

Hard-Label Cryptanalytic Extraction of DNNs

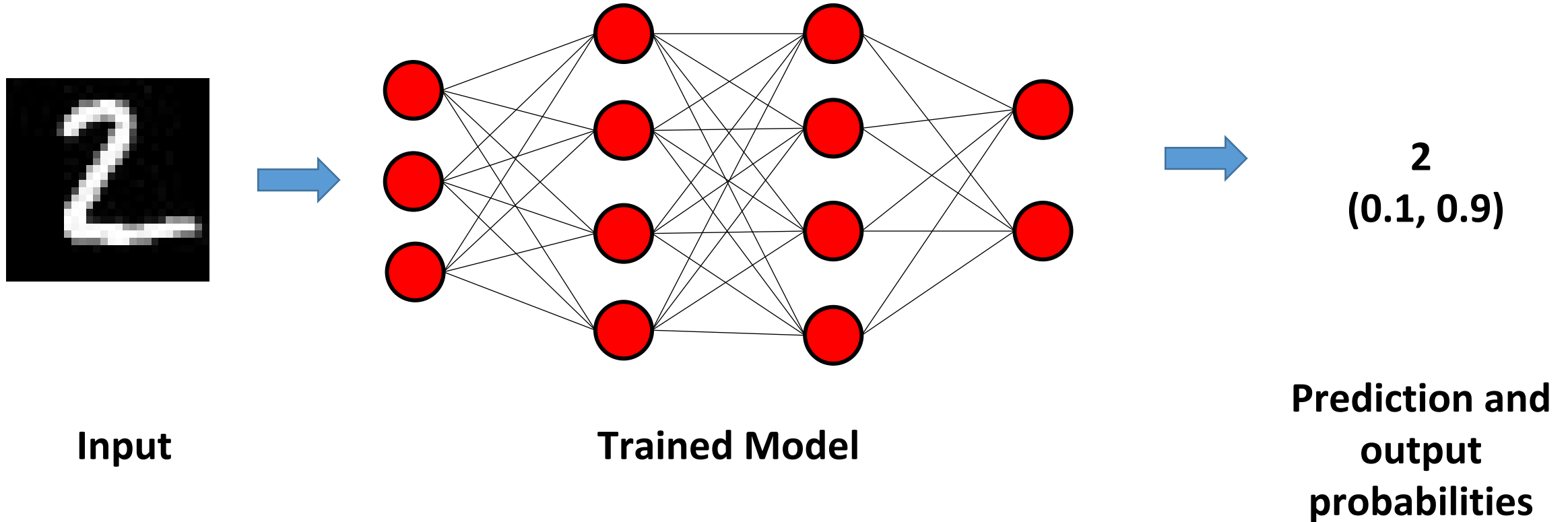
Benoit Coqueret^{1&2}, Mathieu Carbone¹, Olivier
Sentieys², Gabriel Zaid¹

1. CESTI Thales
2. University of Rennes, INRIA, IRISA



SRE

inria

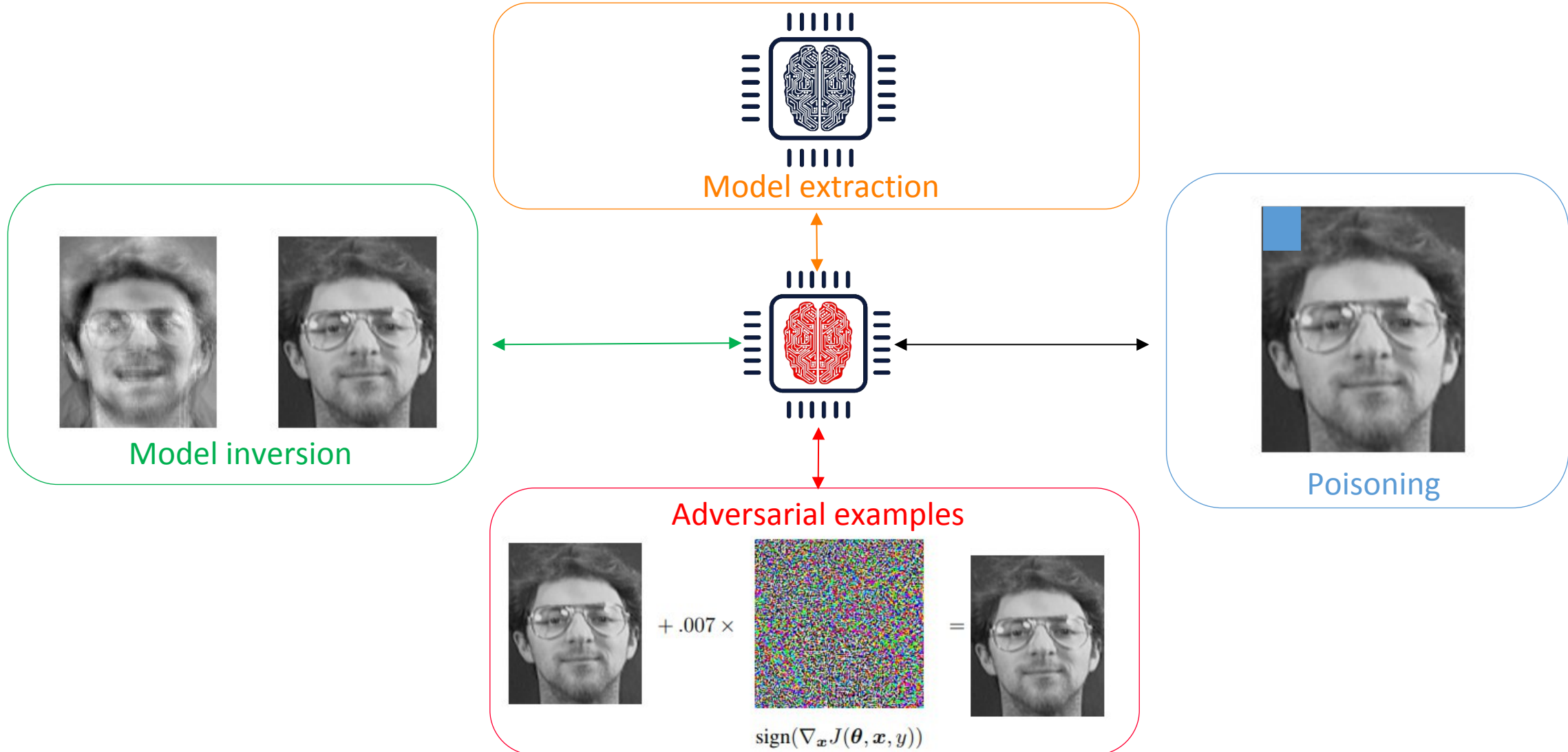


Attacks Against Deep Neural Networks



SRE

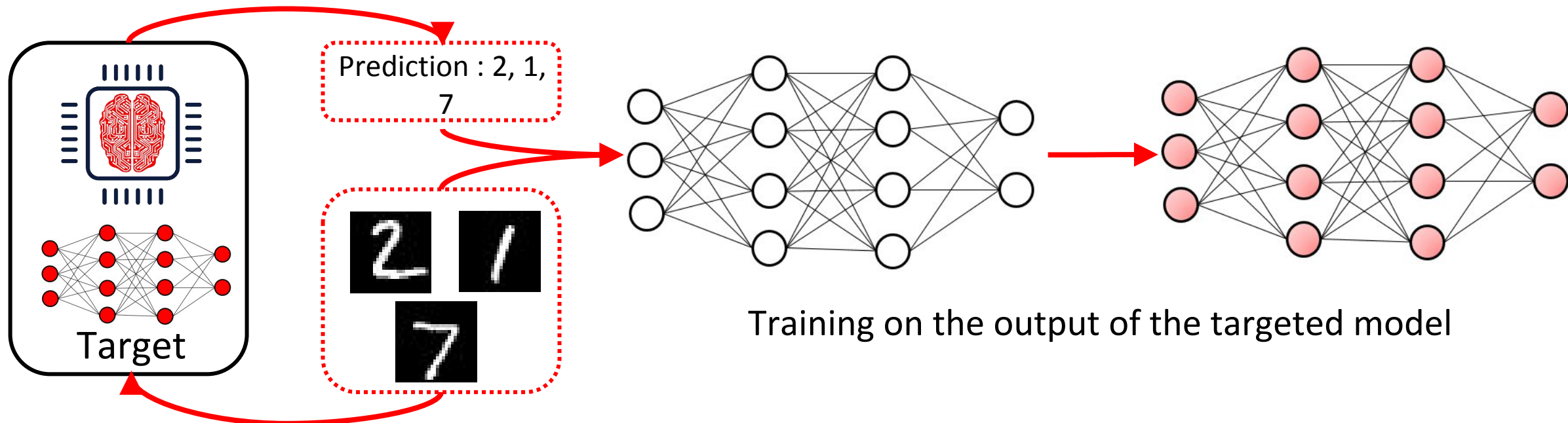
Inria



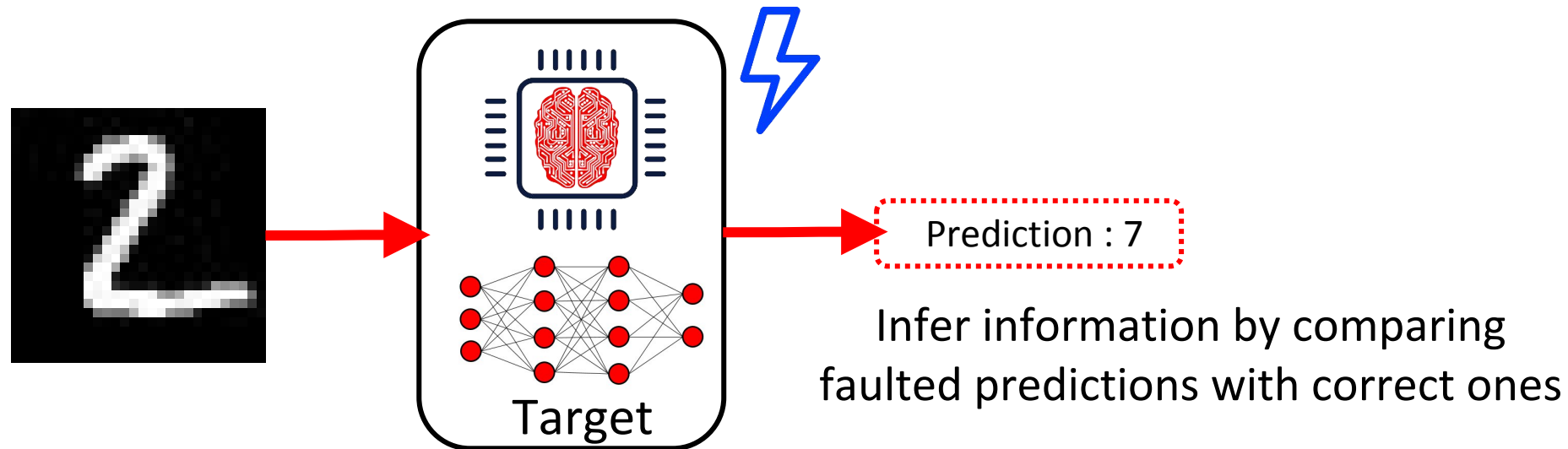
Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures, ACM SIGSAC 2015

- ❖ Obtain a copy of the targeted DNN
 - Stealing the Intellectual Property
 - Possibility to mount more powerful attack on the targeted DNN
- ❖ 3 broad methodologies

- ❖ Obtain a copy of the targeted DNN
 - Stealing the Intellectual Property
 - Possibility to mount more powerful attack on the targeted DNN
- ❖ 3 broad methodologies
 - Active learning [1]

