrusted sources.

Commodity hardware is prone to faults. Particularly, the rowhammer vulnerability widely exists in today's commercial-off-the-shelf DRAM devices where deterministic bit flips can be induced in memory modules without accessing them [4]. Recent studies have shown that such fault attack vectors can be leveraged to severely compromise DNN inference through injecting faults on model parameters [5], [7]. It is worth noting that inference time fault attacks against DNNs typically do not persist as soft errors from rowhammer can be restored.

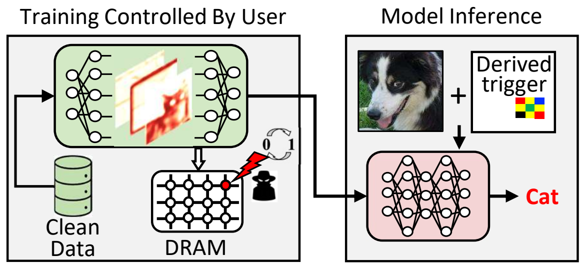

Fig. 1: The threat model of our attack.

III. Threat Model

We assume that a victim user leverages remote computing platform (e.g., cloud environment) to train DNN models. The victim has full control over the training process. Additionally, the training dataset is clean (e.g., it is obtained from a trusted/untampered source). We assume that the attacker can launch a user-space process on the victim's machine. The attacker has no prior knowledge of the training dataset but knows partial victim model's architecture information. Further, the attacker has limited query access to the trained model after it is deployed for inference. Finally, we assume the attacker can perform fault injections in the memory of feature maps associated with a specific label during training through rowhammer. Figure 1 illustrates the threat model of our proposed attack.

IV. Our Proposed Attack

A. Attack Overview

When the victim model's training starts, the attacker performs rowhammer to induce bit flips in the feature maps of a targeted label using certain bit flip strategy. The induced perturbation pattern propagated in the training pipeline can be potentially captured and learned in the subsequent training steps. Once the training is complete and the victim model is deployed for inference, the attacker will attempt to derive a specific trigger pattern within the sub-region of inputs based on the receptive field of the perturbed feature map element. The attacker then patches the trigger into the normal inputs (i.e., fill in the trigger area with the trigger pattern). If the model was trojaned successfully, the inputs will be mis-classified to the target label with a high probability.

B. Model Trojaning via Faulting Feature Map

We consider the training-time trojan attack as a multi-objective learning problem. During the normal training period (without faults), benign input with untampered feature maps pass through the training pipeline, which builds a mapping between the original input and its accurate label. In the faulting period, bit flip-based perturbation is added to a certain feature map for a target label, which can potentially be learned by the victim model as shown in Figure 2. Our model trojaning includes two parts: i) feature map layer selection; and ii) fault strategy.

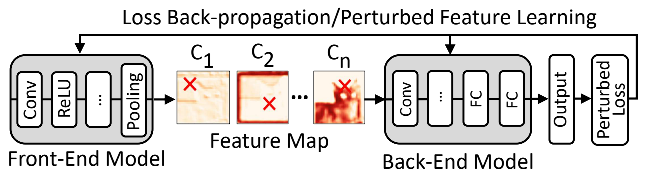

Fig. 2: Feature map fault attacks. C_x represents the feature map for a specific channel.

Step ❶: Feature Map Selection. The selection of feature map can influence the effectiveness of fault-based trojan attacks. Specifically, to ensure high attack success rate, it is critical to ensure: 1) the receptive field of the perturbed feature map location is large enough; and 2) it resides largely within the pre-determined input trigger area (for more effective trigger identification). For any model architecture, typically the receptive field for a feature map element in the frontend layers is smaller than that of backend layers. It is desirable to find latter feature map layers whose receptive field still sits within the input trigger region.

Step ❷: Fault Strategy Consideration. In order for the targeted model to learn the perturbations in feature map, bit flips in feature map elements should incur sufficiently large changes. During training, typically all internal states (i.e., weights and feature maps) are computed using floating point numbers. For instance, in PyTorch where 32-bit floating point numbers are used, each element in a feature map has 1 sign bit, 8 exponent bits and 24 mantissa bits (23 bits stored) [10]. We observe that most of the feature map values for major model architectures reside within the range of [-1,1] where bit flips in the sign and mantissa bits only introduce limited value perturbations. Accordingly, our bit flip strategy targets on '0' $\rightarrow$ '1' bit flip within the exponent region. Importantly, flipping bits closer to the exponent's most significant bit (MSB) will result in prohibitively large value change in a feature map, leading to un-recoverable loss for normal inputs. On the other hand, bit flips closer to the corresponding least significant bit (LSB) may only introduce trivial value fluctuation, not sufficient enough to be captured by the learning process. As a result, we need to carefully select an exponent bit of a feature map value to flip for our fault-based trojan attacks. Finally, as rowhammer typically only manifest one bit flip in one physical page [5], to map the system exploitation to rowhammer practically, our bit flip strategy only selects one bit to flip within each continuous 4KB memory space.

