Side Channels and Deep Neural Network Weights

Attacks, Defences and the Future to Come

–

Journées GT SSLR 2025

Lorenzo CASALINO

CentraleSupélec, IRISA, Inria (SUSHI Team)
lorenzo.casalino@{centralesupelec.fr, irisa.fr, inria.fr}

ANR-21-CE39-0018

Agenda

Table of Contents

Side-channel Analysis

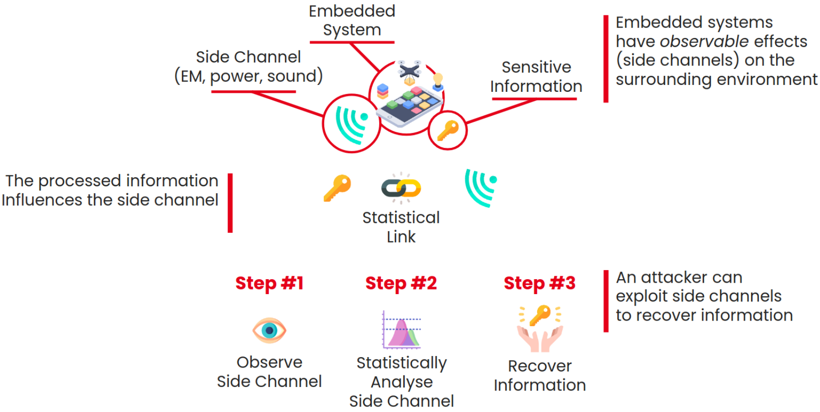


Figure: Information Recovery Through Side-channel Analysis

Deep Neural Networks (DNNs)

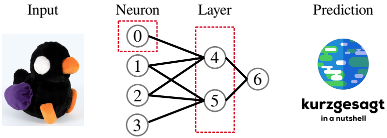


Figure: A simple DNN brand classifier[1].

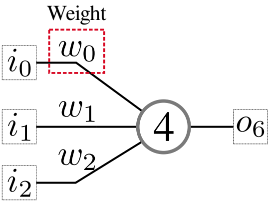


Figure: A neuron computes a weighted sum of its inputs (Eq. 1).

$$o_6 = \overbrace{\underbrace{i_0 \cdot w_0}_{\text{multiply}} + i_1 \cdot w_1}^{\text{accumulate}} + i_2 \cdot w_2 \qquad (1)$$

[1]Duck and Kurzgesagt Logo belong to Kurzgesagt

Motivation

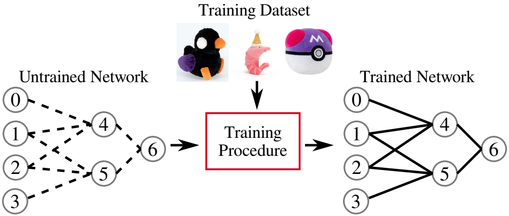


Figure: The training process.[2]

DNN Training is Expensive

- Expensive hardware (e.g., GPUs), time-intensive (e.g., days)

Weights Piracy

A non-negligible **economic damage**

[2] Duck (Kurzgesagt), Shrimp (Jellycat London), Masterball (Nintendo)

Table of Contents

Recovery of Weights – Single Neuron

Weight Recovery Attack

Retrieve **correct** weight value *among all* the accepted ones.

$$o_6 = \overbrace{i_0 \cdot \underbrace{w_0}_{\text{target}}}^{\text{target oper.}} + i_1 \cdot w_1 + i_2 \cdot w_2$$

Attack Complexity for a Neuron

- **Attack Complexity**: $O(N_{weights})$, $N_{weights} = \#\text{weights}$
  - Typical $N_{weights}$: 9 (MobileNet-v2), 25 (GoogleLeNet)
  - Already a non-negligible effort
- But actually ...

Recovery of Weights – Single Neuron

Weights and Data Types

- Weights data type: floating-point or integer
- Weights may have wider or narrower bitwidths (e.g., 32 bits)
- For each data type and bitwidth, attack strategies and complexities change

Work	Type/Width	Complexity (at least)
[Jou+23]	Float/32	$O(2^{16} \cdot N_{\text{weights}})$
[Yos+21]	Int/8	$O(2^{8k} + N_{\text{weights}})$
[Gon+24]	Int/8	$O(2^{16} + N_{\text{weights}})$

Table: Complexity of State-of-the-Art Weight Recovery Attacks (One Neuron).

