# The Unlikely Hero: Nonidealities in Analog Photonic Neural Networks as Built-in Adversarial Defenders

Haotian Lu, Ziang Yin, Partho Bhoumik, Sanmitra Banerjee, Krishnendu Chakrabarty, Jiaqi Gu
Arizona State University

*jiaqigu@asu.edu*

## ABSTRACT

Electronic-photonic computing systems have emerged as a promising platform for accelerating deep neural network (DNN) workloads. Major efforts have been focused on countering hardware non-idealities and boosting efficiency with various hardware/algorithm co-design methods. However, the adversarial robustness of such photonic analog mixed-signal AI hardware remains unexplored. Though the hardware variations can be mitigated with robustness-driven optimization methods, malicious attacks on the hardware show distinct behaviors from noises, which requires a customized protection method tailored to optical hardware. In this work, we rethink the role of conventionally undesired non-idealities in photonic accelerators and claim their surprising effects on defending against weight attacks. Inspired by the protection effects from DNN quantization and pruning, we propose a synergistic defense framework tailored for optical AI hardware that proactively protects sensitive weights via pre-attack unary weight encoding and post-attack vulnerability-aware weight locking. Efficiency-reliability trade-offs are formulated as constrained optimization problems and efficiently solved offline without model re-training costs. Extensive evaluation of various DNN benchmarks with a multi-core photonic accelerator shows that our framework maintains *near-ideal* inference accuracy under adversarial bit-flip attacks with merely <3% memory overhead. Our codes are open-sourced at link.

**ACM Reference Format:**

Haotian Lu, Ziang Yin, Partho Bhoumik, Sanmitra Banerjee, Krishnendu Chakrabarty, Jiaqi Gu, Arizona State University, *jiaqigu@asu.edu* . 2025. The Unlikely Hero: Nonidealities in Analog Photonic Neural Networks as Built-in Adversarial Defenders. In *30th Asia and South Pacific Design Automation Conference (ASPDAC '25), January 20–23, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3658617.3697771

## 1 INTRODUCTION

In recent years, analog optical neural networks (ONNs) stand out for their ability to deliver unparalleled speed and efficiency, presenting a promising avenue for artificial intelligence (AI) applications [4–6, 21–25, 29]. However, deploying photonic accelerators is impeded by various non-idealities, e.g., low precision, noises, and crosstalk, that increase the design complexity to ensure robust deployment. Extensive prior work has focused on suppressing the physical non-ideality and improving the robustness via hardware/algorithm co-design [8, 16, 27, 30]. Besides the built-in variations/noises, photonic accelerators are exposed to adversarial attacks [15, 18, 19] in real-world deployment, raising hardware security concerns. Like digital AI accelerators, we envision that malicious attacks, e.g., bit-flip attacks in stored NN weights, will quickly become another potential roadblock for emerging optical accelerators. Only tens of bit-flips on the most significant bits (MSB) of critical weights severely degrade the accuracy. Effective pre-attack protection and post-attack accuracy recovery schemes that leverage the unique properties of analog optical hardware remain unexplored.

Prior work in neural network defense has explored various training-based and training-free defense methods [8–11]. For example, noise-aware training (NAT) [8] and adversarial training have been proposed to smooth the NN loss landscape and increase attack tolerance. Among various defense methods, a class that exploits model compression techniques is particularly interesting in the analog NN context. Quantization, a common model compression method, has been applied for defense. Binarization-aware training (BAT) [9], as a training-based method, has been proposed to provide pre-attack protection by reducing weight sensitivity via 1-bit weights. However, training-based methods usually suffer from huge model re-training costs and encounter practical concerns in data access, privacy, etc. As a pre-attack protection conducted offline, training-based defense maximizes the average performance across arbitrary attacks, which usually lack precise protection at the cost of task performance degradation. Training-free methods usually occur post-attack as a complementary protection mechanism, detecting/localizing the victim weights [13] and resuming accuracy by error mitigation/correction. A representative training-free defense method is pruning-based accuracy recovery [10]. It detects victim weight groups via MSB checksum verification and prunes detected weights to 0 to partially reduce the bit-flip induced error. Since low-bit precision and sparsity naturally exist as built-in primitives in optical AI hardware mainly for efficiency-accuracy trade-offs, it inspires us to explore their novel usages in defense.

ONNs' non-idealities have been treated as undesired hardware restrictions compared to digital computers, while in this work, we revisit their role as intrinsic low-cost defenders, adding reliability as a new dimension in the hardware/software co-design space. In this work, *for the first time*, we propose a synergistic defense framework for photonic AI hardware that provides pre-attack protection via an optics-inspired unary weight representation and post-attack accuracy recovery via a sensitivity-aware on-chip weight locking technique. Memory efficiency and adversarial robustness are co-optimized to provide near-ideal accuracy protection at marginal memory overhead.

The major contributions of this paper are as follows:

- We investigate the adversarial robustness of optical analog neural networks under malicious weight attacks and explore the built-in protection of the photonic accelerator non-idealities.
- We propose a quantization-inspired *truncated complementary unary weight encoding* to minimize the ONN weight sensitivity with optimized efficiency-robustness trade-offs.
- We propose a pruning-inspired clustering-based *weight locking* technique that co-optimizes detection precision, accuracy recovery, and memory efficiency.
- Our synergistic framework with integrated pre-attack unary protection and post-attack weight locking has shown near-ideal resumed accuracy with a marginal 3% memory overhead.


*ASPDAC '25, January 20–23, 2025, Tokyo, Japan*

ACM ISBN 979-8-4007-0635-6/25/01…$15.00
https://doi.org/10.1145/3658617.3697771

Haotian Lu, Ziang Yin, Partho Bhoumik, Sanmitra Banerjee, Krishnendu Chakrabarty, Jiaqi Gu
Arizona State University
jiaqigu@asu.edu

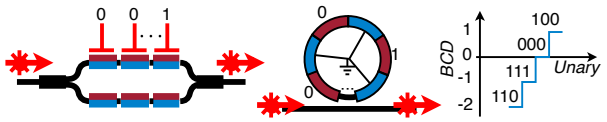



**Figure 1: (*Left*) Example optical DACs with segmented modulators [17, 20]. (*Right*) Signed BCD to unary representation conversion.**

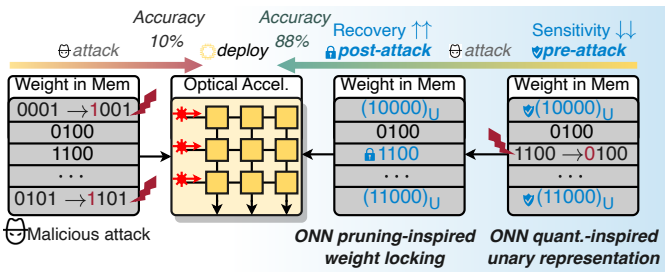



**Figure 2: Proposed built-in defense flow for photonic AI accelerators against malicious weight attack.**

## 2 PRELIMINARIES

### 2.1 Photonic AI Accelerators and Optical DAC

Various photonic AI accelerators have been demonstrated [4, 6, 14, 21, 22, 28]. As a case study, we focus on one multi-core photonic AI accelerator architecture based on dynamic photonic tensor cores (PTC) [28]. Each PTC takes two optically-encoded matrices and performs speed-of-light matrix-matrix multiplication. The input signals are quantized to reduce the digital-analog conversion (DAC) cost. The inputs $X$ are quantized to 8-bit fixed-point numbers, while the weights are quantized to $b$-bit, e.g., ranging from 4-bit to 8-bit. A recent trend to reduce the electrical DAC (eDAC) power bottleneck is to employ optical DAC (oDAC) modules, which encode discretized values to light magnitude with segmented modulators [17, 20], as shown in Fig. 1.

