The Dark Side of Neural Networks: an Advocacy for Security in Machine Learning

Pierre-Alain Moëllic

CEA Tech, Centre CMP, Equipe Commune CEA Tech - Mines Saint-Etienne,
F-13541 Gardanne FRANCE
pierre-alain.moellic@cea.fr

Abstract. The revival of neural networks has significantly accentuated the development and proliferation of Machine Learning (ML) systems. This trend is intensifying with major focus on embedded systems with sometimes strong architectural constraints. As often happens with such technological disruptions, the main efforts are massively focused on performance relegating security in the background. Yet, since neural networks-based systems will be ubiquitous, it is essential to wonder whether such systems are not intrinsically sensitive to threats targeting their integrity, confidentiality or accessibility. Thanks to previous works, we know that an attacker can fool a supervised classifier or even extract confidential data or IP leaking from a model. In this paper, we propose an overview of attacks that threaten ML algorithms and particularly neural networks with a focus on *adversarial examples* which have strong properties like transferability. Physical attacks (side-channel analysis, fault injection) will also be presented as upcoming attack vectors against embedded ML. Then, we discuss some state-of-the-art protection schemes as well as the importance of methods and tools to evaluate the robustness of models that could help the dissemination of good practice for the design and development of durable and secure ML systems.

Keywords: Machine Learning · Deep Neural Networks · Security · Attacks · Adversarial Examples · Protections · Evaluation · Review

1 Introduction

A.I. deployment. The context of this paper is the growing importance of security purposes for machine learning (ML) systems. Yet, this recognition is insufficiently widespread, particularly if we jointly consider the expected level of dissemination within a large variety of critical devices and infrastructures and the significant body of publications describing severe security issues.

Even if ML is not in its early years, the so-called *Deep Learning* has significantly accentuated its development and deep neural networks (DNN) are planned to be deployed at large-scale, offering "intelligence" for a wide range of devices with, obviously, IoT as a major market. However, an overwhelming part of the research and technical efforts are related to performance and nothing but performance: ML systems have to learn (even more) *better* and *faster* and, now, *lighter*.

In such a context and whatever the alarms raised by the scientific community, security still seems to be an option. Yet, with ubiquitous DNN-based systems the question of their sensitivity to critical threats becomes crucial. The answer is already known in the ML community thanks to major works that sound the alarm and invite not only researchers but also system developers, providers and end-users to make security a core subject. This understanding is fundamental to foresee and thwart – as far as possible – threats that could have disastrous impacts on the development, deployment and durability of ML systems.

Machine Learning System In this review, we focus on systems processing large amount of data to perform a specific task through algorithms which are said to *learn*, in that their performance in achieving this task improves with experience [1]. Learning is achieved using (1) examples – then, *learning* is *training* – associated to (*supervised learning*) or without (*unsupervised learning*) knowledge related to the task (e.g., discrete labels for classification) or using (2) environment rules and rewards to perform actions (*reinforcement learning*).

The output of learning is a *model* that is later applied to perform the task. Traditionally, a ML system can be decomposed into two components: one produces the model during the learning process and a second performs the task (*inference* or *test*). This decomposition makes sense since training usually requires significantly more computing resources than inference: once the model is built, it could be deployed in platforms with limited computing, precision or memory capabilities. For the sake of clarity, we will focus this paper on supervised learning and DNN since these algorithms meet considerable interests for some years. However, we will see that most of the attack principles described thereafter are suitable to a large variety of ML algorithms as well as other learning paradigms.

Notation Despite some exception, a supervised ML algorithm is a parametric mapping function f_θ from the input space $X = \mathbb{R}^d$ (e.g., images, speech) to the output space Y. Θ is the set of parameters to be learned. For image classification into C classes, f_θ maps images to the space of labels $Y = \{1, ..., C\}$. The goal of learning is to find the optimal parameters minimizing a cost function $\mathcal{J}$ that quantifies the error between the prediction $\hat{y} = f_\theta(x)$ and the *true* output, y, over the training dataset. The output $f_{\theta^*}(x)$ is the so-called *learned model*:

$$\Theta^* = \arg\min_\theta(\mathcal{J}(f_\theta(X^{train}), Y^{train})) \tag{1}$$

The *machine learning pipeline* (see Fig. 1) is as follows: the data used for training are samples of X and composed the *training dataset*: X^{train}. A fraction of that dataset, X^{val}, can be used to intrinsically validate the learning process in order to evaluate the behavior of f_θ on *unseen* data. At inference time, the learned model f_θ is used on new samples from X: X^{test}. For classification tasks, the output $f_\theta(x)$ is a probability vector that x belongs to each class of Y.

We start this review by modeling the treaths (Section 2), then we analyze the main attacks against ML systems (Section 3) with a particular focus on

adversarial examples (Section 4). The defense strategies (and evaluation works) will be detailed in Section 6 after the presentation of new attack vectors based on the physical implementation of ML algorithm in Section 5.

2 Threat Modeling

The first step of any security analysis is the identification of the goals and capabilities of attackers as well as the intrinsic vulnerabilities of the system. We highlight the remarkable work of Papernot et al. [2] that proposes an in-depth description of the threat scenarios with related references.

2.1 Objective of an attack.

The goal of an adversary is traditionally well defined by the information security triad: Confidentiality–Integrity–Availability.