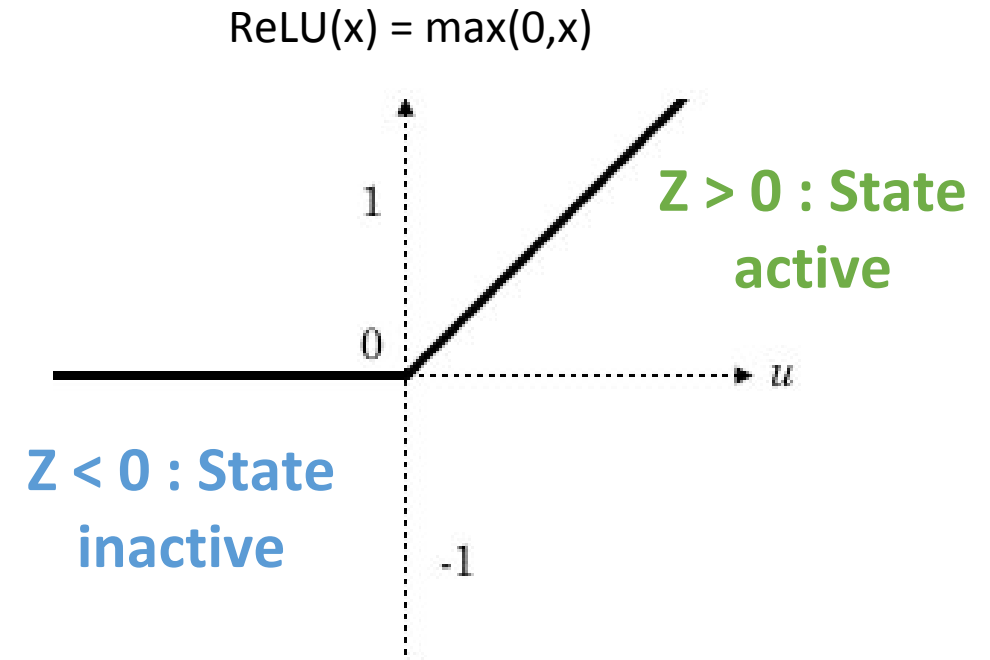
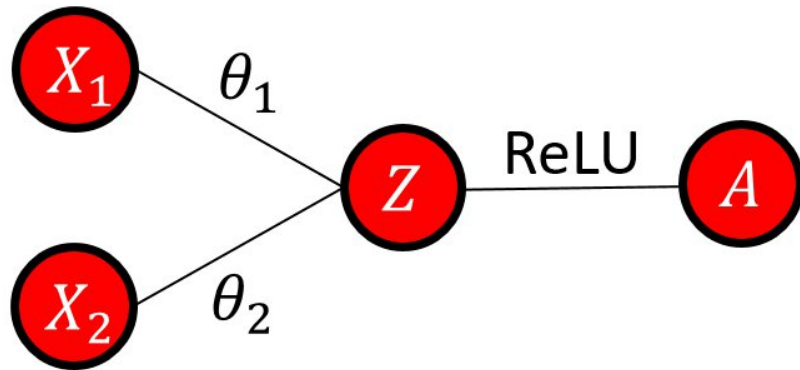


- ❖ Obtain a copy of the targeted DNN
 - Stealing the Intellectual Property
 - Possibility to mount more powerful attack on the targeted DNN
- ❖ 3 broad methodologies
 - Active learning [1]
 - Hardware attacks (Fault Injection [2] or Side Channel [3])

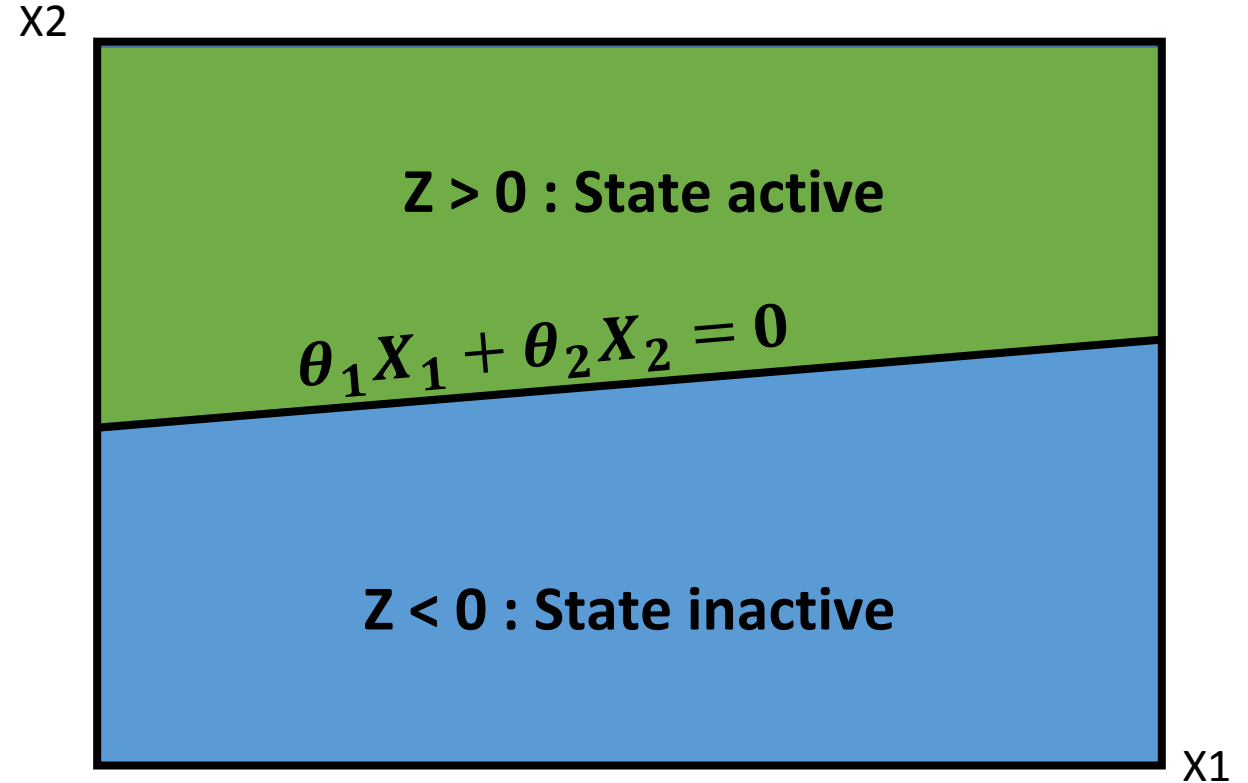
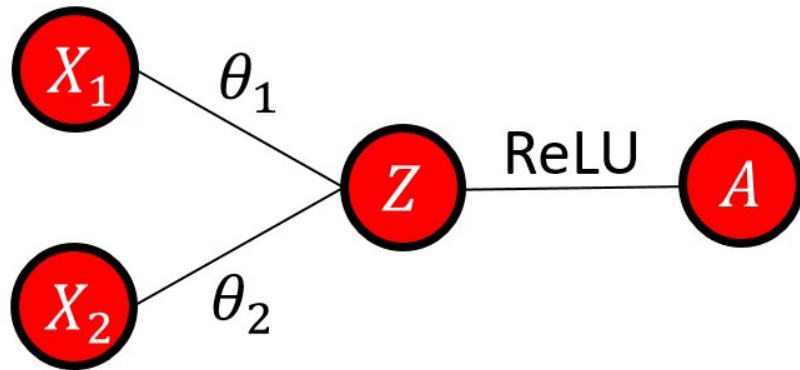


- ❖ Obtain a copy of the targeted DNN
 - Stealing the Intellectual Property
 - Possibility to mount more powerful attack on the targeted DNN
- ❖ 3 broad methodologies
 - Active learning [1]
 - Hardware attacks (Fault Injection [2] or Side Channel [3])
 - Cryptanalytical extraction[4, 5, 6]
 - Analogy between the weights and the key
 - Input becomes the message
 - Output is equivalent to cipher text

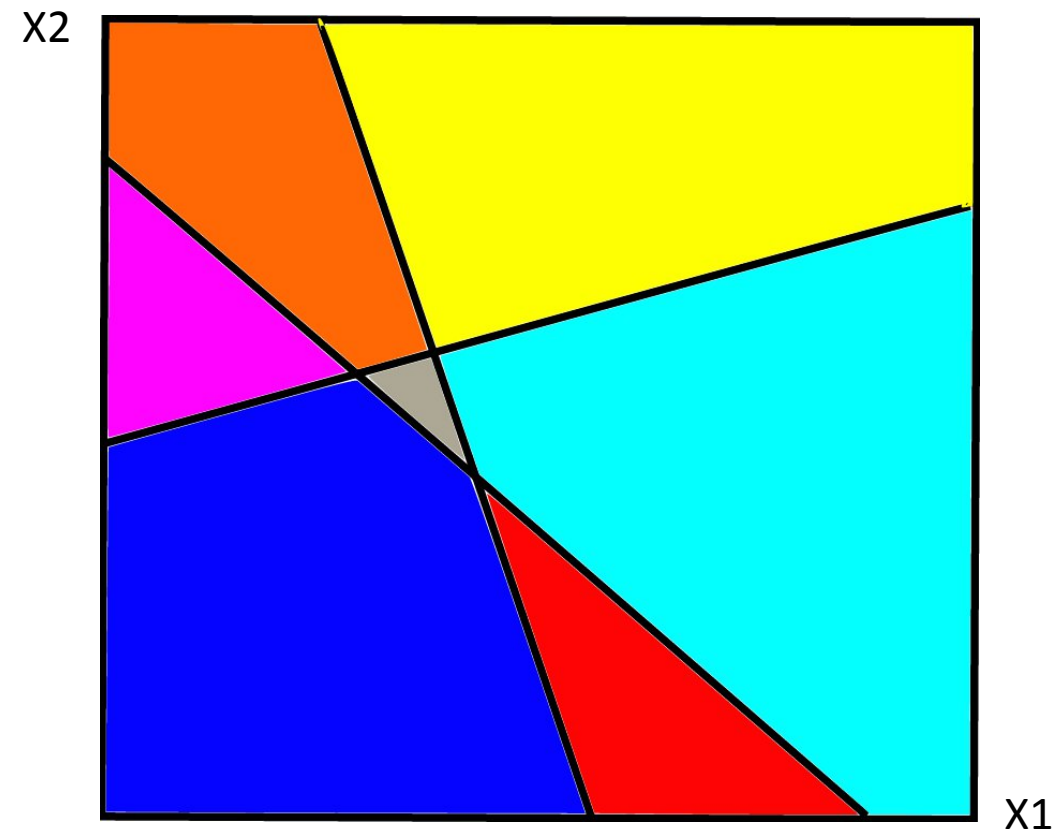
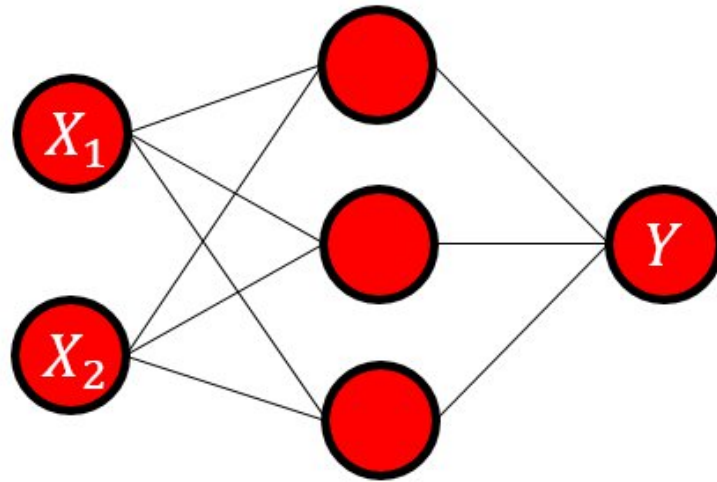
❖ Special case of networks using ReLU function