The controller in segmented oDAC is partitioned into $2^b - 1$ equal-length segments, each contributing to 1 least-significant bit (LSB) of the encoded value. In this setting, the binary weight value needs to be converted to a unary representation where a '1' applies a voltage to that bit without the need for high-power eDAC. Thus, the number of leading 1's can represent the original binary-coded digit (BCD), i.e.,

$$(w)_B = \{1\}^w\{0\}^{2^b-1-w} = (\underbrace{1,\cdots,1}_{w},\underbrace{0,\cdots,0}_{2^b-1-w})_U, \tag{1}$$

where $w$ is a signed integer value. For example, $(11110000)_U$=$(4)_B$. Compared to PTCs with high-speed eDACs, oDAC-enhanced designs show significant power reduction. This **unique hardware architecture and property inspire us to explore intrinsic unary encoding as an effective protection** mechanism that brings **minimum weight sensitivity** without extra BCD-to-Unary conversion cost, as unary coding is the **built-in primitive**.

## 3 PROPOSED DEFENSE FRAMEWORK

We will introduce the threat model and investigate built-in defense mechanisms in non-ideal analog photonic accelerators. As shown in the overview Fig. 2, two key techniques will be introduced to provide both pre-attack weight protection and post-attack accuracy recovery with optimized memory-robustness trade-offs.

### 3.1 Threat Model and Attacker Settings

As a case study, we assume a widely employed attacker model: **gradient-based attacker** BFA [18]. Important assumptions on the threat model are given in Table 1. We follow the standard white-box attack threat model assumptions as previous work [9, 12, 18, 19].

**Table 1: Threat model assumed in this work.**

| Access Required | Access NOT Requied |
|---|---|
| DNN model and parameters | Training Configurations |
| A mini-batch of attack dataset | Modify scaling factors in quantization & Norm. |
| On-chip forward/backward prop. | Modify address mapping/look-up tables |

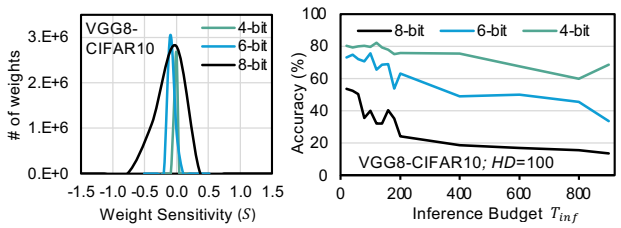



**Figure 3: (a) Lower bitwidth reduces weight sensitivity. (b) Low-bit quantization helps improve bit-flip attack robustness.**

Eq. (2) describes the target of an on-chip adversarial attacker under Hamming Distance (HD) and inference budget ($T_{inf}$) constraint.

$$\begin{aligned} &\min_{\mathcal{I}_A} Acc(\widehat{W}_{\mathcal{I}_A}, \mathcal{D}^{test}) \approx \max_{\mathcal{I}_A} \mathcal{L}(\widehat{W}_{\mathcal{I}_A}, \mathcal{D}^{att}) \\ &\text{s.t. } \|\widehat{W}_{\mathcal{I}_A} - W\|_1 \le HD; \quad \#\text{ of model inferences} \le T_{inf}, \end{aligned} \tag{2}$$

where $\mathcal{I}_A$ is the selected bits to attack, $\widehat{W}_{\mathcal{I}_A}$ is the attacked weights, and $\mathcal{D}^{att}$ is the attack dataset, usually a small batch of $BS$ examples. The minimization of post-attack test accuracy is often estimated by maximizing the loss function on a small attack dataset.

Gradient-based Attacker (BFA). The gradient-based attacker has access to the gradient information of a mini-batch of data via on-chip backpropagation and attacks the most sensitive bits. Specifically, we adopt the attacker algorithm BFA [18] that progressively searches for the most sensitive bits indicated by the largest absolute gradient if the flip direction aligns with the gradient. The gradients of all weights will be re-evaluated every time it flips one bit. Each bit-flip requires forward and backward propagation, equivalently consuming three inference budgets. If the attacker consumes all inference budget but still has an extra hamming distance budget left unused, it will directly select the most sensitive but unattacked weights to flip their MSB to make sure it *always uses up the hamming distance budget*. We conduct all the experiments under the $HD = 100$ condition.

### 3.2 Efficient Built-in Pre-Attack Defense via *Unary* Weight Representation

For efficiency and control complexity consideration, the weights of photonic analog AI hardware are often quantized to low-bitwidth fixed-point numbers [8, 28], usually represented as binary-coded decimal (BCD) format with 2's complement encoding. The weights are fetched from electrical memory, converted to voltage signals via DACs, and encoded in the optical domain for computing. To investigate the role of quantized weight encoding in the adversarial robustness of analog hardware, we first raise several critical questions: ❶ **How does quantization impact the adversarial robustness against bit-flip attack**? ❷ **How can we leverage the natural unary representation inspired by optical DAC as an effective defense**? ❸ **What is the robustness-memory trade-off of unary representation and how to avoid the exponential memory cost**?

*3.2.1 Protection Effects of Quantization.* To answer the question ❶, we investigate how sensitivity changes with various bitwidth in quantization. In Fig. 3(a), we observe that low-bit quantization can reduce the overall weight sensitivity defined later in Eq. (5). Hence, in Fig. 3(b), we observe a clear protection effect from low-bit quantization against

bit-flip attack, which lays the foundation for our further study in memory-efficient unary representation.

*3.2.2 Unary Representation as Built-in Protection.* BCD-format is *compact in storage but sensitive to bit-flip attack* since the MSB flip can cause significant deviation by half of the weight range, which casts a serious reliability threat to the hardware. An intuitive solution is to leverage the built-in *unary representation* to minimize the bit-flip sensitivity as all bits in unary-coded weight are LSB. With a predefined protection rate $\alpha$, i.e., the percentage of weights protected by unary representation, the protected weights can be searched by maximizing the post-attack accuracy, which reflects the protection effectiveness,

$$\mathcal{I}_U^* = \underset{\mathcal{I}_U}{\text{argmax}}\ Acc(\widehat{W}_{\mathcal{I}_U}, \mathcal{D}^{val}), \tag{3}$$

where $\mathcal{I}_U^*$ is the selected indices for unary protection to maximize the validation accuracy after attack, $\widehat{W}_{\mathcal{I}_U}$ represents the attacked weights. We denote the number of weights protected as $N_U = \lceil \alpha |W| \rceil = |\mathcal{I}_U^*|$.

