A Review of Confidentiality Threats Against Embedded Neural Network Models

Raphaël Joud, Pierre-Alain Moellic, Rémi Bernhard, Jean-Baptiste Rigaud

2021 IEEE 7th World Forum on Internet of Things (WF-IoT) | 978-1-6654-4431-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/WF-IoT51360.2021.9595434

A Review of Confidentiality Threats Against Embedded Neural Network Models

Raphaël Joud*†, Pierre-Alain Moëllic*†, Rémi Bernhard*†, Jean-Baptiste Rigaud‡

*CEA Tech, Centre CMP, Equipe Commune CEA Tech - Mines Saint-Etienne, F-13541 Gardanne, France
†Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France
Email: {raphael.joud, pierre-alain.moellic, remi.bernhard}@cea.fr
‡Mines Saint-Etienne, CEA Tech, SAS, Centre CMP, Gardanne, France
Email: {jean-baptiste.rigaud}@emse.fr

***Abstract*—Utilization of Machine Learning (ML) algorithms, especially Deep Neural Network (DNN) models, becomes a widely accepted standard in many domains more particularly IoT-based systems. DNN models reach impressive performances in several sensitive fields such as medical diagnosis, smart transport or security threat detection, and represent a valuable piece of Intellectual Property. Over the last few years, a major trend is the large-scale deployment of models in a wide variety of devices. However, this migration to embedded systems is slowed down because of the broad spectrum of attacks threatening the integrity, confidentiality and availability of embedded models. In this review, we cover the landscape of attacks targeting the confidentiality of embedded DNN models that may have a major impact on critical IoT systems, with a particular focus on model extraction and data leakage. We highlight the fact that Side-Channel Analysis (SCA) is a relatively unexplored bias by which model's confidentiality can be compromised. Input data, architecture or parameters of a model can be extracted from power or electromagnetic observations, testifying a real need from a security point of view.**

***Index Terms*—Machine Learning, Deep Neural Networks, Security, Attacks, Side-Channel Analysis**

I. Introduction

An increasing number of edge applications can take advantage of the high effectiveness of Deep Neural Networks (DNN) such as image classification or natural language processing. Based on that, deploying DNN models on hardware platforms catch more and more interest. Running DNN on edge devices offers many advantages. A first one is *power consumption efficiency*, since high-frequency communication with a cloud-based ML services can be prohibitive. Model compression (e.g., quantization, pruning) and inference optimization (e.g., low bitwidth arithmetic) enable to keep reducing models processing consumption costs. Then, model's embedding solves *latency* issues that may occur for applications that rely on important amount of data or need to work at high frequency. Another advantage is linked to *privacy* and *confidentiality* concerns by keeping locally critical information and avoid their transmission to potentially flawed networks.

These advantages explain the massive effort for the deployment of models on a wide variety of devices, especially for IoT applications. For now, this deployment is essentially focused on inference purposes since training still requires important computing capabilities. However, the concerned hardware platforms gather a broad range of integrated circuits, from constrained 32-bit microcontrollers, low-power processors, system-on-chip with HW accelerators to high performance FPGA.

A. Attacks on the Machine Learning pipeline

Among the major obstacles of the deployment of ML models, security issues are notably critical and concern the whole ML pipeline. The traditional ML pipeline gathers the *learning step*, where training data are processed to optimize a (usually) parametric model such as a neural network and the *inference step* where the learned model is used to achieve its task on unseen test inputs. An important amount of works raises major security flaws everywhere in the pipeline with threats targeting integrity (e.g., adversarial examples), confidentiality (e.g., model extraction) and availability (e.g., over-power consumption).

However, contrary to secure applications such as cryptographic modules, designed to be robust against theoretical attacks (i.e. cryptanalysis), embedded ML models suffer from a twofold attack surface:

- *Algorithmic attack surface*: several theoretical flaws have been successfully exposed such as the well-known adversarial examples that enable to fool a model with imperceptible perturbations of the inputs. This kind of attacks are regularly referenced as *Application Programming Interface (API) -based* attacks and we use this notation in this work.
- *Physical attack surface*: being embedded in a device (hardware platform), a model can also be attacked by taking advantage of flaws related to its particular physical implementation, such as side-channel (SCA) and fault injection analysis (FIA).