Recovery of Weights – Whole Network

Attacking the Whole Network

- Attacker can independently target neurons (of the same layer)
- Attack cost linear with number of neurons (N_{neurons})
- DNNs with millions of neurons $\implies$ millions weights ($N_{\text{weights,net}}$)
  - Examples: $\sim 3,4M$ (MobileNet-v2), $\sim 6,8M$ (GoogLeNet)

Work	Type/Width	Complexity
[Jou+23]	Float/32	$O(2^{16} \cdot N_{\text{weights,net}})$
[Yos+21]	Int/8	$O((2^{8k} \cdot N_{\text{neurons}} + N_{\text{weights,net}}))$
[Gon+24]	Int/8	$O((2^{16} \cdot N_{\text{neurons}} + N_{\text{weights,net}}))$

Table: Complexity of State-of-the-Art Weight Recovery Attacks (Whole Network).

Attacks: to Sum Up

Weight Recovery – A Challenging Task

- Weight recovery linear in number of weights
  Example:
  - DNN with 600k weights (in total)[3]
  - Weight Recovery Time: 10 seconds/weight
  - **Recovery time: 69 days**
  - Hidden constants increase the recovery time
  - Not considering other costs (e.g., side-channel acquisition time)
- Attacking beyond input layer adds further difficulty

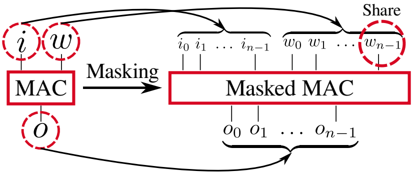
State-of-the-Art Limitations

- Methodologies proved only on really small networks
- Very few works target beyond input layer

[3] Reasonable for microcontroller-oriented DNNs (https://github.com/mit-han-lab/mcunet)

Table of Contents

Masking



Masking

Replace the weight-dependent signal with N random ones (the *shares*)

Advantages

Provably secure side-channel countermeasures

Difficulties

1. Slower, huge (code size/silicon area), and energy-ravenous design
2. Physical non-idealities may lead to information leakage [Cas+23]
3. Huge design limits security evaluation

Shuffling

$$\text{Inference } \#0 : i_0 \cdot w_0 + i_1 \cdot w_1 + i_2 \cdot w_2$$
$$\text{Inference } \#1 : i_1 \cdot w_1 + i_2 \cdot w_2 + i_0 \cdot w_0$$
$$\text{Inference } \#2 : i_2 \cdot w_2 + i_0 \cdot w_0 + i_1 \cdot w_1$$

Shuffling

- Randomly shuffle operations to bury weight-dependent signal in signal noise

Advantages

Less expensive than masking

Difficulties

1. No formal security guarantees
2. Operations (e.g., division) may lead to unintended information leakage [Puš+25]
3. No generic security projections (attacker dependent)

Other Approaches

DNN-Tailored Countermeasures

- Current defences come from cryptanalysis
- But DNNs $\neq$ cryptosystems!
- DNNs exhibit particular characteristics (e.g., error resilience)

Approximate-Computing (AxC)-based Countermeasures

- Trade accuracy for better energy efficiency, size and execution time
- Recently considered as a counteremeasure [Din+25; Jap+25][Cas+26][4]

[4] Paper just accepted at HOST'26

Table of Contents

Challenges

Attack Methodologies

- Linear complexity with #weights
  - But million of weights
  - Non-negligible hidden constants
- No attempts on full DNN models

Defence Methodologies

- Too expensive too deploy, design and evaluate (masking)
- Provide few security guarantees (shuffling)
- Few works proposing countermeasures
- Few security analyses of countermeasures

Narrow Set of Targets

- Most works consider really simple MLPs and CNNs
- No attempts on state-of-the-art DNN models
- Marginal focus on other NNs (e.g., Spiking NNs [PBS25]).

The Future to Come

Better Evaluation Methodologies

- Efficient and Comprehensive (e.g., analyse deeper layers, use all leaked information)
- Explainable (i.e., precisely identify the leakage root cause)

$\downarrow$ to have $\uparrow$

Better Defence Methodologies

- Efficient (i.e., minimise performance overhead)
- Effective (i.e., protect against state-of-the-art attacks)

That's All Folks

Thank You!