- **threaten confidentiality or privacy** concerns both the data and the model. An adversary may want to extract secret or sensitive information (e.g., banking information, medical records, classified data) processed during training [3] or inference. If the model is proprietary or restricted to specific users, the attacker may want to steal (even partially) information about the model thanks to reverse engineering techniques [4] such as the DNN structure and its parameters (e.g., number of layers, weights). The attack surface of this leakage scenario is also discussed in Section 5.
- **targeting the integrity** means that the attacker wants to deflect the nominal behavior (e.g., classify data with a certain confidence) by altering the expected outputs. Several tampering modalities can be defined according to the precision and ambition of the attack (e.g., target or non-targeted adversarial examples).
- **threaten the availability** means that the adversary targets the overall system or the environment hosting the ML algorithms so that it will become unavailable (like a *Denial-of-Service* attack) or the attacker deteriorates so seriously the system performance or behaviors that it becomes useless.

Figure 1 (from [5]) illustrates the principal threats on the ML pipeline.

2.2 Adversary's knowledge and ability.

To achieve his goal, an adversary has limited or, on the contrary, extended capabilities. The first important criterion is his ability to perform an attack at training or/and at inference time. In the first case, the attacker directly targets the model's building, e.g. by poisoning X^{train}. For inference-based attacks, he corrupts data to fool the model. The level of knowledge available about f_θ is another essential criterion for defining the attack scenario: in a *white-box* attack the adversary has strong knowledge of f_θ, and even X^{train}, contrary to the *black-box* paradigm where the attacker has no (or very limited) information about the model, i.e. investigating the model properties with mirroring or blinded approaches is required.

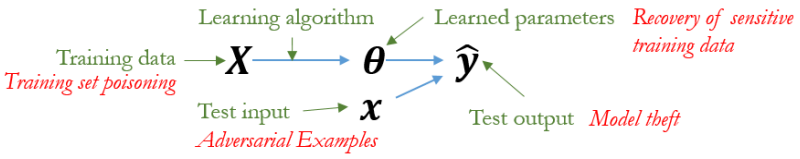

Fig. 1. From I. Goodfellow [5]: attacks on the machine learning pipeline.

3 Attack Deep Neural Networks

3.1 Targeting integrity

The most studied integrity-based attacks are *adversarial examples* and will be detailed in section 4. Symmetrically, another relevant integrity-based attacks are *poisoning attacks* that work at training time by tampering X^{train}. Poisoning attacks are well-known for SVM [6] and have been rapidly proposed for DNN [7, 8]. Basically, the attack aims to deteriorate the relevance of f_θ by adding poisoned samples in X^{train}. An important remark is that poisoning attacks enable to target integrity or availability depending on the level of alterations. If the adversary aims to corrupt the integrity of the DNN, he will perform focused misclassification on a precise subset of the labels in such a way the system stays within an overall nominal behavior. As for adversarial examples, the methods use the gradient information of the cost function with several optimization strategies (see [8]).

3.2 Targeting confidentiality and privacy

Data leakage. Both learning data and model represent confidential and valuable information. Extracting data captured or memorized within a model (a strong issue for DNN) can be critical in several domains such as medical applications. For example, Carlini et al. [9] show that secret information (namely social security and credit card numbers) can be extracted from a model (a classical LSTM-based language model) trained with large-scale emails collections.

Another important issue is known as *membership attacks* where an attacker tries to know if a target input belongs to X^{train} which, in some particular case, can lead to serious privacy breach. Membership inference is not exactly the same as *model inversion* [3] which infers some characteristics of an input thanks to the model's output but cannot provide the membership information $x \in X^{train}$. Shokri et al. [10] propose a reference study of membership attacks targeting well known ML-as-a-service platforms (Google Prediction API, AmazonML): thanks to a shadow dataset, $\widetilde{X}$, the adversary is able to train *and* test a set of k shadow models, $\{f_k^{shadow}\}$, to perform the same task as the target model, f^{target}. The prediction outputs for $\widetilde{X}_k^{train}$ and $\widetilde{X}_k^{test}$ are respectively used as the *in* (membership) and *out* (not membership) labeled data that feed the training of a final attack model, f^{attack} . Finally, the adversary can query f^{target} with

a target input, x^*, and uses the attack model (in/out), $f^{attack}(f^{target}(x^*))$ to know if x^* belongs to the original X^{train} (see Fig. 2).

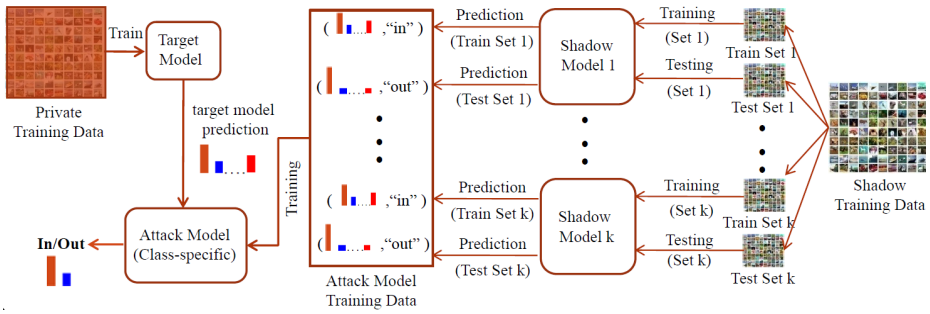

Fig. 2. From [11]: an illustration of the membership attack proposed in [10].

