

Careful What You Wish For: on the Extraction of Adversarially Trained Models

Kacem Khaled
Département GIGL
Polytechnique Montréal
Montreal, Canada
kacem.khaled@polymtl.ca

Gabriela Nicolescu
Département GIGL
Polytechnique Montréal
Montreal, Canada
gabriela.nicolescu@polymtl.ca

Felipe Gohring de Magalhães
Département GIGL
Polytechnique Montréal
Montreal, Canada
felipe.gohring-de-magalhaes@polymtl.ca

Abstract—Recent attacks on Machine Learning (ML) models such as evasion attacks with adversarial examples and models stealing through extraction attacks pose several security and privacy threats. Prior work proposes to use adversarial training to secure models from adversarial examples that can evade the classification of a model and deteriorate its performance. However, this protection technique affects the model’s decision boundary and its prediction probabilities, hence it might raise model privacy risks. In fact, a malicious user using only a query access to the prediction output of a model can extract it and obtain a high-accuracy and high-fidelity surrogate model. To have a greater extraction, these attacks leverage the prediction probabilities of the victim model. Indeed, all previous work on extraction attacks do not take into consideration the changes in the training process for security purposes. In this paper, we propose a framework to assess extraction attacks on adversarially trained models with vision datasets. To the best of our knowledge, our work is the first to perform such evaluation. Through an extensive empirical study, we demonstrate that adversarially trained models are more vulnerable to extraction attacks than models obtained under natural training circumstances. They can achieve up to $\times 1.2$ higher accuracy and agreement with a fraction lower than $\times 0.75$ of the queries. We additionally find that the adversarial robustness capability is transferable through extraction attacks, i.e., extracted Deep Neural Networks (DNNs) from robust models show an enhanced accuracy to adversarial examples compared to extracted DNNs from naturally trained (i.e. standard) models.

Index Terms—Deep Learning, Model Extraction, Adversarial Training, Privacy, Security

I. INTRODUCTION

Given the success of Deep Learning (DL) in achieving the state-of-the-art and sometimes human-competitive performance in several computer vision tasks [1]–[3], DL became the core of several critical applications such as autonomous vehicles and robotics. Machine Learning (ML) providers offer their fine-tuned models to users as a service (Machine Learning as a Service (MLaaS)) so they can benefit from its outstanding performance in a specific prediction task. The architecture of the models is usually not revealed, instead, users are only provided with an Application Programming Interface (API) to query them [4].

Models are often proprietary and are a business advantage to their owner as they are expensive to obtain [5]. In fact, when a model owner shares an internal knowledge about a model such as the architecture or weights, a malicious user can infer sensitive information in the training data which raises privacy concerns in certain cases [6]. Additionally, revealing the trained models poses security concerns, since an adversary can craft adversarial examples evading the classification by a white-box victim model easily [7].

Prior work shows that an adversary with a query access to a model is able to extract it and construct a very similar model to the victim’s functionality [4], [8]–[19]. These attacks help an adversary gain more knowledge about the victim model and therefore raise different concerns; for example, using an extraction attack, an adversary can craft adversarial examples evading the extracted model that are transferable for the evasion of the original one [4]. In contrast, recent work shows attempts to defend against these extraction attacks and mitigate their risk [20]–[24]. They often work on slightly changing the output prediction values without highly affecting the model’s performance. This may hinder the possibility of an attack, or at some points make it require more queries to achieve a sufficient extraction.

In addition to the extraction attacks that compromise the privacy of models, the ML security gained significant attention during the last decade especially with the serious threats of adversarial attacks [25]. Recently, model owners find themselves required to modify their regular training process that results in unprotected models referred also as *natural* models. They apply a defense technique to their Deep Neural Networks (DNNs) such as *adversarial training* before deploying them as final oracles [26]. Nevertheless, existing extraction attacks do not take into consideration these security-imposed and uncommon scenarios that deviate from the normal training process, and this is still, to the best of our knowledge, an open research problem. Since researchers tend to work on different attack objectives separately and evaluate different potential threats independently, this may raise concerns about the robustness under these circumstances. Hence, we don’t know how the risk of model extraction would develop in the case of adversarial training and whether models get more or less vulnerable to these attacks. As a matter of fact, a recent

This research was funded by Synopsys Inc. and the Natural Sciences and Engineering Research Council of Canada (NSERC).

To appear in Proceedings of PST 2022, Fredericton, Canada ©2022 IEEE

work showed that when working on improving the security of a model, they may have increased the risk of the training data's privacy and leaked information about it [27].

In this paper, we explore important intersections between extraction attacks and an unavoidable security-imposed scenario which is protecting natural models (i.e., unmodified models) against adversarial examples. We assess the robustness against extraction attacks of DL models under different learning circumstances. Thus, we make the following contributions:

- an open source framework to assess a model's extractability against extraction attacks under different unexplored situations in the state-of-the-art using potential defenses against adversarial attacks¹;
- an evaluation of the extraction risk of models with increased robustness against adversarial examples showing that adversarially trained models can achieve up to $\times 1.2$ higher accuracy and agreement (i.e., fidelity) of natural models with a fraction lower than $\times 0.75$ of the queries, and;
- an empirical proof that the adversarial robustness capability of adversarially trained models is transferable through models extraction attacks.

The remainder of the paper is organized as follows: Section II introduces the basic notions to ML and DL; Section III reviews the background and the state-of-the-art that relates to privacy and security threats and attacks; Section IV details our methodology; we include our experiments and obtained results in Section V; Section VI summarizes potential defenses and countermeasures against extraction attacks; and Section VII concludes this paper.

II. MACHINE LEARNING AND NEURAL NETWORKS

A *machine learning* algorithm is an algorithm that can learn from data to perform a task. The data includes examples with quantitatively measured features [28]. An example is typically represented as a vector $\mathbf{x} \in \mathcal{R}^n$ where each entry of the vector x_i is another feature. ML algorithms can be broadly categorized as supervised and unsupervised:

- supervised learning algorithms involve working on annotated datasets which means every example \mathbf{x} in the dataset is associated with a provided label \mathbf{y} . The algorithm learns to predict \mathbf{y} from \mathbf{x} by estimating $p(\mathbf{y}|\mathbf{x})$ [28], and;
- unsupervised learning algorithms experience an unlabeled dataset containing a collection of examples, and try to learn interesting properties about the dataset's structure [28]. The algorithm wants to learn implicitly or explicitly a probability distribution that generated the data $p(\mathbf{x})$.