This extended attack surface provides many possibilities to an attacker since it opens door towards two highly critical problems. First, combining the theoretical evidences from API-based attacks and side-channel analysis may reduce the complexity of a *Model Extraction* task that consists in Intellectual Property theft, as Neural Network can be long, expensive to train and represent a real asset for its owner. On another

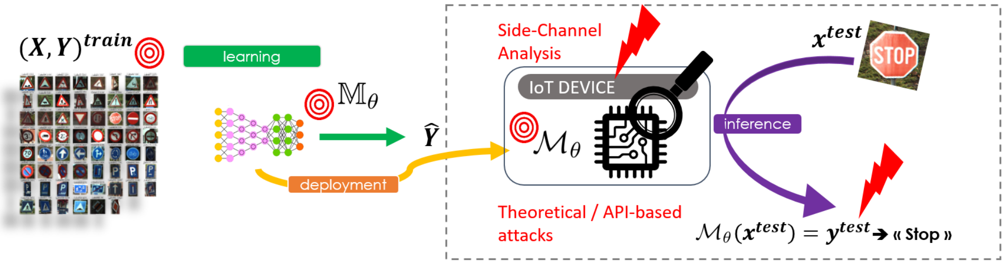

Fig. 1: The traditional (supervised) ML pipeline and the scope of this review (gray dotted area). The threats concern the exploitation of algorithmic flaws as well as SCA techniques to extract model's information or exploit data leakage (red targets).

hand, a successful model extraction attack changes the level of control an attacker has over the system. Attacker can then shift from a black-box context to an almost (if not complete) white-box one. This kind of threat can be dramatic as it would drastically ease the research for adversarial examples [1], [2], membership inference attacks [3] or data leakage exploitation [4]. This context change is a critical risk regarding model's integrity or user's data confidentiality because the attacks previously mentioned keep high effectiveness when transferred from a model to a similar one.

B. Side-channel analysis for embedded DNN models

Physical attack surface leverage model's implementation weaknesses, which can be exploited through Side-Channel Analysis (SCA). Every running program has a specific impact on physical quantities such as power consumption or electromagnetic emanations. SCA aims at exploiting these physical signatures to extract sensitive information related to program's execution. Initially oriented against cryptographic devices, SCA appeared to be extremely powerful over the years and could be used against embedded DNN for model extraction. Several techniques have been proposed to exploit side-channel leakages:

- *Simple Power Analysis* (SPA) is only based on the visual interpretation of *patterns* using a single or few observations (hereafter, called *traces*). This analysis, also mentioned as *Timing Analysis*, relies on some knowledge of the studied program.
- *Differential Power Analysis* (DPA) is focused on the statistical dependencies between several physical measurements and some assumptions about the targeted program. For example, in key-recovery challenge, the assumptions are made by the attacker based on key hypothesis (e.g., 256 for one key byte) and a *leakage model* such as the popular Hamming Weight or Hamming Distance models that link power consumption with the energy used to process or shift one bit.
- *Correlation Power Analysis* (CPA): it is a specific DPA implementation in which the statistical technique used to recover dependencies is the computation of Pearson Correlation Coefficient.

For more details about SCA, we refer interested readers to [5].

C. Contributions and scope

This review aims at alerting to the criticality of confidentiality-based threats against embedded models by highlighting the case of black-box model reverse engineering that may jointly exploits API-based and physical attacks. For physical threats, we limit the scope of our review on SCA since it gathers the most significant efforts. However, Breier *et al.* [6] pave the way to model extraction techniques exploiting fault attacks (bit flip model in [6]). We illustrate the scope of the paper in Figure 1 with a representation of the ML pipeline. This document is organized as follow: after introducing some notations and formalism used throughout this work in Section II, we detail the considered threat model in Section III. Section IV is dedicated to confidentiality issues with the exploitation of training data leakages. The last section (V) proposes state-of-the-art perspectives regarding model extraction techniques.

II. Notations and Formalism

We distinguish a DNN model, $\mathbb{M}$ as an abstract algorithm from its physical implementations $\mathcal{M}$. One model $\mathbb{M}$ (e.g., a convolutional neural network trained on ImageNet in order to classify colored images) may be deployed for inference purpose in a 32-bit microcontroller ($\mathcal{M}^1$), a computer with an advanced CPU ($\mathcal{M}^2$) or a FPGA ($\mathcal{M}^3$). From a purely functional point of view, the models $\mathcal{M}^*$ rely on the same abstraction $\mathbb{M}$ but strongly differ in terms of implementation and their respective hardware environments. Moreover, because of some hardware constraints, the parameters of $\mathcal{M}$ may differ from $\mathbb{M}$ (e.g., weights quantization) and therefore the functionality equivalence is not possible.

