Evaluation of Parameter-based Attacks against Embedded Neural Networks with Laser Injection

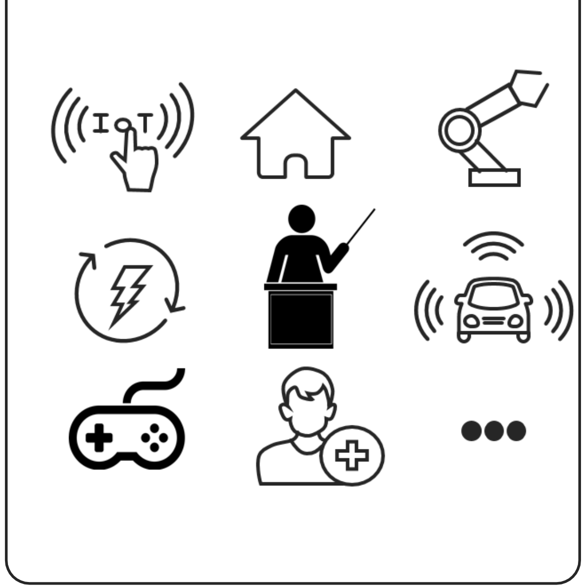
M. Dumont*, K. Hector*, P-A. Moellic*,

J-M. Dutertre+, S. Pontié*

*** CEA LETI, + Mines Saint-Etienne***

Context

Model Deployment



Security

Regulation

Ubiquitous AI
ML models everywhere…

A founding principle of
***Trustworthy* AI**

European **AI Act:**
upcoming security certification actions

Context

AI grows too fast without safety and security concerns

- Lot of activities related to *Cybersecurity of AI & Standardization*
- GDPR, AI Act, Cyber Res. Act, NIS2, Cyber Act…
- ➔ ENISA reports focused on Cybersecurity of AI Systems

AI system certification: critical challenges

- ❖ Urgent needs to develop robust evaluation protocols
- ❖ **practical** evaluations

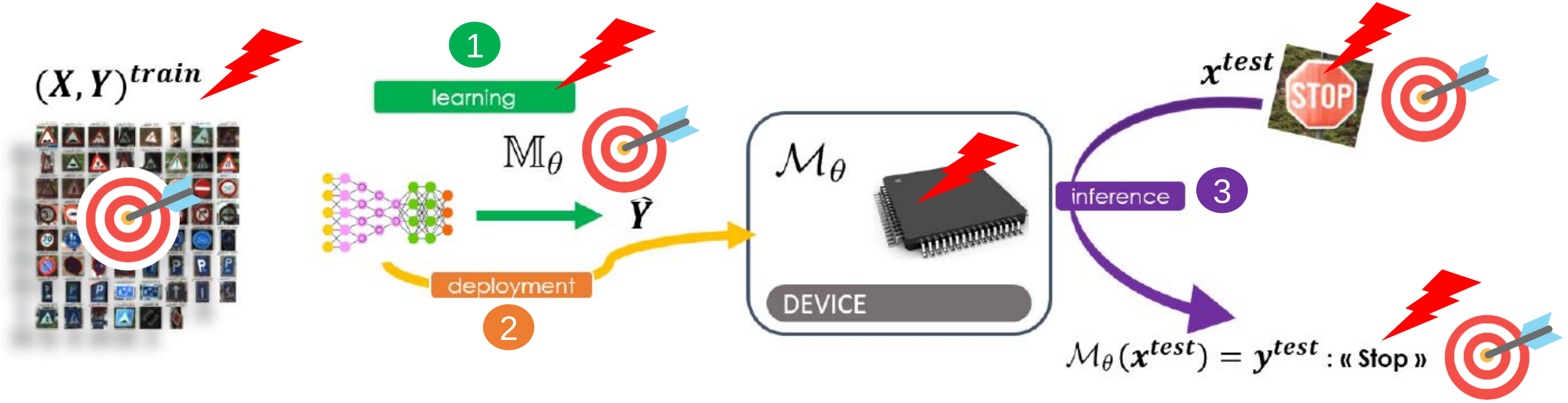
Regulation

European **AI Act** & Cybersecurity-based regulatory frameworks
upcoming security certification actions