Other ML algorithms do not work on just fixed data, such as reinforcement learning algorithms where the learning system interacts with a dynamic environment in which it must perform a certain goal. The system is provided with a feedback loop

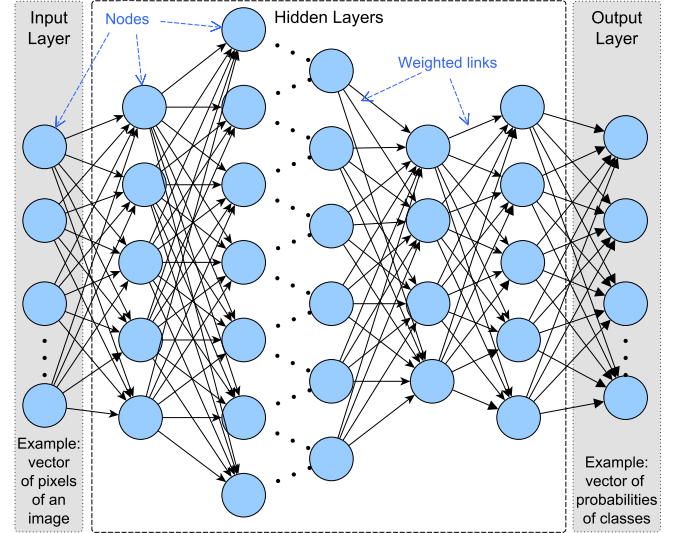


Fig. 1. An illustration of a neural network with multiple layers. The input values are fed to the first layer and propagated to the following layers multiplied by weights. For each layer, the nodes output the results of an activation function to the sum of their entries. The last layer outputs probabilities for each class at the corresponding node.

in terms of rewards as it takes actions in the environment, and through experience the agent learns a policy that maximizes that reward [29].

Deep learning is a subset of ML involving techniques based on Artificial Neural Networks (ANN) [28]. An example of a deep learning model is a feed-forward deep network, or a Multilayer Perceptron (MLP), which is a mathematical function defined as $\mathbf{y} = F(\mathbf{x})$ mapping input vectors $\mathbf{x} \in \mathcal{R}^n$ to output values $\mathbf{y} \in \mathcal{R}^m$. A Neural Network (NN) has L layers characterizing the model's depth where each layer has a width and is characterized with a number of nodes called neurons. Fig 1 illustrates an example architecture where the neurons connect the layers together through weights \mathbf{W} . Each neuron perform an activation function $g(\mathbf{z})$ to the weighted sum of the output values of the previous layer's neurons: $\mathbf{z} = \mathbf{W}^T \mathbf{h} + \mathbf{b}$.

To obtain a meaningful set of weights, a training process is performed and the parameters are optimized via approaches such as stochastic gradient descent (SGD) in order to minimize a certain loss function [28].

Convolutional Neural Networks (CNNs) are a particular type of feed-forward NNs, that are mostly suitable for image classification tasks, with the requirement that at least one of the layers performs a mathematical operation called convolution operation followed by a non-linear activation. After enough training, CNNs are able to learn filters and characteristics relevant to the task [30].

III. BACKGROUND AND RELATED WORK

Recently, ML models including DNNs were shown to be vulnerable to several attacks [7], [31]–[34]. In this section we taxonomize these attacks depending on the adversary's goals

¹Our code is available at the GitHub repository: <https://github.com/KacemKhaled/model-stealing>

and capabilities as well as the underlying assumptions of his knowledge about the victim.

A. Adversary's capability

It defines the actions that the attacker could perform against the victim model. These capabilities could be related to the training phase or the testing phase depending on the attacker strategy and purposes. In the training phase, the attacker may be able to inject new data to the training set, or add malicious modifications to the current one. In the testing phase, the attacker can only control the test data, for example inject well crafted examples to the model in order to lead it to misclassification.

B. Adversary's knowledge

Models could be affected by different attacks depending on the knowledge level of the adversary which reflects the strength of the attack. Taking the attack settings into consideration, we can cite three types of attacks:

- *White-box attacks*: the adversary has full access and knowledge about the model (e.g. network architecture, parameters, training set...);
- *Gray-box attacks*: the adversary assumes a partial knowledge about the targeted model (e.g. the distribution of the training set, the classifier type), and;
- *Black-box attacks*: the attacker has a very limited access to the target model. He can observe the inputs and outputs, but has no access to the internal working of the model neither the training set.

C. Adversary's goals (attacks on machine learning models)

The attacker can exploit several parts of the machine learning pipeline to achieve various malicious goals [35]. Attacks can be performed in the training phase or during the inference phase. We present the different threats depending on the adversary's objective.

1) *Poisoning attacks*: refer to causative attacks in which specially crafted attack points are injected into the training data in order to increase the model's test error [33] and corrupt it. "*Causative attacks alter the training process through influence over the training data*" [31]. Poisoning attacks assume that the attacker knows the learning algorithm and can draw points from the data distribution. Finding such an attack point can be formulated as an optimization problem. In fact, for a poisoning attack against ML model, the adversary's goal is to find an attack point whose addition to the training set maximally decreases the model's classification error [32]. Poisoning attacks were demonstrated against several applications such as spam filters, PDF malware detection, Denial-of-Service (DoS) attack detection, handwritten digits recognition and healthcare [36]–[39].

2) *Membership inference attacks*: in these attacks, the adversary aims to determine whether an individual data record is part of the training set of the attacked model. This attack reflects the leaked information about the training set from the model [27]. This attack leverages the inference part of

the model, so it does not interfere with the training phase, but instead the adversary uses test data in order to achieve his goal. This attack poses a high risk on the privacy of the data when sensitive information is used to train the model (e.g. medical records). In some cases, this attack relies on a confidence thresholding technique, in which the adversary concludes about a membership of data through the prediction confidence given by the model about this input.

3) *Evasion attacks (i.e., adversarial examples attacks)*: this attack happens on the inference part of the model, so it does not interfere with its training phase. Instead, it interferes with the test data through creating adversarial examples. "*An adversarial example is a sample of input data which has been modified very slightly in a way that is intended to cause a machine learning classifier to misclassify it*" [40]. Usually these modifications are not subtle to humans, but they cause a huge error in machine learning models and it was shown that even state-of-the-art algorithms can be fooled by this malicious input. The methods of crafting adversarial examples rely on optimization problem that aims to maximise the networks prediction error [7].