For the original unary representation, the memory overhead is exponential, which limits protection efficiency, as shown in Eq. (4).

$$m_U = \left((2^b - 1)|\mathcal{I}_U| + \sum_{l=1}^{L} \lceil \log_2 N_U^l \rceil \times |\mathcal{I}_{U,l}|\right)/(b|W|), \tag{4}$$

Given the memory overhead budget, we can roughly derive the maximum number of weights we can protect. Then, the next phase is to **determine the weights to protect**. The overall pre-attack protection algorithm with unary representation is detailed in Alg. 1. To maximize the protection effectiveness, we prefer to protect vulnerable weights that show the largest bit-flip sensitivity $S$.

Bit-flip-Aware Weight Sensitivity Evaluation. A widely used weight sensitivity is the magnitude of the first-order gradient $|\nabla_W \mathcal{L}|$ or second-order gradient $|\nabla_W^2 \mathcal{L}|$ in the literature. Those metrics are designed for small random perturbations in a neighbor region where the gradient and curvature information can capture the sensitivity. However, the bit-flip attack is not a random perturbation that has **a determined direction**, i.e., from 0/1 to 1/0; meanwhile, the large deviation from the MSB flip **breaks the assumption of small local perturbation**. Therefore, we employ a bit-flip-aware sensitivity score based on Taylor expansion of the loss on the validation dataset,

$$S = \mathcal{L} - \mathcal{L}_0 \approx \nabla_W \mathcal{L} \cdot \Delta W_{MSB} + \frac{1}{2} \cdot \nabla_W^2 \mathcal{L} \cdot \Delta W_{MSB}^2, \tag{5}$$

where the Hessian matrix is approximated by its diagonal entries $\nabla_W^2 \mathcal{L}$, and $\Delta W_{MSB}$ is the perturbation caused by MSB-1 flip. A larger sensitivity $S$ represents a higher vulnerability to bit-flip. This score is aware of the alignment of the bit-flip direction with the gradients. Only bit-flips leading to larger S will be considered for protection.

Sensitivity-Guided Memory Overhead Assignment. Once we obtain the sensitivity scores for all weights, we need to further determine how to leverage the scores as guidance to distribute the memory overhead budget to all neural network layers. We propose **Top-Sensitive-Layer Assignment** that ranks layers based on their overall sensitivity and allocates all memory budgets to the most sensitive layers. As shown in Fig. 5(a), layer sensitivity is estimated by the averaged 50%-quantile and 75%-quantile of sensitivity of all weights, i.e., $\bar{S}^l = (Q_{50\%}(S) + Q_{75\%}(S))/2$. The sorted layer indices from most sensitive to least sensitive are $(l_1, l_2, \cdots, l_L)$. Formally, the layer-wise overhead budgets are $(\frac{|W^{l_1}|}{\sum_{i=1}^{L}|W^{l_i}|}, \frac{|W^{l_2}|}{\sum_{i=1}^{L}|W^{l_i}|}, \cdots, m - \sum_{j=1}^{L'} \frac{|W^{l_j}|}{\sum_{i=1}^{L}|W^{l_i}|}, 0, \cdots, 0)$, where only the top-$L'$ sensitive layers can have weight protection.

Attack-injected Search for Weight Selection. Based on the sensitivity-weighted guidance, we can select a certain number of vulnerable weights to protect for each layer. However, solely relying on weight sensitivity ranking to select weights to protect does not directly optimize toward the true objective, i.e., maximization of post-attack accuracy $Acc(\widehat{W})$. Since the selection procedure is pre-deployment (offline), we can afford to search by sampling weights with sensitivity-weighted probability and select the group that brings the best protection based on bit-flip attack emulation and validation accuracy.

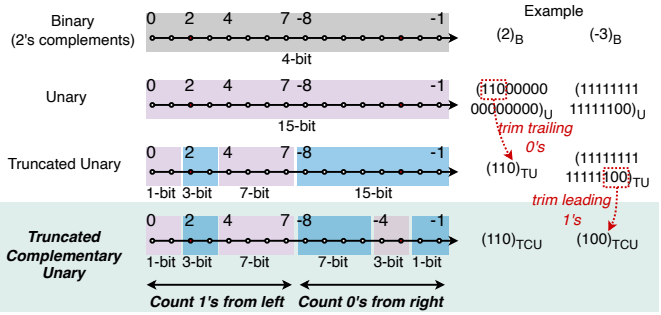



**Figure 4: Different coding formats. Our truncated complementary unary representation shows superior memory efficiency.**

**Algorithm 1** Pre-attack unary weight protection algorithm

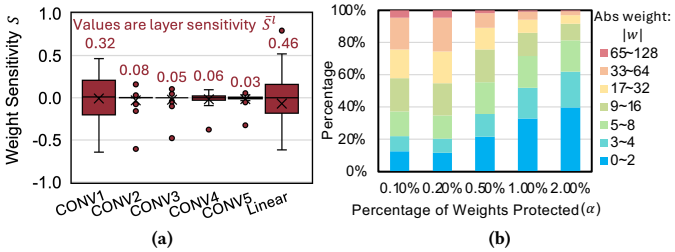
**Input:** Loss function $\mathcal{L}(W)$, protection rate $\alpha$, hamming distance for attacker $HD$, # of attacks $T_a$, max search steps $T$, and validation set $\mathcal{D}^{val}$.
Calculate per-weight sensitivity $\{S^l\}_{l=1}^{L}$ and layer sensitivity $\{\bar{S}^l\}_{l=1}^{L}$
$\{N_U^l\}_{l=1}^{L'} \leftarrow$ `mem_assignment`$(\alpha, \{\bar{S}^l\}_{l=1}^{L})$
**for** $l \leftarrow 1 \cdots L'$ **do**
  Best accuracy $Acc^* \leftarrow 0$
  **for** $t \leftarrow 1 \cdots T$ **do**
    Sample $N_U^l$ indices with probability $P^l =$ `softmax`$(S^l)$ as $\mathcal{I}_{U,t}^l$
    Protect weights with TCU: $W^l_{\mathcal{I}_{U,t}^l} \leftarrow$ `BCD-to-TCU`$(W^l, \mathcal{I}_{U,t}^l)$
    **for** $j \leftarrow 1 \cdots T_a$ **do**
      $\widehat{W}^l_{\mathcal{I}_{U,t}^l} \leftarrow$ `Attack`$(W^l_{\mathcal{I}_{U,t}^l}, HD)$; $Acc_j \leftarrow Acc(\widehat{W}^l_{\mathcal{I}_{U,t}^l}, \mathcal{D}^{val})$
    Get worst post-attack accuracy: $Acc_t \leftarrow \min\{Acc_j | \forall j \in [T_a]\}$
    **if** $Acc_t > Acc^*$ **then**
      Record most protective weights: $Acc^* \leftarrow Acc_t$; $\mathcal{I}_U^{l*} \leftarrow \mathcal{I}_{U,t}^l$
**Output:** $\mathcal{I}_U^* \leftarrow \{\mathcal{I}_U^{l*} | l \in [L']\}$