Security of Machine Learning

Adversarial & Privacy-Preserving Machine Learning

State-of-the-Art: attacks everywhere, everything



Confidentiality / Integrity / Availability

Security of Machine Learning

A Complex ATTACK SURFACE

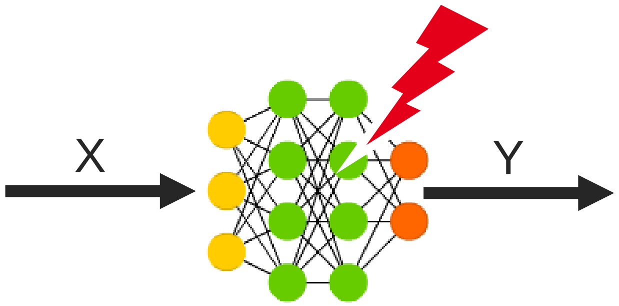
OUR CLAIM

A model is not *just* an abstraction

ATTACK SURFACE

ALGORITHM / ABSTRACTION

API-based Attacks
White-Box / Black-Box



Security of Machine Learning

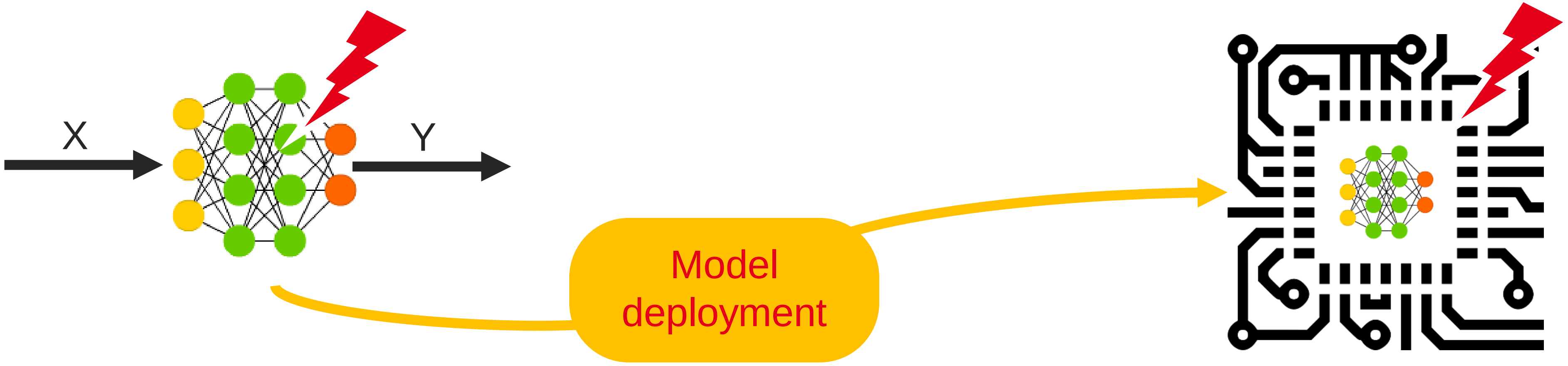
A Complex ATTACK SURFACE

OUR CLAIM

A model is not *just* an abstraction ➔ SW / HW implementations

ATTACK SURFACE

ALGORITHM / ABSTRACTION	IMPLEMENTATION / PHYSICAL
API-based Attacks White-Box / Black-Box	Implementation-based Attacks Physical Attacks (side-channel, fault injection)

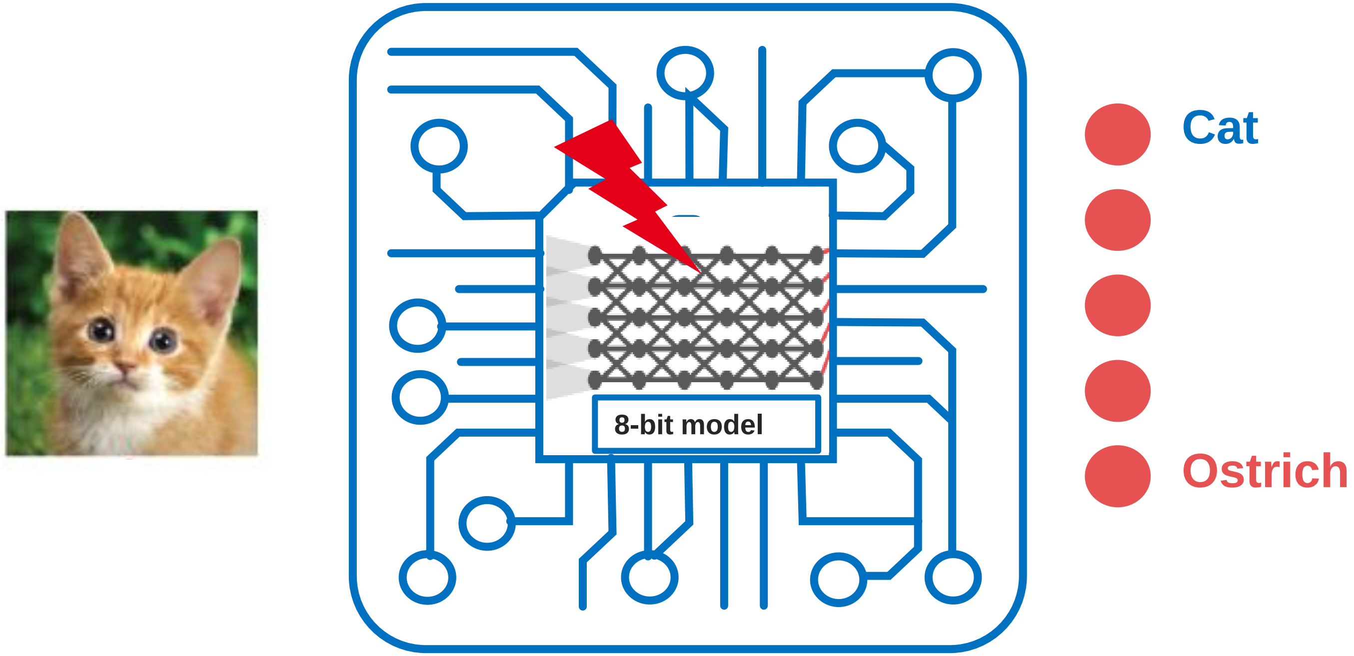


Background & Positioning

Weight-based Adversarial Attacks

Target internal parameters stored in memory

- Deep Neural Network parameters: quantified and stored in memory (e.g., DRAM, Flash)
- Fault Injection Attacks: precisely alter the value of a parameter ➔ bit-level



Weight-based Adversarial Attacks

Target internal parameters stored in memory

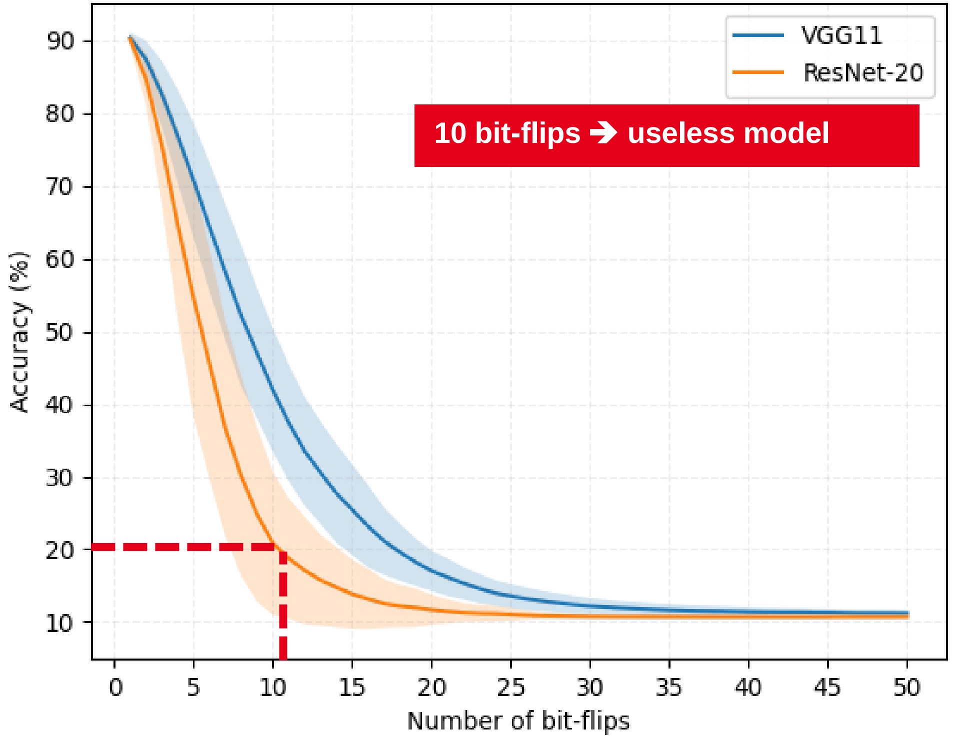
- Main reference: Bit-Flip Attack – **BFA**[1]
  - First demonstration: RowHammer[2] attack (CPU, **DRAM**)
  - Former works[3] on evaluating BFA

