The Unlikely Hero: Nonideality in Analog Photonic Neural Networks as Built-in Defender Against Adversarial Attacks

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

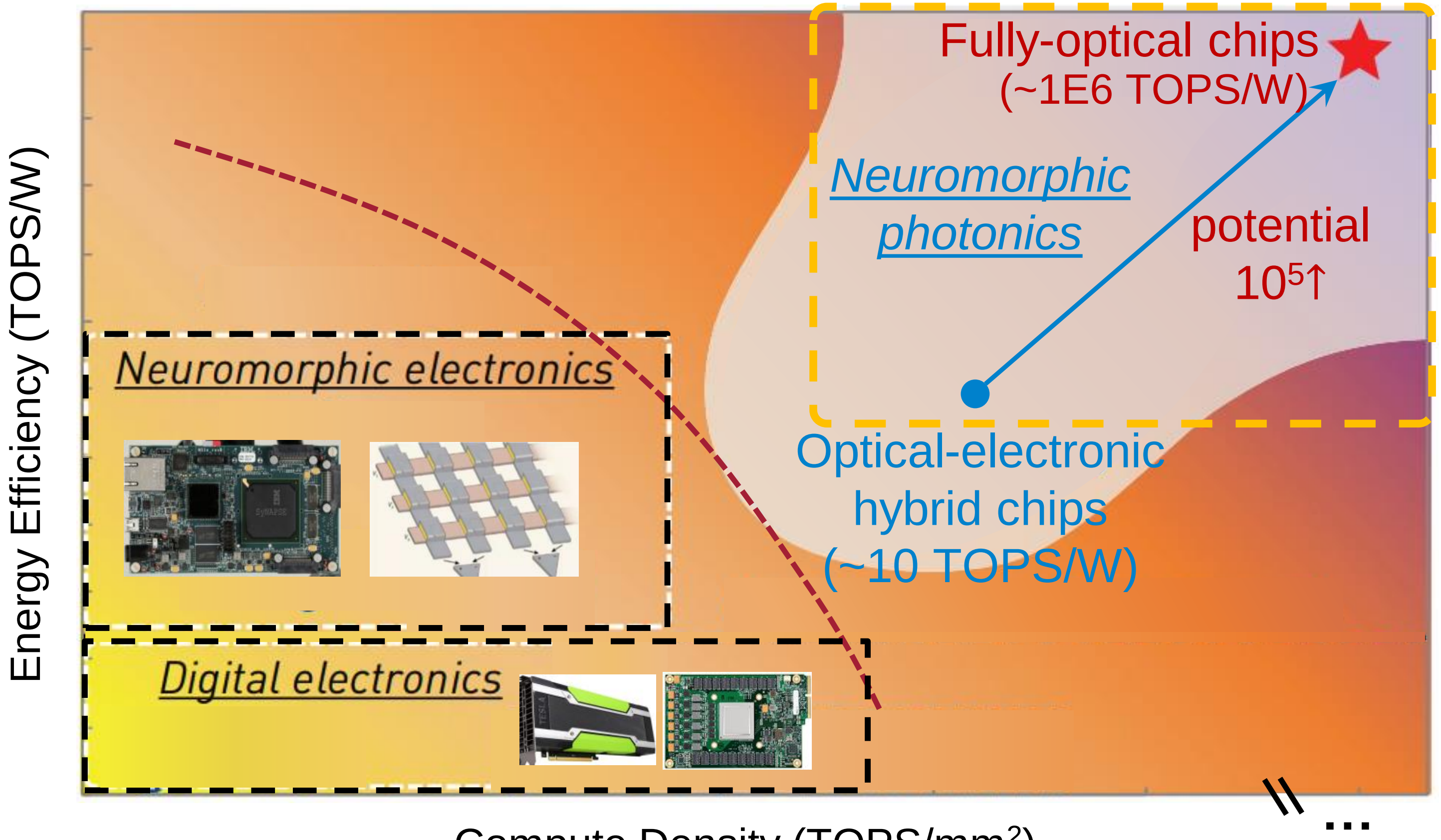
[1]Arizona State University
School of Electrical, Computer and Energy Engineering
jiaqigu@asu.edu | scopex-asu.github.io

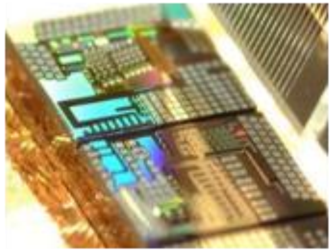
March 7, 2025

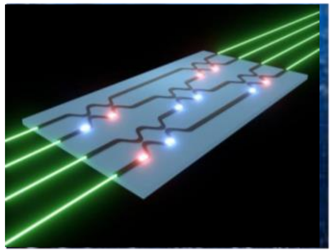
Photonic ML Accelerators

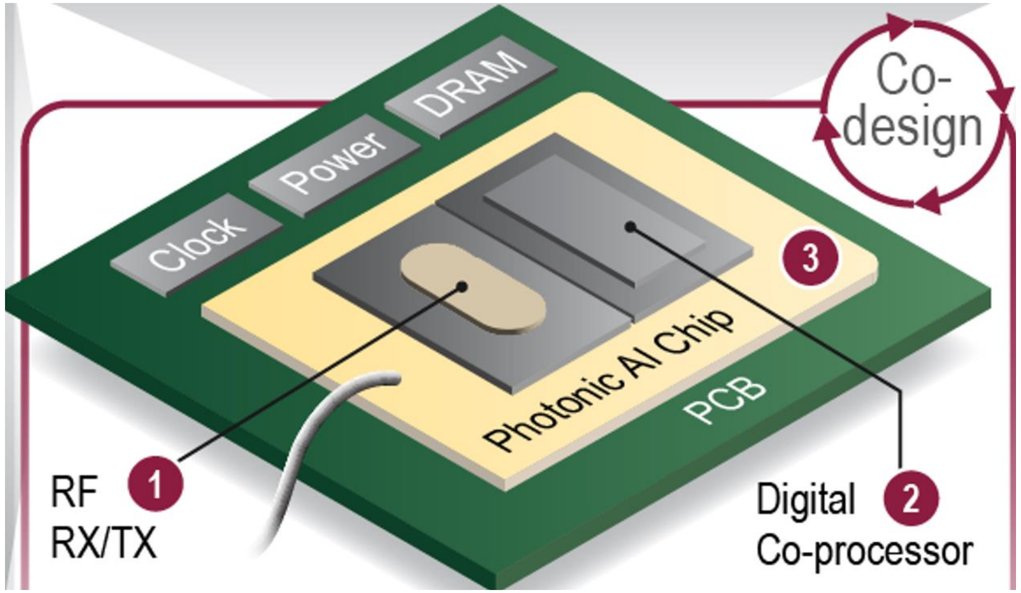
- Evolve from electronics to *heterogenous electronics-photonics*