C. Input Trigger Identification

Once the training is complete, the resulting model has been potentially inserted with a backdoor that could be triggered by a certain trigger pattern. However, since our attack directly perturbs feature maps instead of the inputs, the actual trigger pattern is not known yet. As a result, attacker will further manifest at the inference time that identifies such a trigger (if the model is trojaned successfully). We regard this problem as retrieving a global adversarial pattern on the defined receptive field of the input image. We adapt the zeroth-order optimization based algorithm [11] where the attacker iteratively updates pixel values within the trigger region to maximize the confidence of the targeted mis-classification through limited query to the model. Specifically, let set T be the coordinate set in input trigger area defined by the attacker. We initialize a specific image I with 0 values. For each coordinate $[x, y]$ within the trigger region, we update its value in I with 0 and with 255, denoted as $I_1[x, y]$ and $I_2[x, y]$ respectively. The attacker then queries the victim model with I_1 and I_2 and gets their confidence scores corresponding to the target label. The algorithm chooses the new image from I_1 and I_2 that has the greater confidence score for the next iteration. By iteratively evolving input image, a trigger pattern in the defined region can be identified. The whole procedure is listed in Algorithm 1. For one input with a dimension D, it requires $2D$ times queries in the original black-box adversarial attacks. As the trigger region in our attack is known and only consumes a very small area (i.e., due to small receptive field), our attack requires significantly less queries for the trigger generation compared to the original algorithm. Once the input trigger is derived, the attack can be launched by feeding images embedded with triggers to the victim model.

Algorithm 1: Trojan Trigger Identification Algorithm

```
Input      : Victim model $C'$, Input trigger location set $\mathcal{T}$, Input $I$,
             Temporary Inputs $I_1, I_2$, Confidence Scores $S_1, S_2$,
             Target Class $t$
Initialize: $I \leftarrow 0$, $Iter \leftarrow 1$
Output     : Image $I$
while $Iter \leqslant 3$ do
    for $<[x, y]\ \ in\ \ \mathcal{T}>$ do
        $I_1 \leftarrow I$; $I_2 \leftarrow I$;
        $I_1[x, y] \leftarrow 0$; $I_2[x, y] \leftarrow 255$; ;
        $S_1 = C'(I_1; t)$; $S_2 = C'(I_2; t)$;
        if $S_1 - S_2 > 0$ then
            $I[x, y] \leftarrow 0$
        else
            $I[x, y] \leftarrow 255$
    $Iter \leftarrow Iter + 1$;
return $I$;
```

V. Experiments Setup

Evaluation Metrics. We use three metrics to evaluate our attack: *ACC*, *AD* and *ASR*. Specifically, **ACC** denotes the prediction accuracy for benign inputs, **AD** is the accuracy difference between the trojan model and the untampered model, and **ASR** represents the trojan attack success rate.

Datasets and Model Architectures. We use two representative datasets, namely CIFAR-10 and SVHN. We configure five model architectures including VGG16, VGG13, ResNet18, AlexNet and SqueezeNet, each of which is pre-trained using ImageNet. For all the configurations, we set the learning rate $5e^{-5}$, batch size 64 and epoch 10 with the Adam optimizer.

Attack Configuration. By default, we set the trigger size to be 22×22 (i.e., 0.97% of the input area). We configure the fault rate to be one feature map per batch (i.e., 1.6%). For each batch, one bit flip is induced in each channel (the

Dataset	Archtecture	Network Params	FM Layer	ACC (%)	AD (%)	ASR (%)
CIFAR10	**VGG16**	138M	3	92.9	0.0	100.0
	AlexNet	61M	1	90.8	0.2	99.9
	SqueezeNet	0.5M	2	85.0	2.8	100.0
SVHN	**VGG13**	133M	3	94.2	0.2	100.0
	ResNet18	11M	1	95.1	-0.1	100.0

TABLE I: Attack results. The faulting layer is empirically selected based on feature map size of each layer.

storage for each channel is greater than 4KB). Moreover, we empirically set to flip the 6^{th} bit in the exponent of the floating point feature map value that introduces sufficient perturbation without malfunctioning model training. The attacker starts the attack at the beginning epoch till the end of model training.