- Safety analysis ➔ Random bit-flips
- BFA = **Adversarial** bit-flips ➔ Faults on the most sensitive parameters

$$\underbrace{\max_{W'} \sum_{i=0}^{N-1} \mathcal{L}\Big(M(x_i; W'), y_i\Big)}_{\text{mispredictions}} \quad \text{s.t.} \quad \overbrace{HD(W', W) \leq S}^{\text{adv budget}}$$

No more than S bit-flips

Gradient-based ranking of *w*
$\nabla_w \mathcal{L}$



(1) Rakin et al., ***Bit-flip attack: Crushing neural network…*** IEEE/CVF ICCV 2019
(2) Yao, et al. ***DeepHammer…*** USENIX 2020
(3) Hector *et al.*, ***A closer look at evaluating the BFA…*** IEEE IOLTS 2022

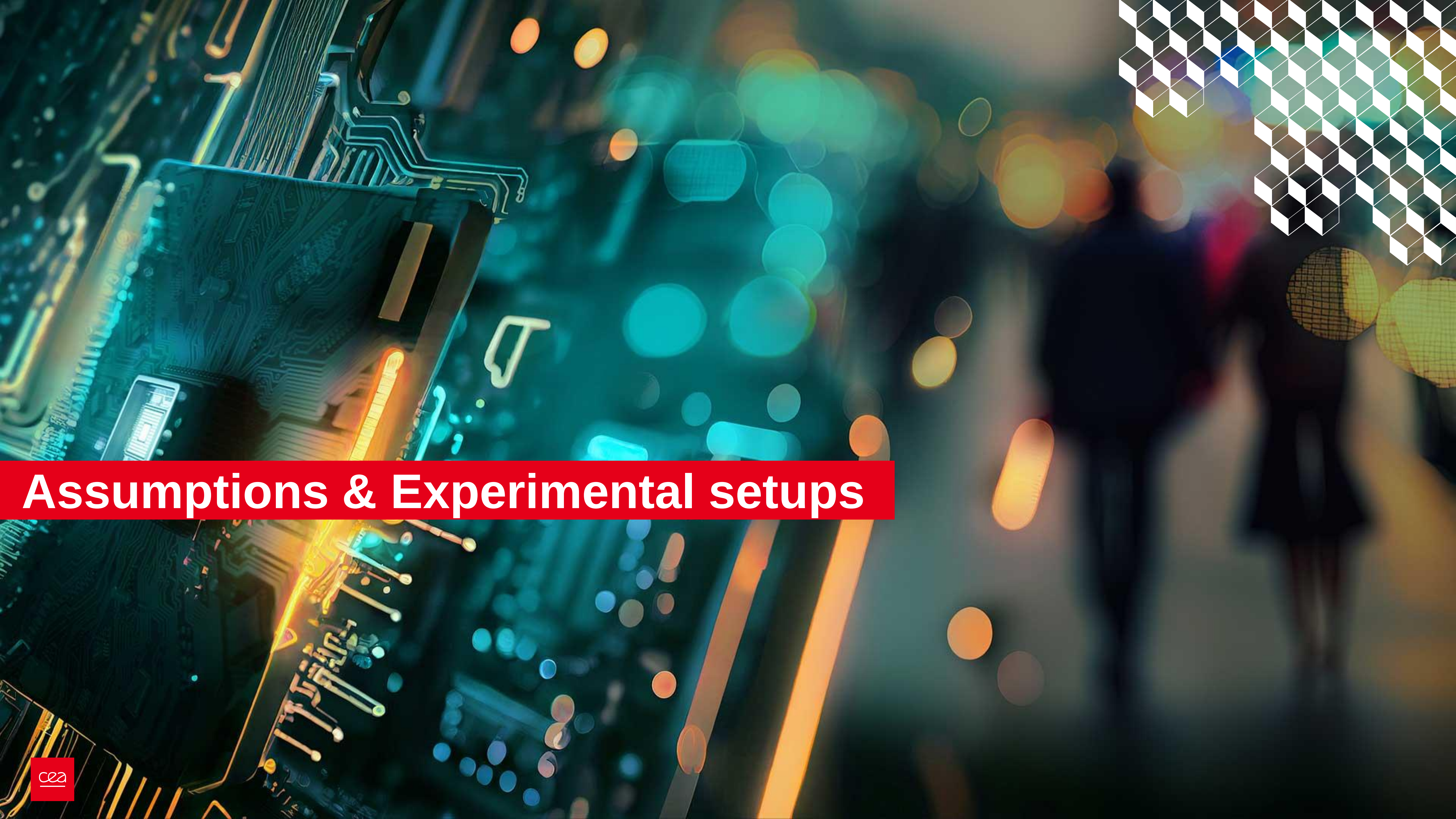
Positioning

OUR SCOPE

- Security evaluation and characterization context ➔ security evaluator point of view
- Parameter-based threats for NN embedded in 32-bit MCU, Cortex M.
  - e.g., widely used in IoT applications
  - **Flash memory ➔ other fault model**
- Laser Fault Injection (**LFI**)
  - Advanced, very spatially and temporally accurate injection means
  - reference technique for many HW security evaluation centers

- State of the Art
  - Most efforts rely on simulation only
  - Practical exp: RowHammer attacks (CPU, **DRAM**)
  - Very few and partial works on LFI on MCU against embedded DNN[1]

(1) Hou, et al. Security Evaluation of Deep Neural Network Resistance against Laser Fault Injection, IPFA 2020



Assumptions & Experimental setups

Assumptions & Experimental Setups

Evaluator assumptions

- **OBJECTIVES**
  - Evaluate model's robustness vs precise fault injections
  - Decreasing the average accuracy (test set)
  - Generic untargeted scenario

- **HYPOTHESIS** Security testing context ➔ evaluator simulates **worst-case adversary**
  - Perfect knowledge of the model (white-box attack)
  - Query the model without limitation
  - Full access to the device (or clones of the device)
  - Can perform elementary characterizations (adapt & optimize the fault injection set-up)