Model leakage. In the same way, an attacker may reverse-engineer a protected model by extracting information about its structure or parameters. This point is particularly interesting since physical attack vectors are emerging, taking advantage of the large background of side-channel and fault injection analysis (see section 5). In [4] a *model extraction* attack is performed against online ML platforms (BigML and Amazon, both offering multilayer perceptron, decision tree and logistic regression) by simply querying – as less as possible – the target model to efficiently approximate a shadow model. The authors show that even with minimal outputs (only the label with the maximum confidence), partial information about the model can be extracted with optimized selection of queries.

4 A critical threat: adversarial examples

4.1 Definition

An adversarial example is an input x^*, crafted from an initial sample x, by an optimized process that is misclassified by a DNN even though it is the result of small (even imperceptible) perturbations (see Fig. 3). Since the definition of adversarial examples clearly evolved over time, I. Goodfellow [5, 12] highlights the fact that the *imperceptible* property that is usually assigned is not required because even a visible perturbation can fool the human perception if it is semantically coherent in the context of the task. Thus, the relevance of adversarial examples have to be judged with accuracy-based metrics and beyond the human perception since it is far to be an absolute truth [13] . A taxonomy of adversarial examples is illustrated in Fig. 4.

Formally, Szegedy et al., who first mentioned the terms "*adversarial examples*" [14], proposed the following box-constrained optimization problem:

$$\arg\min_{r} f_\theta(x+r) = l \quad (l \neq f_\theta(x)) \text{ with } x^* = x + r \in X \tag{2}$$

With, r a small perturbation applied to an input x so that x^* is still in the input space X, and l the (mis)classification output.

At first glance, adversarial examples go against the capacity of a DNN to generalize. Several works [14, 15] try to explain this phenomenon and demonstrate that adversarial examples are not related to any overfitting problem neither are concentrated in some blind holes, pockets or other peculiarities related to the complex geometry of the decision boundaries. On the contrary, adversarial examples have been shown to be well distributed in X but very precisely. I. Goodfellow gives a pertinent representation: "*adversarial examples finely tile space like the rational numbers among the reals*". From a statistical point of view [16], X^{train} cannot perfectly sample the complexity of the high-dimensional space X, the model manages to generalize (i.e. the model decision boundary partially fit the *real* one, see Fig. 3) and correctly classifies unseen inputs because they are *likely* samples but it fails with adversarial examples that are statistically unlikely to occur.

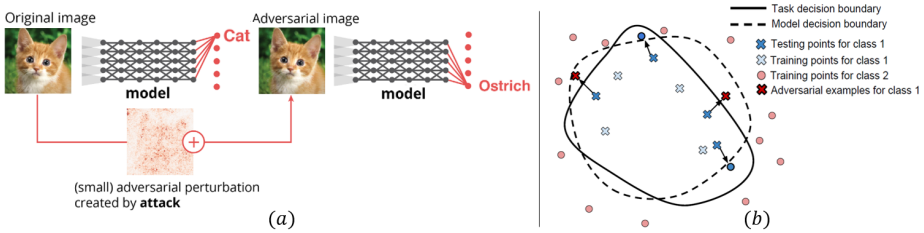

Fig. 3. (a) Illustration of an adversarial example (from NIPS 2018 Challenge). (b) Proposed by N. Papernot in [17]: adversarial examples as a generalization error.

4.2 Crafting adversarials

In the reference paper [14], a box-constrained L-BFGS method is used to solve (2). Subsequently, several works [15, 18] propose efficient crafting methods based on a first-order linear approximation of the model in the neighborhood of x using the gradient of the cost function[1] $\nabla_x \mathcal{J}$ (computed with back-propagation) and the l_∞ norm known as the *Fast Gradient Sign Method* (FGSM) [15]. A generalization of FGSM for l_2 is simply referenced as *Fast Gradient Method* (FGM). Note that FGSM leads to a non-targeted misclassification i.e. the adversarial label is not chosen by the adversary. Looking at (3), the FGSM method tries to directly cross the decision boundary defined with the true label $y = f_\theta(x)$, thus for targeted attack it can be easily modified:

$$\begin{aligned} &\textit{FGSM, } l_\infty\text{:} && x^* = x + \epsilon \ \text{sign}(\nabla_x \mathcal{J}(x, y)) \\ &\textit{targeted attack:} && x^* = x - \epsilon \ \text{sign}(\nabla_x \mathcal{J}(x, y_{target})) \\ &\textit{FGM, } l_2\text{:} && x^* = x + \epsilon \frac{\nabla_x \mathcal{J}(x,y)}{||\nabla_x \mathcal{J}(x,y)||_2} \end{aligned} \tag{3}$$

[1] Note that Carlini et al. [19] propose an effective loss function for adversarial examples crafting (compared to the traditional cross entropy loss): $\mathcal{L}(x, y) = log(1 - x.y)$.