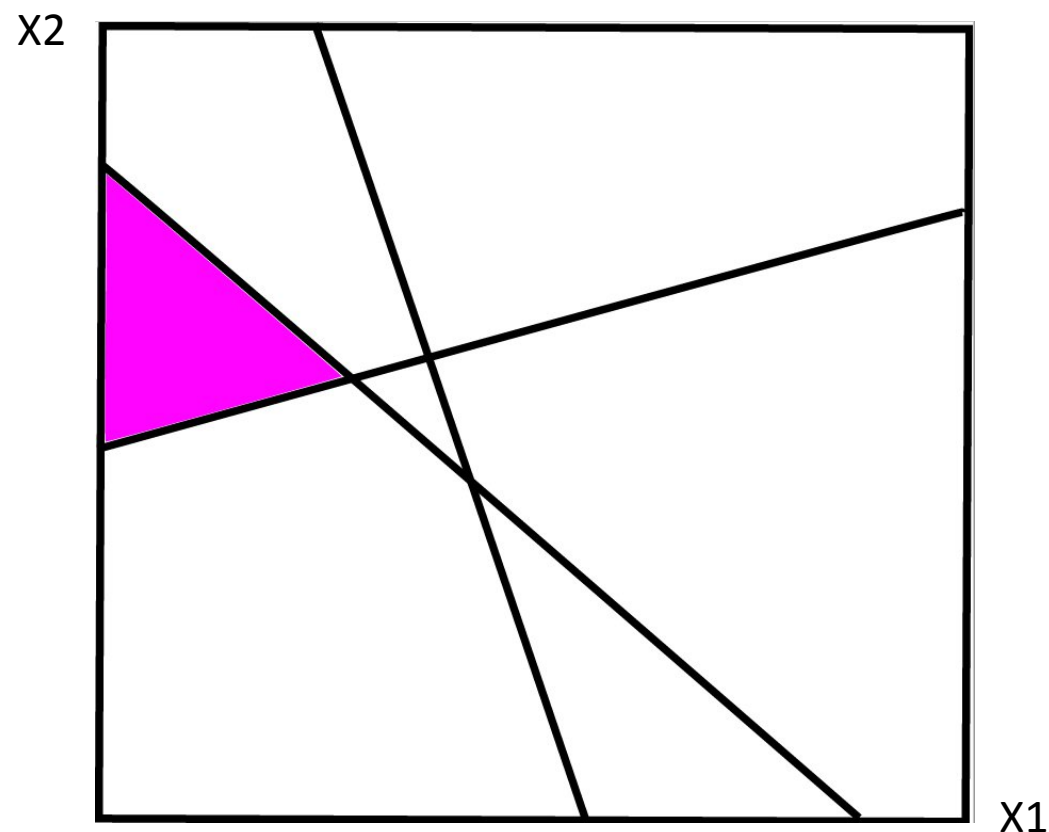
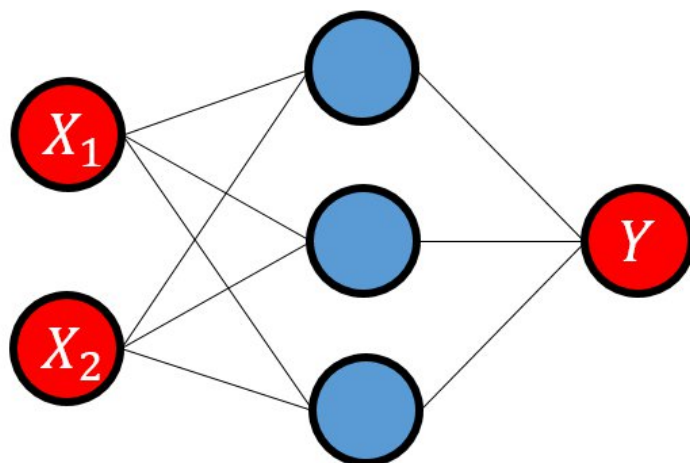
- ❖ Special case of networks using ReLU function



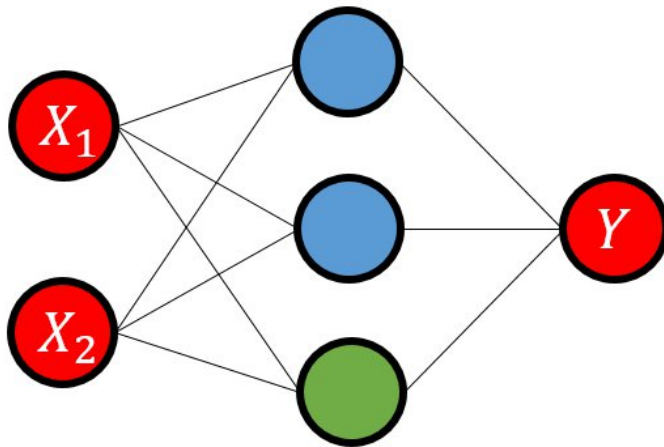
- ❖ Special case of networks using ReLU function



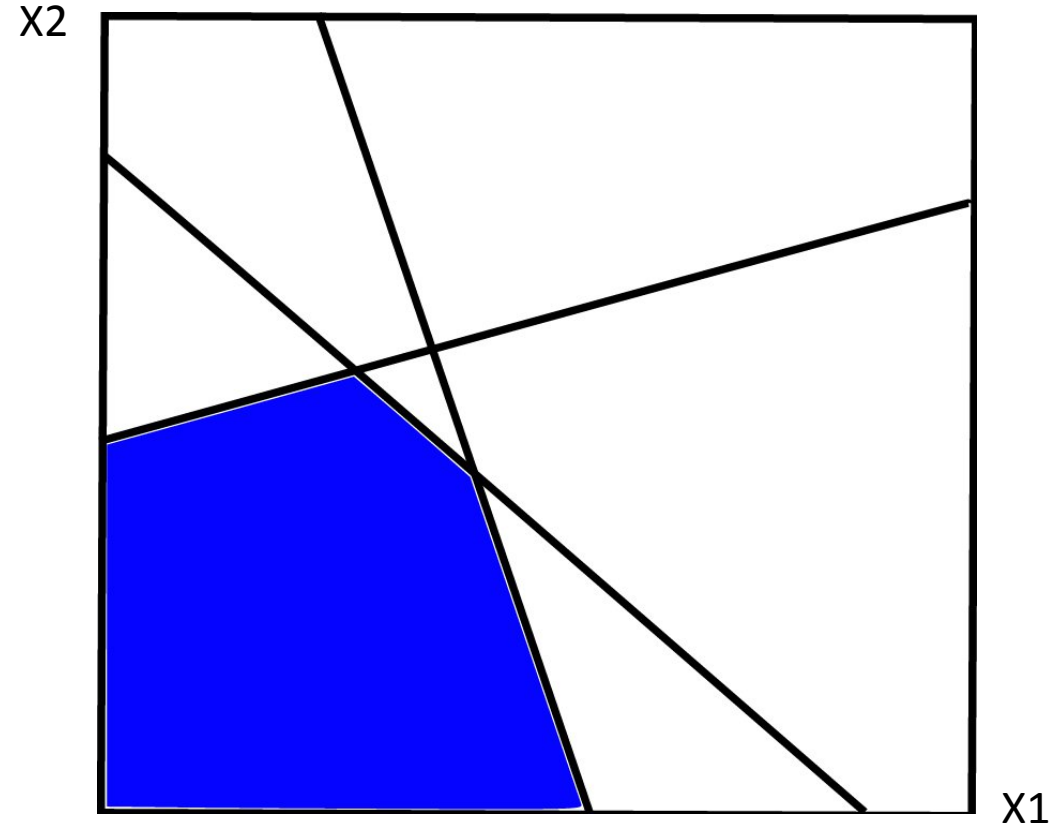
❖ Special case of networks using ReLU function



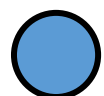
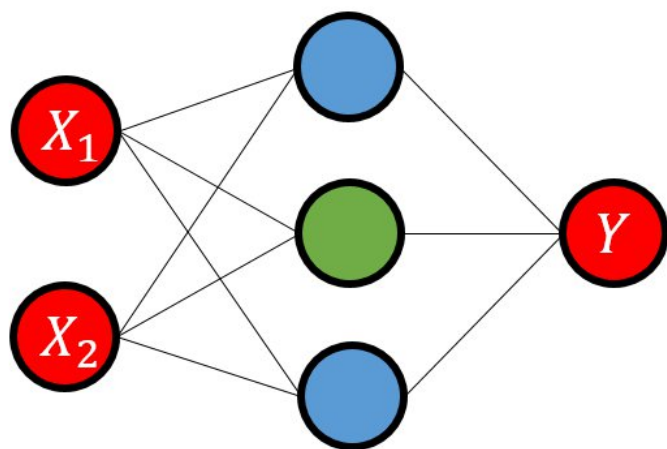
❖ Special case of networks using ReLU function



 Inactive neuron  Active neuron



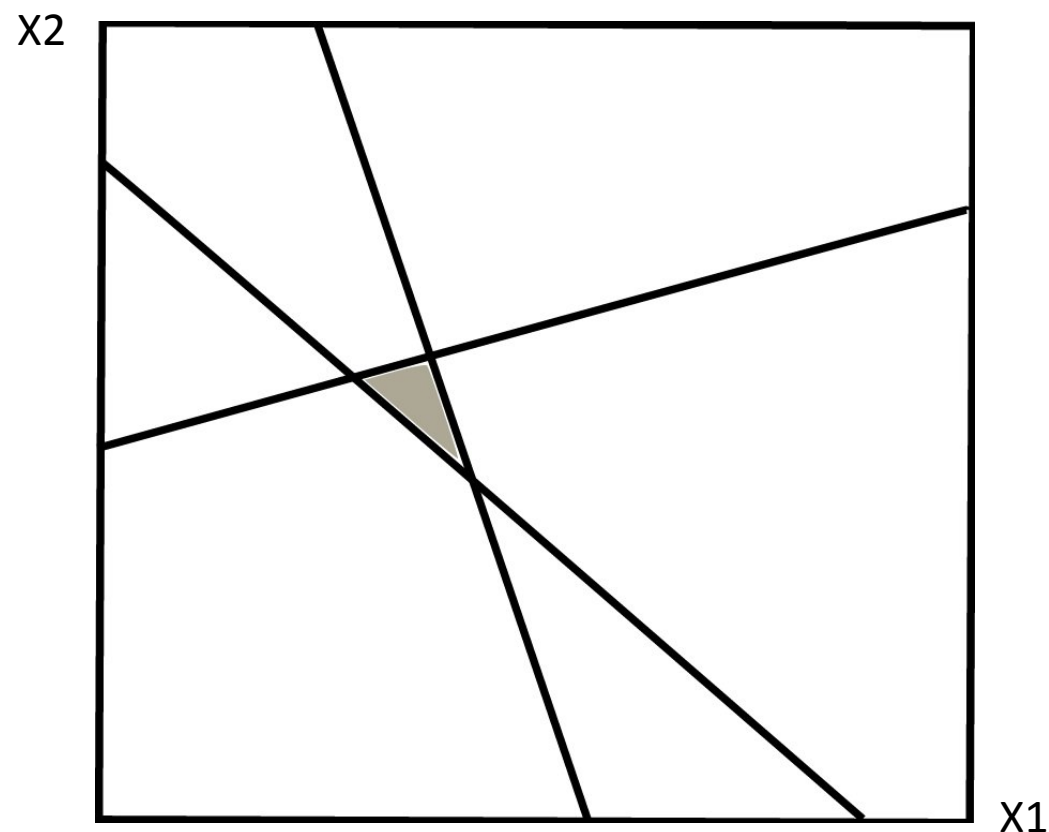
❖ Special case of networks using ReLU function



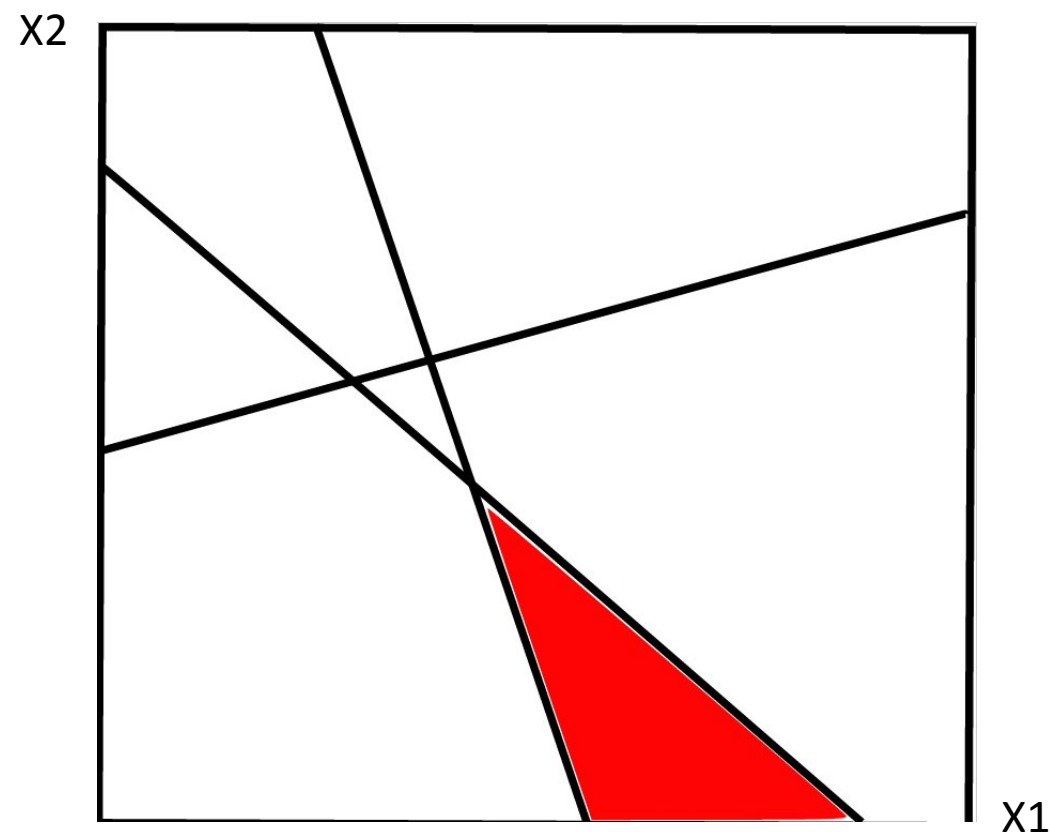
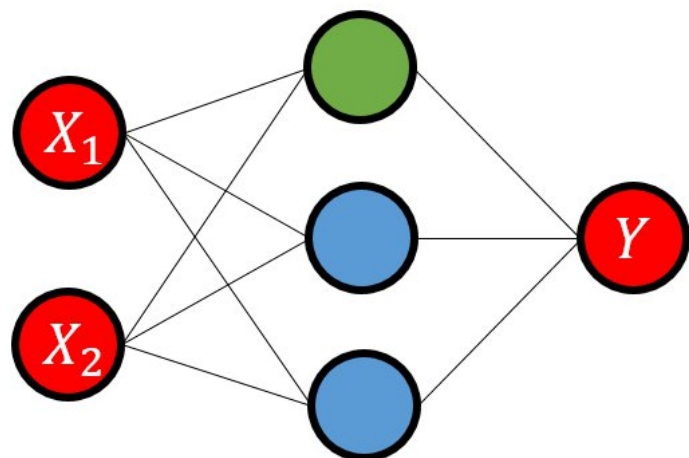
Inactive neuron