Speed-of-light, Massive parallelism, low-power





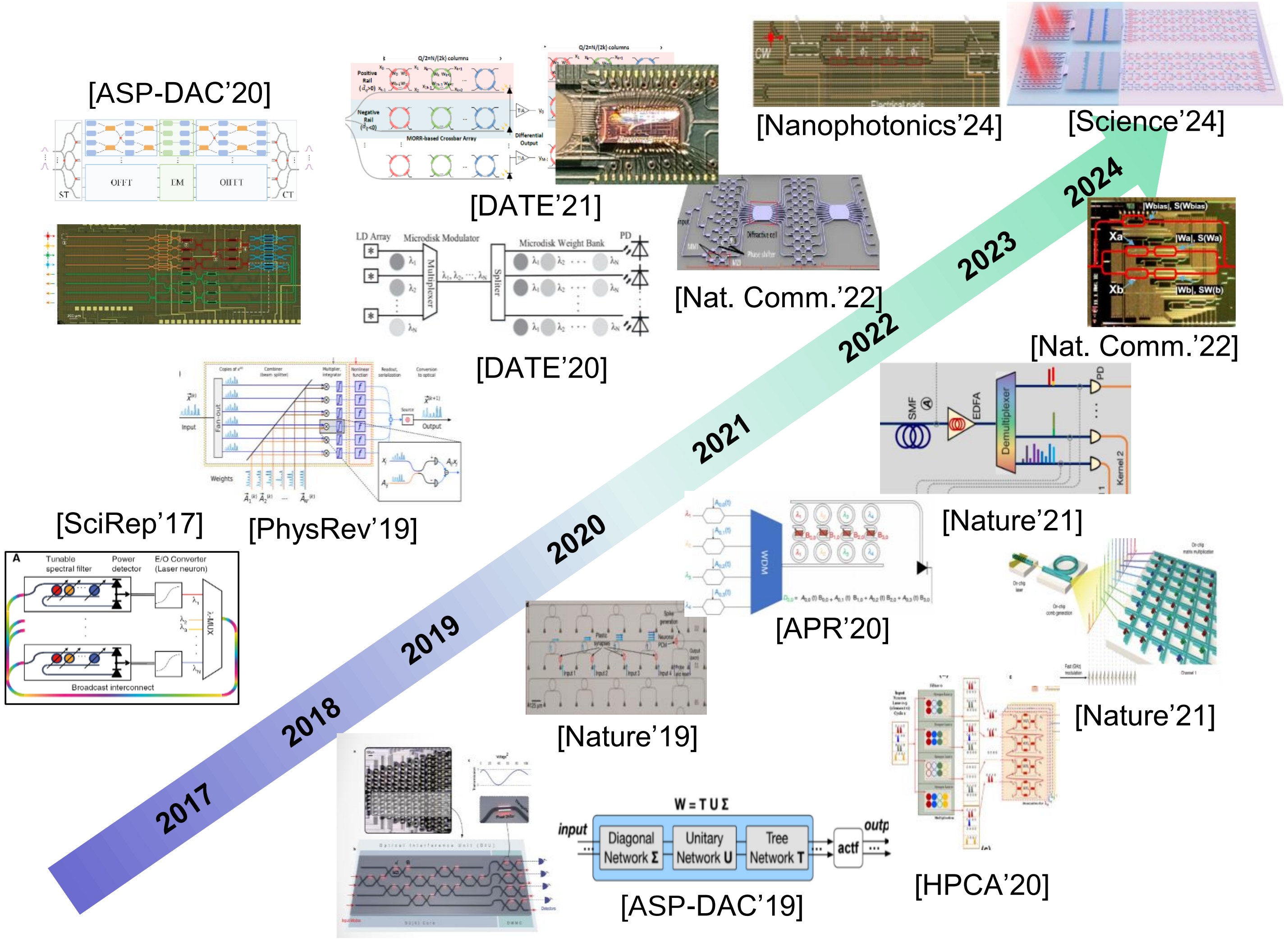




Source: Mitchell A. Nahmias, Bhavin J. Shastri, Alexander N. Tait, Thomas Ferreira de Lima and Paul R. Prucnal, "Neuromorphic photonics," Optics & Photonics News, Jan 2018.

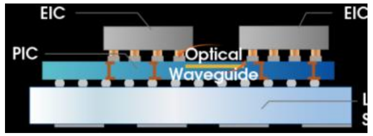
Photonic AI System is Booming

Photonic Neural Network Trends in Academia



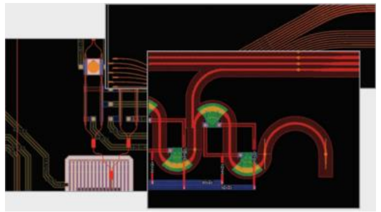
Foundry / EPDA Support in Industry

Photonic **Computing** Chip + Optical **Interconnects**



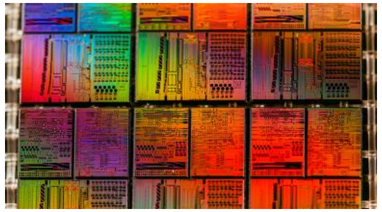
LIGHTMATTER
Lighton
XANADU
LIGHTELLIGENCE
OPTELLIGENCE
iPRONICS Programmable Photonics
OPTIUS light powered computing
SALIENCE LABS

Electronic-Photonic Design Automation Tools



Ansys LUMERICAL
SYNOPSYS PHOTONIC SOLUTIONS
cadence PHOTONICS
LUCEDA

PDK / Tape-out / HI / E-O Co-Packaging Support



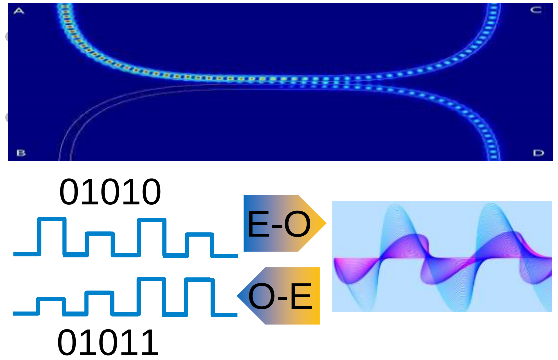
amf ADVANCED MICRO FOUNDRY
AIM photonics
SiEPIC
GlobalFoundries
tsmc
Tower Semiconductor

Gaps in Electronic-Photonic AI Eco-systems

- EPIC AI ecosystem is in *early* stage, many new challenges

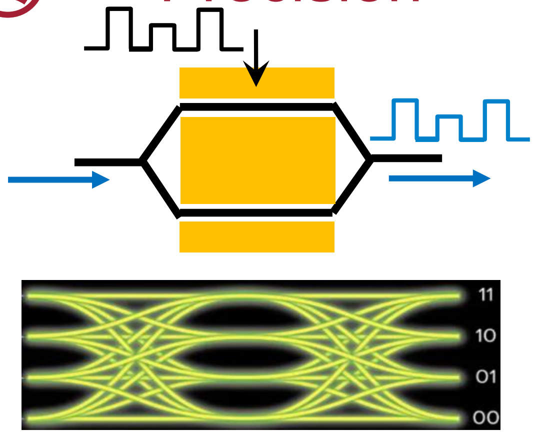
Area/E-O Cost

~40×200 μm^2



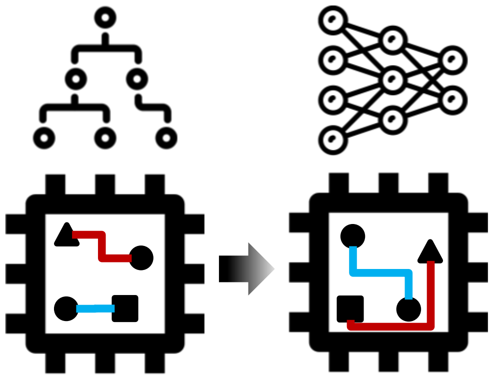
Large spatial footprint
E-O/A-D conversion