Several flavors of this *one-step* gradient methods have been proposed, particularly iterative variants which take T gradient steps of magnitude $\lambda = \epsilon/T$:

$$x_0^* = x,\ x_{t+1}^* = x_t^* + \lambda.\text{sign}(\nabla_x \mathcal{J}(x_t^*, y)) \tag{4}$$

Obviously, the choice of the norm applied to the perturbation has a strong impact. l_∞ is widely used with gradient-based methods. However, with l_0, the resulting perturbation is applied on a minimum number of features as proposed in the Jacobian-based Saliency Map Approach (JSMA) in [18] where saliency maps identify the most sensitive features. In [18] only 4% of the input features (i.e. pixels) are perturbed for 97% of adversarial accuracy (MNIST dataset) but the level of the perturbation is stronger and obviously *visible*.

Finally, other methods propose to use generative models. The idea is to train an *Adversarial Transformation Networks* (ATN, such as adversarial autoencoders) to learn to craft adversarial examples from clean inputs [20]. Once the ATN model is trained, such techniques could be robust and as fast as gradient-based methods.

4.3 Transferability and black-box attacks

Transferability is undoubtedly the most important and intriguing property of adversarial examples [14, 21]. Let's consider a set of adversarial examples crafted using a model f_θ^A. Suppose that another model f_θ^B is learned to solve the same classification task but with a different structure and parameters than f^A. The transferability property – also called *cross-model generalization* ability – sets that some adversarial examples computed from f^A will also fool f^B.

Transferability is a powerful property for an attacker since it paves the way to black-box attacks [22, 23]: an adversary trains a model f_θ^{mirror} that simulates the target and unknown model, f_θ^{target}. Then, adversarial examples are crafted on this new model and applied to the target one. Papernot et al. [22] first propose a method to train f_θ^{mirror} based on data augmentation: the attacker first collects a small but representative training dataset labeled by f^{target}, then f^{mirror} is trained using this training dataset which is iteratively extended with synthetic samples, x', using a Jacobian-based technique.

In [23], Liu et al. consider five reference DNNs on the ImageNet database and show that an average of 85% of non-targeted adversarial images produce for one model also lead to misclassification on another model. The challenge is more difficult for targeted adversarial images but the authors manage to reach 77% of targeted misclassification and almost a perfect 100% transferability for non-targeted adversarial examples using a so-called *ensemble-based approach* that simply optimizes the crafting of adversarials considering a linear combination of k models in the adversarial optimization problem (Eq. 2).

4.4 Digital and Physical Adversarial Examples

Some works shift the digital adversarial examples into the physical world [25], i.e. physically modify (or create using 3D-printing as in [26]) an object to make

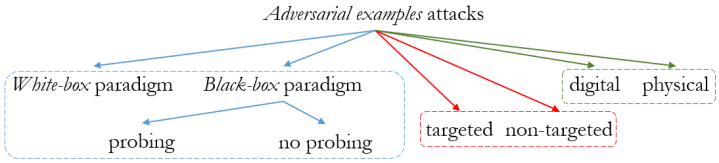

Fig. 4. Different classifications of adversarial examples attacks [24]. Black-box attacks without probing means that the adversary cannot probe or query the target model.

it misclassified by a supervised model. For example, Eykholt et al. [27] fool a road signs classifier with posters or stickers-printed perturbations as illustrated in Fig. 5 and [28] fools a face recognition system. All these works enable to better understand and validate some hypotheses on the inherent mechanisms of adversarial examples and participate in the global understanding of the crucial DNN interpretability issue.

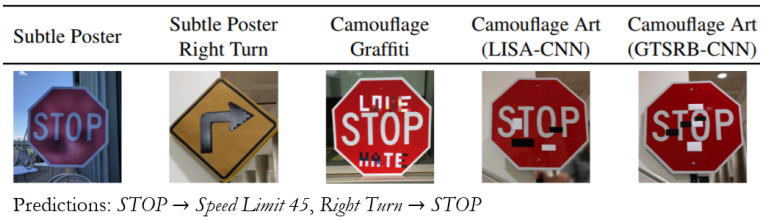

Fig. 5. Physical printed perturbations (from [27]).

4.5 Beyond supervised neural networks and computer vision

Predominantly, adversarial examples studies are focused on supervised DNN for computer vision but they have also been successfully crafted for other applications like speech-to-text [29], Q&A systems, malware detection [30] and even for reinforcement learning [31, 32]. Similarly, other machine learning methods [33, 34] are concerned by adversarial examples such as linear models, SVMs or Decision Trees. More important, the transferability property is not constrained by the nature of the ML algorithm: Papernot et al. [35] demonstrate – a significant result – how adversarial examples transfer between five ML algorithms (see Fig. 6).

5 Physical attacks as new attack vectors

Attacks presented above target the ML pipeline at the algorithmic level. They are possible because of theoretical flaws which are independent of the implementation of the algorithms on a physical device (e.g., computer, smart phone). However, other critical systems (such as cryptographic systems) show the importance of not limiting the attack surface by bridging the physical and the algorithmic world.