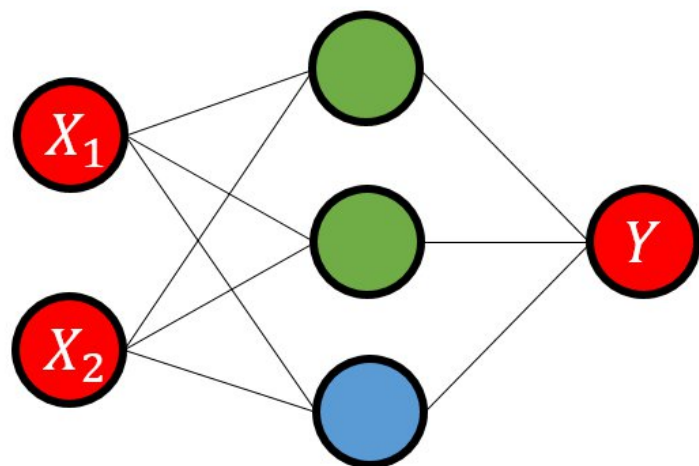
Active neuron



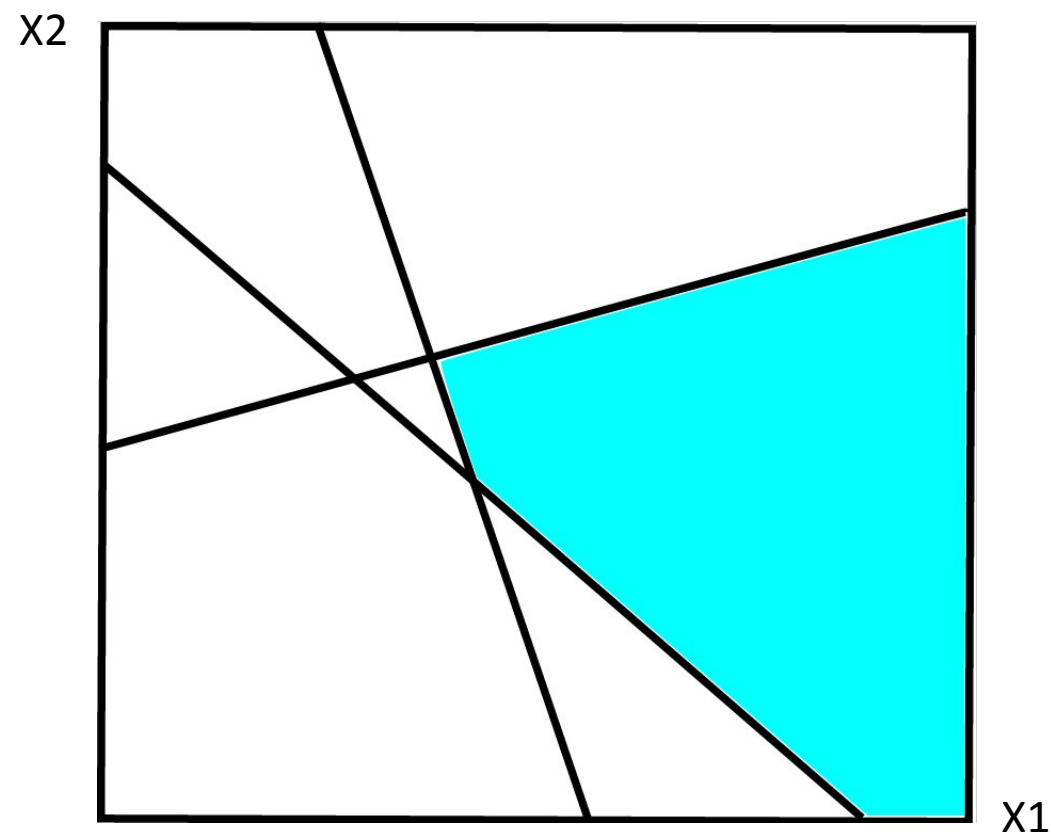
❖ Special case of networks using ReLU function



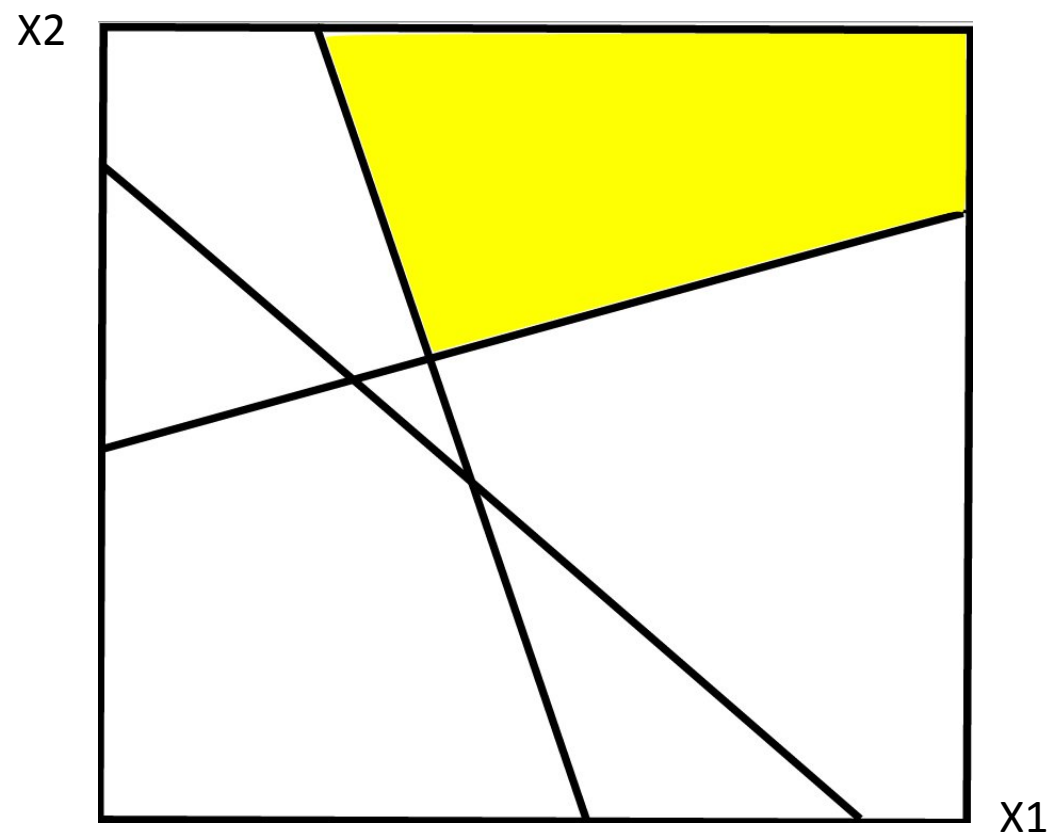
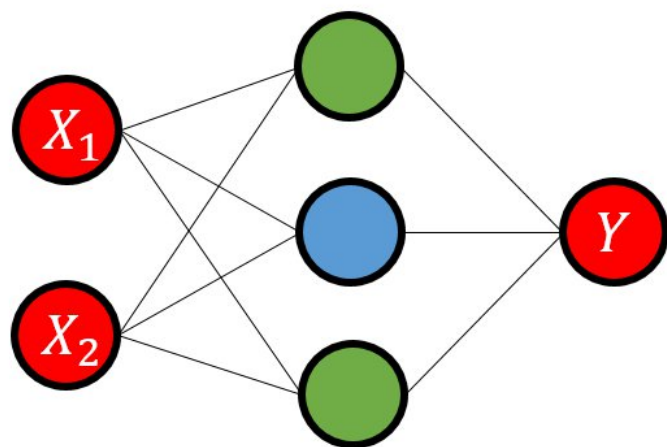
❖ Special case of networks using ReLU function



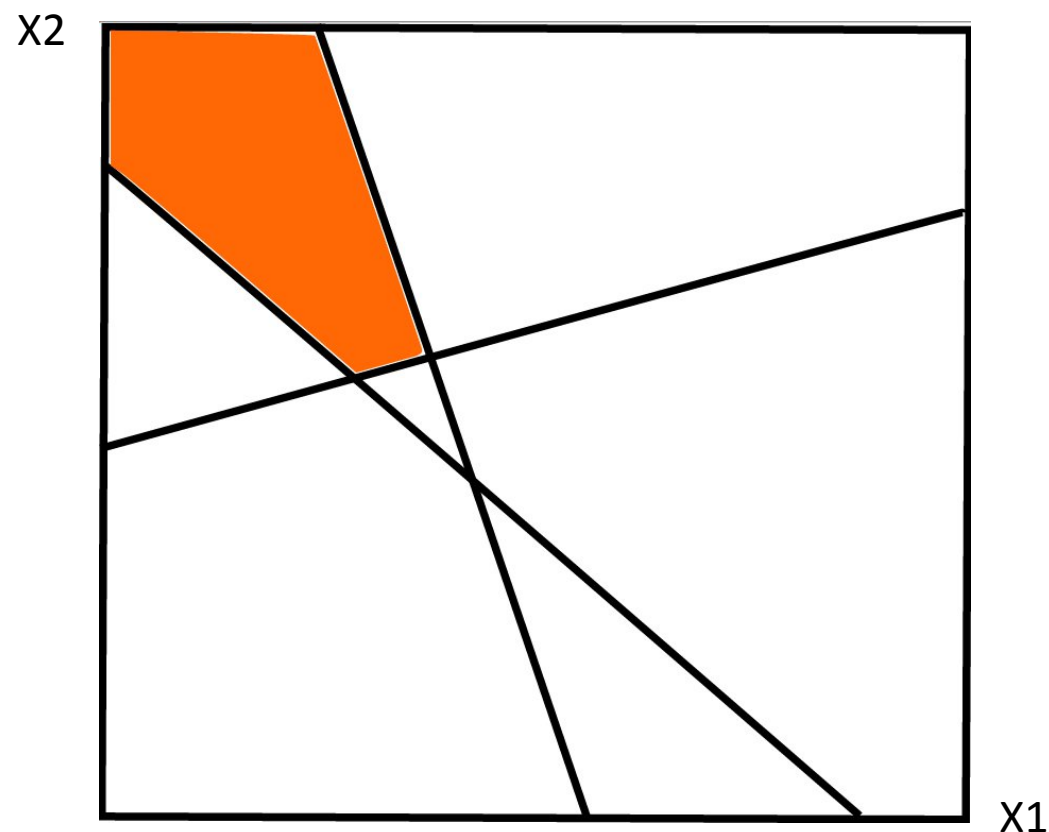
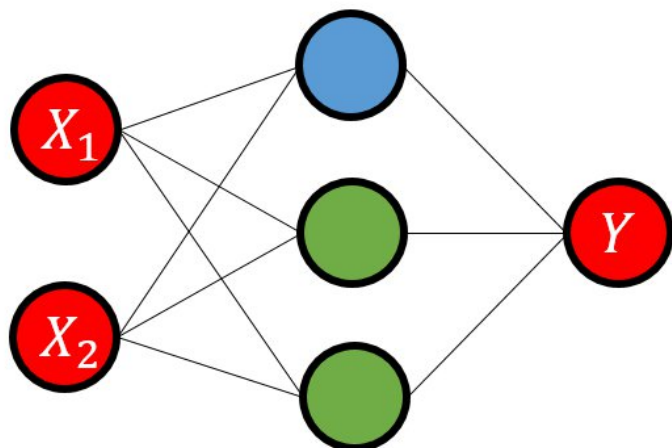
 Inactive neuron  Active neuron