Precision



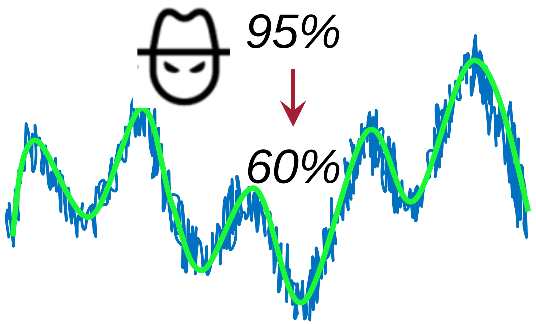
Low **precision** in encoding

Reconfigurability



Lack of **versatility** for diverse workloads

Reliability



Variations + **Attacks**
Accuracy loss

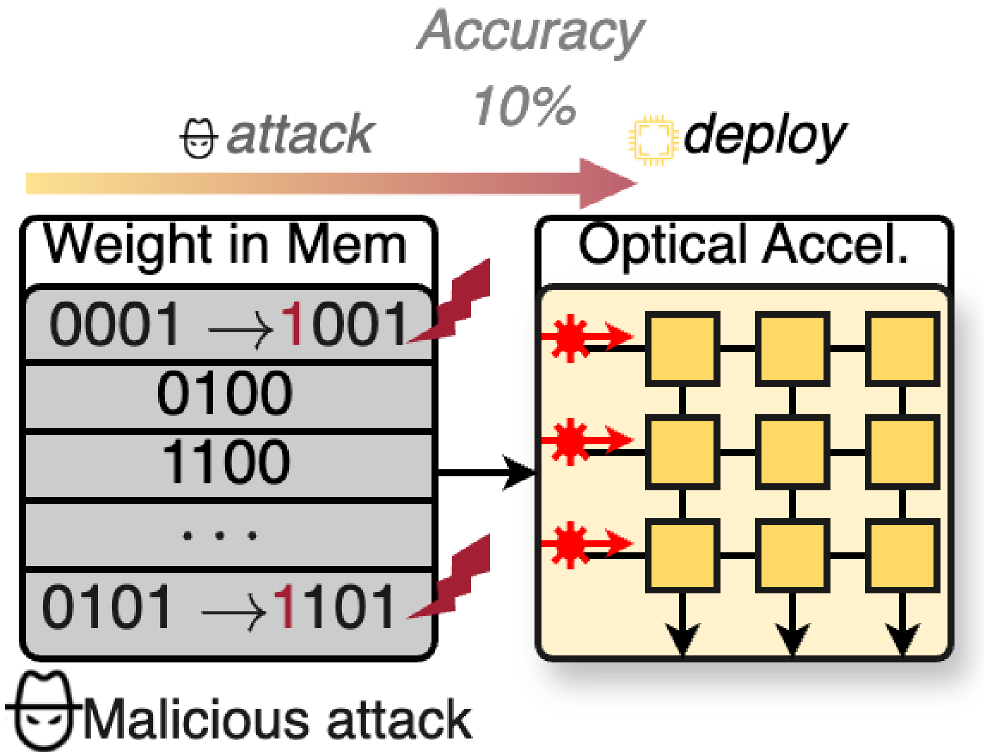
Our concern in this paper

??

Reliability is Severely Challenged by Attack

- **Security** problem is under-explored for AMS photonic AI hardware
- Serious reliability concerns with **<u>two</u>** enemies?

Malicious attack + Hardware non-ideality



Bit-flip Attack in Photonic AI Hardware

- Bit-flip (PBFA) [Rakin+, ICCV'19] poses great threat to Photonic AI HW
- White-box attack, arbitrary bit-flip available in model weights
- **First-order Gradient-based**, **Progressive**
  - Search the most vulnerable bit index to flip / iteration → largest acc. drop & loss

$$\min_{I_A} Acc\left(\widehat{W_{I_A}}; D^{test}\right) \approx \max_{I_A} \mathcal{L}\left(\widehat{W_{I_A}}; D^{attack}\right)$$
$$s.t. \left\|\widehat{W_{I_A}} - W\right\|_1 \leq HD;\ \#\ \text{model inf.} \leq T_{inf}$$

- *Hamming Distance* (HD) and *Inference Budget* (T_{inf}) constraint
- Hint: Attacks happen on **MSB mostly**

Threat Model

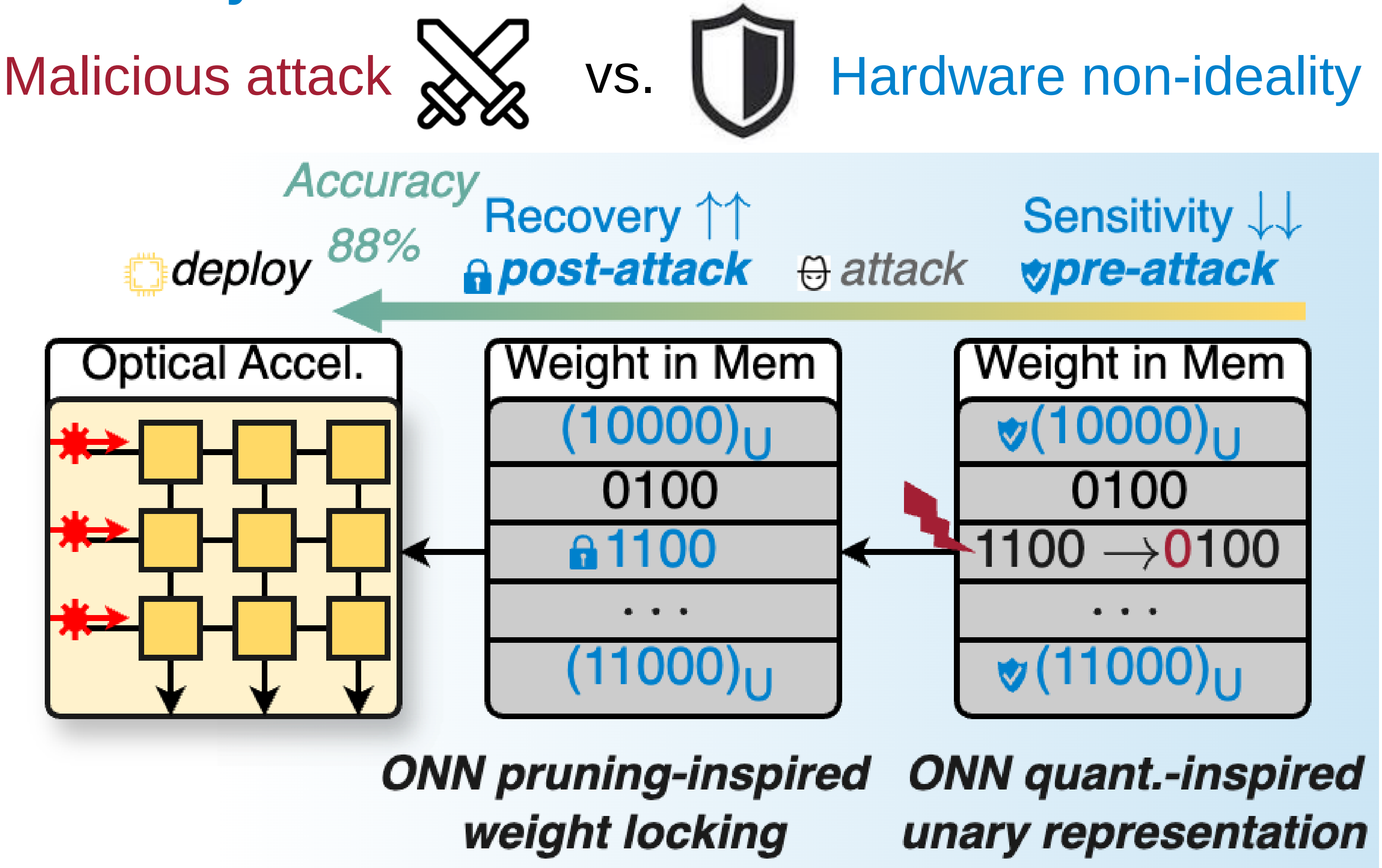
	Access Required	Access NOT Requied
	DNN model and parameters	Training Configurations
D^{attack} ←	A mini-batch of attack dataset	Modify scaling factors in quantization & Norm.
	On-chip forward/backward prop.	Modify address mapping/look-up tables