Papernot et al. [41] demonstrated that these adversarial attacks against feed-forward neural networks can be adapted to fool Recurrent Neural Networks (RNNs). They particularly demonstrated an attack against the Long-Short-Term-Memory (LSTM) architecture. Adversarial attacks were also shown to be applied in the physical world, Kurakin et al. [42] proved that adversarial images printed on paper can be misclassified when they are fed to a classifier through a cellphone camera. Furthermore, Eykholt et al. [43] explored the use case of safety-critical situations where road signs could be manipulated to fool the autonomous vehicles classifiers. They succeeded in proposing physical perturbations such as stickers and graffiti to road signs that were misclassified by the target classifier in lab settings as well as in field test with captured video frames obtained from a moving vehicle.

Several countermeasures has been proposed to mitigate these attacks, for example adversarial training, which rely on creating a bigger dataset rich with adversarial examples through generating adversarial examples using the methods that could be used by the attacker and then, retrain the model using that dataset [26], [27]. Another way to defend against adversarial examples is through modifying the model with network distillation [44] which relies on transferring the knowledge from an initial network to a distilled network.

4) *Model extraction attacks (model stealing attacks)*: in these attacks the adversary exploits the inference phase of the machine learning pipeline. Through observing the prediction outputs of his queries, even in black-box settings, the attacker aims to steal the functionality or parameters of a ML model [8]. The adversary attempts to learn a classifier \hat{f} that matches or closely approximates a target classifier f . These attacks have a great risk on the Intellectual Property (IP) of model owners. Additionally, they facilitate other attacks such as membership inference and evasion attacks.

In the literature, extraction attacks can broadly be categorized as:

- attacks that exploit hardware access, or side-channel extraction attacks: when a ML model is deployed in a hardware platform where the user has access to it, for example on an FPGA, on a NN accelerator or sometimes the same host machine as the victim, the adversary has more access than the API software level case. Leveraging the leaked information through side-channels (e.g. cache, memory, power, electromagnetic, timing), the adversary can gain more knowledge about a NN architecture, its parameters, or even duplicate its functionality [45]–[52], and;
- attacks that leverage API query access, through exploiting the input-output predictions on a software level [4], [8]–[16]. These attacks are the main focus of our paper, therefore, we describe them in more details in the next section.

5) *API based extraction attacks*: Jagielski et al. [16] taxonomize model extraction around the two adversarial objectives: *accuracy* and *fidelity*. Accuracy measures how well the extracted model is performing on the underlying learning task and the goal is to extract a model that tries to make accurate predictions. Fidelity measures the matching between the predictions of the extracted model and the victim model on any input.

Initial works on stealing attacks have been demonstrated by Lowd & Meek [53] where they extract linear classifiers such as support vector machines (SVM) with linear kernels and logistic regressions (LR). They assume that the adversary has a black-box access to the oracle and the queries return just the predicted class label. Tramer et al. [8] propose more attack techniques that works on simple linear and non-linear architectures such as SVMs, LRs, decision trees and simple neural networks. They present the scenario of ML model stealing where a data owner has a trained model f and allows others to make prediction queries, an adversary uses q prediction queries to extract an $\hat{f} \approx f$. Some of their attacks are based on equation-solving to find the parameters of a target model relying on the queries. They also suggest a path-finding algorithm to extract decision trees.

Papernot et al. [4] propose an extraction attack technique that facilitates crafting transferable adversarial examples fooling the target victim. The idea of their attack is to select a substitute Deep Neural Network (DNN) architecture to the attacked DNN and train it in a way that imitates the target model using synthetic data generation and labeling the data with the target model. They assume that the attacker has a black-box capability, but initially draw a dataset from the same distribution to query the target model, then using a Jacobian-based data augmentation approach, they find examples defining the decision boundary of the target model. Finally, using these labeled examples they obtain a high-fidelity surrogate model. Then, they craft adversarial examples that are transferable for evading the classification on the original target model. Juuti et al. [22] present a concurrent framework which

relies on synthetic data generation with Jacobian-based data augmentation and randomly perturbing color channels. They further investigate selecting hyperparameters for the surrogate model instead of using fixed ones.

Correia-Silva et al. [9] demonstrate that a Convolutional Neural Network (CNN) can be copied using public data to query the target model. Their technique "*Copycat CNN*" leverages a mix of problem domain and non-problem domain data. They follow the same concept of generating a fake dataset labeled by the target network, then use it to train the copycat network. They successfully extracted models on problems such as facial expression, object and crosswalk classification. Orekondy et al. [10] propose "*Knockoff Nets*" attack where the adversary is lacking knowledge about the train/test data used by the target model and its internal architecture. They assume that the attacker is only capable to interact with the victim model through querying it with images and observing the predictions. Like previous work, their attack is based on querying the target model with samples and constructing a dataset which serves to train the knockoff model. They investigate more complex DNNs and rely on publicly available datasets to steal the victim model such as ILSVRC [54], [55] and OpenImages [56].

In addition to computer vision tasks, model extraction is also effective on natural language processing (NLP) tasks. Krishna et al. [57] demonstrate an extraction attack against pretrained and fine-tuned large language models such as BERT [58]. The attacker uses the same pretrained model as the one assumed to the victim and fine-tunes it on his obtained fake dataset. In order to build this dataset, the adversary queries the target model with random sequences of words coupled with task-specific heuristics and does not need to have grammatical nor semantical meaning, and then uses the victim outputs as labels.

Recently, other works in extraction attacks focus on the techniques used to generate the adversary's dataset instead of relying on public datasets [19], [59]. In addition to synthetic data generation and data augmentation, they leverage generative models to generate data with an objective that enables a better extraction [18].