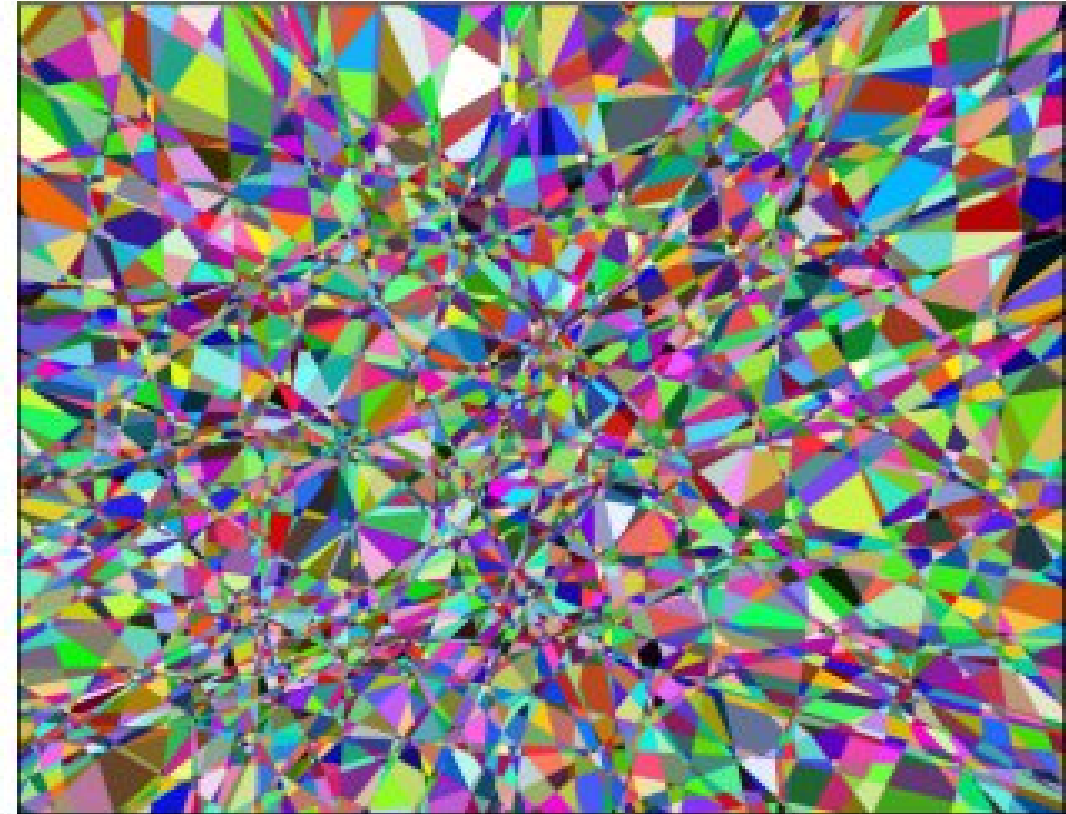
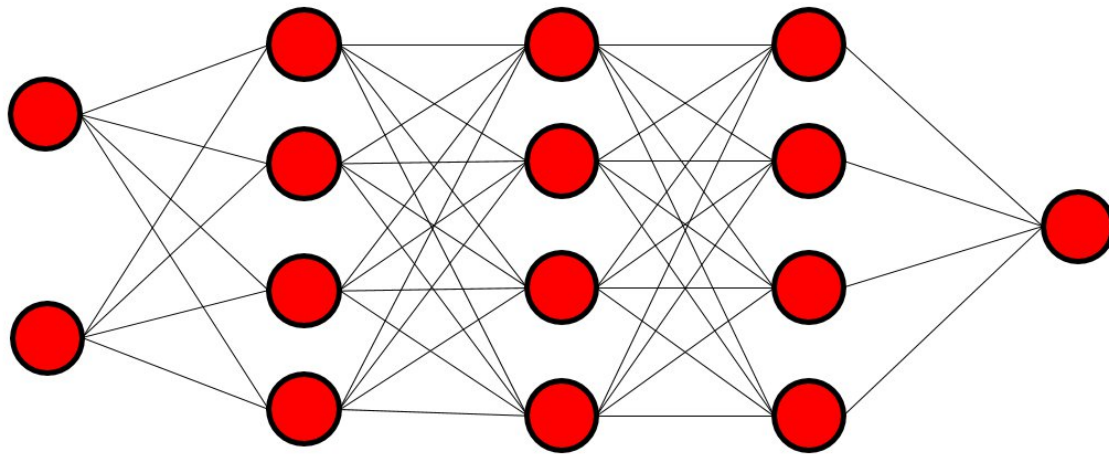
❖ Special case of networks using ReLU function



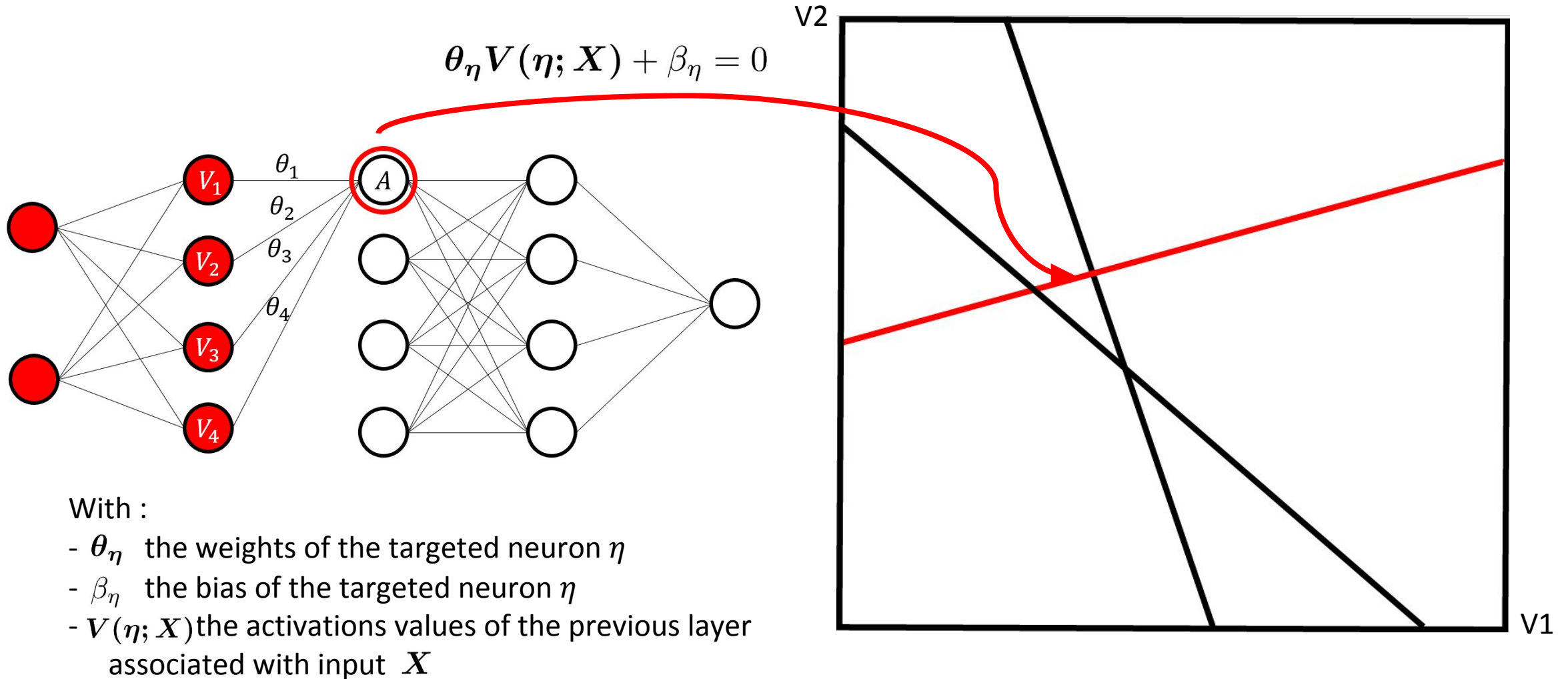
❖ Special case of networks using ReLU function