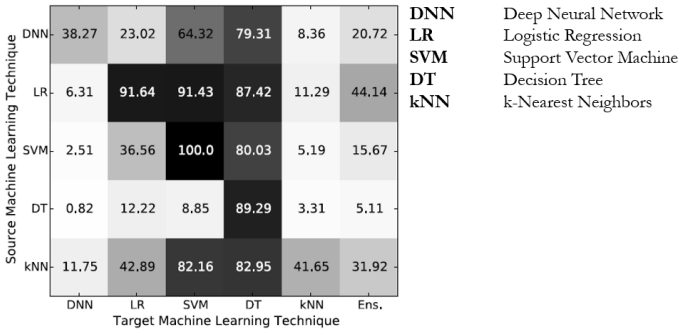

Fig. 6. From [35]: cross-technique transferability of adversarial examples. For example, 64.32% of adversarial examples originally crafted with DNN mislead SVM.

5.1 Expand the attack surface

Let's make a parallel between a ML algorithm and a cryptographic primitive, for example the AES (Advanced Encryption Standard). As an abstract algorithm, AES has been designed to be perfectly robust against theoretical attack (i.e. cryptanalysis). Today, the only way to find the secret (the encryption key) is a brute force attack. But, since the late nineties, we know that secret information leak from the physical implementation (e.g., a software AES on a standard microcontroller). Indeed, an attacker who has a physical access to the device can exploit so-called *side-channels* like the power consumption or the electromagnetic (EM) emanation of the chip that are dependent of the running process as well as the processed data (including the secret). Combined with theoretical knowledge of the algorithm, these information are powerful attack vectors to recover the AES secret key. An other way to proceed is to physically tamper the algorithm by injecting faults [36], e.g. by glitching the clock or pulsing EM radiation. The differential between the correct and the faulted processing is also a powerful attack vector to guess the secret.

Today, one of the most important trend in ML is to embed algorithms and models. Every chip manufacturers announce new products or tools "*A.I. suitable*" even for targets with limited resources (among others, see STMicroelectronic[2] or ARM[3]). Logically, high interests are shown on porting and applying models for inference purpose but more complex embedded platform plan to perform both training and inference tasks. Thus, in short term, ML system will rapidly be included in a large variety of devices (smart phones, connected objects, probes/sensors...) and then, will be exposed to the same attack surface as the one previously described for cryptographic primitives.

[2] ST at CES 2018 – Deep Learning on STM32, http://bit.ly/ST-home-tag
[3] ARM-NN: developer.arm.com/products/processors/machine-learning/arm-nn

5.2 Exploiting leakage with side-channel analysis

Recently, first works exploiting side-channel analysis (SCA) for embedded DNN have been published. The first study [37] – to our knowledge and claimed by the authors – shows how SCA enables to retrieve information about the model thanks to EM emanations of the device. The authors use a popular Atmel ATmega328P (8bits microcontroller) and a classical setup with an EM probe and an oscilloscope (see Fig.7). Several trained networks (simple multilayer perceptron) are implemented in C and compiled on the chip. They show that with basic SCA techniques they can profile and extract the nature of the activation function (*ReLu*, *sigmoid*, *tanh*, *softmax*), recover estimation of the weights and reverse engineering the number of neurons and layers. Admittedly, the models are small shallow neural networks on a simple unprotected target but this work paves the way to further experimentations on more complex models (e.g. CNN) and targets (e.g., 32-bits microcontrolers, FPGA). Furthermore, as it is traditionally the case in the hardware security community, the authors conclude with well-known protection schemes in SCA that may be relevant (not tested in [37]) for embedded ML system such as shuffling, masking or constant time implementation.

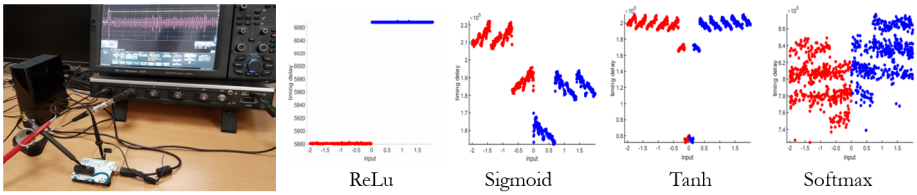

Fig. 7. From [37]. SCA setup (left); timing behavior for four activation functions (right)

5.3 Fault injection

We are clearly at the early beginning of the use of fault injection techniques against embedded neural networks. Among the very first studies on that field, Breier et al. [38] deal with the practicability of fault injection by attacking several activation functions (implemented in C) with a laser beam. The target is the same as in [37], i.e. ATmega328P. The fault model is a single instruction skip during the execution of the activation function. Successful attacks has been performed on *ReLu* (leading to a zero function i.e. an inactive neuron), *sigmoid* and *tanh* (horizontal flipping leading to decreasing functions) but not, intriguingly, for *softmax*. Simulation based on these results show that for a simple multilayer perceptron, at least half of the neurons in the target layer have to be tampered in order to have a reasonable misclassification success rate ($> 50\%$).

In parallel to such practical studies several simulation-based efforts evaluate the behaviors of neural networks facing fault injections. For example, [39] proposes to analyze the impact of fault injections into memory by simulating faults

against two (white-box) CNN trained on the well-known MNIST and CIFAR10 image datasets. The threat model is based on the (strong) assumption that the adversary can modify any parameters of the models (i.e. any weight and bias). A first strategy is based on the fact that the output neuron with the higher value has the largest probability after the softmax layer (which is also true when targeting ReLu-like hidden layer for a sufficiently higher bias), then, by increasing the bias of a specific output neuron, an adversary can switch the classification to this target label. The attack rests upon large necessary perturbations of the parameters (namely the bias) because neural networks are intrinsically robust against small deviations. That's why a second attack targets several parameters (i.e. neurons) with minimal perturbation (with a classical gradient descent-based optimization problem). The practicability of these attacks with real fault injection techniques is obviously questioned but the study has the strong merit of showing and proving that numerous attack paths can be exploited.