D. Positioning with the state-of-the-art

Stealing attacks are diverse and were proven to be successful in many cases, yet they were only demonstrated on unaltered and standard state-of-the-art models that were trained under natural training circumstances. They still do not consider security-imposed scenarios that modify the model's architecture or its behavior for instance towards adversarial examples. Adversarial training impacts the prediction probabilities that could be leveraged by malicious users to perform other attacks through observing the model's response to their queries. Particularly, adversarially robust models were proven to be more vulnerable to membership inference attacks due to their wider generalization compared to natural models which raises concerns about the training data privacy [27]. Extraction attacks rely on the predictions probabilities of the victim model to the adversary's queries and have not yet been evaluated on

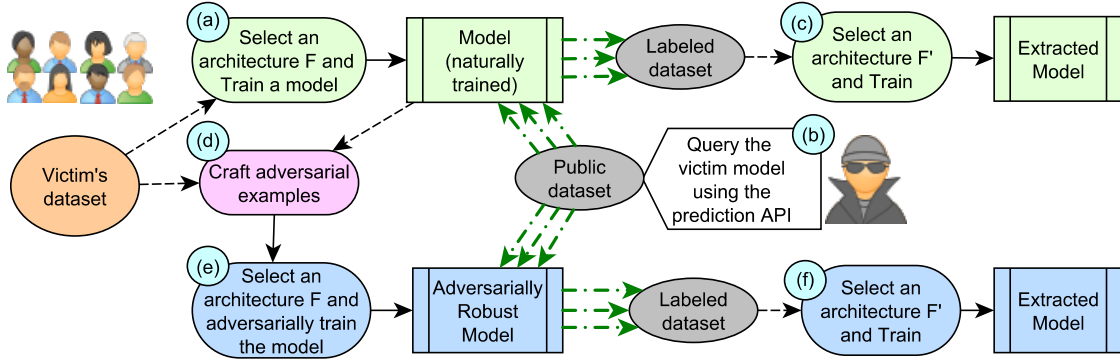


Fig. 2. Evaluation of the extraction risk of natural models versus adversarially trained models. First, model owners train an oracle under natural training circumstances (a) and offer their model for queries with an API. The adversary labels a public dataset through observing the queries responses from the victim model (b) which he uses later to train a surrogate, called extracted model (c). Alternatively, a model owner generates adversarial examples (d) to adversarially train their model (e) which the adversary would extract using the same process (b), (f).

adversarially robust models, hence, the extraction robustness of the latter is still an open research problem. It is unknown if we alter the models through adversarial training, we increase or not their vulnerability to extraction. In order to answer to this question, we propose a new methodology and a significant empirical study.

IV. METHODOLOGY

Consider a victim classifier trained solely on benign (i.e., natural) examples and another victim trained on both benign and adversarial examples to increase its adversarial robustness. We call the former *Natural model* and the latter *Adversarially (Adv.) trained model*.

A. Threat model

In our work, we consider the state-of-the-art extraction attack KnockoffNets [10] that aims to steal the functionality of black-box model trained with an architecture F through leveraging the API access. Fig. 2 gives an overview of our methodology. Following the attack strategy, we query a victim, trained on a dataset D_V , with a different dataset D_A and through observing the obtained prediction probabilities for each sample, we obtain a labeled *transferset* (Fig. 2 (a), (b)). Then, we select an architecture F' to train the surrogate model leveraging the probabilities as soft labels for the adversary's dataset D_A (Fig. 2 (c)). The attack is performed in black-box settings, i.e., the adversary has no prior knowledge about the victim's architecture nor the training set.

B. Extraction of Adversarially trained models

We adopt different state-of-the-art techniques to generate adversarial examples, namely Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD):

- FGSM [60] uses the gradient of the loss with respect to the input image to craft new adversarial image that maximizes the loss.

$$x_{adv} = x + \varepsilon \text{sign}(\nabla_x L(\theta, x, y)) \quad (1)$$

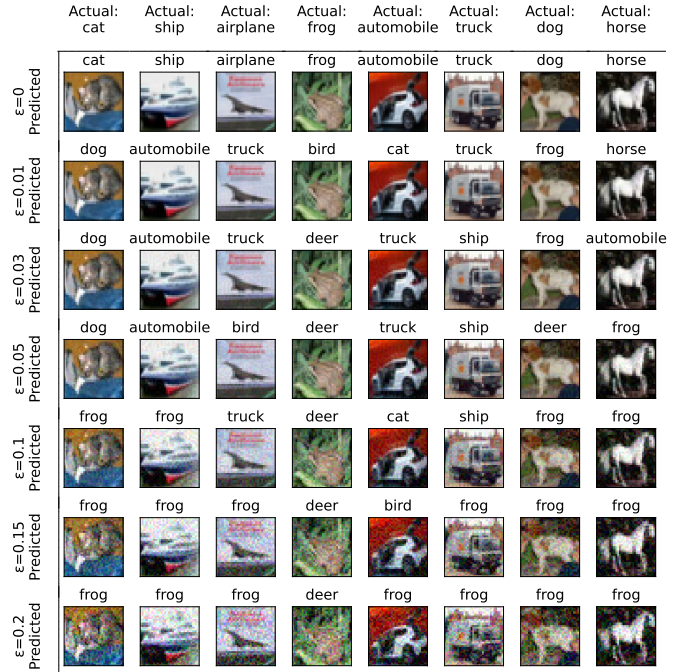


Fig. 3. A demonstration of PGD evasion attack on a ResNet-34 architecture trained on CIFAR-10 dataset. The header shows the actual ground truth labels. Then, for each row, we visualize a set of the resulting adversarial images and their prediction labels with relation to the attack perturbation level ε . For lower ε values, the attack is imperceptible, yet effective in changing the predicted labels. The higher the perturbation level ε , the more visible the distortion on the adversarial examples.

The equation 1 summarizes this technique, where x_{adv} is the adversarial image, x the input image, y the input label, ε a parameter to control the perturbations amplitude, θ the model parameters, L the loss of the trained model and ∇_x the gradient with respect to x .

- PGD [26] is an iterative method that is considered as a multi-step variant of the previous technique, which is a projected gradient descent on the negative loss function.

The PGD attack contains T gradient descent steps.

$$x^{t+1} = \Pi_{x+S}(x^t + \varepsilon \text{sign}(\nabla_x L(\theta, x, y))) \quad (2)$$

Equation 2 summarizes this technique, where Π_{x+S} denotes the projection onto the perturbation constraint $x+S$ and S is the set of allowed perturbations.

We demonstrate in Fig. 3 a PGD attack that generates an adversarial example for each image by adding an imperceptibly small vector. The resulting malicious input changes the prediction of each image by ResNet-34 [61], one of the state-of-the-art classifiers for images.