❖ Special case of networks using ReLU function

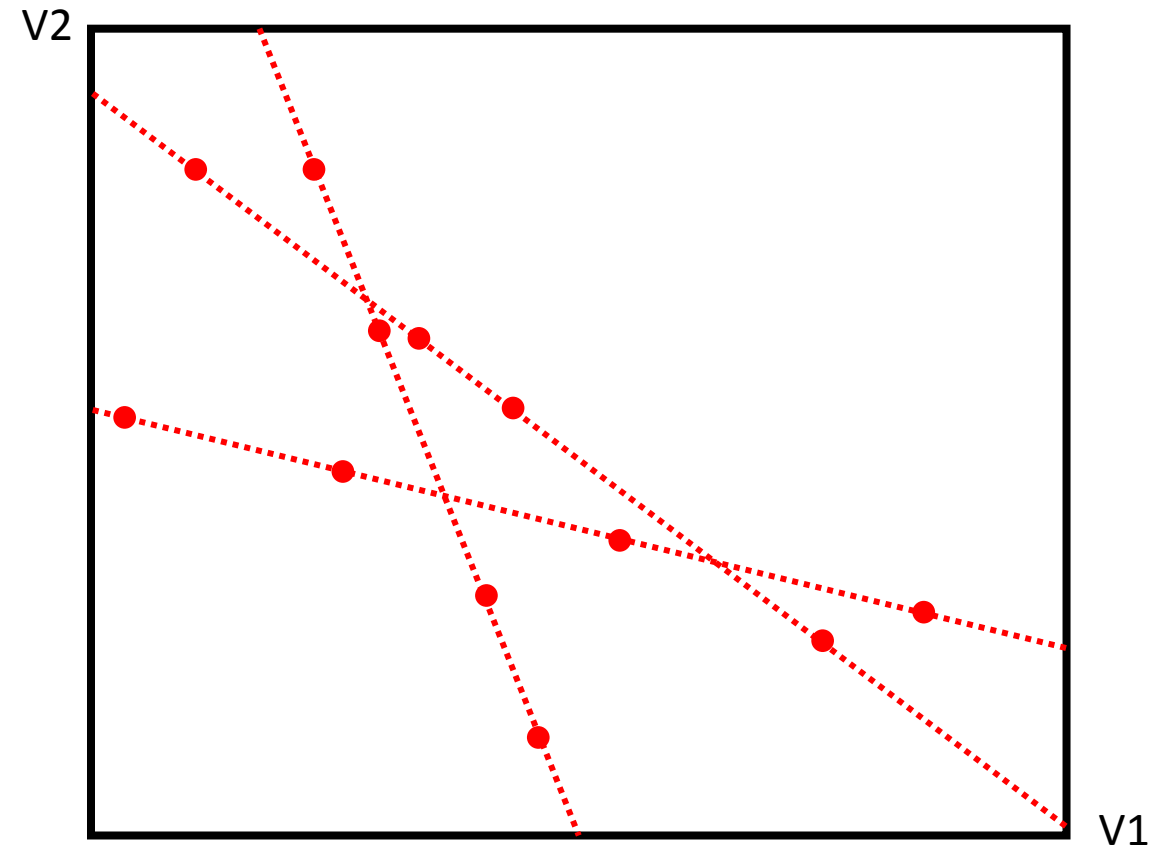


❖ Special case of networks using ReLU function



❖ Global methodology

- Search for points on the hyperplanes: the critical points

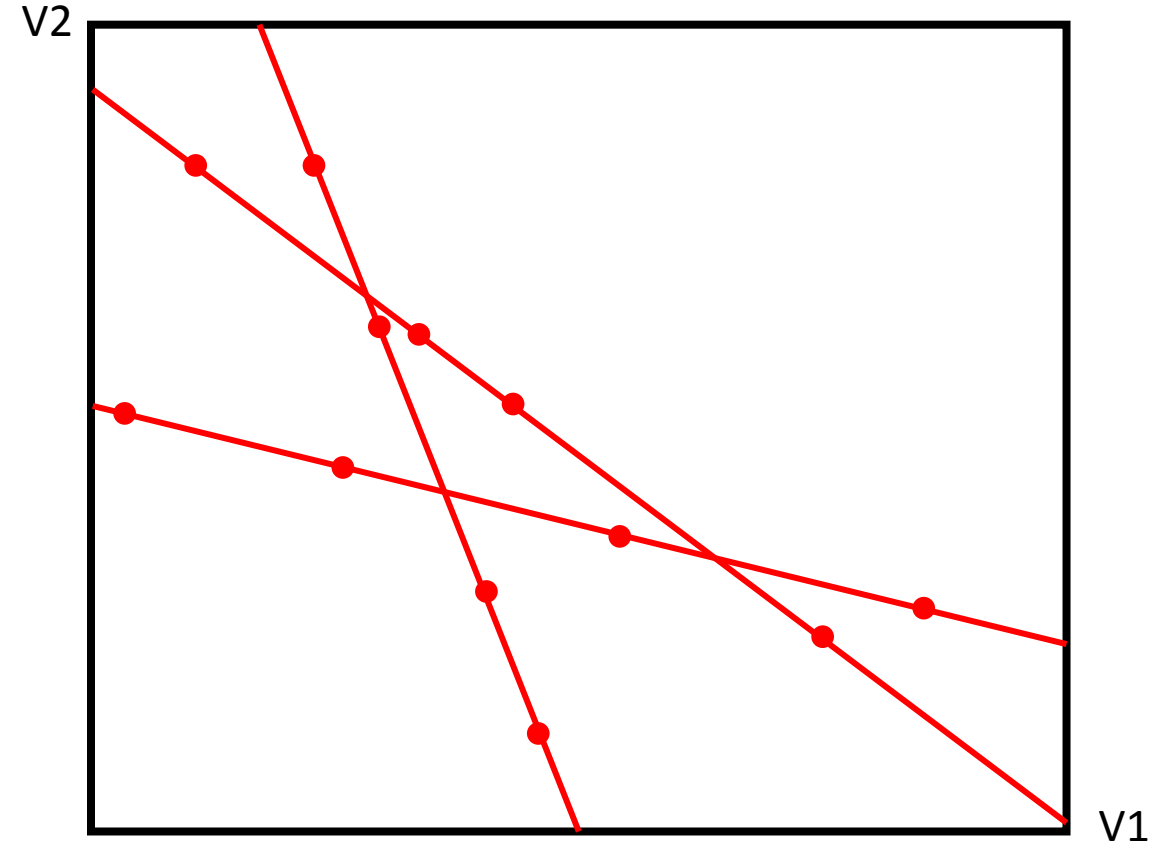


❖ Global methodology

- Search for points on the hyperplanes: the critical points

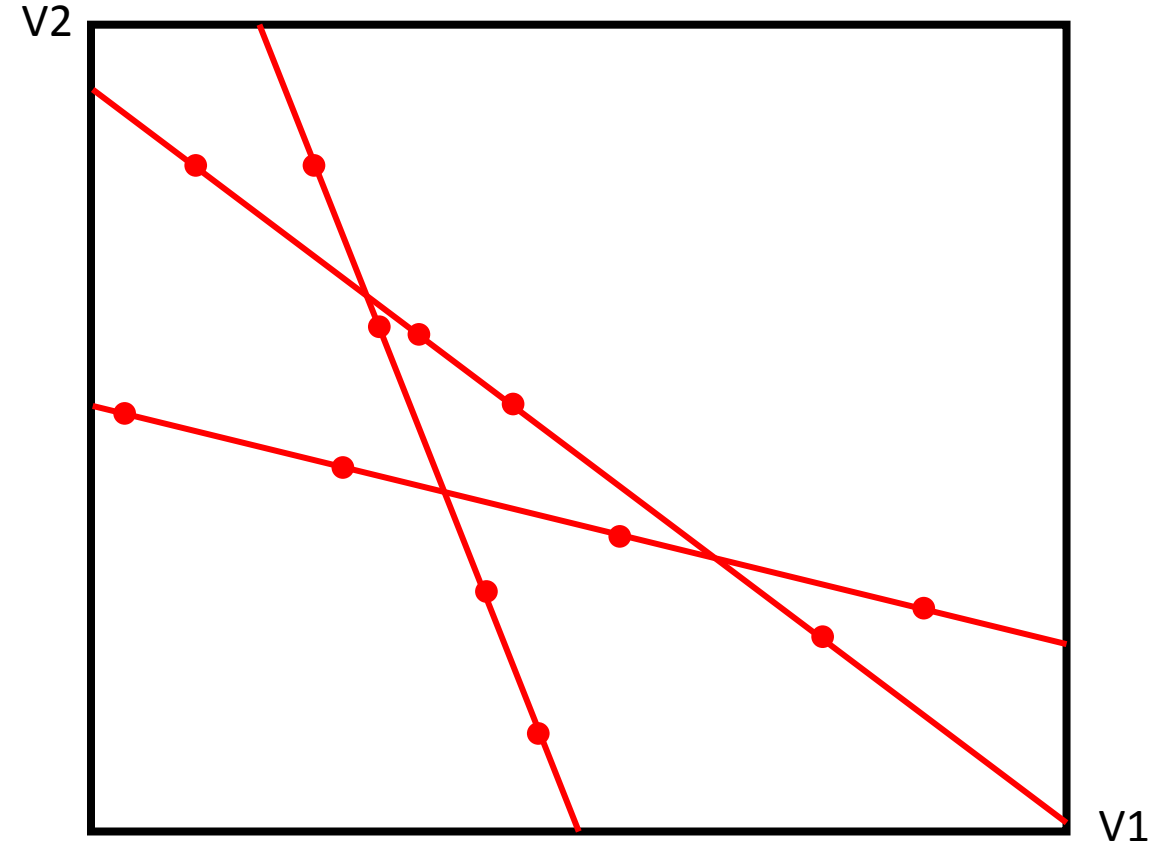
- Retrieve the equations of the hyperplane and the weights

$$\theta_{\eta} V(\eta; X) + \beta_{\eta} = 0$$



❖ Global methodology

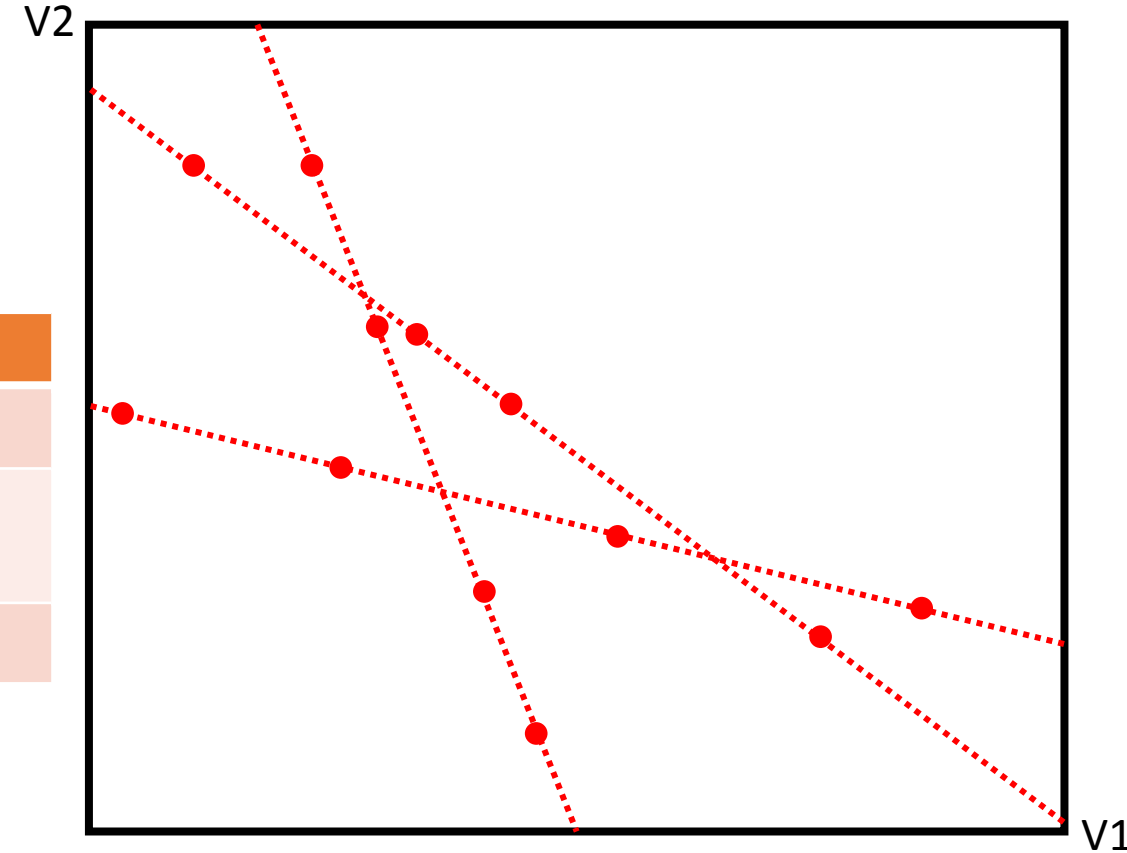
- Search for points on the hyperplanes: the critical points
- Retrieve the equations of the hyperplane and the weights
$$\theta_{\eta} V(\eta; \mathbf{X}) + \beta_{\eta} = 0$$
- Get the sign of the neuron



- ❖ Search for the critical points is the crucial step
 - Highly dependent on the gradient

❖ Current limitations

Issue	Solution
Hard-label settings	Adaptation with dual points
Restriction to fully connected layers	None
Special cases of neurons	None



❖ ReLU implementation

- ARM CMSIS-NN, open source

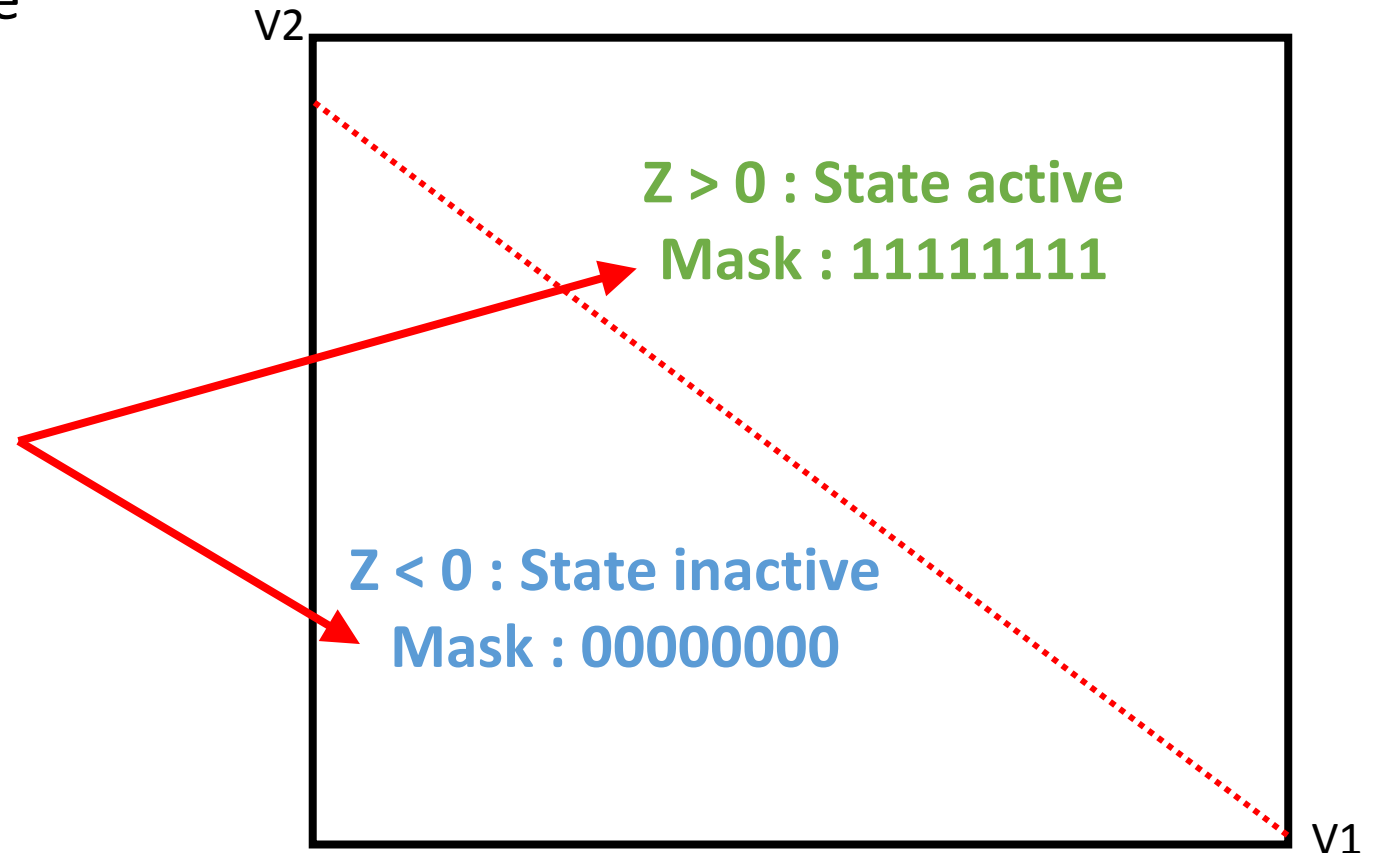
```
while (i)
{
    in = arm_nn_read_s8x4_ia((const int8_t **)&input);

    /* extract the first bit */
    buf = (int32_t)ROR((uint32_t)in & 0x80808080, 7);

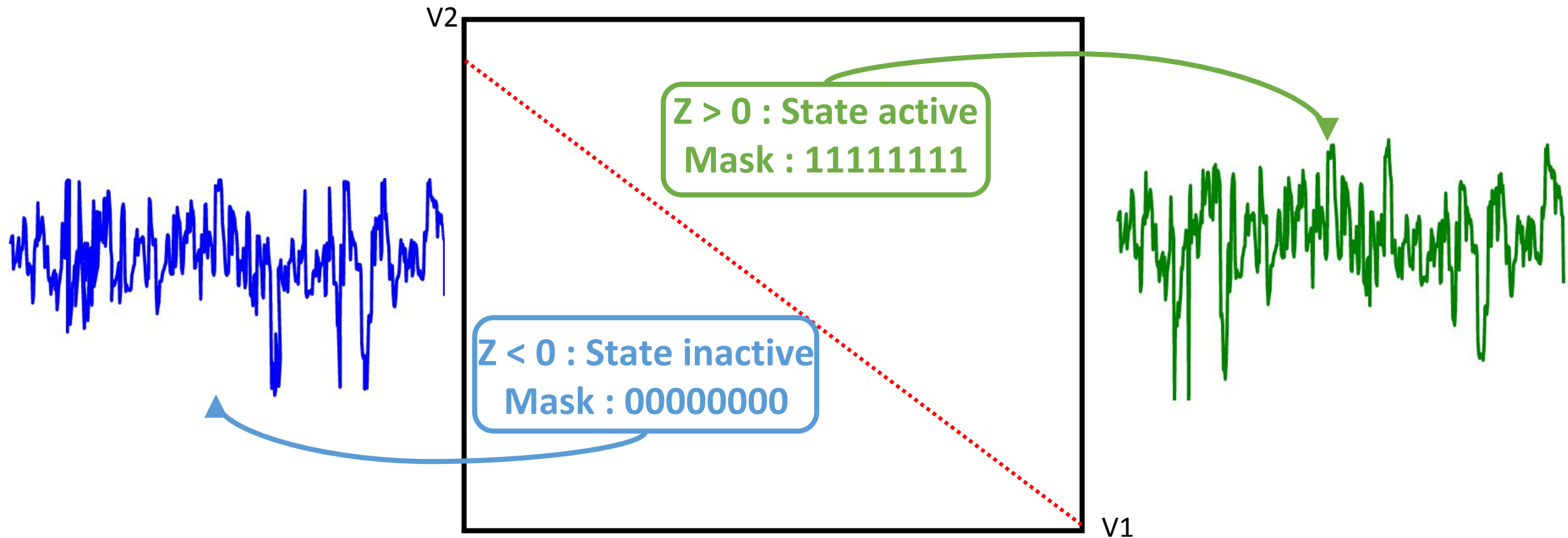
    /* if MSB=1, mask will be 0xFF, 0x0 otherwise */
    mask = QSUB8(0x00000000, buf);

    arm_nn_write_s8x4_ia(&output, in & (~mask));

    i--;
}
```