Prior Defense Methods for Photonic AI HW

<table>
<tr><th>Existing Defense v.s. Attack</th><th>NAT [Gu+,DATE’20]</th><th>BAT [He+, CVPR’20]</th><th>Pruning [Li+, DATE’21]</th><th>Common Challenges</th></tr>
<tr><td>Require Training?</td><td>Training-based</td><td>Training-based</td><td>Training-free</td><td rowspan="4">Either Pre- or Post-attack protection
Lack of effective while efficient defense framework targeted on photonic AI hardware
Novel defense method needs: <u>Training-free</u> + <u>Pre & Post Protection</u> + <u>High Mem. Efficiency</u> → <u>High Acc. Recovery</u></td></tr>
<tr><td>Occurance</td><td>Pre-attack</td><td>Pre-attack</td><td>Post-attack</td></tr>
<tr><td>Mem. Overhead</td><td>0</td><td>0</td><td>Relatively High</td></tr>
<tr><td>Recovery Performance</td><td>Low</td><td>Relatively High</td><td>Relatively High</td></tr>
</table>

Analog AI Accel. Nonideality: Double-edged Sword

- **Security** problem is un-explored for photonic AI hardware
- Insight: **Hardware nonideality can be built-in defender**
- Nonideality:
  - Quantization
  - Sparsity
  - On-chip Noise
  - etc …



Proposed Synergistic Defense Framework

1. Quantize-inspired Pre-attack Defense

- Protect via optics-specific *encoding*
- Memory efficiency optimization

+

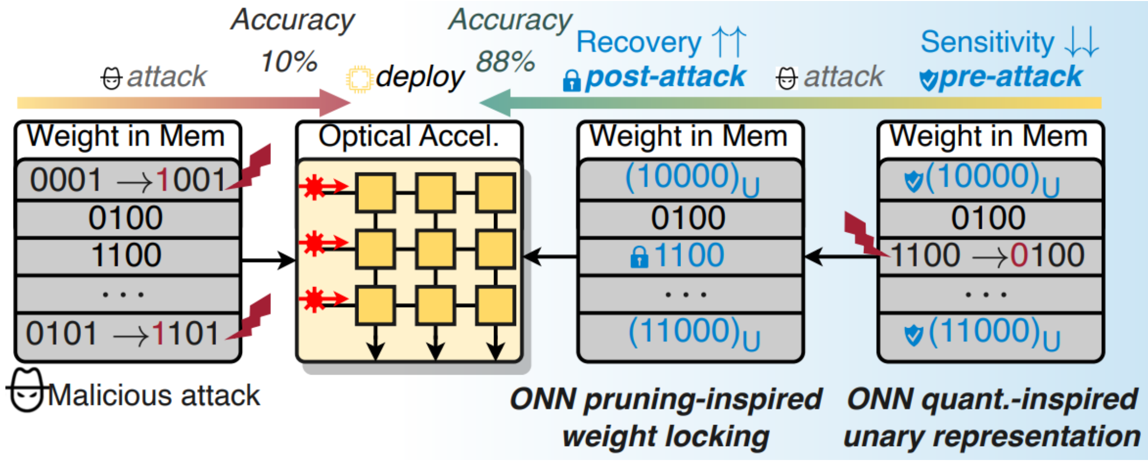
2. Prune-inspired Post-attack Recovery

- Efficient detection of bit-flipped weights
- Error correction via *weight locking*

=

Full Protection

- near-ideal acc. recovery (<2% drop)
- marginal memory cost (~2% ovhd)



Proposed Synergistic Defense Framework

1. Quantize-inspired Pre-attack Defense

- Protect via optics-specific *encoding*
- Memory efficiency optimization

Full Protection

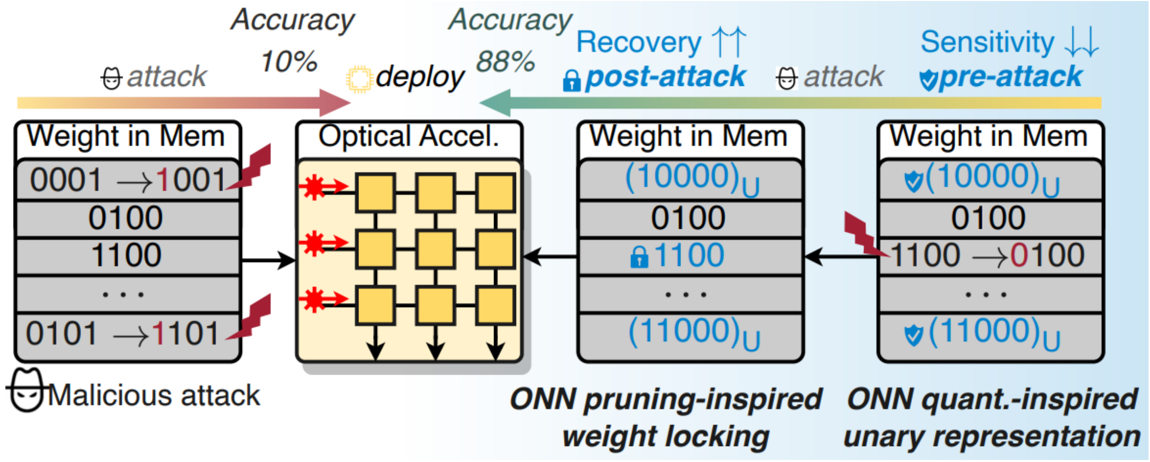
- near-ideal acc. recovery (<2% drop)
- marginal memory cost (~2% ovhd)

+

=

2. Prune-inspired Post-attack Recovery

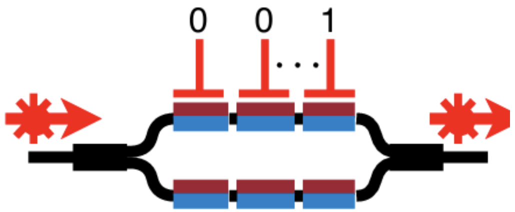
- Efficient detection of bit-flipped weights
- Error correction via *weight locking*

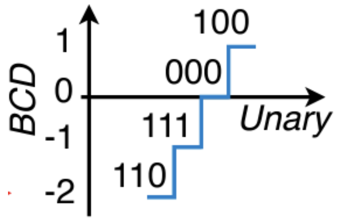


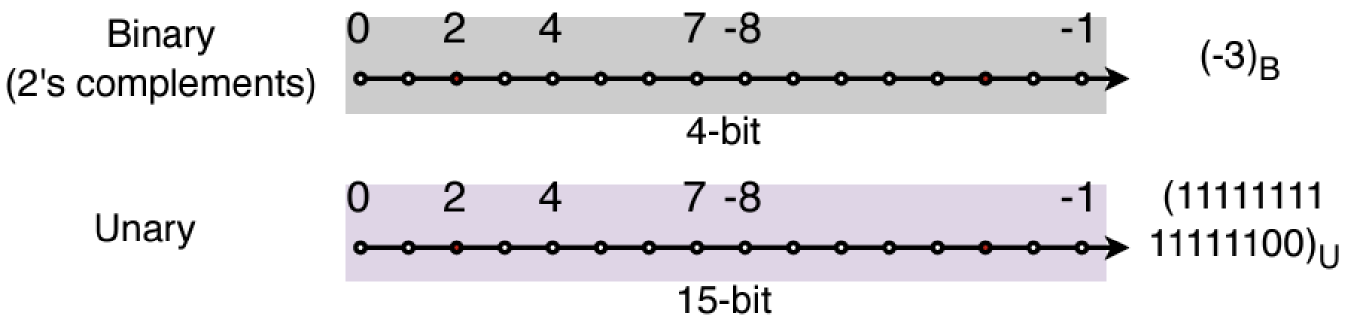
Minimize Weight Sensitivity by Unary Represent.