Assumptions & Experimental Setups

Fault Model

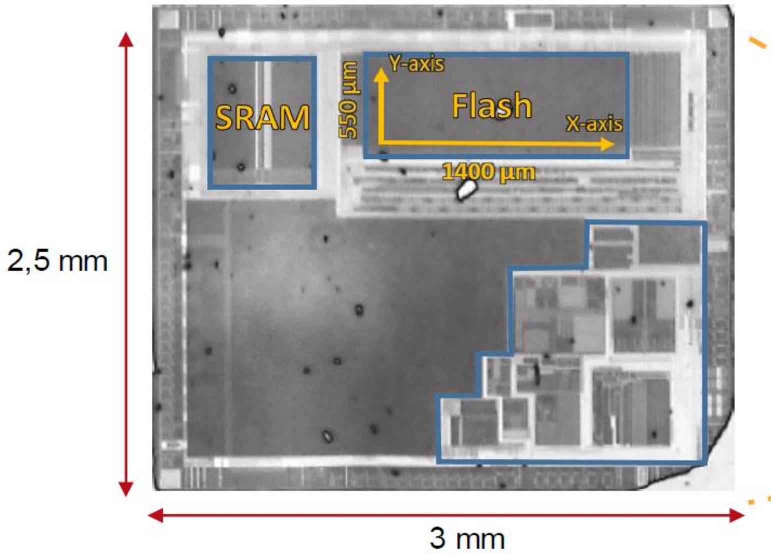
- Single **bit-set fault model** on Flash memory **[0 → 1 | 1 → 1]**
  - Accurate fault model relevant for LFI
  - Explained and demonstrated for NOR-Flash memory of Cortex-M MCU by Colombier *et al*. [1]

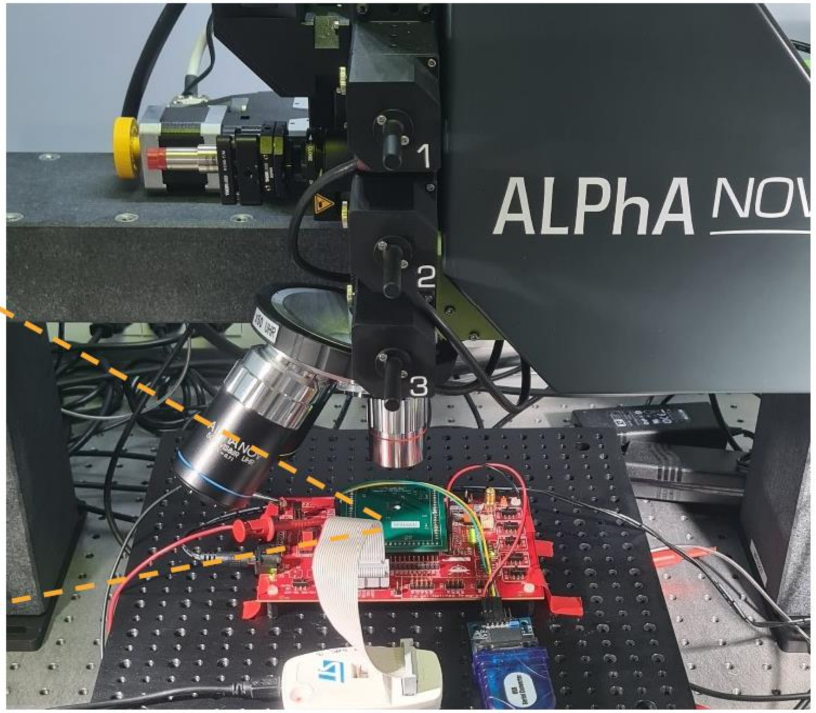
Target & Laser bench

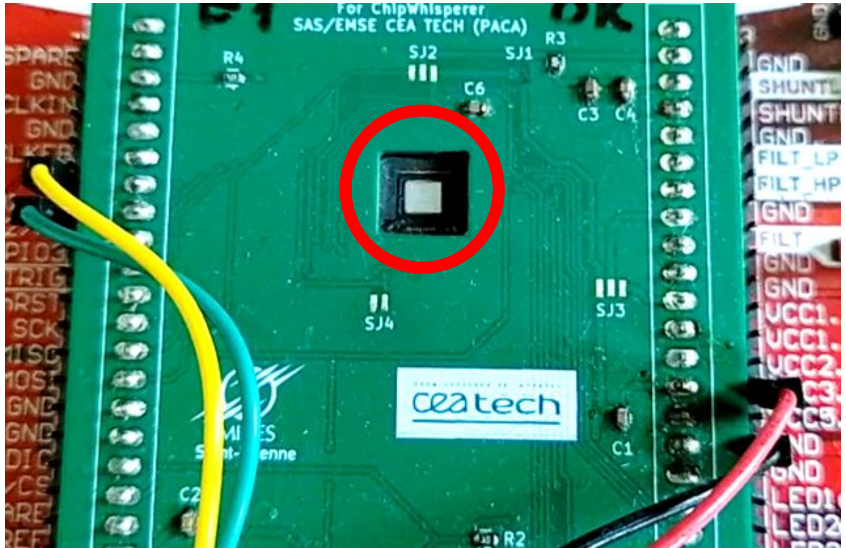
- ARM Cortex-M3 (90nm CMOS) | 8 MHz | 128 kB of Flash memory | Chip = 3 x 2.5 mm
- For LFI: MCU packaging is opened (engraving tools, acid…)
- Double spots laser platform
  - Near infrared (IR), λ=1, 064 nm, Laser spot diameter [1.5 - 15] µm. Max power = 1, 700mW.
  - Delay (trigger/shot) = few nanoseconds
  - Infrared camera

(1) Colombier, et al. Laser-induced Single-bit Faults in Flash Memory…, IEEE HOST 2019.

Assumptions & Experimental Setups



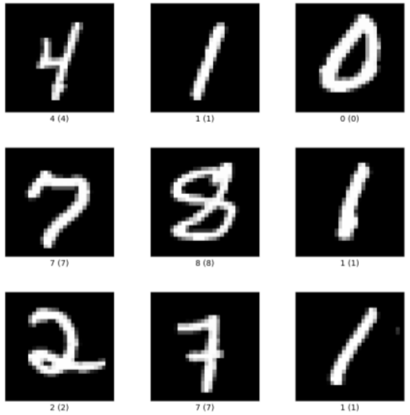




Assumptions & Experimental Setups

Datasets & Models

- Cortex M3 with high memory constrains
- Work with 8-bits quantized models
- Deployment: open-source platform NNoM: *Neural Network on Microcontrollers*
- Use of a Multilayer-Perceptron (MLP) trained on **MNIST**
  - Input compression on $\mathbb{R}^{50}$ (PCA)
  - 1 hidden layer with 10 neurons
  - 1 output layer with 10 neurons



Weight-based adversarial attack with LFI

What, when, where to shoot?