- ❖ Different states have different electromagnetic traces

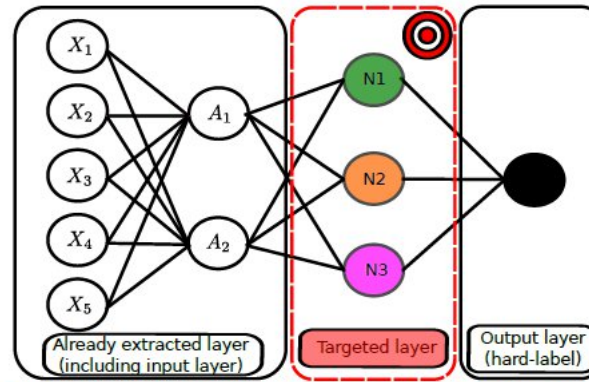


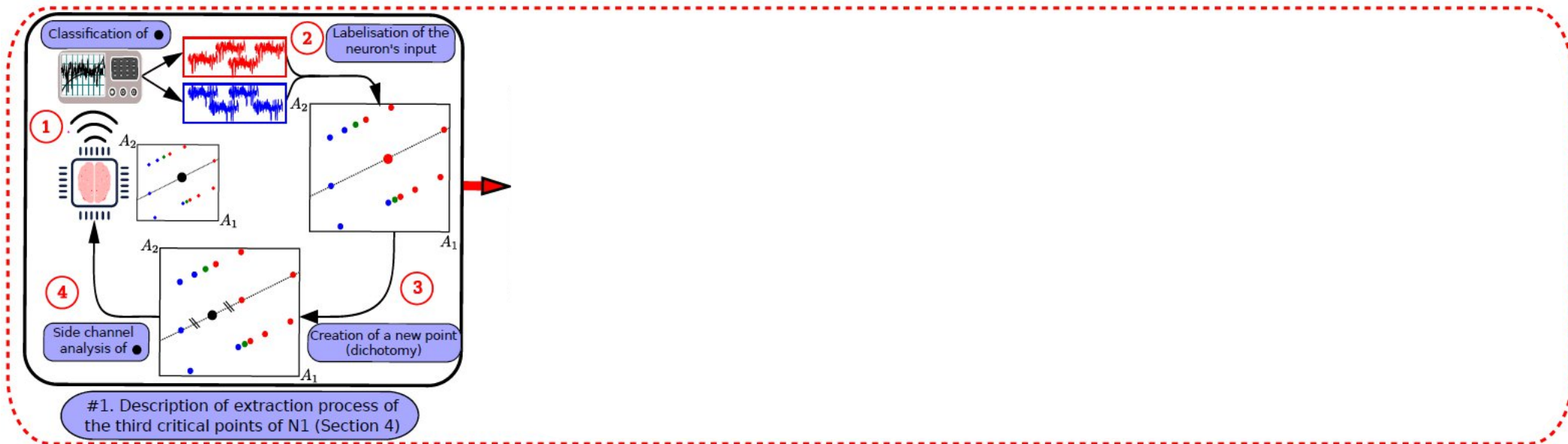
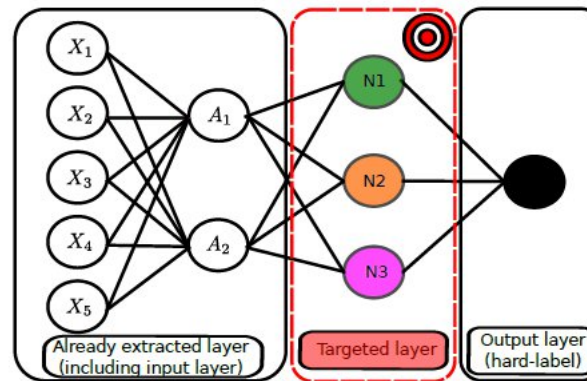
Model Extraction: Our Method



SRE

Inria



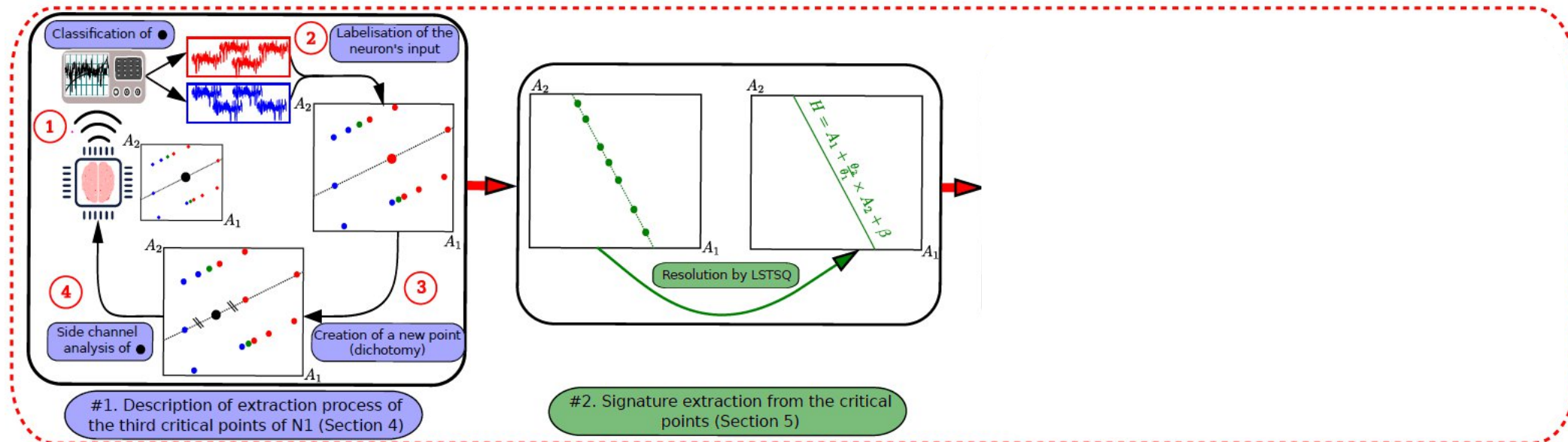
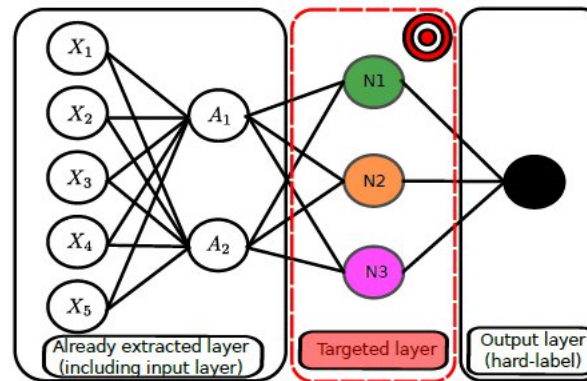