- **Electro-optic DAC**: unary encode → min sensitivity (LSB) → built-in defender

b bit BCD number W_B → $2^b - 1$ bit Unary number W_U $\boldsymbol{W_B = \#\ \mathbf{1s\ in\ W_U}}$: No MSB in W_U, **all LSB**







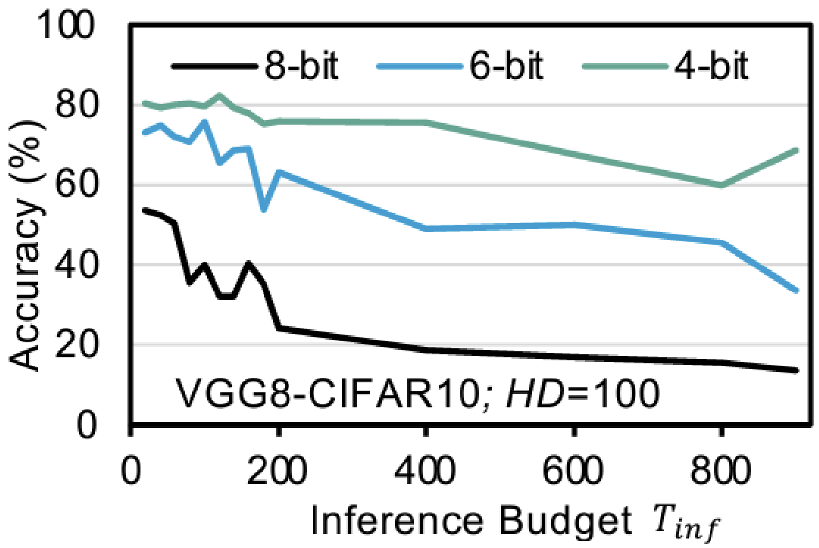
- **Exponential memory overhead...**

$$Mem_{OV} = \frac{2^b - 1}{b} \quad \longleftarrow \quad \approx 32 \times \text{OVHD for } b = 8\text{-bit}$$

How to reduce the memory overhead required by Unary Representation?

Memory-Efficient Unary Enc.: Low-bit Quant

- **Sol 1: Low-bit quantized model**
- Low-bit models are robust against attack
- Trade-off among (*mem-efficiency*, *robustness*, *expressivity*)

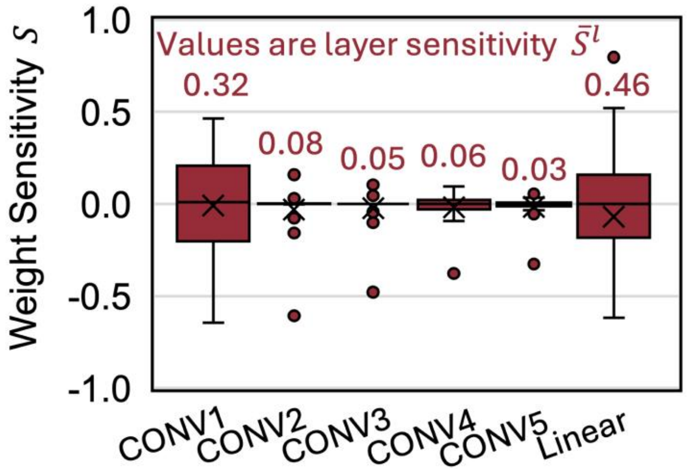


Memory-Efficient Unary Enc.: Vulnerable Weights

- **Sol 2: Only protect vulnerable weights**
- How to <u>identify</u> vulnerable weights?
  - Weight sensitivity represented by second-order Taylor Expansion
  - **Bitflip-injection** during sensitive weight search

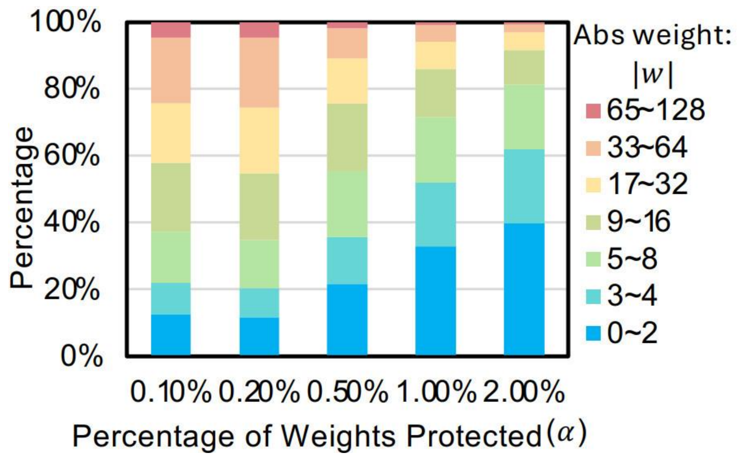
$$S = \nabla_W \mathcal{L} \cdot \Delta W_{MSB} + \frac{1}{2} \nabla_W^2 \mathcal{L} \cdot \Delta W_{MSB}^2$$

- How to <u>assign</u> limited memory budget?
  - **Uneven** sensitivity distribution in layers
  - **Top-Sensitive-Layer Assignment** for given mem. budget α: <u>Fill most sensitive first!</u>



Memory-Efficient Unary Enc.: Fold & Truncation

- **Sol 3: Fold & Truncate the encoding**
- Observation: Sensitive weights have small abs values (**Gaussian-like** Distribution)
  - *Waste* to store trailing 0s

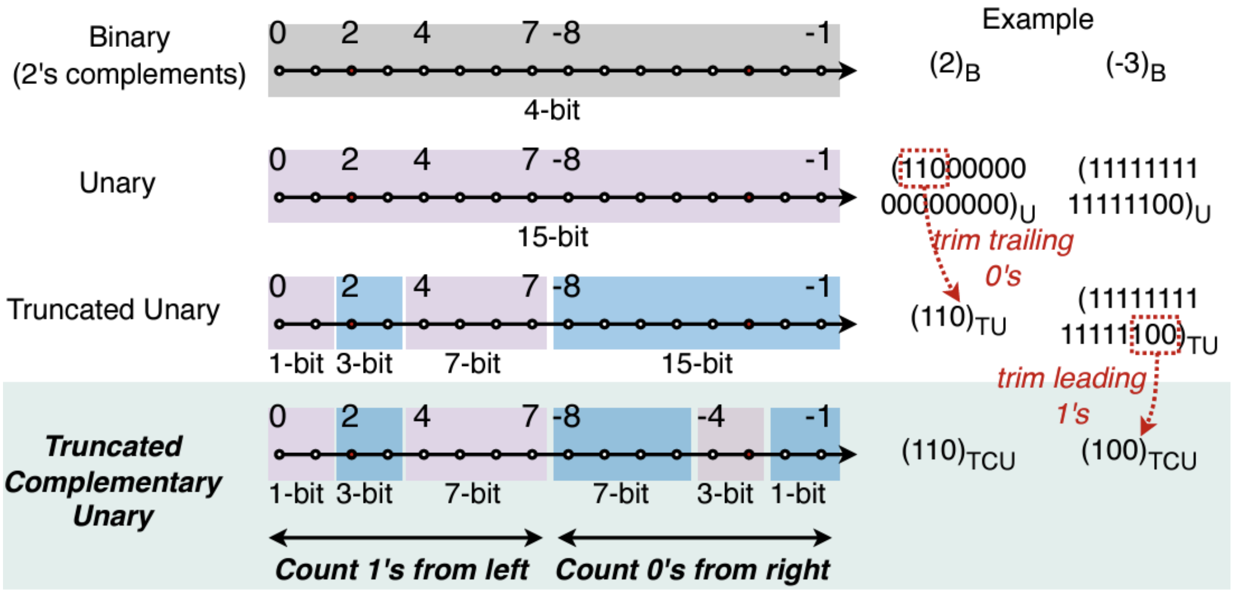


- **Truncate** unnecessary 0s to bins (TU)