Leveraging the generated adversarial examples from either the PGD technique or the FGSM technique, we expand the training set and retrain the models (Fig. 2 (d), (e)). We obtain adversarially robust models. We evaluate the robustness of obtained models against adversarial examples generated with both techniques. Finally, we evaluate the extraction risk of these obtained adversarially robust models (Fig. 2 (b), (f)).

C. Evaluation metrics

To evaluate the success of extraction attacks, we calculate the *accuracy* of the surrogate model on the victim’s test set, which is the fraction of correct predictions from all predictions. This metric measures the stolen functionality from the victim, i.e., how well the surrogate model performs on the same task. In addition, we measure the *agreement* (i.e., fidelity) between the victim model and the extracted model through calculating the fraction of matching predictions (both correct and wrong) from the victim and the surrogate model, this is similar to calculating the accuracy of the extracted model using the predictions of the attacked victim as ground truth labels. Both metrics are calculated using a heldout labeled test set that was not seen before by the victim (during the training) nor the extracted model.

Then, we evaluate the success of extraction attacks against *Adv. trained models* versus *Natural models* and study whether making the models robust against adversarial examples makes them more vulnerable to extraction attacks. Finally, we measure the robustness of victims and surrogates against evasion attacks. We compute the *adv. accuracy* of each model against various adversarial examples generated with FGSM and PGD techniques using multiple levels of perturbations ε .

V. EXPERIMENTS

A. Setup

In our experiments we tackle DNNs trained on 3 benchmark images datasets: MNIST [62], CIFAR-10 [63] and SVHN [64]. We describe these datasets as well as the adversary’s datasets in Table I. Following our methodology, we begin by evaluating the extraction risk of naturally trained models, then of adversarially robust models.

First, we train a DNN on a given dataset. In our work, for both CIFAR-10 and SVHN we fine-tune a ResNet-34 [61] pretrained model on ImageNet. For MNIST, we train a simple CNN architecture composed of two 2D-convolutional blocks, max pooling, dropouts and fully connected layers [65]. In

TABLE I
SELECTED DATASETS IN OUR EXPERIMENTS. THE FIRST THREE ROWS ARE THE VICTIM’S DATASETS AND THE FOLLOWING ROWS ARE THE ADVERSARY’S DATASETS.

Dataset	Description	Image size	Nb. of samples	Nb. of classes
MNIST	Handwritten grayscale digits	28×28	Train: 50k Test: 10k	10
CIFAR-10	Images of animals and vehicles	32×32	Train: 60k Test: 10k	10
SVHN	Images of street view house numbers	32×32	Train: 73k Test: 26k	10
Fashion MNIST	Grayscale images of clothes	28×28	Train: 50k Test: 10k	10
ImageNet	Various real-world images belonging to 1000 different categories	224×224	Train: 1.2M Test: 50k	1000

training, we use an SGD optimizer with an initial learning rate of 0.01 that is decreased by a factor of 10 every 60 epochs over up to 200 epochs [66]. Then, we extract the victim model using the extraction attack KnockoffNets [10]. We experiment with multiple query budgets for up to 50000 samples. We use as an adversary dataset, to query both CIFAR-10 and SVHN victims, a subset of images from ImageNet resized to 32×32 pixels using bilinear interpolation to match our input shapes for both victims networks. For the MNIST victim extraction, we leverage the FashionMNIST dataset (Table I).

Using the victim’s prediction *probabilities* as labels for the adversary’s dataset, we train our surrogate models with a different architecture. We steal ResNet-34 victims with VGG-16 [67] and the MNIST CNN with LeNet architecture [68]. Surrogate models are trained using an SGD optimizer with an initial learning rate of 0.01 that is decreased by a factor of 10 every 60 epochs over up to 100 epochs. Next, we evaluate the extraction success for each of these attacks.

After that, we assess the extraction of adversarially robust models. Therefore, we take the training dataset and craft adversarial examples using one of the state-of-the-art techniques in order to use them for retraining. In our work, we use two techniques Fast Gradient Sign Method (FGSM) [60] and Projected Gradient Descent (PGD) [26] with 8 different perturbation levels $\varepsilon \in [0.01, 0.3]$. Fig. 3 illustrates some adversarial examples generated for the CIFAR-10 dataset using the PGD attack.

Using the generated adversarial examples from one of the techniques for a selected perturbation levels $\varepsilon \in [0.01, 0.15]$, where the noise added to the images does not fully deteriorate its visual perception, we augment the training set and retrain our models using the same training hyperparameters for the naturally trained victims. We obtain 8 *adversarially robust* models for each ε with PGD and FGSM techniques. Then, we perform the same extraction attack procedure on these models. Finally, we compare the success of each attack on natural models versus their adversarially robust versions.

All experiments were performed on a Ubuntu 20.04 operating system with 8-core processor (3.7GHz ×8) and a GPU NVIDIA Quadro RTX 6000. We used Weights & Biases

TABLE II

RESULTS OF EXTRACTION ATTACKS WITH SELECTED BUDGETS (B) AGAINST NATURALLY TRAINED DNN AND ADV. TRAINED DNNs WITH DIFFERENT LEVELS OF ε (DATASET: SVHN). THE BEST ACCURACY AND AGREEMENT OF EXTRACTED MODELS FOR EACH BUDGET B ARE IN **BOLD**.

Victim Model	ε	Test Acc.	Extracted Models Accuracy & Agreement					
			B=15k		B=25k		B=50k	
Natural	-	96.14	57.17	57.50	71.65	72.01	84.29	84.84
Adv. trained FGSM	0.03	96.33	59.36	59.73	72.08	72.62	85.69	86.29
	0.05	96.54	61.07	61.42	76.34	76.79	86.46	86.92
	0.10	96.41	62.91	63.35	78.12	78.74	86.36	86.97
	0.15	96.37	61.48	61.85	75.52	76.03	84.07	84.62
Adv. trained PGD	0.03	96.53	59.36	59.73	72.08	72.62	84.57	85.17
	0.05	96.40	63.09	63.37	75.21	75.59	86.76	87.37
	0.10	96.47	65.39	65.83	76.26	76.76	85.72	86.28
	0.15	96.42	62.76	63.34	74.81	75.29	84.81	85.54

(WandB) [69] for experiment tracking and visualizations to develop insights for this paper ².