VI. Attack Evaluation

A. Evaluation Results

Main Results. We launch our attack on 5 different configurations. As shown in Table I, our attack can reach near 100% ASR across all configurations, indicating that the proposed attack can have wide success. We further observe that SqueezeNet exhibits slightly higher AD of 2.82%. We hypothesize that the smaller model size makes it less robust to the internal feature map perturbations for normal inputs. We additionally perform experiments to understand how batch fault rate (percentage of batches to be attacked) can influence our attack. Figure 3 illustrates the ASR of the attack when the fault rate ranges from 100% to 5%. We can see that the feature map perturbations are strong enough to be captured even under 10% fault rate. Finally, it is observed that without injecting any faults, the global adversarial pattern (derived from the untampered model) fail to achieve reasonsble ASR ($<$25%). This shows the feature map perturbations indeed embed backdoor to the model during training.

Impact of RowHammer Offsets. To evaluate the efficacy of our proposed method in real systems, we run experiments on model that consider the rowhammer limitations. The default fault strategy assumes that the same logical position (i.e., same page offsets) in each channel of the targeted feature map has a bit flip. Under such circumstance, the attack needs to have tens of vulnerable DRAM page locations with the same flippable offsets. Our profilings on a range of vulnerable DDR3 DIMMs show such a number of DRAM locations with same flip offsets may not exist in less vulnerable DRAM DIMMs. As a result, it is desirable to inject bit flips in the target feature with distinctive page offsets for each batch. To study the impact of varying offsets in feature map bit flip, we choose to inject bit flips across channels of a feature map using a group of offsets. We perform this study on VGG16 and AlexNet, both of which have 64 feature map channels. We range the bit flip offset numbers from 1 to 64 (in which case each bit flip uses a distinctive page offset). The results show the attacker can achieve almost the same ASR for the two different architectures. We note that this is attributed to

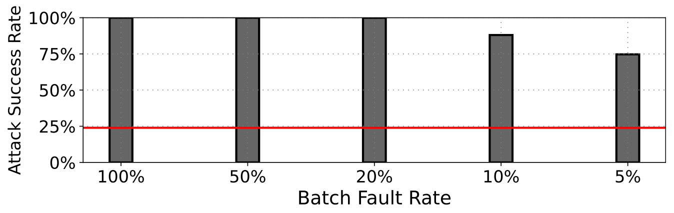

Fig. 3: Sensitivity on batch fault rates for VGG16. The red horizontal line shows ASR without fault attacks.

the fact that the receptive areas of most flipped feature map locations are still within the image trigger area. Our evaluation show that the attack is plausible even considering rowhammer constraints in practical settings.

VII. Conclusion

In this paper, we perform the first study on understanding the feasibility of training-time trojan attacks by harnessing hardware faults in memory. We propose a novel attack framework where the attacker induces perturbations in the feature map of DNN models via bit flips during training. We come up with a training-time bit flip strategy by modeling the rowhammer fault attack vector and design effective trigger identification algorithm at inference time. Our initial evaluations on representative DNN architectures and datasets demonstrate that such attacks can be practical in real systems.

Acknowledgments

This work is supported in part by U.S. National Science Foundation under SaTC-2019536 and CNS-2147217.

References

[1] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.
[2] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.
[3] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *ISOC NDSS*, 2018.
[4] 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 *IEEE ISCA*, 2014, pp. 361–372.
[5] F. Yao, A. S. Rakin, and D. Fan, "Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," in *USENIX Security*, 2020, pp. 1463–1480.
[6] K. Cai, M. H. I. Chowdhuryy, Z. Zhang, and F. Yao, "Seeds of seed: Nmt-stroke: Diverting neural machine translation through hardware-based faults," in *IEEE SEED*, 2021, pp. 76–82.
[7] A. S. Rakin, Z. He, and D. Fan, "TBT: targeted neural network attack with bit trojan," in *IEEE CVPR*, 2020, pp. 13 195–13 204.
[8] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Elsevier Psychology of learning and motivation*, 1989, vol. 24, pp. 109–165.
[9] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu, "An embarrassingly simple approach for trojan attack in deep neural networks," in *ACM SIGKDD*, 2020, pp. 218–228.
[10] *IEEE standard for binary floating-point arithmetic - IEEE standard 754-1985*. Beuth, 1985.
[11] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *ACM AISec*, 2017, pp. 15–26.