  $(W)_U$: $2^b - 1$ bit → $(W)_{TU}$: $2^{\lceil \log_2 |W| \rceil} - 1$ bit
  - Negative values still take large #bits

- **Fold** symmetric encoding (TCU)
  - pos.: count 1's; neg.: count 0's
  - Truncated **Complementary** Unary

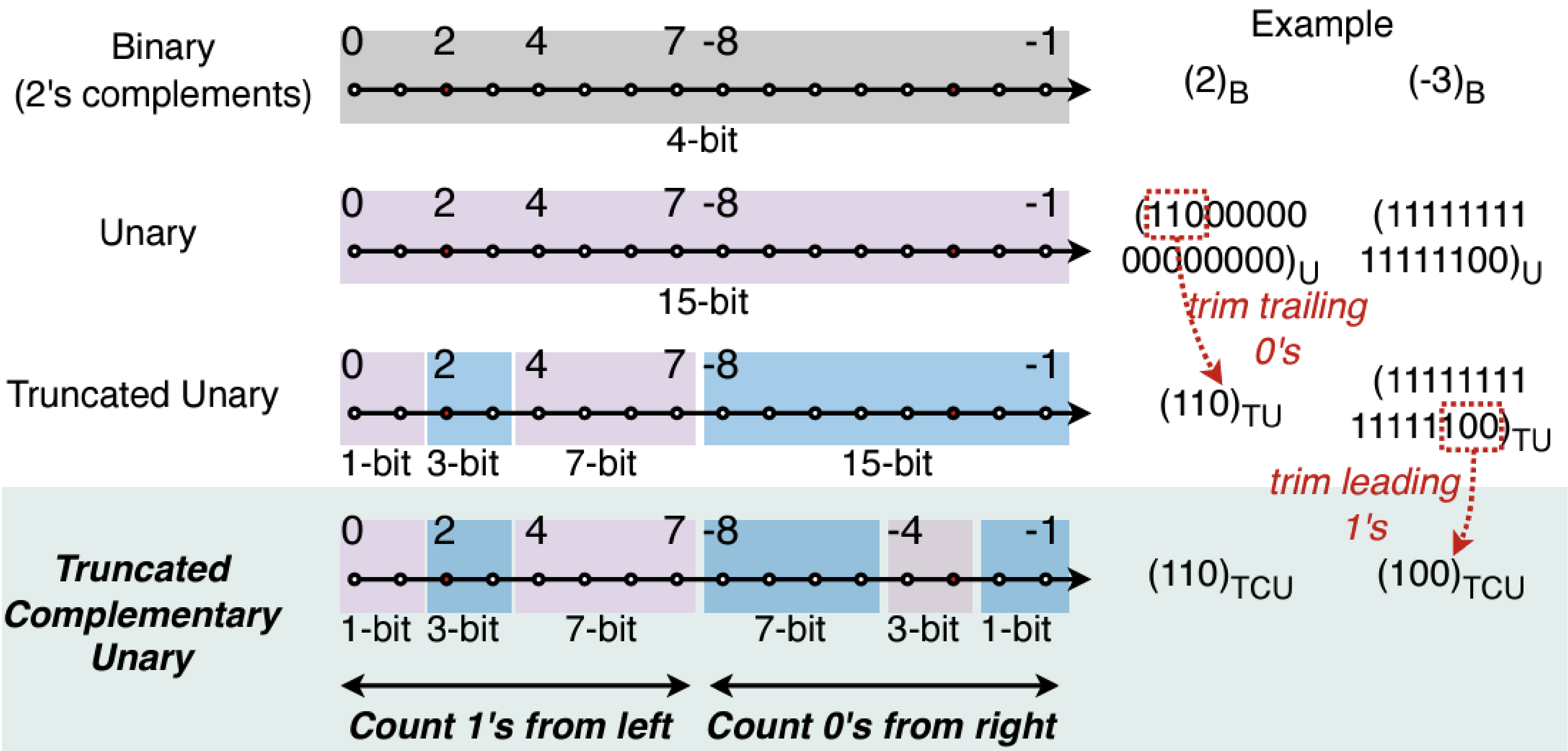
$(W)_U$: $2^b - 1$ bit → $(W)_{TCU}$: $2^{\lceil \log_2 \min\{|W|, |2^b - W|\} \rceil} - 1$ bit



Quantization-Inspired: Pre-deploy Protection

- Truncated complementary unary (TCU) + protect ***vulnerable*** weights
- Attack-injected Weight Protection Search for ***vulnerable*** weights

  → mem.-efficient & secure



Pre-deploy provides ***insufficient target-less*** protection.

How to compensate for potential protection miss?

Proposed Synergistic Defense Framework

1. Quantize-inspired Pre-attack Defense

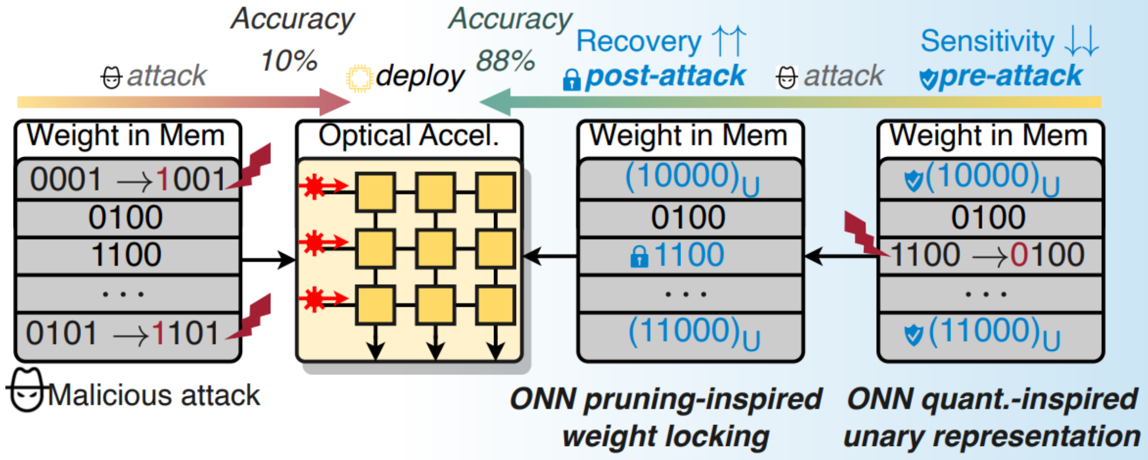
- Protect via optics-specific *encoding*
- Memory efficiency optimization

2. Prune-inspired Post-attack Recovery

- Efficient detection of bit-flipped weights
- Error correction via *weight locking*

Full Protection

- near-ideal acc. recovery (<2% drop)
- marginal memory cost (~2% ovhd)