A supervised neural network model $\mathbb{M}_\theta$ is a parametric model that aims at mapping the input space $\mathcal{X} = \mathbb{R}^d$ (e.g., images, speech) to the output space $\mathcal{Y}$. For classification, $\mathcal{Y}$ is a finite set of labels. $\mathbb{M}_\theta$ is trained by minimizing a loss function $\mathcal{L}$ that quantifies the error between a prediction $\hat{y} = \mathbb{M}_\theta(x)$ and the ground-truth label y. At inference time, when feed by a input $x \in \mathcal{X}$, the learned model outputs a set of raw predictions (the *logits*) that are mapped in a probabilistic formalism thanks to a softmax function.

III. Threat Model

In this section, we detail the threat model used in this paper as a compulsory step to define properly the adversary's goals and capacities as well as the defender requirements.

Target	Goal	Critical IoT domains
Model ($\mathcal{M} \rightarrow \mathbb{M}$)	IP theft, steal performance, shift to white-box context	Transport, Industry, (Cyber)Security
Training data, Membership	Corrupt privacy	Medical, Biometry, Finance, Defense

TABLE I: Summary of the targets and goals of a confidentiality attack against an embedded DNN with representative related IoT domains.

Attack surface. As previously mentioned, the attack surface in this work is related to an adversary that has physical access to the target device in which a DNN model is implemented.

Adversary's goal. The main objective of the attacker is to strike the confidentiality/privacy of the system. Therefore, he may has several goals:

- target the model by itself: extract the architecture of the model as well as all the parameters (see Section V)
- extract information related to the data (see Section IV), predominantly the training data.

We summarize these goals and targets in Table I with a set of representative critical IoT domains.

Adversary's capacity and knowledge. The more realistic setting is to consider a black-box model, i.e. the adversary has access neither to the internal parameters of the models nor to its architecture.

On the contrary, the adversary can query the device by feeding the model with input data and collect the corresponding outputs. Note that these outputs may be minimal information (only a label) or complete one (e.g., all the prediction probabilities). Moreover, the adversary can perform side-channel analysis on the device (or by using a clone of the device) by observing the power consumption or the electromagnetic emanation of the target thanks to a probe (see a typical SCA set-up illustration in Figure 2). We assume that the attacker targets the inference. Then, he/she has no impact on the training process neither on the deployment phase. However, according to the nature and the context of the task performed by the model, the adversary may have some assumptions on the training data and even on the nature of the model (e.g., computer vision tasks are likely to use convolutional neural networks or model language recurrent neural networks). In the following sections, according to this threat model, we discuss the different adversary's goals by highlighting state-of-the-art results on both theoretical flaws that *directly* exploit input/output pairs and assumptions about the (black-box) target model $\mathbb{M}$; and physical attacks (SCA) that exploit physical observation of the implementation $\mathcal{M}$.

IV. Data Leakage

A. Extraction of Training Data

DNN tend to encapsulate information about the training data that may leak *intrinsically* from the model (i.e. through the exploitation of the model's parameters) or through the model's outputs (labels or scores). Carlini *et al.* [4] highlight critical *unintended memorization* issues for language models trained on a large-scale text data set. Such issue may be solved by Differential Privacy (DP) [7], which aims at limiting the information one may derive from the output about the input.

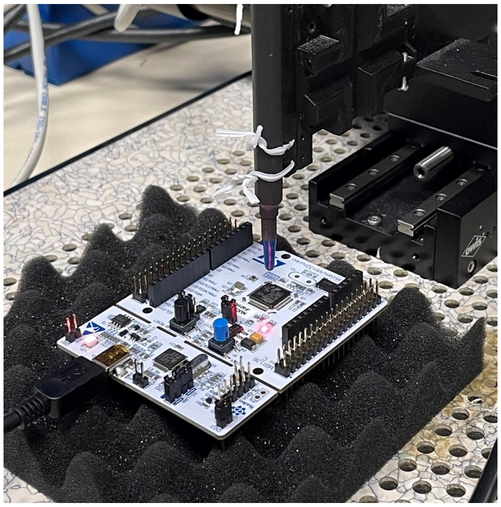

Fig. 2: Classic setup for electromagnetic emanations recording (target is a Cortex M4 32-bit microcontroller.