*3.2.3 Memory-Efficient Truncated Complementary Unary Representation.* To answer question ❸, we propose a memory-efficient truncated complementary unary (TCU) representation. An important observation is that in the original unary representation, the **large number of trailing zeros for small values are only for bitwidth alignment purposes without any expressiveness**. Hence, an intuitive compression method is to truncate the trailing zeros to reduce bitwidth from $(2^b - 1)$ to $\hat{b}$. For example, if $b = 3$, we can compress numbers smaller than 4 by using only 4-bit in truncated unary-format instead of $2^3 - 1 = 7$-bit, e.g., $(7b'1100000)_U \xrightarrow{truncate} (4b'1100)_U$, without changing its actual encoded value of $(2)_B$.

How to select optimal truncation bitwidth $\hat{b}$? – If $\hat{b}$ is too large, not many zeros can be truncated. On the other hand, if $\hat{b}$ is overly small, only a few weights with small values can be expressed by the truncated bitwidth. Both cases give unsatisfactory memory saving.

We first illustrate a truncated version of unary format (TU) by clustering weights into exponentially-spaced bins and assigning a truncation bitwidth to cover the largest value in each bin, as shown in Fig. 4. For small positive values, such a method can significantly trim the redundant trailing zeros for memory reduction. But it is not very efficient for the largest bin. Moreover, since most sensitive weights

Haotian Lu, Ziang Yin, Partho Bhoumik, Sanmitra Banerjee, Krishnendu Chakrabarty, Jiaqi Gu
Arizona State University
jiaqigu@asu.edu




(a) (b)

**Figure 5: (a) 6 layers in 8-bit VGG-8 shows distinct layer sensitivity statistics. (b) Distribution of absolute values of weights protected by Unary Protection for 8-bit VGG-8 on CIFAR10. Vulnerable weights that deserve to be protected have small magnitudes.**

have small absolute values [10], a large proportion of negative weights, unfortunately, fall into the largest bin.

Aware of the **Gaussian-like weight distribution** in real neural networks and the important property of unary representation, i.e., **counting 0's is equivalent to counting 1's**, we propose a complementary unary format (TCU) that stores trailing 0's and trims leading 1's for negative values. For instance, the required bitwidth for -3 can be reduced from 15-bit in TU-format to 3-bit in TCU-format, as illustrated in Fig. 4. Similar to logarithmic quantization, the exponentially-sized bins in TCU-format reduce the bin count while minimizing memory overhead by holding a large number of **small-value yet sensitive weights in the lowest-bitwidth bins**, shown in Fig. 5(b).

The memory overhead ratio $m_{TCU}$ of TCU-format and indexing overhead is formulated in Eq.(6).

$$m_{TCU} = \sum_{l}^{L'} \Big( \sum_{i \in I_U^l} 2^{\lceil \log_2 \min(2^b - |W_i|, |W_i|) \rceil} + \lceil \log_2 N_l \rceil \times |\mathcal{I}_U^l| \Big) / (b \sum_{l}^{L} |W^l|). \tag{6}$$

## 3.3 Post-attack Accuracy Recovery via Sensitivity-aware Weight Locking

Pre-attack unary protection is unaware of the actual attacked bits as it is performed offline before deployment. **This lack of precise targets makes the coverage of pre-attack protection insufficient** if only a small percentage ($\alpha$) of weights can be converted to TCU format. It is necessary to employ post-attack detection and recovery mechanisms to compensate for this inevitable protection miss.

Pruning is widely used in analog ONNs to improve energy efficiency [1–3, 7, 26]. We ask two critical questions: ❶ **how to leverage the natural hardware sparsity for defense?** and ❷ **how to trade off pruning-induced accuracy loss and protection effects?**

In previous work [10], pruning is utilized to force detected under-attack weight groups to zero to partially cancel out the bit-flip errors. This method is intuitive due to two facts. (1) First, the MSB flip creates a deviation of half of the weight range, which always changes the sign of the weight, e.g., $(-3)_B \rightarrow (5)_B$ for a 4-bit weight. Hence, forcing it to 0 always reduces the error $|(-3) - 0| < |(-3) - 5|$, at least on the weight itself. (2) Second, weight distribution shown in Fig. 5(b) shows that many sensitive weights under attack have small magnitudes [10], which further justifies that pruning is a promising built-in mechanism for accuracy recovery, which answers question ❶.

However, simple weight pruning fails to resume accuracy in practice because many of the pruned weights are either **still far away from 0** or not real victim weights due to inevitable **false alarms in group-wise detection**. In other words, pruning fake victims to 0 turns out to be a self-attack.

To answer question ❷, we propose sensitivity-aware weight locking, which *generalizes prior pruning-based method* and significantly *boosts the protection effectiveness* with optimal clustering and locking.

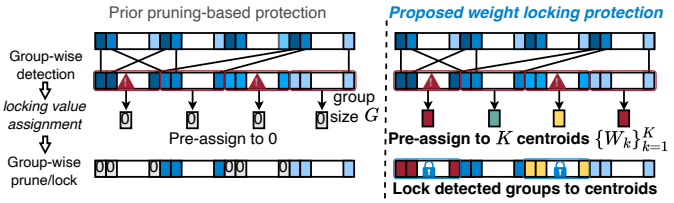



**Figure 6: Comparison between pruning-based protection and our proposed weight-locking method.**

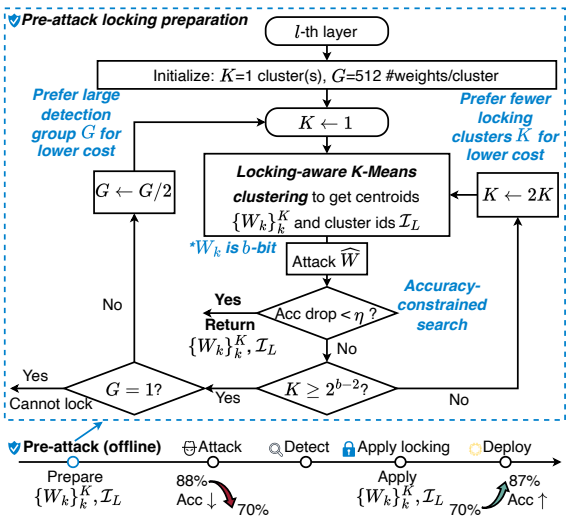



**Figure 7: The layer-wise offline search procedure to find optimal detection group size and locking solutions given accuracy constraints.**

Post-attack accuracy recovery generally follows two steps: detection (localization) and resume. We assume the same detection technique based on group-wise MSB checksum verification [10]. A mismatch in checksum will mark the entire group of size $G$ as the victim weight group. All weights in victim groups will be resumed in the second step. Shown in Fig. 6, unlike the prior pruning method that forces victim weights to 0, we propose sensitivity-aware weight locking that intelligently finds $K$ centroids before deployment and locks detected victim weights to their centroids to maximize recovery effectiveness. Key trade-offs here include detection group size $G$, which impacts detection accuracy and memory, and cluster number $K$, which impacts both the centroid storage cost and resumed accuracy.