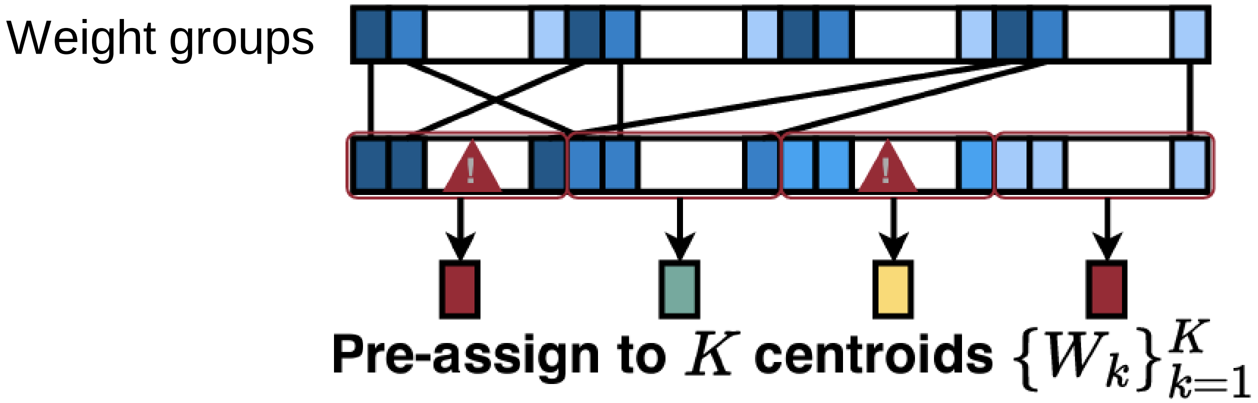


Group-based Detection: Checksum

- Post-deploy protection: detect → correct
- **Detection** of Attacked Weights
  - Interleaving weight group, layer-wise
  - MSB checksum verification [Li+, DATE'21]
  - 2-bit checksum for a group of G Weights

    → **Pinpoint MSB-targeted attacks with high coverage**

    (might miss attacked weights; cannot localize specific weight in a group)
- How to correct detected weights?
  - No access to original values anymore…
  - → assign a value to wipe out attacks (prior work prunes it to 0, not good, like self-attack…)



Which values should we assign?
(Trade off accuracy vs. robustness)

Preparation for Group-based Weight Recovery

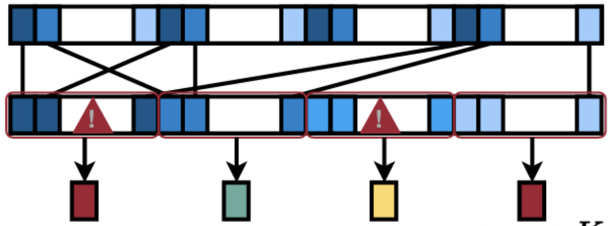
- Smart values to assign: group centroid
- Sensitivity-aware cluster centroids
  - Total **K** clusters in $\widetilde{W}$ (cluster size: **G**)
  - K-Means clustering $\{\widetilde{W}\}$ → K centroids $\{\widetilde{W}_K\}$
  - Assign one centroid $\widetilde{W}$ → each group

- How to make it attack-aware?
  - Sensitivity-aware distance in K-means

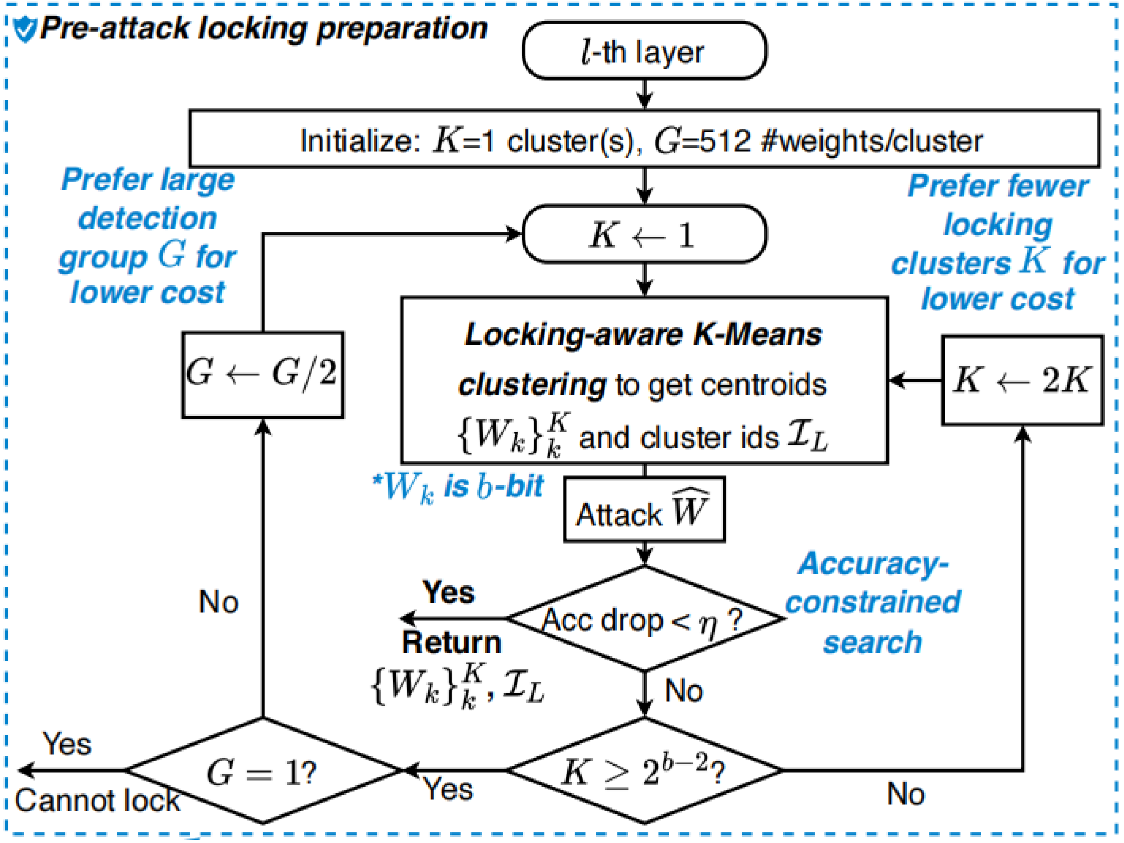
$$\min_{\widetilde{W}} \sum_{i=1}^{G} d_i = \sum_{i=1}^{G} \nabla_W \mathcal{L} \cdot (W_i - \widetilde{W}) + 0.5 \cdot \nabla_W^2 \mathcal{L} \cdot (W_i - \widetilde{W})^2$$

  - Attack injection to evaluate post-attack acc.