Model Extraction: Our Method



SRE

Inria

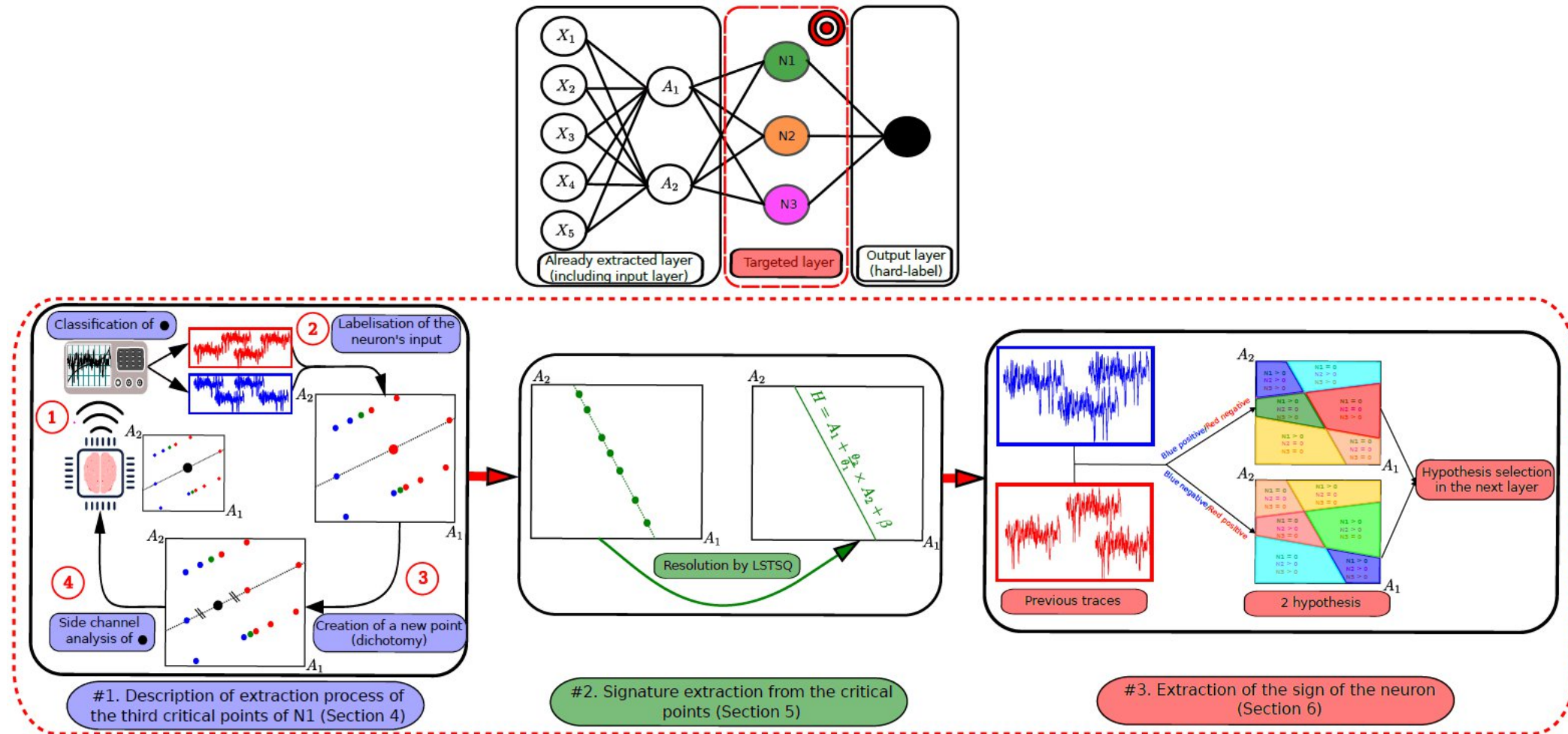


Model Extraction: Our Method



SRE

Inria

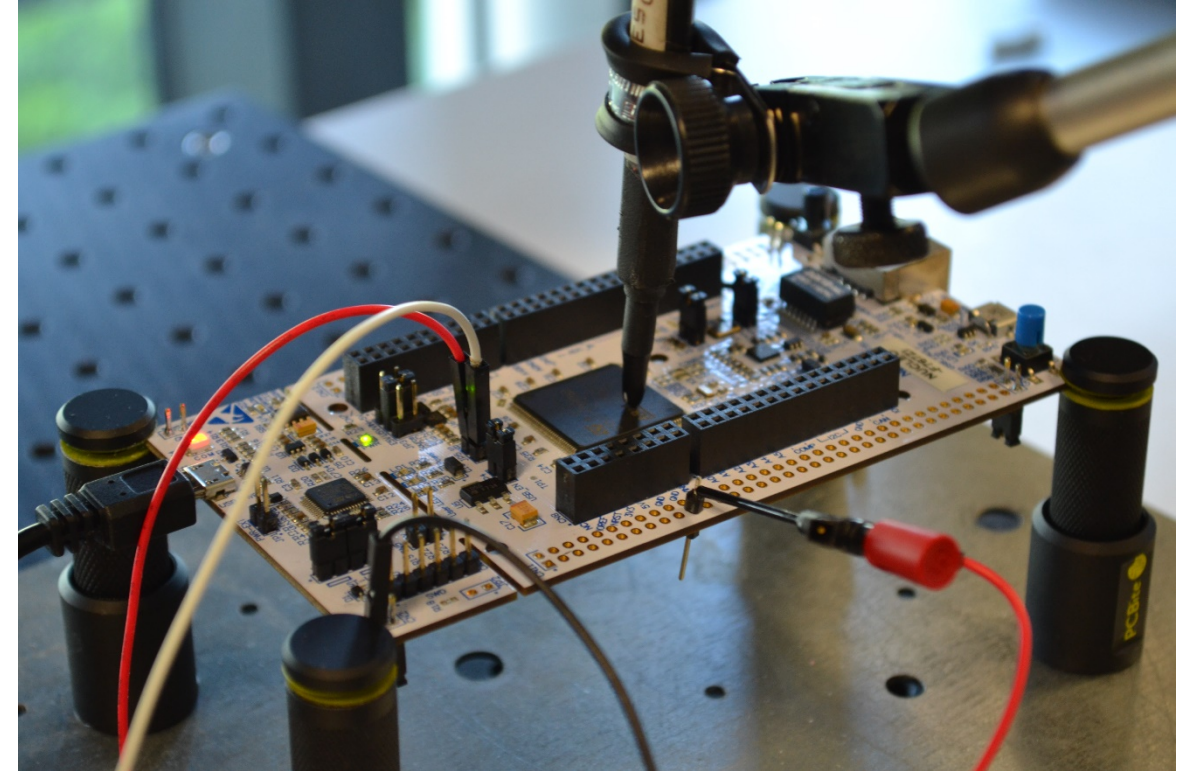


❖ Targeted DNN

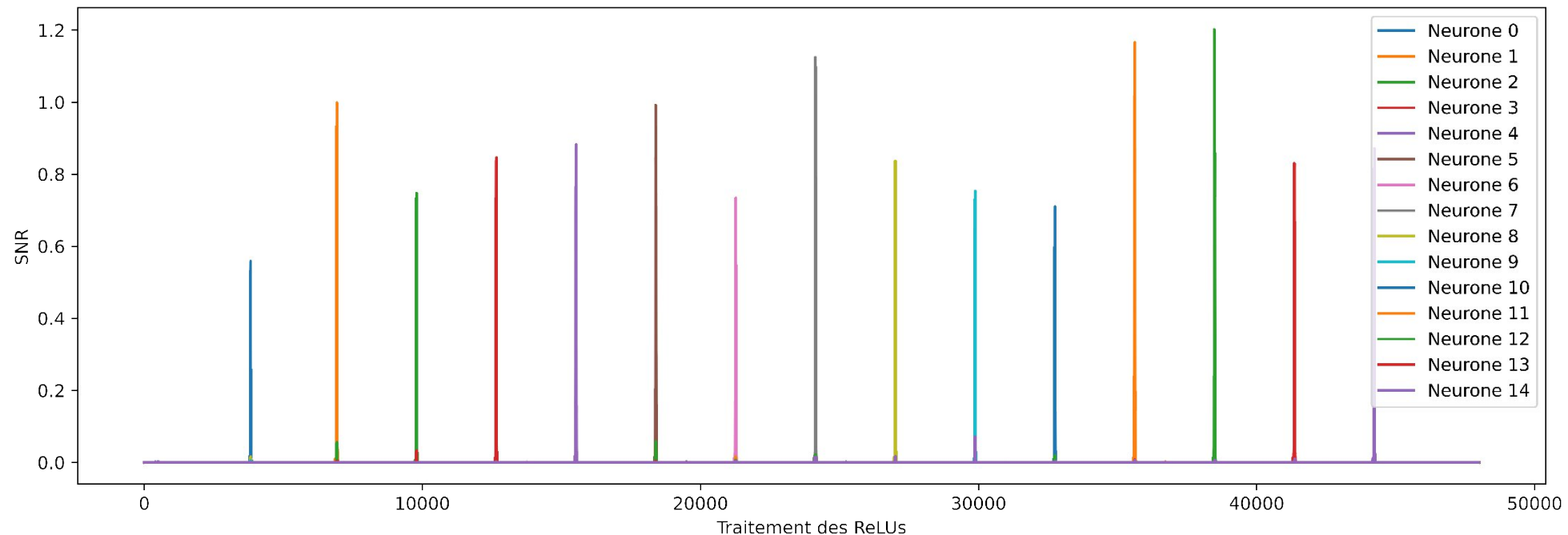
- Truncated version of MobileNetv1
- 11 layers (Depthwise Separable convolutions + batchnorm + ReLU)

❖ Hardware

- STM32F767ZI
- X-Cube-AI



- ❖ State extraction for 15 neurons in a layer
 - Signal to noise ratio on the state of the neuron



- ❖ Success rate in one EM trace: 86.3% (k-means algorithm)


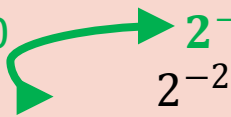

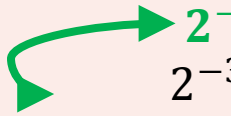


- ❖ Metrics used for classifier: Fidelity, Accuracy Under Attack and Number of queries
 - Fidelity: percentage of label agreement between the stolen and the targeted model (different from accuracy)
 - Accuracy Under Attack: transfer rate of adversarial examples generated on the stolen model to the target
 - Number of queries: number of random queries made to the targeted model (results are given under the assumption that the state of the neuron is obtained in one trace)

- ❖ Metrics used for classifier: Fidelity, Accuracy Under Attack and Number of queries

Architecture	Parameters	Number of queries	Fidelity	Accuracy Under Attack
3072-256-256-256-64-10	935 370	$2^{26.2}$	97.2%	98.6%
3072-512-256-64-10	1 721 802	$2^{26.0}$	93.2%	96.7%
Truncated MobileNetv1	5 234	$2^{18.8}$	88.4%	95.7%

- ❖ One query corresponds to a prediction made by the model on random data ($2^{20} \sim 1\,000\,000$)

❖ Results from simulation with 64-bits data for regression tasks

Architecture (Regression task)	Parameters	Number of queries	$\max \theta - \hat{\theta} $
784-128-1	100 480	x2  $2^{22.6}$ $2^{21.5}$ [5]	x2 700  $2^{-40.8}$ $2^{-29.4}$ [5]
10-20-20-1	620	x4  $2^{15.6}$ $2^{17.1}$ [5]	x700  $2^{-46.5}$ 2^{-37} [5]
40-20-10-10-1	1 110	x2  $2^{16.8}$ $2^{17.8}$ [5]	x32 000  $2^{-42.0}$ $2^{-27.1}$ [5]

❖ One query corresponds to a prediction made by the model on random data ($2^{20} \sim 1\,000\,000$; $2^{-41} \sim 4 \times 10^{-13}$)

❖ Conclusion

- Fidelity-based model extraction of a complex DNN in hard-label settings
- Complementarity between hardware and software attacks
- Paper under review
- Extend this work on more complex architecture
- Evaluate the impact of the data representation on the attack

❖ ST was noticed in September 2024

- [1] Tramèr, Florian et al. “Stealing Machine Learning Models via Prediction APIs.” *USENIX Security Symposium* (2016).
- [2] Rakin, Adnan Siraj et al. “DeepSteal: Advanced Model Extractions Leveraging Efficient Weight Stealing in Memories.” *2022 IEEE Symposium on Security and Privacy (SP)* (2021): 1157-1174.
- [3] Batina, Lejla et al. “CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel.” *USENIX Security Symposium* (2019).
- [4] Jagielski, Matthew et al. “High Accuracy and High Fidelity Extraction of Neural Networks.” *USENIX Security Symposium* (2019).
- [5] Carlini, Nicholas et al. “Cryptanalytic Extraction of Neural Network Models.” *Annual International Cryptology Conference* (2020).
- [6] Shamir, Adi et al. “Polynomial Time Cryptanalytic Extraction of Neural Network Models.” *IACR Cryptol. ePrint Arch.* 2023 (2023): 1526.
- [7] Rolnick, David and Konrad Paul Kording. “Reverse-engineering deep ReLU networks.” *International Conference on Machine Learning* (2019).
- [8] Carlini, Nicholas et al. “Polynomial Time Cryptanalytic Extraction of Deep Neural Networks in the Hard-Label Setting.” *IACR Cryptology ePrint Archive* (2024).

❖ Complete results with 32-bit data

Architecture	Parameters	Queries	$\max \Delta_\theta ^L$
784-32-1	25, 120	$2^{19.8}$	$2^{-17.7}$
784-128-1	100, 480	$2^{21.7}$	$2^{-17.4}$
10-10-10-1	210	$2^{13.0}$	$2^{-18.2}$
10-20-20-1	620	$2^{14.5}$	$2^{-17.8}$
40-20-10-10-1	1, 110	$2^{16.4}$	$2^{-12.1}$
80-40-20-1	4, 020	$2^{19.1}$	$2^{-14.8}$

❖ Complete results with 64-bit data

Architecture	Parameters	Approach	Queries	$\max \Delta_\theta ^L$
10-10-10-1	210	[5]	$2^{16.0}$	$2^{-36.0}$
		[7]	$2^{22.0}$	$2^{-12.0}$
		This work	$2^{15.6}$	$2^{-46.2}$
10-20-20-1	620	[5]	$2^{17.1}$	$2^{-37.0}$
		This work	$2^{15.6}$	$2^{-46.5}$
40-20-10-10-1	1, 110	[5]	$2^{17.8}$	$2^{-27.1}$
		This work	$2^{16.8}$	$2^{-42.0}$
80-40-20-1	4, 020	[5]	$2^{18.5}$	$2^{-39.7}$
		This work	$2^{18.3}$	$2^{-44.2}$
784-32-1	25, 120	[5]	$2^{19.2}$	$2^{-30.2}$
		[4]	$2^{18.2}$	$2^{-1.7}$
		This work	$2^{20.6}$	$2^{-43.5}$
784-128-1	100, 480	[5]	$2^{21.5}$	$2^{-24.7}$
		This work	$2^{22.6}$	$2^{-40.8}$