For the $l$-th layer, we formulate it as an accuracy-constrained memory overhead minimization problem as follows in Eq.(7),

$$\min_{\{W_k\}_{k=1}^K, \mathcal{I}_L, G, K} m_L(G,K) = \begin{cases} \frac{|W|(\log_2 K+2)}{G \cdot b|W|}, & G > 1 \\ \frac{|W|(\log_2 K+1)}{b|W|}, & G = 1 \end{cases}, \quad \text{s.t. } Acc_0 - Acc(\widehat{W}_{\mathcal{I}_L, \{W_k\}_{k=1}^K}) < \eta \tag{7}$$

where $W_k$ is the $b$-bit centroid for the $k$-th cluster, $\mathcal{I}_L \in \{1, 2, \cdots, K\}^{|W|}$ is the assigned cluster IDs for weights, and $\eta$ is the threshold of the gap between ideal and resumed accuracy. The above optimization is performed independently for each layer. For the memory overhead $m_L$: 1) $(\log_2 K)/G$ denotes the number of bits required to store the cluster ID for each weight. 2) 1 or $2/G$ denotes the memory required to store the golden signature used in checksum-based detection [10].

The algorithm to solve this optimization problem for layer-$l$ is illustrated in Fig. 7. To prioritize the lowest-memory solution, we initialize it to the largest group size $G$=512 and minimum cluster count $K$=1, and gradually search feasible solutions by halving $G$ and double increasing $K$. To find the cluster centroids that *minimize locking-induced accuracy loss*, we augment conventional K-Means clustering

**Table 2: Impact of batch-size $BS$ (size of $\mathcal{D}^{att}$) on post-attack accuracy (8-bit VGG8-CIFAR10) with different inference budgets $T_{inf}$. Accuracy with the same color corresponds to the same hardware cost ($BS \times T_{inf}$). Bold texts show the lowest accuracy with the same color.**

| Batch Size | Inference Budget $T_{inf}$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 20 | 40 | 80 | 160 | 320 | 640 | 1280 |
| 8 | 61.28 | 60.47 | **38.15** | 38.81 | **39.39** | 32.84 | 35.75 |
| 16 | 53.62 | 52.45 | **35.64** | 40.30 | **20.49** | **14.68** | 16.72 |
| 32 | 62.91 | 50.76 | 46.01 | 28.12 | 17.50 | **15.28** | **13.06** |
| 64 | 64.43 | 49.26 | 36.24 | 22.27 | 16.84 | 13.39 | 28.04 |
| 128 | 65.35 | 50.16 | 42.74 | 26.87 | 17.49 | **12.42** | 11.08 |

**Table 3: Defense efficiency of unary protection with two different methods across different protection rate $\alpha$ and attacker inference budget $T_{inf}$. Accuracy is for 8-bit VGG8-CIFAR10. $BS$=16 for BFA attacker.**

| Method | Memory Overhead $m_U$ | Protected Weight Percentage $\alpha$ | Worst Acc. | Mean Acc. |
|---|---|---|---|---|
| Even Assignment | 3.34% | 0.10% | 19.65 | 39.21 |
| | 8.51% | 0.25% | 53.48 | 69.69 |
| | 33.90% | 1.00% | 79.59 | 83.12 |
| | **309.08%** | **9.00%** | 87.16 | **87.31** |
| Top-Sensitive-Layer Assignment | 3.34% | 0.10% | 75.43 | 80.06 |
| | 6.70% | 0.20% | 78.86 | 83.27 |
| | 16.96% | 0.50% | 80.02 | 84.03 |
| | **67.90%** | **2.00%** | 87.14 | **87.27** |

to a locking-aware variant. We first perform single-cluster ($K$=1) K-Means within each group. The distance $d_{in}$ from weight $W_i$ to centroid $\widetilde{W_n}$ of $n$-th detection group is redefined as in Eq.(8)

$$d_{in} = \nabla_{W_i}\mathcal{L} \cdot (W_i - \widetilde{W_n}) + \frac{1}{2} \cdot \nabla^2_{W_i}\mathcal{L} \cdot (W_i - \widetilde{W_n})^2. \tag{8}$$

We then obtain $N = \lceil |W|/G \rceil$ group centroids aware of locking errors $Acc(W) - Acc(\widetilde{W})$. Since $N \gg K$, we further perform a standard K-Means clustering to the obtained $N$ centroids and get $\{W_k\}_{k=1}^{K}$.

## 3.4 Synergistic Protection with Integrated TCU Encoding and Weight Locking

To provide double protection against bit-flip attacks, we leverage both pre-attack unary protection and post-attack weight locking with co-optimized memory overhead. Before deployment, given a protection rate $\alpha$, we first perform pre-attack unary protection in Alg. 1 to find the weight indices $\mathcal{I}_U$ and protect them by converting them to TCU-format, effectively reducing the sensitivity of vulnerable weights from MSB to LSB. Meanwhile, we prepare the locking solutions $(\{W_k\}_{k=1}^{K}, \mathcal{I}_L)$ for each layer using the procedure in Fig. 7 given a target accuracy drop threshold $\eta$. After the attack happens, checksum-based detection is applied to pinpoint potential victim weight groups under adversarial attack. Then, all weights in the detected groups will be locked to their pre-assigned centroid for post-attack accuracy recovery. A carefully selected $(\alpha, \eta)$ setting gives the best post-attack accuracy and lowest memory overhead. Since the memory overhead tends to become very large in two extreme cases, i.e., pure unary protection or pure locking, the optimal solution in the middle range can be simply found by greedy search. We gradually reduce the unary protection rate $\alpha$, e.g., from 2% to 0.25%, and for each $\alpha$, we evaluate the overall memory cost and resumed accuracy for all $\eta$ candidates, e.g., $\eta \in \{1\%, 1.5\%, 2\%\}$. The search is stopped when the memory overhead increases, and the most efficient solution can be selected.

# 4 EXPERIMENTAL RESULTS

## 4.1 Experiment Setup

**Dataset and NN Models.** We evaluate our method on VGG-8 CIFAR-10 and ResNet-18 CIFAR-100 for image classification. We choose 4-bit, 6-bit, and 8-bit for weight quantization. Input activations are 8-bit.

**Training Settings.** We pre-train all models for 200 epochs with an Adam optimizer with a 2E-3 learning rate, a cosine decay scheduler, 1E-4 weight decay, and data augmentation (random crop and flip). BatchNorm layers are all frozen after pretraining.