- How to make it memory-efficient?
  - Prefer **larger G** and **smaller K** for lower cost



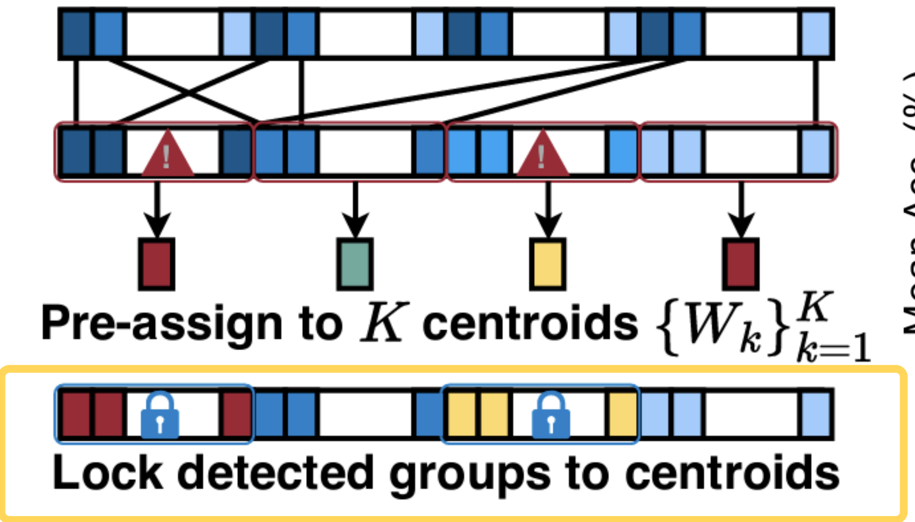
Pre-assign to K centroids $\{W_k\}_{k=1}^{K}$

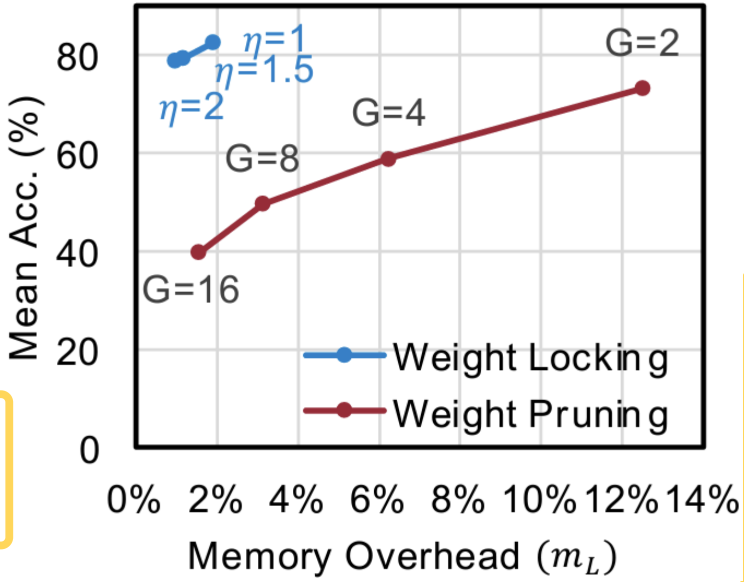


Proposed Weights Grouping and Vulnerability-aware Clustering

Pruning-Inspired: Post-deploy Recovery

- **Pruning-Inspired** protection → weight locking
- Smartly group weights and lock to centroids (vulnerability-aware K-Means)
  → ***wipe*** attacked weights & maintain acc & mem-efficient





Locking provides less "self-attack" compared with Pruning
→ ***Larger G, less mem. overhead***

Insight
- "Pruning" wipes out attack
- Locking generalizes pruning
- Low overhead but w/ acc cost

Synergistic Pre-/Post-Deploy Protection

Quantize-inspired unary encoding (eoDAC)
→ min sensitivity (LSB) → built-in defender

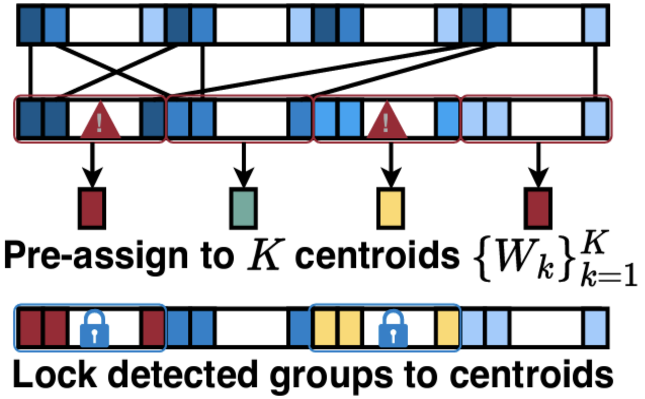
Binary (2's complements)	$(2)_B$	$(-3)_B$
Truncated Complementary Unary	$(110)_{TCU}$	$(100)_{TCU}$

Insight

- Low-bit model is more robust to attack
- Statistics-aware unary encoding balances *memory vs. security*

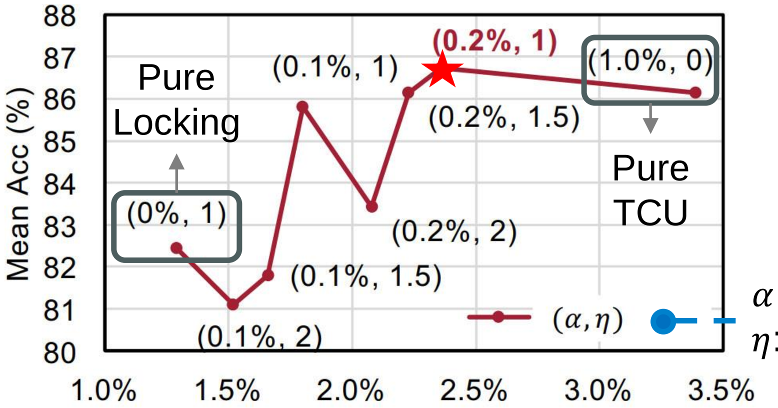
✚

Pruning-inspired weight locking
→ group&lock attacked weights to centroids



Insight

- "Locking" wipes out attack
- Low mem. overhead at small acc. drop



α: Protect. ratio of TCU
η: Accep. Acc. Drop of Locking

★ ***Optimal Mem. budget allocation***

Large mem. for TCU +
small mem. for Weight Locking

Train-Free Memory-Efficient Build-in Defender

- Prior methods (Noisy train/Quantize/Prune): only 25~80% acc w/ 3 hr train cost
- **Our method: 83~86.7% acc @ 2% memory overhead w/ <1 hr search**
- Provide **Near-ideal Accuracy Recovery** with **Marginal** Memory Budget

Category	Quant. Bit	Defense Method	Prior-attack Accuracy	Worst Acc.	**Mean Acc.**	Training/Searching Runtime	Memory Overhead Pre (m_{TCU})	Post (m_L)	Total
w/o Def	4-bit	-	87.73	59.87	75.85	-			
	6-bit	-	88.00	33.58	61.74				
	8-bit	-	88.00	13.52	32.89				
Training-based	1-bit	BAT [19]	87.09	74.62	80.39	0.33 hrs	-		
	4-bit	NAT [10]	87.96	66.71	77.06	2.8 hrs			
	6-bit	NAT [10]	87.19	33.39	64.78	2.8 hrs			
	8-bit	NAT [10]	85.91	26.14	55.88	2.8 hrs			
Training-free	4-bit	Pruning [19]	87.73	57.23	70.68	-	3.13% (G=16)		
		Ours	87.73	83.08	**84.74**	0.03hrs + 0.33 hrs	0.84%	0.000%	0.84%
	6-bit	Pruning [19]	88.00	40.74	66.14	-	4.17% (G=8)		
		Ours	88.00	86.25	**86.48**	0.03hrs + 0.50 hrs	0.93%	1.11%	2.04%
	8-bit	Pruning [19]	88.00	18.68	48.59	-	3.13% (G=8)		
		Ours	88.00	86.08	**86.73**	0.03hrs + 0.75 hrs	1.07%	1.29%	2.36%

Model: VGG8, Dataset: Cifar-10

Open-Source
ONN Defender

ONN Defender against Adversarial Attacks

Thank you!
Q & A?

The Unlikely Hero: Nonideality in Analog Photonic Neural Networks as Built-in Defender Against Adversarial Attacks

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

arXiv Preprint

Arizona State University

ASU Center for Semiconductor Microelectronics (ACME)

SCOPE