Another critical threat is known as *attribute inference attack* (or *model inversion*) [8] and mainly concerns applications where input data includes *non-sensitive* features (e.g., public knowledge) and some *sensitive* features that are not available for the adversary. The goal of the attack is to extract the value of these sensitive features with the knowledge of the model output and the *non-sensitive* features. Interestingly, a SCA experiment has been proposed in [9] for a pure model inversion use case, by recovering MNIST (grayscale image of digits) images from the inference power traces of a CNN deployed on a FPGA-based accelerator.

B. Membership Inference

The goal of an adversary mounting a Membership Inference Attack (noted MIA hereafter) against a model $\mathbb{M}$ is to decide whether or not an example x was used or not at training time. For example, considering a model trained with health information gathered from people suffering from a particular disease, this type of attack therefore results in the disclosure of the identity of some people having this disease.

Yeom *et al.* [10] introduce two simple MIAs, which require the adversary to know at least the loss value. The first one is the the "0-1" baseline attack: an example is decided to belong to the training set if it is well predicted. The second one ("Loss-attack") directly links the probability to belong to the training set to the loss value.

Advanced MIA principles have been proposed by [3] with an adversary having access to the confidence output scores and a set of data closed to the ones used for the training. The idea is to train a *shadow* model $\mathbb{M}_c^s$ for each possible label class c. The adversary uses each M_c^s to build a set $((c, \hat{y}), in/out)$, by querying train (in) or test (out) points used to train M_c^s, to get the confidence vector $\hat{y}$. This set, consisting of confidence score vectors labeled by their membership to the train or test

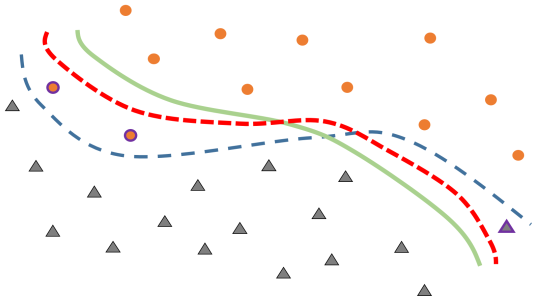

Fig. 3: Illustration from [22]. Original model's decision frontier is modeled with the green line. Red-dashed line is the *Fidelity*-oriented reconstruction as it matches predictions in the same manner as the original green curve. Whereas blue-dashed line is an *Accuracy*-oriented approach: incorrect prediction from green curve are corrected.

is then used to train an *attack model*, taking as inputs the $\hat{y}$ and as outputs the in or out. [11] shows that this attack is also feasible when relaxing assumptions on the strength of the adversary, keeping high efficiency rate.

To mitigate this type of attacks, many defense schemes aim at adding noise to the confidence output vector [12]–[14]. However, these protections have been thwarted by so-called ”Label-Only” MIAs, which rely only on the output label [15]. The efficiency of this attack compared to white-box attacks has already been observed experimentally [16]. Moreover, the optimal strategy to perform MIAs has been formally derived, and it shows that an adversary only needs to have access to the classification loss value [17]. Therefore, the white-box setting does not bring any advantage in terms of MIA success.

Many efforts have been devoted to identify factors that influence the success of MIAs [18]: the size of the training set, the number of classes and the intra-class standard deviation, the mutual information between outputs and inputs, or the amount of generalization. However, all these factors are just a cause for the reduction of the generalization capacity [19]. Thus, [20] proposes to reduce the generalization gap (difference between train and test set loss) by adding to the training loss a measure of the performance gap between training and validation samples. This simple procedure aims at minimizing the difference between probability scores for training and validation examples.

Another line of defense against MIAs focuses on Differential Privacy. However, such defenses face a classical issue with DP since it has been observed that for a certain privacy level needed to thwart MIAs, the accuracy of the target model was decreased in a non-acceptable way [21].

V. Model Extraction

Being an important intellectual property, a DNN model is a privileged target that an adversary aims at reverse engineering especially if the black-box model is insufficiently protected. In the following part, we present different shades of model extraction attacks starting with algorithmic-oriented techniques. Then, we broach SCA-based attacks. All the attacks mentioned in the following part are gathered in table II.