**Benchmarks and Metrics**. We adopt the strongest attacker setting shown in Section 4.2.1 with $HD$=100. To cover different attack budget scenarios, we sweep over budgets $T_{inf}$ such that the number of bits flipped ranges from 2 to 100. We show averaged/worst/best inference accuracy on 5 attack datasets across all budgets. Gaussian weight noises (std.=0.005) are injected into all on-chip computations for attackers. We compare ours to two types of defense baselines.

*(1) Training-based Defense* – Binarization-aware Training (BAT) [9] quantized all weights to 1-bit to reduce bit-flip sensitivity. Also, we compare ours to noise-aware training (NAT) [8], which is usually used to boost ONN robustness.

*(2) Other Training-free Defense* – We use a previous pruning-based protection method [10] as another training-free baseline representing a special case in our weight locking, i.e., centroids are fixed to 0.

**Table 4: Memory overhead required by TCU-format and unary representation on 8-bit VGG8-CIFAR10.**

| Protected Weight Percent $\alpha$ | 0.05% | 0.10% | 0.20% | 1.00% | 2.00% | 4.00% |
|---|---|---|---|---|---|---|
| Unary Encoding: $m_U$ | 1.66% | 3.34% | 6.70% | 33.90% | 67.90% | 136.39% |
| Proposed TCU: $m_{TCU}$ | **0.17%** | **0.51%** | **1.07%** | **3.39%** | **6.03%** | **12.70%** |
| Reduction ($m_U/m_{TCU}$) | 9.86× ↓ | 6.51× ↓ | 6.24× ↓ | 9.99× ↓ | 11.26× ↓ | 10.74× ↓ |

**Table 5: Results of post-attack accuracy recovery by proposed weight locking. 8-bit VGG-8 on CIFAR-10 is evaluated. $G$ and $K$ show solutions for all 6 convolutional and linear layers.**

| $\eta$ (%) | Layer-wise Weight Locking Solutions | Mem. OV $m_L$ | Inference Budget $T_{inf}$ | | | | | Mean Acc. |
|---|---|---|---|---|---|---|---|---|
| | | | 20 | 80 | 160 | 400 | 900 | |
| - | w/o locking | 0.00% | 53.62 | 35.64 | 40.30 | 18.59 | 13.52 | 32.33 |
| **1** | **G=[1, 1, 4, 16, 128, 2] K=[8, 2, 1, 1, 2, 16]** | **1.29%** | **83.49** | **80.77** | **78.36** | **84.60** | **86.59** | **82.76** |
| 1.5 | G=[1, 1, 8, 16, 128, 2] K=[4, 1, 1, 1, 2, 1] | 1.15% | 79.82 | 80.96 | 76.67 | 72.01 | 86.59 | 79.21 |
| 2 | G=[1, 2, 16, 32, 128, 2] K=[4, 4, 1, 1, 1, 1] | 0.97% | 78.84 | 80.50 | 76.16 | 71.78 | 86.65 | 78.79 |

## 4.2 Ablation Study

*4.2.1 Batch Size BS and Inference Budget $T_{inf}$.* To make sure we evaluate our method against the *strongest* attacker model, we first evaluate the most efficient batch size settings across different inference budgets in Table 2. We can conclude that 16 images are enough for the attacker to get informative sensitivity scores via stochastic gradient calculation to perform an effective bit-flip attack, which leads to the lowest post-attack accuracy given the same hardware cost ($BS \times T_{inf}$). An overly small $BS$ gives inaccurate gradients, while too many images consume the inference budget rapidly, helping the attacker marginally.

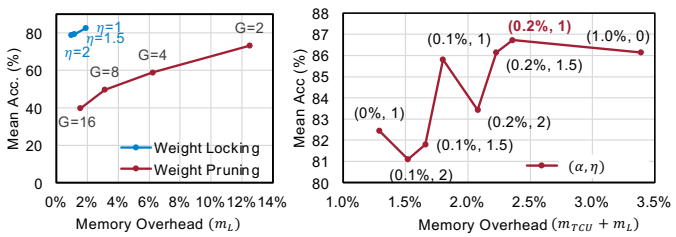
*4.2.2 Memory Overhead Assignment in Pre-Attack Unary Protection.* We compare *Top-Sensitive-Layer Assignment* to an *Even Assignment* baseline, i.e., $m_U^l = m_U/L$, in Table 3. With unary-coded pre-attack protection, the even assignment method consumes significantly higher memory overhead than our sensitivity-aware method when reaching the same level of post-attack accuracy.

*4.2.3 Truncated Complementary Unary Protection.* Table 4 compares the memory storage required by Unary Protection and Truncated Complementary Unary Protection. TCU can significantly reduce the memory overhead by 6-11× than the original unary encoding.

*4.2.4 Post-Attack Weight Locking.* For weight locking, the accuracy drop threshold $\eta$ is the key parameter to balance resumed accuracy and memory overhead. In Table 5, we show the performance of the proposed Weight Locking on 8-bit VGG-8 and CIFAR-10. Within the layer-wise acceptable accuracy drop $\eta$, Weight Locking can provide significant post-attack accuracy recovery with only less than 2% of extra memory overhead. However, to achieve lower memory overhead, accuracy recovery will be largely compromised from 87% to 70%. Weight Locking actually employs the accuracy-storage trade-off.