Although not focused specifically on adversarial attacks but on fault tolerance and resilience for hardware optimization purpose, another simulation work has to be highlighted with "*a light-weight, DNN-specific fault injection framework*" (Ares, [40]). This work offers interesting conclusions about fault tolerance that can vary across models (and across layers) by several orders of magnitude as well as the stronger resilience of activations compared to weights.

6 Protections and evaluation

For now, there is an incontestable imbalance between attacks and defenses. One reason is that, obviously, security of ML systems is in its infancy [41] and better defense strategies are closely related to the ongoing understandings of intrinsic properties of these security flaws. It is particularly true for adversarial examples which undoubtedly suffer from the lack of robust defenses in that some protections are too attack-dependent and rapidly obsolete when put to the test of new attacks. Precursory recent research on *provably secure machine learning* tackle the problem of assessing defenses in case of worst-case scenarios such as [42, 43] with certified defenses for data poisoning and adversarial attacks. This effort is a strong and logical necessity to answer the critical issue summarized by J. Steinhardt [42, 43]: "*How can we design ML systems that are not just empirically secure against known attacks, but provably secure against all attacks under a meaningful and well-defined threat model?*".

6.1 Protection schemes

Protect confidentiality and privacy. As previously seen, a majority of attacks query the model with one or a set of target inputs $\{x_i\}$ and use the outputs of the model $\{f_\theta(x_i)\}$, which usually gather all the probability vector that x_i belongs to each class, to infer information about X^{train} or f_θ. Thus, the first and basic defense strategy is to only output the label with the best confidence or, at least, to reduce the precision of the confidence scores. However, [4] demonstrates the limit of this countermeasure when an adversary carefully optimizes

the selection of $\{x_i\}$. Another (classical) strategy to improve the robustness is to use several models (*ensemble approach*) in such a way the final prediction is the mean or the aggregation of different models.

Privacy preserving machine learning is currently a hot topic with some interesting research based on cryptographic approaches such as homomorphic encryption or garbled circuit (interested readers can refer to [44]). A promising field concerns *Differential Privacy* (DP) for DNN. Simply put, DP gathers perturbation approaches that aim at making the outputs of a learned model useless for attackers who try to infer information about the training data. Recently, techniques propose to use DP directly and efficiently in the gradient descent as in [45, 46]. Another strategy [47] is to process the private data on disjoint *teacher* models and to learn a *student* model to imitate the teachers thanks to non-private data with a similar distribution as the private set, which is a strong assumption and therefore the limitation of this method. However, as demonstrate in [11], DP based deep learning models are still too vulnerable against strong inference-based attacks such as [10].

Defense against adversarial examples. Adversarial examples have been and are still widely analyzed to better understand their inherent intriguing mechanisms. Simultaneously, the ML community proposes numerous protection schemes that we present as follows.

Denoising approaches use signal preprocessing methods (compression [48], filtering, quantification [49]) to alter the inputs and are mainly effective for black-box scenarios (with a perfect knwoledge of the model, optimized crafting techniques enable to thwart these protections).

Gradient masking aims at making the gradient information useless or nonexistent. A first idea is to use non differentiable functions or models, but it appears ineffective against the cross-technique transferability. More sophisticated white-box techniques [50–52] have been proposed such as *shattered gradients* (non-differentiable operations or numerical instability), *stochastic gradients* (randomness injection) and *vanishing/exploding gradients*. For example, [50] is a real step forward that considers the training of a robust classifier as a min-max problem and uses PGD[4] to solve the corresponding constrained optimization problems. However, [53] show that most of these strategies are not fully efficient against advanced attacks that approximate the missing or decayed gradient information.

Adversarial training basically injects adversarial examples at training time so that the model can enhance its robustness against adversarial examples [15]. This principle is particularly studied despite some drawbacks related to overfitting issues and similarity (in the effects) with gradient masking but, for now, it performs state-of-the-art performance like the ensemble-based method in [54] (see NIPS challenge below).

Detection-based defenses do not make the model robust or the crafting of adversarial examples impossible but try to detect any tampering of the inputs, e.g. with statistical approaches [16]. The efficiency of these strategies is closely

[4] Projected Gradient Descent

linked to the assumption that adversaries are not informed of the detector otherwise it could be easily thwarted. Moreover, recently, Carlini et al. bypass ten recent detection methods by building new loss functions [55].

Even though the multitude attack/defense propositions can be seen as a vain *arms race*, strong advances have been achieved to built robust models that are not only efficient against a very specific class of attacks. This topic is highly dynamic and even very recently, new approaches appeared with state-of-the-art results such as the logit pairing [56] which forces similarity between logits for pairs of examples.