Most of published works using side-channel analysis against embedded neural networks are focused on model extraction. However, it is important to distinguish what kind of extraction is to be performed. Jagielski *et al.* [22] define a clear difference between attacks targeting **Fidelity** and **Accuracy**:

- Fidelity measures how well extracted model predictions match those from the original model (victim target of the extraction attack).
- Accuracy aims at performing well over the underlying learning task of the original model based on a testing data-set.

Figure 3 illustrates these concepts (from [22]).

A. API-based model extraction attacks

According to previous notions, it is possible to distinguish different goals regarding extraction strictly speaking. An adversary may want to precisely extract model's characteristics in order to obtain a kind of clone. By achieving this goal, the attacker tremendously enhances his level of control over the system allowing to go from a black-box context to a white-box one. Attacker would then be able to easily design other attacks harming model's integrity and data confidentiality. On the other hand, attacker might want to steal the “performance” of an already-trained model for least possible effort without paying much interest to the exact value of model's characteristics. Indeed, high-performance neural network can take a significant amount of time and investment to be trained and deployed. Simply stealing its performance would avoid the attacker to go through a long and expensive training phase and harm the individuals or company behind the original model. This kind of attack is largely underrepresented in model extraction related publications. However, in [22], Jagielski *et al.*take an interest about these attacks with the idea that model predictions are more informative than training data labels. They compare the performance achieved when using a fully-supervised learning or semi-supervised learning upon part of the original training data set. Performance reached when using semi-supervised learning are significant and according to the publication, could be further improved by combining active-learning techniques with semi-supervised learning.

Considering fidelity-oriented model extraction attacks in the black-box context, the attacker can interact with a reduced number of elements. Generally, he can provide inputs to the model and receive model's prediction, then the adversary could be able to craft special inputs that would reveal highlighted information. Based on this idea, [22] work on models using ReLU activation function in order to use specially crafted inputs that put exactly one neuron of the hidden layer in a critical state. This critical state forces one neuron to change sign and allows to identify the weight associated to it. By doing this for every neurons, the authors are able to reconstruct the entire weight matrix of the hidden layer, noting that models involved in this work are structured with only one hidden layer. This methodology is extended to models with more than one hidden layer in [23] by considering the model extraction as a cryptanalytic problem with a differential attack.

Attack	Implementation	$\mathbb{M}$	$\mathcal{M}$	Specificity
Carlini *et al.* [23]	N.S.	✓		For ReLu Multi Layer Perceptron
Jagielski *et al.* [22]	N.S.	✓		For ReLu Multi Layer Perceptron
Oh *et al.* [24]	N.S.	✓		ML model able to recognize other models
Batina *et al.* [25]	µC		✓	SCA
Regazzoni *et al.* [26]	µC		✓	SCA
Maji *et al.* [27]	µC		✓	SCA
Xiang *et al.* [28]	µC	✓	✓	SVM to classify DNN architecture from SC traces
Hua *et al.* [29]	FPGA		✓	Memory-access pattern and timing SC
Dubey *et al.* [30]	FPGA		✓	CPA and advanced leak-model
Yu *et al.* [31]	FPGA	✓	✓	EM SC traces (model's architecture) and AE to train substitute model
Breier *et al.* [6]	N.S.		✓	Fault Injection Analysis (FIA)

TABLE II: Summary table of model extraction attacks mentioned in this work and their specificity. Abbreviations: NS = Not Specified, µC = Microcontroller, $\mathbb{M}$ refers to API-based attacks and $\mathcal{M}$ to Physical Attacks (e.g., SCA)

Chosen or crafted-query approach is also exploited in [24] but the attack relies on a *metamodel* trained to predict the attributes of a black-box model, as a *classifier of classifier*. Working with MNIST classifiers, different versions of this *metamodel* are developed and combined in a way that, in the end, the *metamodel* is able to reveal information about model's architecture and hyperparameters even with restricted output information (label-only) or difficult black-box contexts.

B. Side-Channel Analysis for Model Extraction

From a pre-trained DNN model, an attacker with a fidelity-based objective would want to extract the model's architecture, the parameters or the activation functions.