Strike parameters stored in Flash memory

- Flash Memory ➔ Model architecture and internal parameters
- SRAM ➔ intermediate calculations (e.g., activations)

Neuron ➔ weighted sum

```
while (rowCnt){
    //pA : address, stored input
    //pB : address, stored weight
    for (int j = 0; j < dim_vec; j++)
     { //loop on all neuron
     parameters
         q7_t inA = *pA++;    //load
     input to inA, address increment
         q7_t inB = *pB++;   //load
     weight to inB, address increment
         ip_out += inA * inB; //neuron
       weighted sum
    }
    *p0++ = (q7_t)__NNOM_SSAT((ip_out
       >> out_shift), 8);
    rowCnt--;}
```

C code, weighted-sum in a dense layer

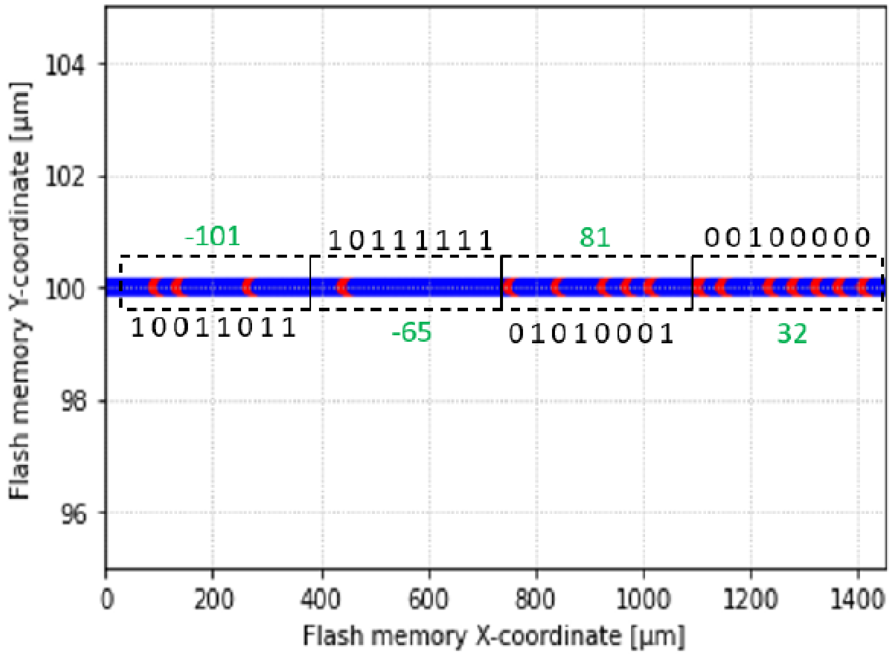
```
;q7_t inB = *pB++          ;Weight n+1
     initialization
ldr       r3, [r7, #80]   ;Loading the
     address of the weight n
adds      r2, r3, #1      ;Next weight
     address
str       r2, [r7, #80]   ;Input value
     loading into r2 reg
ldrsb.w r3, [r3]          ;Weight value
     loading. LASER SHOT
strb      r3, [r7,#23]    ;Store of the
     weight in SRAM reg
```

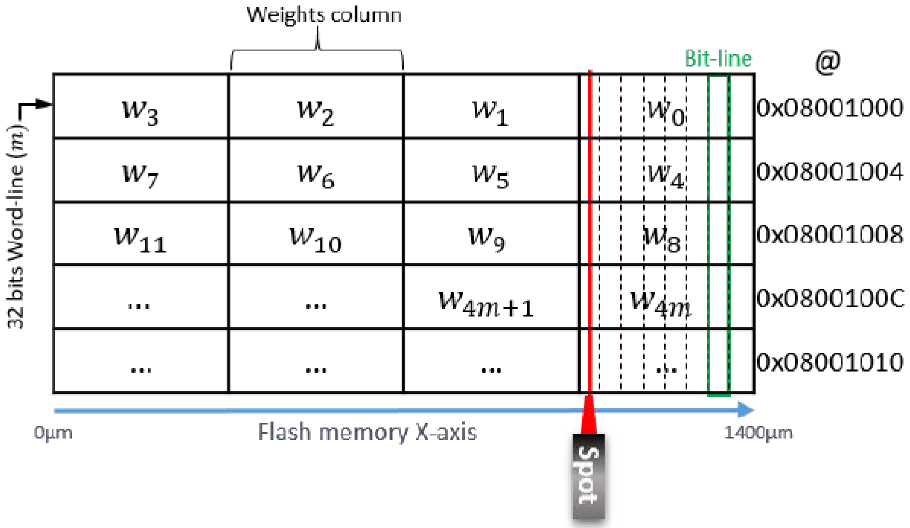
Assembly code, of line 6

What, when, where to shoot?

Mapping the Flash memory

- First experiment with a 4-weight neuron ➔ alter all the bits following the **bit-set model**
  - 0 → 1 // 1 → 1





m = mth word-line

What and where to shoot?