B. Results

1) Extraction of natural and adversarially trained models:

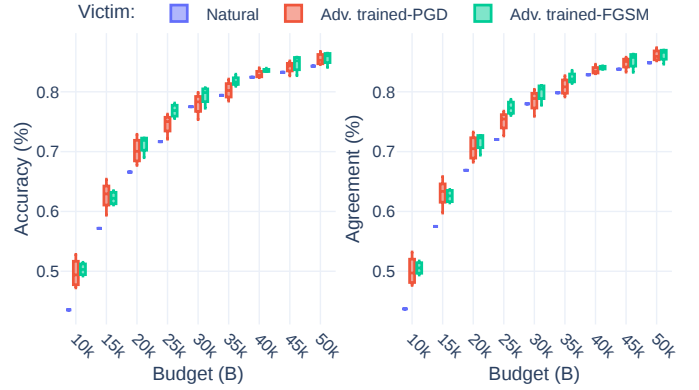
Table II shows detailed results about the extraction attacks on naturally trained and adv. trained models on SVHN dataset. Each adv. trained victim is either an FGSM or PGD Adv. trained model with a specific $\varepsilon \in [0.03, 0.15]$. Each victim is extracted using a set of queries budget (B). We visualize the *accuracy* and the *agreement* results of stolen models from different types of victims for SVHN, CIFAR-10 and MNIST datasets in Fig. 4. We observe that adv. trained models have higher accuracy and higher agreement than naturally trained models. This gap is more noticeable with smaller budgets. This entails that adv. trained models are faster to extract compared to naturally trained models, i.e., adv. trained models require less number of queries in a stealing attack to have a higher extraction results.

We quantify in Fig. 6 the accuracy and agreement gains for the extraction of adversarially trained models with respect to naturally trained models depending on the noise level ε that we use to generate adversarial examples for adversarial training. For both types of adv. trained models with different levels of ε , in most cases the average gain in accuracy and agreement is higher than a naturally trained model. We find that surrogate models from the adv. trained models reach up to $\times 1.2$ the extraction metrics of models stolen from a normally trained victim.

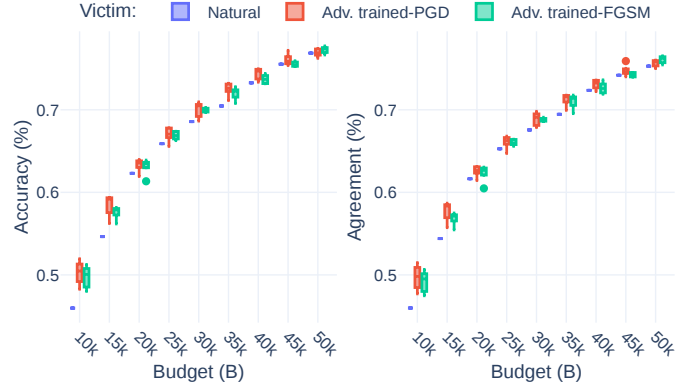
2) Adversarial robustness of victims and surrogate models:

We evaluate the robustness of all models against several sets of adversarial examples with different levels of ε . Fig. 5 shows the adversarial accuracy of each model. Note that naturally trained models are the most vulnerable to evasion attacks. We discover that an extracted model have a higher robustness against adversarial examples. The adversarial training of our victims is verified to be effective as a defense against evasion attacks as it shows an exceedingly better adversarial

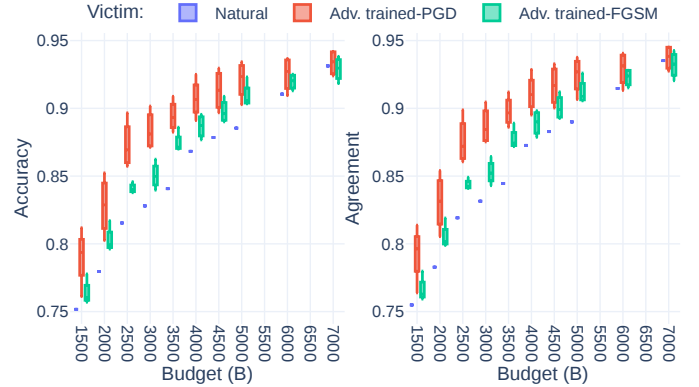
²Our project runs and visualizations are available at the WandB link: <https://wandb.ai/kacem/MSF>



(a) Dataset: SVHN



(b) Dataset: CIFAR10



(c) Dataset: MNIST

Fig. 4. Impact of the adversarial training on the extraction attack success. For each victim model type, multiple extraction attacks are performed with different numbers of queries (B). The adv. trained models from FGSM and PGD results are grouped in box plots. For each technique, the boxes present results from victims with different levels of $\varepsilon \in [0.03, 0.15]$ that were used in the adversarial training.

accuracy compared to naturally trained victims. Additionally, we discover that stolen models from adv. trained victims are as robust as the target models against adversarial examples and their adv. accuracy is higher than one of a stolen model from a natural victim. This demonstrates that the capability to