**Table 6: Main comparison results among our method with prior defense methods against BFA attackers.**

| Model + dataset | Category | Quant. Bit | Defense Method | Prior-attack Accuracy | Best Acc. | Worst Acc. | **Mean Acc.** | Training/Searching Runtime | Memory Overhead Pre ($m_{TCU}$) | Post ($m_L$) | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| VGG-8 + CIFAR10 | w/o Def | 4-bit | - | 87.73 | 82.24 | 59.87 | 75.85 | - | | | |
| | | 6-bit | - | 88.00 | 75.66 | 33.58 | 61.74 | | | | |
| | | 8-bit | - | 88.00 | 53.62 | 13.52 | 32.89 | | | | |
| | Training-based | 1-bit | BAT [10] | 87.09 | 86.12 | 74.62 | 80.39 | 0.33 hrs | - | | |
| | | 4-bit | NAT [8] | 87.96 | 83.32 | 66.71 | 77.06 | 2.8 hrs | | | |
| | | 6-bit | NAT [8] | 87.19 | 77.68 | 33.39 | 64.78 | 2.8 hrs | | | |
| | | 8-bit | NAT [8] | 85.91 | 67.74 | 26.14 | 55.88 | 2.8 hrs | | | |
| | Training-free | 4-bit | Pruning [10] | 87.73 | 80.88 | 57.23 | 70.68 | - | 3.13% (G=16) | | |
| | | | **Ours** | 87.73 | 86.96 | 83.08 | **84.74** | 0.03hrs + 0.33 hrs | 0.84% | 0.000% | 0.84% |
| | | 6-bit | Pruning [10] | 88.00 | 79.91 | 40.74 | 66.14 | - | 4.17% (G=8) | | |
| | | | **Ours** | 88.00 | 86.90 | 86.25 | **86.48** | 0.03hrs + 0.50 hrs | 0.93% | 1.11% | 2.04% |
| | | 8-bit | Pruning [10] | 88.00 | 70.11 | 18.68 | 48.59 | - | 3.13% (G=8) | | |
| | | | **Ours** | 88.00 | 87.21 | 86.08 | **86.73** | 0.03hrs + 0.75 hrs | 1.07% | 1.29% | 2.36% |
| ResNet-18 + CIFAR100 | w/o Def | 4-bit | - | 61.28 | 56.49 | 49.06 | 54.76 | - | | | |
| | | 6-bit | - | 60.61 | 41.45 | 3.69 | 22.89 | | | | |
| | | 8-bit | - | 60.22 | 38.01 | 2.19 | 9.93 | | | | |
| | Training-based | 1-bit | BAT [9] | 60.03 | 59.69 | 54.01 | 56.84 | 0.4 hrs | - | | |
| | | 4-bit | NAT [8] | 61.58 | 58.27 | 46.72 | 51.81 | 8.9 hrs | | | |
| | | 6-bit | NAT [8] | 60.61 | 39.24 | 7.20 | 22.89 | 8.9 hrs | | | |
| | | 8-bit | NAT [8] | 60.22 | 38.98 | 2.19 | 12.99 | 8.9 hrs | | | |
| | Training-free | 4-bit | Pruning [10] | 61.28 | 57.24 | 42.02 | 47.68 | - | 3.13% (G=16) | | |
| | | | **Ours** | 61.28 | 60.20 | 53.78 | **57.40** | 0.04 hrs + 1.17 hrs | 1.12% | 1.05% | 2.17% |
| | | 6-bit | Pruning [10] | 60.61 | 54.96 | 10.99 | 44.76 | - | 4.17% (G=8) | | |
| | | | **Ours** | 60.61 | 59.46 | 58.21 | **58.88** | 0.04 hrs + 1.62 hrs | 1.06% | 1.20% | 2.26% |
| | | 8-bit | Pruning [10] | 60.22 | 53.62 | 16.33 | 39.72 | - | 3.13% (G=8) | | |
| | | | **Ours** | 60.22 | 58.82 | 57.21 | **58.10** | 0.04 hrs + 3.01 hrs | 1.26% | 1.77% | 3.03% |




(a) (b)

**Figure 8: (a) Weight locking outperforms pruning [10] with higher resumed accuracy and lower memory overhead. (b) Average resumed accuracy and memory overhead with various protection rates $\alpha$ in TCU and accuracy drop thresholds $\eta$ in locking with 8-bit VGG8-CIFAR10.**

We also compare our method with Weight Pruning proposed in [10]. Fig. 8(a) shows the effectiveness and memory overhead of protection by Weight Pruning under different detection group sizes $G$. Weight Locking can achieve higher accuracy recovery with more than 10× reduction in memory consumption.

*4.2.5 Optimal Combination of TCU and Locking.* Fig. 8(b) presents the searching process with different combinations of $(\alpha, \eta)$. Pure unary protection will consume high memory overhead to achieve effective protection, while pure locking cannot offer comparable accuracy recovery. $(\alpha, \eta) = (0.2\%, 1)$ will give the optimal solution considering both the accuracy recovery and memory overhead.

## 4.3 Main Results

In Table 6, we compare our TCU+Locking scheme with NAT [8] (std.=0.005 weight noise injection), BAT [9], and pruning [10]. Our method can resume accuracy with only a 2% drop after BFA attacks at a marginal 3% memory overhead, significantly outperforming all prior arts. Our method is training-free, which also saves significant runtime compared to training-based methods.

## 4.4 Discussion: Can Noises Become Defender?

Besides low-bit quantization and sparsity, on-chip hardware noises are the main source of non-idealities. While noises often degrade the inference accuracy, they also hinder the attack process by adding uncertainty to loss functions or gradients. To reduce the uncertainty,

**Table 7: BFA attack performance with different samples $N_S$ on 8-bit VGG8-CIFAR10. $BS$=16.**

| Sample Times $N_S$ | Inference Budget $T_{inf}$ 60 | 120 | 240 | 400 | 480 | Mean Acc |
|---|---|---|---|---|---|---|
| 1 | **44.94** | **34.83** | **20.88** | **16.14** | **13.61** | **26.08** |
| 2 | 56.24 | 47.32 | 29.79 | 25.19 | 20.14 | 35.74 |
| 3 | 60.47 | 50.86 | 39.58 | 35.46 | 25.84 | 42.44 |

**Table 8: Performance of adversarial attacker with different on-chip Gaussian weight noise level (std.) on 8-bit VGG8-CIFAR10.**

| noise std. \ $T_{inf}$ | 0 | 40 | 180 | 400 | 600 | 800 | 900 |
|---|---|---|---|---|---|---|---|
| 0.005 | 87.58 | 52.45 | 35.18 | 18.59 | 17.02 | 15.67 | **13.52** |
| 0.01 | 86.82 | 59.68 | 34.64 | 18.42 | 16.15 | 31.88 | **21.77** |
| 0.02 | 83.81 | 54.87 | 32.12 | 21.97 | 17.21 | 34.19 | **14.48** |
| 0.03 | 74.69 | 41.06 | 25.82 | 20.74 | 17.29 | 20.45 | **18.10** |

attackers might need to average over multiple ($N_S$) samples, which equivalently reduces the attack efficiency. However, we find that sampling the noisy gradients (noise std.=0.005) once without averaging gives the best attack performance.

We increase the noise intensity in Table 8 to create higher uncertainty for the BFA attacker. Unfortunately, adding larger Gaussian noises to the weights during on-chip computations does not provide clear protection effects but leads to severe accuracy drops. It is worth investigating potential protective noise injection and the trade-offs between noise-induced error and protection effects in the future.

## 5 CONCLUSION

In this work, *for the first time*, we investigate the security issue of analog optical neural networks and present a novel nonideality-enabled built-in defender against adversarial bit-flip attacks. We introduce quantization-inspired pre-attack protection based on truncated complementary unary weight representation to minimize the weight sensitivity with optimized memory overhead. A complementary pruning-inspired weight-locking method is introduced to resume accuracy with precise error correction. Our method outperforms prior defense approaches with near-ideal accuracy recovery under bit-flip attacks with marginal (<3%) memory overhead. Our work makes significant strides toward reliable ONN against adversarial weight attacks and unlocking future applications in security-thirst scenarios.

# REFERENCES


[1] Sanmitra Banerjee, Mahdi Nikdast, Sudeep Pasricha, and Krishnendu Chakrabarty. 2022. CHAMP: Coherent hardware-aware magnitude pruning of integrated photonic neural networks. In *Optical Fiber Communication Conference (OFC)*.

[2] Sanmitra Banerjee, Mahdi Nikdast, Sudeep Pasricha, and Krishnendu Chakrabarty. 2022. Pruning coherent integrated photonic neural networks using the lottery ticket hypothesis. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*.