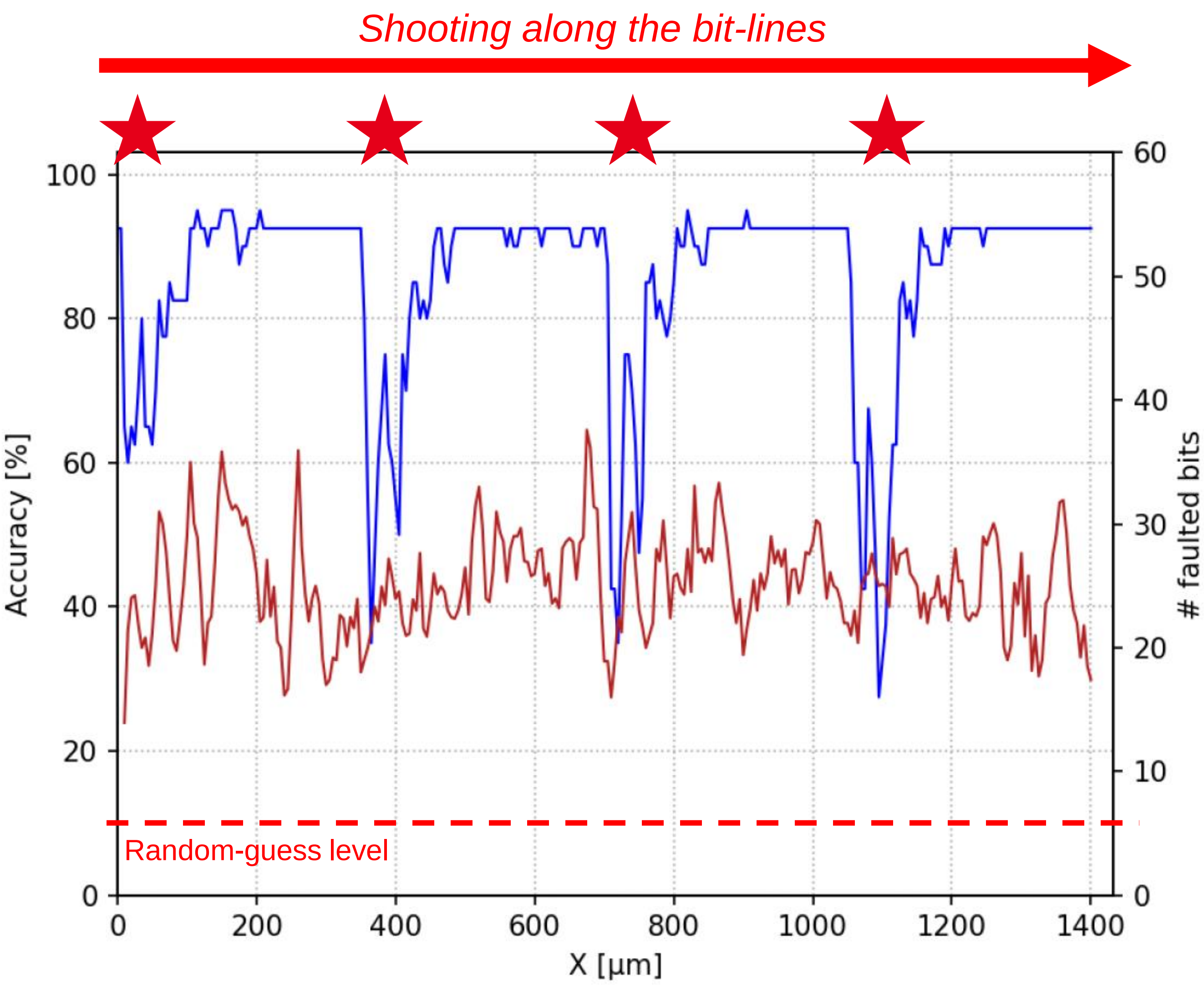
Target a MLP model

- Brute-Force approach

➔ target all the bit from all the weights

- MLP model ➔ 4960 bits
- By targeting 1 bit line at a given X-position of the laser, only weights on the same address column are faulted with a bit-set

- Significant accuracy drops for MSB locations ★



Combining LFI and simulation

Advanced Guided-LFI

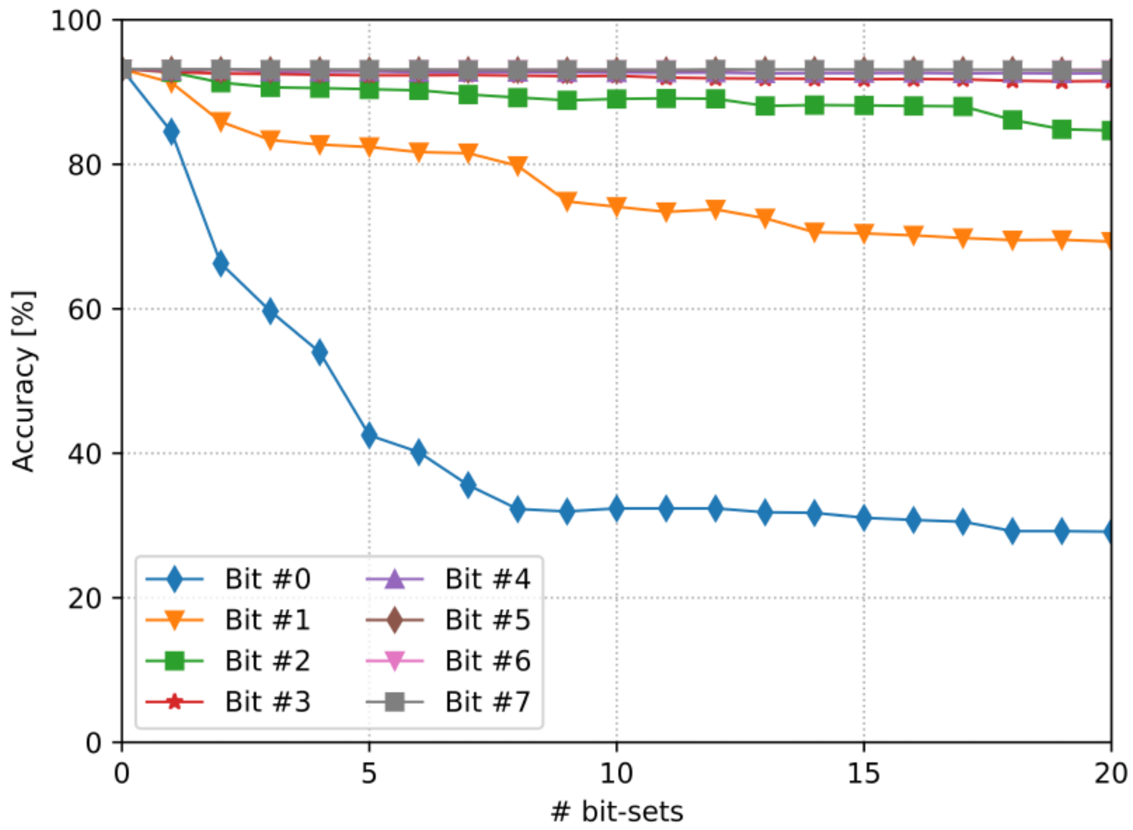
- Brute-Force strategy is impractical with deeper models
- IDEA: adapt the BFA principle to our fault model ➔ BSCA = *Bit-Set Constrained Attack*

- Target model **M**. **W** weights matrix. Adversarial budget **S** (max number of faults)
- FOR EACH weight column index **c**, bit line index **b**
  - #1. BFA ranks the most sensitive bits of **W** according to $\nabla_b \mathcal{L}$
  - #2. Exclude the bits already set to 1 and not related to **c** and **b**.
  - #3. Pick the best bit-set and perform the fault permanently in **M**.
  - #4. Repeat the process until reaching **S**
  - Repeat over **c** and **b** ➔ **KEEP WORST ACCURACY**