6.2 Evaluation

The security issues developed above are not trivial phenomena because they rely on complex and sometimes (still) intriguing properties of ML models more particularly with DNN. Simultaneously to the analysis of attacks and the development of sound protection schemes there is a growing and crucial need for proposing evaluation methodologies in order to efficiently test the robustness of models. Such methodologies and tools [19, 41] are actually compulsory to help the dissemination of good practice towards ML systems designers and providers as well as to anticipate security norms or certifications that will likely incorporate, in the near future, the particularities of ML components as it is the case, for example, for cryptographic based-systems.

For integrity-based attacks based on adversarial examples, the CleverHans platform [41] represents an important step forward for the development of benchmarking tools. This adversarial ML library (Python) proposes adversarial (standardized reference) crafting techniques (FGSM, JSMA) as well as adversarial training features.

In 2017, Google Brain proposes a competition on "Adversarial Attacks and Defense" [24] within the NIPS[5] conference. The competition – which gathered more than 100 teams – consisted of three tracks: (1)*non-targeted adversarial attack*, (2) *targeted adversarial attack*, (3) *defense against adversarial attack*.

The winning proposition [57] for both non-targeted and targeted adversarial attacks is based on the iterative fast gradient approach (see Eq. 4) with a momentum method that accumulates gradient information. The main advantage of this effective technique (named MI-FGSM) is to jointly preserve the attack ability of iterative FGSM (white-box) and the transferability power of one-step FGSM (black-box). Interesting analysis is detailed in [57] like ensemble of models to enhance the transferability. The second best submission [24] also uses an optimized iterative approach on ensemble of models but improves the robustness and transferability with data augmentation (random projective transformation). Other submissions are detailed in [24].

The winning protection [58] to the defense track targets the adversarial perturbation throughout the neural networks (the authors show how this perturbation is amplified layer after layer) at the feature, logits and label (softmax

[5] https://nips.cc/Conferences/2017/CompetitionTrack

outputs) levels and uses denoising autoencoders at each level. Close to the best result, [59] proposed a randomization-based protection (image padding and resizing) combines to an adversarial trained model (based on ensemble of models) proposed in [54] which is known as a state-of-the-art protection.

Such competitions clearly accelerate research and, even if these three tracks are specifically focused on adversarial examples, the success of the NIPS competition (renewed in 2018[6]) show that the scientific community is increasingly concerned by this topic.

7 Conclusion

An increasing number of devices will embed ML algorithms, some will be additionally connected to large-scale online ML infrastructure. It is not an alarmist prophecy to say that all these devices (e.g., cars, smart sensors) could be hacked and, in a worst-case scenario, weaponized, it is a likely prediction which relies on known research efforts or already encountered cyber-attacks that could be easily transposed to ML systems.

Machine Learning is a godsend for adversaries who will never consider ML systems as impassable ramparts, whatever their intrinsic complexity. On the contrary, ML systems offer new powerful attack vectors to directly strike critical processes (e.g., prediction, decision-making) with the ability to operate under the radar (adversarial examples) or bypass difficulties that are finally not (e.g., efficiently thwart the black-box paradigm). Security becomes a critical issue, one of the two major pitfalls that would seriously constrain the deployment of ML and more particularly DNN-based systems with the well-known problem of interpretability (with hundreds of thousands, even millions, of parameters, a DNN is probably the best definition of what is an opaque statistical model) . Both problems seem to be more overlapped as appear at first glance since solving a majority of security flaws (e.g., transferability of adversarial examples) enables to better understand the properties of DNN models. Reciprocally, understand how DNNs work or fail on complex task and high-dimensional data is compulsory for the analysis of attacks and the development of robust defense strategies.

In this article, we show that, thanks to essential works from the ML community, advances have been succeed on the analysis of attacks striking ML and especially DNN (admittedly, several open questions still remain) but the lack of robust defenses is a real issue: simply put, we do not have any security guarantee with too many attack-dependent *testing* for rare guaranteed *verification* over a wide range of well-defined threats [41]. This problem could be solved by sharing skills and expertises between two separate scientific and technical communities: Machine Learning and Security. This collaboration appears as unavoidable if one wants to foresee new powerful threats rather than endure it (e.g., ongoing physical attacks). Both communities have everything to gain: the software and hardware security domains can bring proven and sound concepts as well as

[6] The 2018 competition holds during the redaction of this paper. See www.crowdai.org/challenges/nips-2018-adversarial-vision-challenge for 2018 results

methodologies for threat modeling, attacks analysis, countermeasures development and evaluation that lack to the ML community, who – in return – provides essential theoretical tools to understand the inherent mechanisms of advanced ML techniques such as DNN which never cease to evolve and gain in complexity (very deep networks, generative models...).