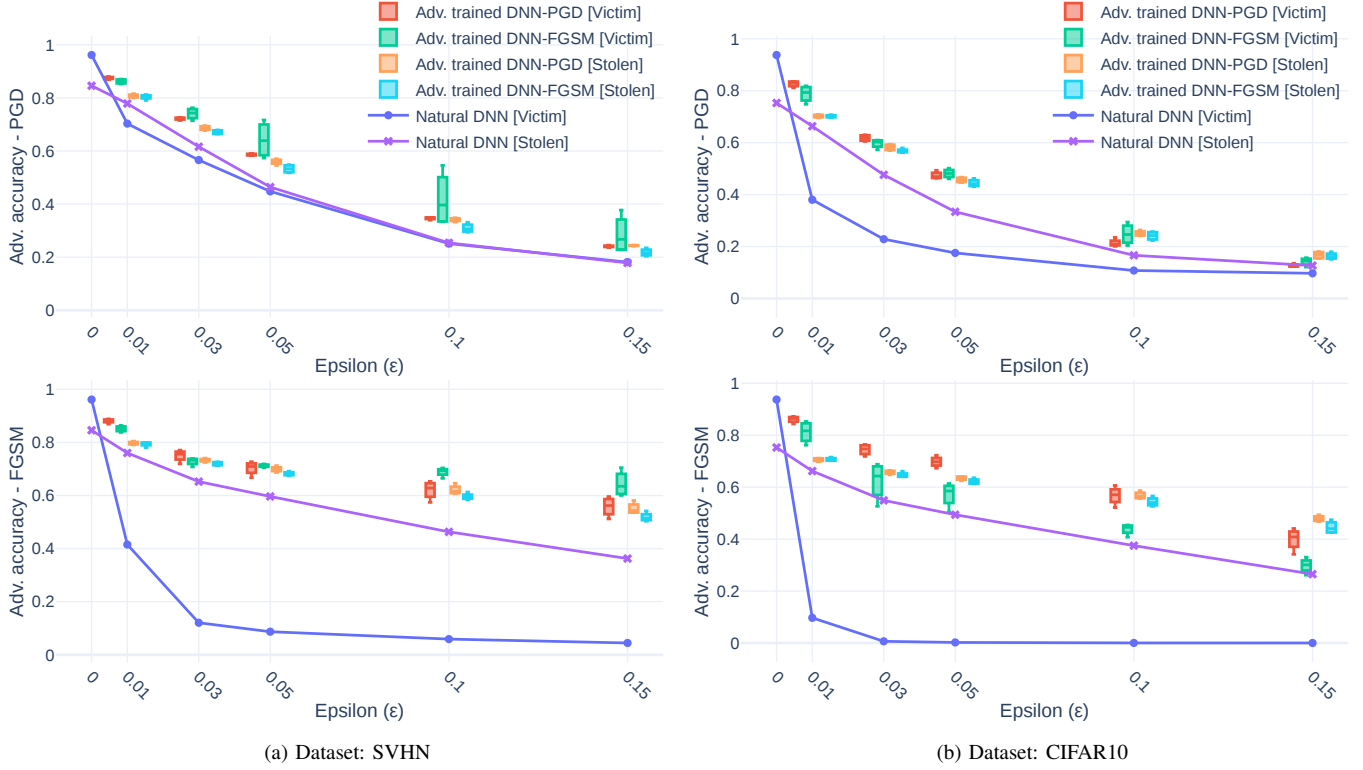


Fig. 5. Adversarial robustness of victims and corresponding stolen models against multiple sets of adversarial examples with different ϵ levels. The top row shows the accuracy against adversarial examples generated with PGD attack, while the second row concerns the FGSM attack. For each victim model type, we select the corresponding stolen model obtained with the highest budget in the performed extraction attack. The adv. trained models from FGSM and PGD trained with different ϵ levels are grouped in box plots as well as each of their corresponding stolen models.

be robust against adversarial examples is transferable through extraction attacks.

C. Discussion

Our evaluation demonstrates that defending against adversarial examples through adversarial training may increase the vulnerability of models against extraction attacks. This proves that following a security-imposed scenario, the privacy of models can be jeopardized through stealing attacks.

In addition, our work finds that stealing adversarially trained models rewards the thief with a surrogate model that is robust to adversarial examples. In fact, this finding may be explained by the generalization capacity of surrogate models. Since stolen models are trained by images from outside the distribution of the attacked victim’s train set, the adversarial examples, which were crafted to evade the classification of the victim, fool less the extracted models. Hence, this makes the latter more robust to adversarial examples. This concept have some intersections with a prior defense against adversarial examples through knowledge distillation of a model to another one using a temperature T [44]. In fact, both distillation and extraction attacks rely on training a model with data labeled by another model. However, the defensive distillation technique relies on the same training data rather than a different out of distribution images. Besides, this technique is performed by

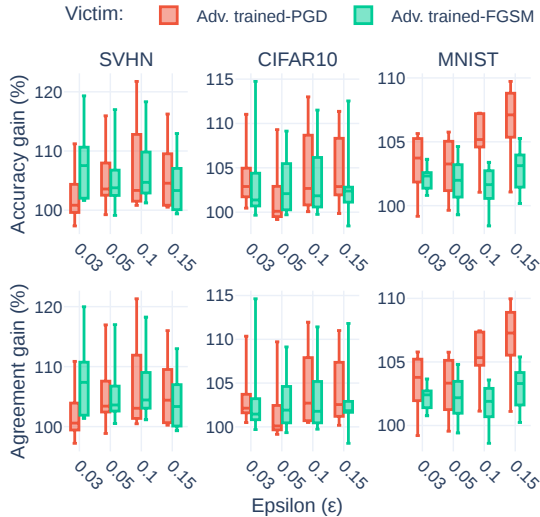


Fig. 6. Accuracy and agreement gains for the extraction of adversarially trained models compared to naturally trained models for different ϵ levels. The adv. trained models from FGSM and PGD results are grouped in box plots. For each technique, the boxes present results from multiple extraction attacks which were performed with different numbers of queries (B).

the model owner (white-box settings), not a malicious user that aims to steal the model (black-box settings).

VI. DEFENSES AND COUNTERMEASURES

Model extraction attacks have several drawbacks on ML security and privacy, therefore, researchers propose different techniques to defend against them and mitigate their risk. Some techniques suggest watermarking the model to help claim it when stolen [70], [71], for example by changing the output probabilities for a small subset of queries (e.g., $\leq 0.5\%$) from API clients which allows model owners to reliably demonstrate ownership. Other defenses can be broadly categorized into two main categories: *proactive* and *reactive* techniques. The former includes techniques to reduce efficiency of an extraction attack such as changing the prediction probabilities [21], [23], [24]. But, they may sometimes degrade the model accuracy and result in inference delays and an increase in computational costs. Reactive defenses are based on the detection of extraction attacks when they are happening which enables the victim to take action immediately. These techniques continually observe the API query and response streams of users in order to either train a substitute model to assess the knowledge gained by the adversaries [20], or analyse the distribution of successive users queries to identify a deviation from a normal (Gaussian) distribution [22] or a large number of out of distribution queries [23]. However, these defenses do not detect all the attacks and they often consume more computational power.

VII. CONCLUSION

Despite the tremendous progress of neural networks and their trending and wide use in critical applications such as robotics and autonomous vehicles, they are still showing flaws and vulnerabilities in both security and privacy. We have systematically explored model privacy threats through extraction attacks on a security-imposed scenario of adversarially trained models on three benchmark vision datasets. We discovered that adversarially trained models might have a higher extraction rate compared to natural models for a lower number of queries. This calls for a required attention from model owners when improving the security of their models against adversarial examples through adversarial training. We additionally found that extracted DNNs from adv. trained models show an enhanced robustness against adversarial examples compared to extracted DNNs from naturally trained models.

REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [2] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR 2011*. IEEE, 2011, pp. 3361–3368.
- [3] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.
- [4] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.
- [5] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3645–3650.
- [6] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.
- [7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
- [8] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.
- [9] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, "Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.
- [10] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4954–4963.
- [11] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, and M. Iyyer, "Thieves on sesame street! Model extraction of bert-based apis," *arXiv*, jan 2019.
- [12] R. N. Reith, T. Schneider, and O. Tkachenko, "Efficiently stealing your machine learning models," in *Proceedings of the 18th ACM workshop on privacy in the electronic society*, 2019, pp. 198–210.
- [13] J.-B. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4771–4780.
- [14] S. Pal, Y. Gupta, A. Shukla, A. Kanade, S. Shevade, and V. Ganapathy, "ActiveThief: Model Extraction Using Active Learning and Unannotated Public Data," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 865–872, 2020.
- [15] X. Yuan, L. Ding, L. Zhang, X. Li, and D. O. Wu, "Es attack: Model stealing against deep neural networks without data hurdles," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.
- [16] 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)*, 2020, pp. 1345–1362.
- [17] X. Yuan, L. Ding, L. Zhang, X. Li, and D. Wu, "Es attack: Model stealing against deep neural networks without data hurdles," *arXiv preprint arXiv:2009.09560*, 2020.
- [18] S. Kariyappa, A. Prakash, and M. Qureshi, "Maze: Data-free model stealing attack using zeroth-order gradient estimation," *arXiv preprint arXiv:2005.03161*, 2020.
- [19] A. Chawla, H. Yin, P. Molchanov, and J. Alvarez, "Data-free knowledge distillation for object detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 3289–3298.
- [20] M. Kesarwani, B. Mukhoty, V. Arya, and S. Mehta, "Model extraction warning in mlaas paradigm," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 371–380.
- [21] T. Lee, B. Edwards, I. Molloy, and D. Su, "Defending against machine learning model stealing attacks using deceptive perturbations," *arXiv preprint arXiv:1806.00054*, 2018.
- [22] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "Prada: protecting against dnn model stealing attacks," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 512–527.
- [23] S. Kariyappa and M. K. Qureshi, "Defending against model stealing attacks with adaptive misinformation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 770–778.
- [24] T. Orekondy, B. Schiele, and M. Fritz, "Prediction poisoning: Towards defenses against dnn model stealing attacks," in *International Conference on Learning Representations*, 2019.
- [25] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

- [26] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.
- [27] 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, pp. 241–257.
- [28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [29] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. USA: Prentice Hall Press, 2009.
- [30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [31] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 2006, pp. 16–25.
- [32] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.
- [33] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrđić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.
- [34] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.
- [35] I. Goodfellow, "Defense against the dark arts: An overview of adversarial example security research and future research directions," *arXiv preprint arXiv:1806.04169*, 2018.
- [36] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. A. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," *LEET*, vol. 8, pp. 1–9, 2008.
- [37] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *International Conference on Machine Learning*. PMLR, 2015, pp. 1689–1698.
- [38] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar, "Antidote: understanding and defending against poisoning of anomaly detectors," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, 2009, pp. 1–14.
- [39] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 19–35.
- [40] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings*, no. c, pp. 1–14, 2017.
- [41] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *MILCOM 2016-2016 IEEE Military Communications Conference*. IEEE, 2016, pp. 49–54.
- [42] A. Kurakin, I. Goodfellow, S. Bengio *et al.*, "Adversarial examples in the physical world," 2016.
- [43] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.
- [44] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks," in *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*. San Jose, CA: IEEE, may 2016, pp. 582–597.
- [45] 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," vol. 14, 2018.
- [46] M. Yan, C. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn DNN architectures," pp. 2003–2020, 2018.
- [47] S. Hong, M. Davinroy, Y. Kaya, S. N. Locke, I. Rackow, K. Kulda, D. Dachman-Soled, and T. Dumitras, "Security analysis of deep neural networks operating in the presence of cache side-channel attacks," 2018.
- [48] L. Batina, D. Jap, S. Bhasin, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," *Proceedings of the 28th USENIX Security Symposium*, pp. 515–532, 2019.
- [49] X. Hu, L. Liang, S. Li, L. Deng, P. Zuo, Y. Ji, X. Xie, Y. Ding, C. Liu, T. Sherwood *et al.*, "DeepSniffer: A dnn model extraction framework based on learning architectural hints," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 385–399.
- [50] S. Hong, M. Davinroy, Y. Kaya, D. Dachman-Soled, and T. Dumitras, "How to own nas in your spare time," *arXiv*, pp. 1–16, 2020.
- [51] Y. Liu and A. Srivastava, "Ganred: Gan-based reverse engineering of dnns via cache side-channel," in *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, 2020, pp. 41–52.
- [52] 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.
- [53] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 641–647.
- [54] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [55] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [56] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Mallocci, A. Kolesnikov *et al.*, "The open images dataset v4," *International Journal of Computer Vision*, pp. 1–26, 2020.
- [57] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, and M. Iyyer, "Thieves on sesame street! model extraction of bert-based apis," *arXiv preprint arXiv:1910.12366*, 2019.
- [58] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [59] P. Han, J. Park, S. Wang, and Y. Liu, "Robustness and diversity seeking data-free knowledge distillation," *arXiv preprint arXiv:2011.03749*, 2020.
- [60] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
- [61] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [62] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, vol. 2, 2010.
- [63] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
- [64] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.
- [65] S. Chintala, "Pytorch mnist example," <https://github.com/pytorch/examples/blob/main/mnist/main.py>, 2020.
- [66] B. G. Atli, S. Szyller, M. Juuti, S. Marchal, and N. Asokan, "Extraction of complex dnn models: Real threat or boogeyman?" in *International Workshop on Engineering Dependable and Secure Machine Learning Systems*. Springer, 2020, pp. 42–57.
- [67] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [68] Y. Lecun, L. Bottou, Y. Bengio, and P. Ha, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, no. November, pp. 1–46, 1998.
- [69] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: <https://www.wandb.com/>
- [70] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, "Dawn: Dynamic adversarial watermarking of neural networks," *arXiv preprint arXiv:1906.00830*, 2019.
- [71] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," *arXiv preprint arXiv:2002.12200*, 2020.