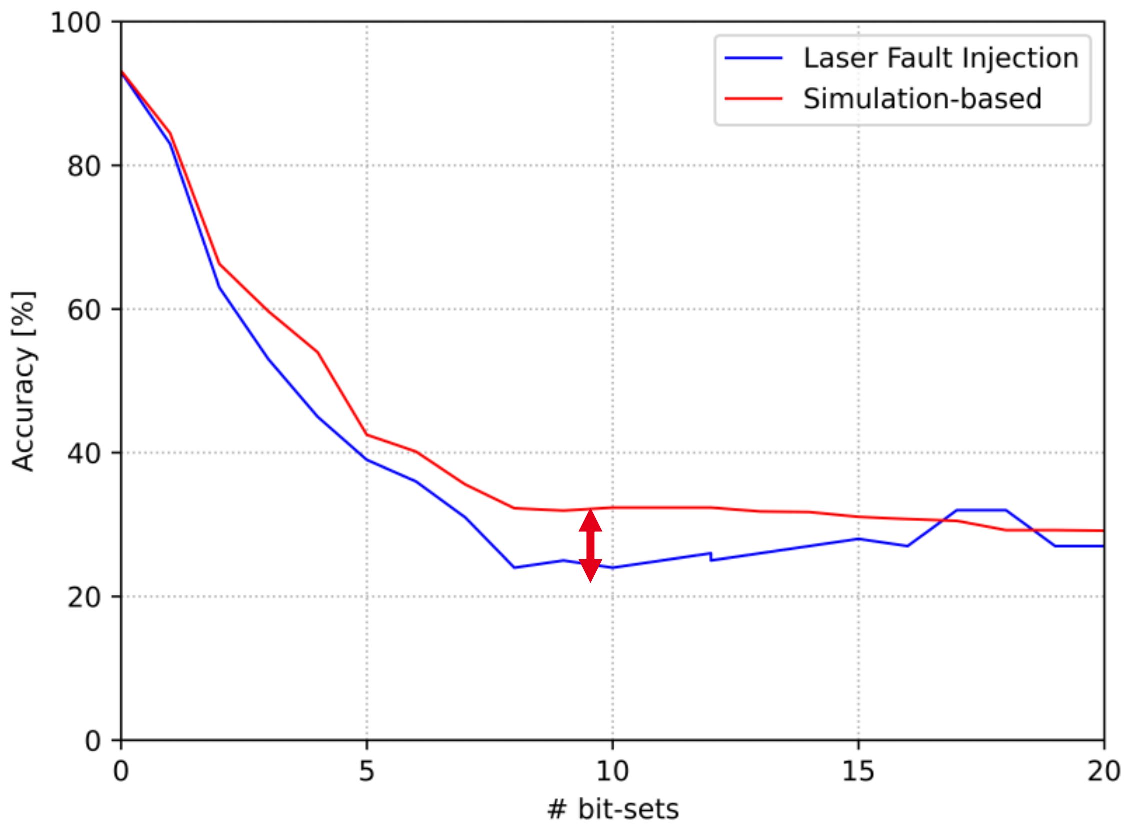
Combining LFI and simulation

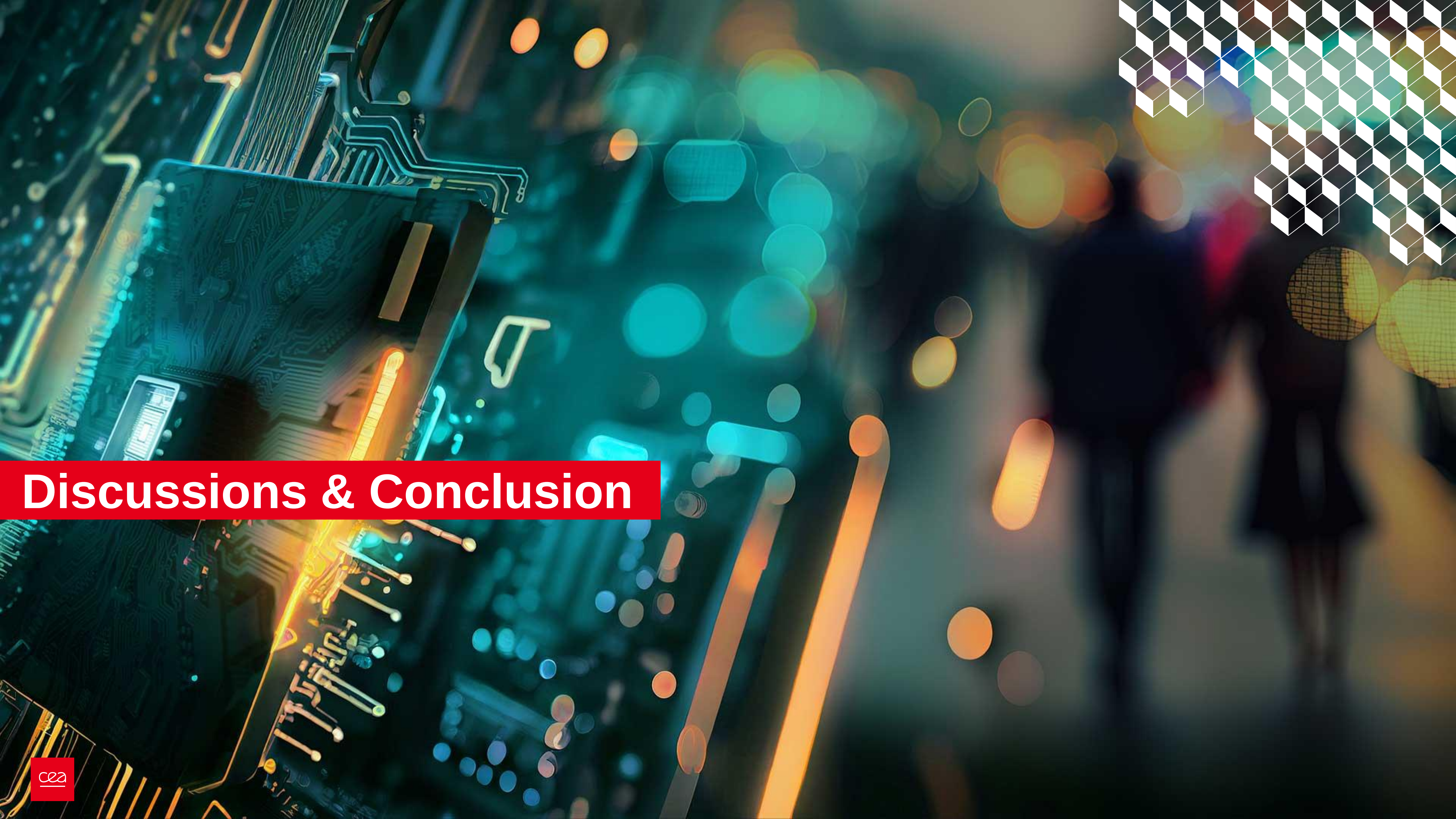
Advanced Guided-LFI

Simulation over weight columns
(« best »: m=2)



How practical LFI fit with simulations (BSCA)?