[3] Sanmitra Banerjee, Mahdi Nikdast, Sudeep Pasricha, and Krishnendu Chakrabarty. 2023. Pruning Coherent Integrated Photonic Neural Networks. *IEEE Journal of Selected Topics in Quantum Electronics* 29 (2023), 1–13.

[4] Q. Cheng, J. Kwon, M. Glick, M. Bahadori, L. P. Carloni, and K. Bergman. 2020. Silicon Photonics Codesign for Deep Learning. *Proc. IEEE* (2020).

[5] Johannes Feldmann, Nathan Youngblood, Maxim Karpov, Helge Gehring, Xuan Li, Maik Stappers, Manuel Le Gallo, Xin Fu, Anton Lukashchuk, Arslan Raja, Junqiu Liu, David Wright, Abu Sebastian, Tobias Kippenberg, Wolfram Pernice, and Harish Bhaskaran. 2021. Parallel convolutional processing using an integrated photonic tensor core. *Nature* (2021).

[6] Chenghao Feng, Jiaqi Gu, Hanqing Zhu, Zhoufeng Ying, Zheng Zhao, et al. 2022. A Compact Butterfly-Style Silicon Photonic–Electronic Neural Chip for Hardware-Efficient Deep Learning. *ACS Photonics* 9, 12 (2022), 3906–3916.

[7] Jiaqi Gu, Zheng Zhao, Chenghao Feng, et al. 2020. Towards Hardware-Efficient Optical Neural Networks: Beyond FFT Architecture via Joint Learnability. *IEEE TCAD* (2020).

[8] Jiaqi Gu, Zheng Zhao, Chenghao Feng, Hanqing Zhu, Ray T. Chen, and David Z. Pan. 2020. ROQ: A Noise-Aware Quantization Scheme Towards Robust Optical Neural Networks with Low-bit Controls. In *Proc. DATE*.

[9] Zhezhi He, Adnan Siraj Rakin, Jingtao Li, Chaitali Chakrabarti, and Deliang Fan. 2020. Defending and Harnessing the Bit-Flip Based Adversarial Weight Attack. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 14083–14091.

[10] Jingtao Li, Adnan Siraj Rakin, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. 2021. RADAR: Run-time Adversarial Weight Attack Detection and Accuracy Recovery. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 790–795.

[11] Jingtao Li, Adnan Siraj Rakin, Yan Xiong, Liangliang Chang, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. 2020. Defending Bit-Flip Attack through DNN Weight Reconstruction. In *Proc. DAC*.

[12] Liang Liu, Yanan Guo, Yueqiang Cheng, Youtao Zhang, and Jun Yang. 2023. Generating Robust DNN With Resistance to Bit-Flip Based Adversarial Weight Attack. *IEEE Trans. Comput.* 72, 2 (2023), 401–413.

[13] Qi Liu, Wujie Wen, and Yanzhi Wang. 2020. Concurrent Weight Encoding-based Detection for Bit-Flip Attack on Neural Network Accelerators. In *Proc. ICCAD*.

[14] W. Liu, W. Liu, Y. Ye, Q. Lou, Y. Xie, and L. Jiang. 2019. HolyLight: A Nanophotonic Accelerator for Deep Learning in Data Centers. In *Proc. DATE*.

[15] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. 2017. Fault injection attack on deep neural network. In *Proc. ICCAD*.

[16] Asif Mirza, Febin Sunny, et al. 2022. Silicon Photonic Microring Resonators: A Comprehensive Design-Space Exploration and Optimization Under Fabrication-Process Variations. *IEEE TCAD* 41, 10 (2022), 3359–3372.

[17] Sajjad Moazeni, Sen Lin, Mark Wade, Luca Alloatti, Rajeev J. Ram, Milos Popovic, and Vladimir Stojanovic. 2017. A 40-Gb/s PAM-4 Transmitter Based on a Ring-Resonator Optical DAC in 45-nm SOI CMOS. *IEEE J. Solid-State Circuits* 52, 12 (Dec. 2017), 3503–3516.

[18] A. Rakin, Z. He, and D. Fan. 2019. Bit-Flip Attack: Crushing Neural Network With Progressive Bit Search. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 1211–1220.

[19] Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. 2022. T-BFA: Targeted Bit-Flip Adversarial Weight Attack. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 11 (2022), 7928–7939.

[20] Alireza Samani, David Patel, Mathieu Chagnon, Eslam El-Fiky, Rui Li, Maxime Jacques, Nicolás Abadía, Venkat Veerasubramanian, and David V. Plant. 2017. Experimental parametric study of 128 Gb/s PAM-4 transmission system using a multi-electrode silicon photonic Mach Zehnder modulator. *Opt. Express* 25, 12 (June 2017), 13252.

[21] Bhavin J. Shastri, Alexander N. Tait, et al. 2021. Photonics for Artificial Intelligence and Neuromorphic Computing. *Nature Photonics* (2021).

[22] Yichen Shen, Nicholas C. Harris, Scott Skirlo, et al. 2017. Deep Learning with Coherent Nanophotonic Circuits. *Nature Photonics* (2017).

[23] Alexander N. Tait, Thomas Ferreira de Lima, Ellen Zhou, et al. 2017. Neuromorphic photonic networks using silicon photonic weight banks. *Sci. Rep.* (2017).

[24] Xingyuan Xu, Mengxi Tan, Bill Corcoran, Jiayang Wu, Andreas Boes, Thach G. Nguyen, Sai T. Chu, Brent E. Little, Damien G. Hicks, Roberto Morandotti, Arnan Mitchell, and David J. Moss. 2021. 11 TOPS photonic convolutional accelerator for optical neural networks. *Nature* (2021).

[25] Zhihao Xu, Tiankuang Zhou, Muzhou Ma, ChenChen Deng, Qionghai Dai, and Lu Fang. 2024. Large-scale photonic chiplet Taichi empowers 160-TOPS/W artificial general intelligence. *Science* 384, 6692 (2024), 202–209.

[26] Ziang Yin, Nicholas Gangi, Meng Zhang, Jeff Zhang, Rena Huang, and Jiaqi Gu. 2024. SCATTER: Algorithm-Circuit Co-Sparse Photonic Accelerator with Thermal-Tolerant, Power-Efficient In-situ Light Redistribution. In *Proc. ICCAD*.

[27] Zheng Zhao, Jiaqi Gu, Zhoufeng Ying, et al. 2019. Design Technology for Scalable and Robust Photonic Integrated Circuits. In *Proc. ICCAD*.

[28] Hanqing Zhu, Jiaqi Gu, Hanrui Wang, et al. 2024. Lightening-Transformer: A Dynamically-Operated Optically-Interconnected Photonic Transformer Accelerator. In *Proc. HPCA*. 686–703.

[29] H.H. Zhu, J. Zou, H. Zhang, et al. 2022. Space-efficient optical computing with an integrated chip diffractive neural network. *Nature Commun.* (2022).

[30] Ying Zhu, Grace Li Zhang, Bing Li, et al. 2020. Countering Variations and Thermal Effects for Accurate Optical Neural Networks. In *Proc. ICCAD*.