References

1. T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., 1997.
2. N. Papernot et al. Towards the science of security and privacy in machine learning. *IEEE European Symposium on Security and Privacy*, 2016.
3. M. Fredrikson et al. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of Conference on Computer and Communications Security*, 2015.
4. F. Tramèr et al. Stealing machine learning models via prediction apis. In *USENIX Conference on Security Symposium*, 2016.
5. I. Goodfellow. Defense against the dark arts: An overview of adversarial example security research and future research directions. *Keynote lecture at the Deep Learning Security workshop, IEEE Security and Privacy*, 2018.
6. B. Biggio et al. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*, 2012.
7. C. Yang et al. Generative poisoning attack method against neural networks. *CoRR*, abs/1703.01340, 2017.
8. L. Muñoz-González et al. Towards poisoning of deep learning algorithms with back-gradient optimization. *ACM Workshop on A.I. and Security*, 2017.
9. N. Carlini et al. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *CoRR*, abs/1802.08232, 2018.
10. R. Shokri et al. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017.
11. A. Rahman et al. Membership inference attack against differentially private deep learning model. *Transactions on Data Privacy*, 11, 2018.
12. I. Goodfellow. OpenAI blog: blog.openai.com/adversarial-example-research/, 2017.
13. F. E. Gamaleldin et al. Adversarial examples that fool both human and computer vision. *CoRR*, abs/1802.08195, 2018.
14. C. Szegedy et al. Intriguing properties of neural networks. *International Conference on Learning Representations*, 2014.
15. I. Goodfellow et al. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015.
16. K. Grosse et al. On the (statistical) detection of adversarial examples. *CoRR*, abs/1702.06280, 2017.
17. N. Papernot. Security and privacy in machine learning, 2017.
18. N. Papernot et al. The limitations of deep learning in adversarial settings. *IEEE European Symposium on Security and Privacy*, 2016.
19. N. Carlini et al. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.
20. S. Baluja et al. Learning to attack: Adversarial transformation networks. In *AAAI Conference on Artificial Intelligence*, 2018.
21. L. Wu et al. Understanding and enhancing the transferability of adversarial examples. *CoRR*, abs/1802.09707, 2018.

22. N. Papernot et al. Practical black-box attacks against machine learning. In *ACM on Asia Conference on Computer and Communications Security*, 2017.
23. Y. Liu et al. Delving into transferable adversarial examples and black-box attacks. *International Conference on Learning Representations*, 2017.
24. A. Alexey Kurakin et al. Adversarial attacks and defences competition. *CoRR*, abs/1804.00097, 2018.
25. A. Kurakin et al. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.
26. A. Athalye et al. Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397, 2017.
27. K. Eykholt et al. Robust physical-world attacks on deep learning visual classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
28. M. Sharif et al. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *ACM SIGSAC Conference on Computer and Communications Security*, 2016.
29. N. Carlini et al. Audio adversarial examples: Targeted attacks on speech-to-text. *CoRR*, abs/1801.01944, 2018.
30. K. Grosse et al. Adversarial examples for malware detection. *Computer Security – ESORICS 2017*, 2017.
31. S.H. Huang et al. Adversarial attacks on neural network policies. *CoRR*, abs/1702.02284, 2017.
32. V. Behzadan et al. Vulnerability of deep reinforcement learning to policy induction attacks. *CoRR*, abs/1701.04143, 2017.
33. L. Huang et al. Adversarial machine learning. In *ACM Workshop on Security and Artificial Intelligence*, 2011.
34. B. Biggio et al. Pattern recognition systems under attack: Design issues and research challenges. *IJPRAI*, 28(7), 2014.
35. N. Papernot et al. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, 1605.07277, 2016.
36. A. Barenghi et al. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, Nov 2012.
37. L. Batina et al. CSI neural network: Using side-channels to recover your artificial neural network information. Cryptology ePrint Archive, 2018.
38. J. Breier et al. Practical fault attack on deep neural networks. *CoRR*, abs/1806.05859, 2018.
39. Y. Liu et al. Fault injection attack on deep neural network. In *International Conference on Computer-Aided Design*, 2017.
40. B. Reagen et al. Ares: A framework for quantifying the resilience of deep neural networks. In *ACM Annual Design Automation Conference*, 2018.
41. Cleverhans blog: http://www.cleverhans.io/.
42. J. Steinhardt et al. Certified defenses for data poisoning attacks. In *Conference on Neural Information Processing Systems*, 2017.
43. A. Raghunathan et al. Certified defenses against adversarial examples. *International Conference on Learning Representations*, 2018.
44. N. Dowlin et al. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, 2016.
45. R. Shokri et al. Privacy-preserving deep learning. In *Conference on Computer and Communications Security*, 2015.
46. M. Abadi et al. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

47. N. Papernot et al. Semi-supervised knowledge transfer for deep learning from private training data. *International Conference on Learning Representations*, 2016.
48. N. Das et al. Keeping the bad guys out: Protecting and vaccinating deep learning with JPEG compression. *CoRR*, abs/1705.02900, 2017.
49. X. Xu et al. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Network and Distributed System Security Symposium*, 2018.
50. A. Madry et al. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.
51. T. Na et al. Cascade adversarial machine learning regularized with a unified embedding. In *International Conference on Learning Representations*, 2018.
52. G.S. Dhillon et al. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018.
53. A. Athalye et al. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, 2018.
54. F. Tramèr et al. Ensemble adversarial training: Attacks and defenses. *International Conference on Learning Representations*, 2018.
55. N. Carlini et al. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, 2017.
56. H. Kannan et al. Adversarial logit pairing. *CoRR*, 2018.
57. Y. Dong et al. Boosting adversarial attacks with momentum. In *Conference on Computer Vision and Pattern Recognition*, June 2018.
58. F. Liao et al. Defense against adversarial attacks using high-level representation guided denoiser. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
59. C. Xie et al. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018.