Discussions & Conclusion

Discussions & conclusion

Exploiting Fault Injection for Confidentiality Threats

- BFA + RowHammer for model extraction scenario ➔ extract partial parameter values[1]
- We demonstrated Model Extraction with the BSCA
  - ESORICS / SECAI Workshop[2]
  - Basic idea: Safe Error Attack principle ➔ guess bit value whether the fault changes the prediction or not (w.r.t. the normal behavior)

Limitations

- Model complexity is not such a problem ➔ complexity of Flash memory is the big challenge
- Further experiments need to be focused on other targets (e.g., Cortex M4 and M7)
- When to shoot ➔ smart trigger with side-channel analysis?

(1) Rakin, et al. DeepSteal. IEEE S&P 2022
(2) Hector, et al. Fault Injection and Safe-Error Attack for Extraction of Embedded Neural Network Models. ESORICS/SECAI 2023.

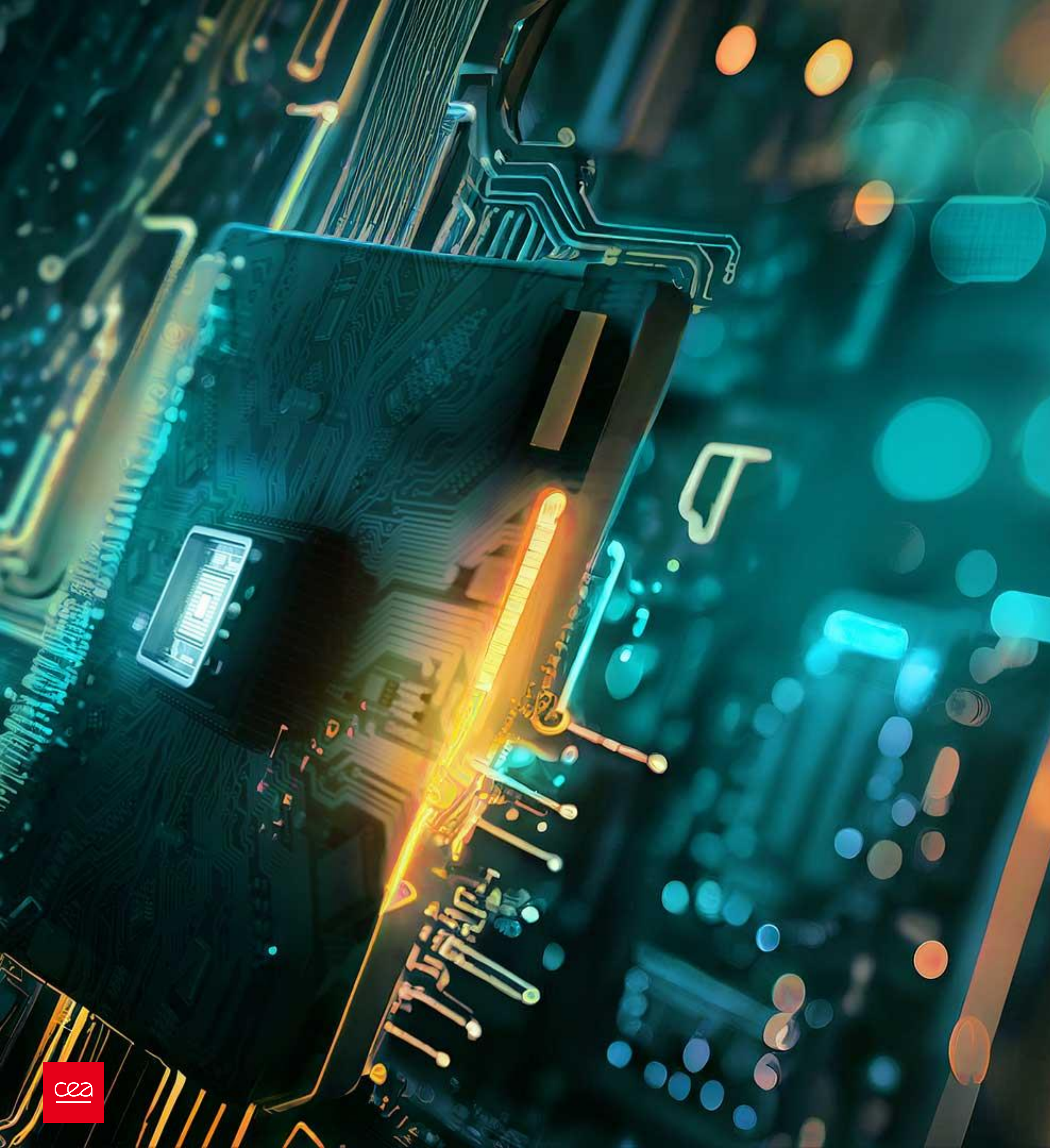
Discussions & conclusion

Maturity of Parameter-based Attacks

- Parameter-based attack still lacks of maturity
- SotA: limitations, improvements, alternatives
- Future works ➔ considering or combining more attack methods to improve evaluation

Extend to practical evaluation of protections

- Are generic countermeasures against fault injection relevant?
- Practical evaluation of specific BFA-oriented defenses
  - *weight clipping*, *clustering-based quantization*, *code-based detectors*, *adv training*…
- As for adversarial examples, the definition of sound evaluations of defenses is highly important to disseminate security guidance & future certification actions



Thank you for your attention

Support & Funding

- ✓ EU project **InSecTT**
- ✓ French ANR, IRT **Nanoelec**
- ✓ French ANR **PICTURE** program

This work benefited from the French

Jean Zay *supercomputer with the AI dynamic access program.*