Fidelity extraction attack can benefit from the information provided by hardware implementation, especially revealed through SCA. Moreover, context restrictions associated to purely software and mathematical extraction can be diminished by using physical attacks. In [25], no specification is imposed to the model topology or activation function. They propose an attack based on electromagnetic emanations emitted by a microcontroller that allows to re-construct a model. The attack is composed of three main steps: *1)* extraction of weights associated to each neurons with method similar to CPA with hypothesis about value of the weight; *2)* distinction of the layer to which the neuron belongs with SCA techniques; *3)* guessing of the activation function based on timing analysis. This procedure is repeated layer after layer to recover the entire architecture. This attack is also extended to CNN in the paper, requiring only small changes. A similar methodology is evoked in [26] and presents comparable results.

In a recent work, Maji *et al.* [27] demonstrate that even simple power and timing analysis enable to recover an embedded neural network model with different precision (floating point, fixed point and binary models) on three microcontroller platforms (8-bit ATmega328P, 32-bit ARM Cortex-MO+ and 32-bit RISC-V RV32IM). Interestingly, to thwart the timing side-channel leakages, they propose two countermeasures based on a constant-time implementation of the ReLU operation (for 8-bit input) and a ad hoc floating-point representation.

From the observation that embedded neural networks can also be based on already-existing and popular architectures (such as MobilNet, AlexNet or ResNet), Xiang *et al.* [28] propose to identify the structure of models deployed on a Raspberry Pi using power consumption traces. The identification, performed by a Support Vector Machine classifier, is allowed by the fact that each architecture is a combination of different layers having their own consumption profile.

Most of the works related to model extraction are focused on CNN embedded on FPGA accelerator. In [29], the authors use the fact that CNN states and parameters are stored in off-chip memory due to their size. They use memory-accesses and timing side-channel to reveal information like architecture (number of layers, data dependencies among layers, size of features maps and kernel filters) or weights from AlexNet and SqueezeNet. Weights extraction is based on utilization of functions that dynamically prune zeros resulting from ReLU activation function. With use of such function, a change in the number of non-zero values in the Output Feature Map (OFM, output of a layer) reveals that a value from the Input Feature Map (IFM, input of the layer) has crossed the 0 boundary. By manipulating content of IFM and controlling apparition of non-zero values in OFM, attacker can recover weights characterizing a layer.

[30] uses CPA to extract weights and bias from hardware implementation (Adder Tree) by focusing on switching activity of registers. The authors propose an advanced leakage model based on Hamming Distance. For example, the registers of the Adder Tree may contain the result of multiplication operation between a secret weight and a known input. An important part of this work presents the implementation of a countermeasure focused on masking all intermediate computations and occurs at different level of the model's implementation (Adder Tree, Activation Function ...).

In [31], Yu *et al.* demonstrate a joint exploitation of SCA and a typical model extraction technique. They first extract the architecture of a black-box binary neural network model embedded on FPGA using electromagnetic traces. Then, a substitute model with the extracted structure is trained with a data set built from prediction returned by the victim model when specifically-crafted adversarial examples were passed to it.

VI. Conclusion

This paper proposes a review of the weaknesses that can critically threaten the confidentiality of neural network models deployed in typical IoT devices. Because the attack surface encompasses theoretical (algorithmic) flaws *and* physical implementations threats, we particularly highlight two major issues – model extraction and training data leakage – that are representative to the use of side-channel analysis. We claim that model extraction is a major threat, with powerful attacks that have been recently demonstrated. Not only a model extraction enables to steal IP but also shifts an adversary into a harmful white-box paradigm. By knowing the internals of a model, an attacker may craft more advanced privacy-based attacks or integrity threats such as adversarial examples. For now, most *fidelity*-oriented model extraction attacks are focused on a non-material attack perspective and few works are dedicated to typical IoT platforms (such as 32-bit micro-controller), mainly by using side-channel analysis to extract the architecture of a target model. More attention has to be paid about the joint exploitation of these techniques that may lead, in a very near future, to powerful attacks for a wide range of state-of-the-art models.

Acknowledgments

This work is a collaborative research action that is partially supported by (**CEA-Leti**) the European project ECSEL InSecTT[1] and by the French National Research Agency (ANR) in the framework of the *Investissements d'avenir* program (ANR-10-AIRT-05, irtnanoelec); and supported (**Mines Saint-Etienne**) by the French funded ANR program PICTURE (AAPG2020).

References

[1] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017.

[2] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Security and Privacy (SP), 2017 IEEE Symposium on*, 2017.