Bibliography I

[Bro+24] Manuel Brosch et al. "A Masked Hardware Accelerator for Feed-Forward Neural Networks With Fixed-Point Arithmetic". In: *IEEE VLSI* (2024).

[Cas+23] Lorenzo Casalino et al. "A Tale of Resilience: On the Practical Security of Masked Software Implementations". In: *IEEE Access* (2023).

[Cas+26] Lorenzo Casalino et al. "Double-Strike: Breaking Approximation-based Side-Channel Countermeasures for DNNs". In: *HOST* (2026).

[Din+25] Ruyi Ding et al. "MACPruning: Dynamic Operation Pruning to Mitigate Side-Channel DNN Model Extraction". In: *HOST* (2025).

[Gon+24] Cheng Gongye et al. "Side-Channel-Assisted Reverse-Engineering of Encrypted DNN Hardware Accelerator IP and Attack Surface Exploration". In: *IEEE S&P*. 2024.

Bibliography II

[Jap+25] Aditya Japa et al. "Security of Approximate Neural Networks against Power Side-channel Attacks". In: *IEEE DAC*. 2025.

[Jou+23] Raphaël Joud et al. "A Practical Introduction to Side-Channel Extraction of Deep Neural Network Parameters". In: *Smart Card Research and Advanced Applications*. 2023.

[PBS25] Matthias Probst, Manuel Brosch, and Georg Sigl. "Side-Channel Analysis of Integrate-and-Fire Neurons Within Spiking Neural Networks". In: *IEEE Transactions on Circuits and Systems I* (2025).

[Puš+25] Leonard Puškáč et al. "Make Shuffling Great Again: A Side-Channel-Resistant Fisher–Yates Algorithm for Protecting Neural Networks". In: *IEEE VLSI* (2025).

Bibliography III

[Yos+21] Kota Yoshida et al. “Model Reverse-Engineering Attack against Systolic-Array-Based DNN Accelerator Using Correlation Power Analysis”. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* (2021).

Recovery of Weigths – Hidden Layers

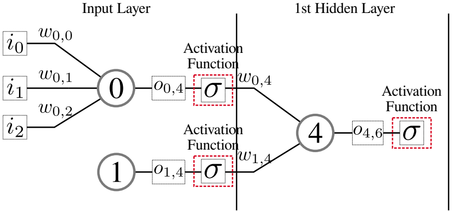


Figure: σ Influences Next Layer's Inputs.

$$\sigma(x) = \begin{cases} x & x \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

Figure: ReLU Activation Function

Attacker Needs Full Input Control

- Hidden layer's input depends on previous layer
- This dependency may forbid hidden layers' weight recovery
  Example:
  - $o_{1,4} = -1.4 \rightarrow \sigma(o_{1,4}) = 0 \rightarrow \sigma(o_{1,3}) \cdot w_{0,4} = 0$
  - Cannot attack $w_{1,4}$!

Recovery of Weigths – Hidden Layers

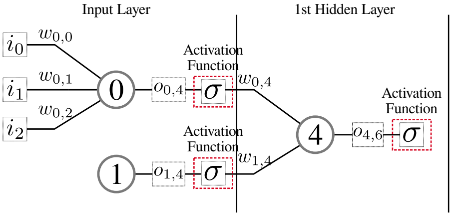


Figure: σ influence Next Layer's Inputs.

$$\sigma(x) = \begin{cases} x & x \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

Figure: ReLU Activation Function

State-of-the-art Solutions [Gon+24; PBS25]

- **Idea**: determine inputs i_j to control $\sigma(o_{h,k})$ (hidden layer's inputs)
- **Caveat**: attack complexity $\propto$ Inputs per layer, Number of layers

Masking – More on the Cost

Table: Software Masked CNN – Execution Time Overheads (Excerpt from [Bro+24]).

Architecture	Masked	Masked (Improved)
(6,5)-(16,5)-256-120-84	$\times$703%	$\times$238%
(16,5)-(32,5)-1568	$\times$306%	$\times$135%

Table: Software Masked CNN – Minimal Storage Requirement (Architectures from [Bro+24]).

Architecture	Original (KBytes)	Masked (2 shares, KBytes)
(6,5)-(16,5)-256-120-84	3,940	7,880
(16,5)-(32,5)-1568	11,072	22,144

Minimal Storage Requirements

$$N_{\text{weights,net}} \cdot N_{\text{shares}}$$