[4] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," 2019.

[5] S. Mangard, E. Oswald, and T. Popp, *Power Analysis attacks: Revealing the secrets of smart cards*. Power Analysis Attacks: Revealing the Secrets of Smart Cards, 2007. Pages: 337.

[6] J. Breier, D. Jap, X. Hou, S. Bhasin, and Y. Liu, "Sniff: Reverse engineering of neural networks with fault attacks," *arXiv preprint arXiv:2002.11021*, 2020 https://arxiv.org/pdf/2002.11021.pdfurl.

[7] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, 2014.

[8] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," 2018.

[9] L. Wei, B. Luo, Y. Li, Y. Liu, and Q. Xu, "I know what you see: Power side-channel attack on convolutional neural network accelerators," in *Proceedings of the 34th Annual Computer Security Applications Conference*, pp. 393–406, 2018.

[10] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," 2017.

[11] A. Salem *et al.*, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018.

[12] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[13] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[14] M. Xue *et al.*, "Use the spear as a shield: A novel adversarial example based privacy-preserving technique against membership inference attacks," *arXiv preprint arXiv:2011.13696*, 2020.

[15] C. A. C. Choo, F. Tramer, N. Carlini, and N. Papernot, "Label-only membership inference attacks," *arXiv preprint arXiv:2007.14321*, 2020.

[16] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," *arXiv preprint arXiv:2003.10595*, 2020.

[17] A. Sablayrolles *et al.*, "White-box vs black-box: Bayes optimal strategies for membership inference," 2019.

[18] S. M. Tonni, F. Farokhi, D. Vatsalan, and D. Kaafar, "Data and model dependencies of membership inference attack," *arXiv preprint arXiv:2002.06856*, 2020.

[19] L. Song, R. Shokri, and P. Mittal, "Privacy risks of securing machine learning models against adversarial examples," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[20] J. Li, N. Li, and B. Ribeiro, "Membership inference attacks and defenses in supervised learning via generalization gap," *arXiv preprint arXiv:2002.12062*, 2020.

[21] M. A. Rahman, T. Rahman, R. Laganiere, N. Mohammed, and Y. Wang, "Membership inference attack against differentially private deep learning model.," *Transactions on Data Privacy*, vol. 11, no. 1, 2018.

[22] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pp. 1345–1362, 2020.

[23] N. Carlini, M. Jagielski, and I. Mironov, "Cryptanalytic Extraction of Neural Network Models," *arXiv:2003.04884 [cs]*, July 2020. arXiv: 2003.04884.

[24] S. J. Oh, B. Schiele, and M. Fritz, "Towards reverse-engineering black-box neural networks," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 121–144, Springer, 2019.

[25] L. Batina, S. Bhasin, D. Jap, and S. Picek, "{CSI} {NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 515–532, 2019.

[26] F. Regazzoni *et al.*, "Machine Learning and Hardware security: Challenges and Opportunities -Invited Talk-," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, vol. 2020-November, pp. 1–6, 2020.

[27] S. Maji, U. Banerjee, and A. P. Chandrakasan, "Leaky nets: Recovering embedded neural network models and inputs through simple power and timing side-channels–attacks and defenses," *IEEE Internet of Things Journal*, 2021.

[28] Y. Xiang, Z. Chen, Z. Chen, Z. Fang, H. Hao, J. Chen, Y. Liu, Z. Wu, Q. Xuan, and X. Yang, "Open DNN Box by Power Side-Channel Attack," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 11, pp. 2717–2721, 2020.

[29] W. Hua, Z. Zhang, and G. E. Suh, "Reverse Engineering Convolutional Neural Networks Through Side-channel Information Leaks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2018.

[30] A. Dubey, R. Cammarota, and A. Aysu, "MaskedNet: The First Hardware Inference Engine Aiming Power Side-Channel Protection," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 197–208, Dec. 2020.

[31] H. Yu, H. Ma, K. Yang, Y. Zhao, and Y. Jin, "DeepEM: Deep Neural Networks Model Recovery through em Side-Channel Information Leakage," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 209–218, IEEE, 2020.

[1] www.insectt.eu, InSecTT: ECSEL Joint Undertaking (JU) under grant agreement No 876038. The JU receives support from the European Union's Horizon 2020 research and innovation program and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands, Turkey. The document reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains.