Who Guards the Guardians? On Robustness of Deep Neural Networks

Misha Glazunov and Apostolis Zarras

Abstract In this chapter, we describe available vectors of attacks against discriminative Deep Neural Networks. We consider a wide range of attacks that aim either to mislead and change the model's behavior or to leak information about the training data and potentially about the model in use. These attacks can be readily mapped within the Confidentiality, Integrity, and Availability triad components. We lay out the potential threat models and include the most prominent examples of malicious exploitation utilizing the artificially crafted adversarial samples provided to the model as input. We cover both types of such inputs: the ones utilized during the training, the so-called poisonous attacks, and the ones applied during the testing, the adversarial examples. Both of these categories cover the wide range of attacks that target changing the behavior of the underlying model. Moreover, we include the often overlooked category in our description: the outliers, which can be exploited as adversarial inputs. In addition, we cover two powerful attacks aimed at breaking privacy: model stealing and membership inference. Finally, we outline the current defenses against these attacks and conclude with a summary.

1 Introduction

Deep learning, powered by Deep Neural Networks (DNNs for short), underlies most modern Artificial Intelligence (AI) algorithms. These models have proven highly successful in various applications, including supervised, unsupervised, and reinforcement machine learning. Furthermore, deep learning models have demonstrated state-of-the-art results in processing diverse data domains such as images, texts, audio recordings, electrocardiograms, malware, games, and many more. Despite their

Misha Glazunov
Delft University of Technology, the Netherlands

Apostolis Zarras
University of Piraeus, Greece

impressive accuracy metrics, the fundamental theory of deep learning is missing. First, the most critical lacking aspect is linked to the generalization capabilities of the modern DNNs. Generalization can be subdivided into (i) generalization from training data to test data and (ii) generalization from inliers to outliers. Currently, there is no satisfying theory, even for the first part, which would explain the surprising generalization capabilities of DNNs. Suppose one employs the classical statistical learning theory based on the Vapnik–Chervonenkis dimension [58]. In that case, it likely indicates that DNNs are overfitting, considering that the number of parameters quite often significantly exceeds the number of data points used for DNN training. Many research works contribute to the gradual progress in this direction [12, 4, 50, 34], but no final theory is currently available. The second part is also an active direction of current research [1, 66, 28]; however, nowadays, there is no theoretically rigorous foundation for that either. As a result, both of the parts require further exploration. Moreover, the second missing cornerstone relates to the optimization routine, e.g., to the convergence regime of DNNs: pointwise vs uniform. There are several studies in this direction [64, 67, 65] and their connections to the generalization [41] but, again, there is still no rigorous theoretical basis for that. Finally, the architectural choices for DNNs are still empirical, and no theory unifies them except for the very coarse results of the well-known Universal Approximation Theorem (UAT) [10] and its variations that basically aim at rigorous bounding either the depth [21, 49, 3] or the width [35] of the DNNs. The lack of rigorous understanding of the theory of deep learning results in many relevant questions that still need to be answered. For instance, how exactly does the model produce the final result? How can we be sure about these results? What are the limits of a particular modeling approach? How can it be rectified in case of malfunction? And many more others.

Therefore, these models are often used as a black-box tool, fed with significant data to achieve the desired performance results. However, this raises serious concerns about the robustness of these models to malicious interventions because if there is no clear understanding of how modern DNNs generalize from training data to test data (i.e., on inliers and, further, on outliers), then it is an obvious direction of exploiting this lack of understanding by probing the models for the potential vulnerabilities, especially when they are used in so many critical domains, ranging from autonomous driving to medical diagnosis. It should be noted that despite the significant progress in the methods of model interpretability such as SHAP [37], GradCAM [54], CAM [69], RISE [48] and CXPlain [53], all of these methods can be applied to any model, and they treat the model in question as a black-box with only potential access to the loss function and/or gradients. The interpretations are subsequently derived from input features that indicate the decision made by the model based on these input features and their corresponding contributions or weights for a particular decision. However, none of these methods tell anything about the model's internal mechanics from the fundamental theoretical perspective, e.g., how exactly did the particular DNN come up with this particular solution and why (all the research dedicated to the generalization and optimization of the DNNs)? How does the network architecture influence the final decision and why (vast volume of different ad-hoc solutions with too few theoretical foundations)? What can be done to alleviate the wrong output and

modify the decision theoretically rigorously (the operational question that logically follows from the first two)? Can we debug the model, identify why these input features are essential for this model, and fix the wrong outputs? (i.e., the practical question based on the previous one).

Consequently, many practical attacks have been discovered that exploit this lack of understanding, which, in turn, assist us a lot in more profound research and comprehension of the internal mechanisms of deep learning. Let us momentarily take the role of an attacker and consider the potentially vulnerable spots that can be exploited from the general perspective in any machine learning modeling approach. Such an approach requires at least two parts: (i) data to be trained on and inferred from and (ii) a particular model to be trained and subsequently used for inference.

Moreover, the attackers may not have access to both model and data. In addition, both the model and data may evolve with time. Furthermore, a model can be a composite one, i.e., consisting of several other machine learning sub-models, and data can be non-homogenous or multimodal. Based on various combinations of these components, different potential scenarios are available for the attacker depending on their interests. For example, the attackers may be interested in modifying the model's behavior for particular inputs, e.g., they want to bypass a spam filter to organize an advertising campaign for an illegal product. Alternatively, they may want to trick a face recognition system to access the victim's smartphone. Conversely, the attackers may be interested in the data; for example, they may have access to the model and can get inference results but lack knowledge about the dataset used to train the model, which could be of particular interest to them. Alternatively, suppose the attackers cannot access the model. In that case, they may be interested in getting it, e.g., the model architecture, the weights obtained after training, and the hyperparameters used for training.

Therefore, grounded in the distinctive attack vectors, we can discern two principal categories of potential attacks. (i) *Functionality-oriented attacks*, encompassing those that seek to manipulate the model's behavior. These may transpire during either the training or inference stages. (ii) *Privacy-oriented attacks*, encapsulating those designed to extract insights into the private data employed during both training and inference stages since models may be trained on private data, such as personal information, medical records, or financial data, which can be extracted from the trained model breaching the privacy. Moreover, even if the training data is anonymized, recovering the data the model was trained on may leak proprietary business information. In addition, this attack category may aim to unveil proprietary facets of the machine learning model itself. If the model is deployed in the cloud and used via the web, then the model architecture and parameters leakage represent a serious proprietary trade secrets privacy breach.

The first kind of functionality-oriented attack strongly relates to the unknown inputs since we assume the model is not under the attacker's control. Hence, something should be done with its input to modify the model behavior. Unknown inputs can be further divided into two main categories. The first category includes inputs that differ significantly from those on which the model has been trained. For instance, if a deep neural network classifier is trained to classify handwritten digits, how will it

process images of unknown numerical systems, languages, or even natural objects such as trees or dogs? Experiments demonstrate that DNNs overconfidently fail when processing such inputs as they cannot distinguish between them and those they have been trained on [47, 60]. Such inputs are closely related to the question of the proper model generalization, and they are referred to as outliers. The model deployed in the wild should be able to deal with the outliers, i.e., with the data points coming from a different distribution than the one the model trained on [19]. In such a case, the question arises of how good the model is in dealing with such inputs and, most importantly, if it can distinguish outliers versus inliers. The second category includes inputs specifically manufactured by adversaries to change the model's output, such as adversarial examples. These examples are becoming increasingly prevalent across all learning approaches, model architectures, and data domains.

The attacker utilizes outliers and adversarial examples in the inference stage when the model is already pre-trained. Alternatively, the poisonous attacks are introduced during the training stage. These are also artificially forged inputs that tend to be covertly introduced in the training dataset with the aim of the subsequent exploit during the inference stage.

The second kind of privacy-oriented attacks includes model stealing and membership inference attacks.

In this chapter, we briefly introduce the available attacking techniques and indicate the potential directions for their mitigation and, if possible, detection.

2 Attacking Deep Neural Networks

In this section, we introduce appropriate threat models together with potential vectors of attacks at DNNs. It includes both functionality-oriented and privacy-oriented threats. The former comprises outliers, adversarial, and poisonous examples. The latter includes attacks that aim at stealing private or sensitive data related either to the model parameters and architecture or to the inputs used during the training of the model, i.e., to the training dataset. Finally, these attacks are mapped within the Confidentiality, Integrity, and Availability triad components.

2.1 Threat Models

We consider the two most common threat models concerning the attacker's abilities: white-box and black-box scenarios. The former relates to the situation when the attacker knows, for instance, the DNN model, including its architecture, learned weights, used hyperparameters, input layer representation, and output layer results. This scenario implies that the attacker has full knowledge of both model and training

data in use.[1] Conversely, the latter applies to cases without direct knowledge of the DNN model. The attackers may only know the inputs used for the model. Sometimes, they can also use the model as a service via a particular API call to infer the outputs. Moreover, it usually denotes the lack of exact knowledge about the data used during the training stage of the model. Nevertheless, the general understanding of the data is particularly always available or can be inferred, such as, for example, in the case of face detection on the image, the training dataset should have included images with faces.

The privacy-oriented attacks imply the black-box scenario. Functionality-oriented attacks, on the other hand, may concern both threat models. The usual approach for them in the case of a black box is to train the so-called surrogate model, which will be used for the proper adversarial or poisonous input generations and the subsequent evaluations of the attack success rates. The properties of this surrogate model depend on the DNN under attack, and it usually entails the initial reconnaissance step with respect to the model in use potentially involving privacy-oriented attacks.

Beyond the available knowledge of the attackers, as we mentioned before, there is also a distinction between their possibilities over data manipulation, namely, if they can manipulate data during training or inference stages. The first type of attacks is called poisonous or sometimes causative, and the second type is called exploratory or evasion attacks. The latter includes both adversarial examples and outliers. We will use both mentioned terms for each of the types interchangeably.

2.2 Functionality-oriented Attacks

First, we explore the most active domain of the current research dedicated to misleading the deep learning model behavior utilizing specific inputs. Since such inputs can be provided during the training or inference stage of the DNN life cycle, we dedicate a separate section to each stage.

2.2.1 Exploratory Attacks on Inference

Inference relates to the stage when the model is already trained and validated on a particular dataset. Hence, it can now *infer* the parameters of the probabilities of the general population based on particular provided inputs. Two types of attacks can be associated with the inference stage of any DNN: (*i*) the attacks utilizing the outliers and (*ii*) the attacks employing the adversarial examples. In the following sections, we consider each of them in more detail.

[1] Please note that it is not in contradiction with our previous assumption about the absence of the control over the model by the attacker since the knowledge of the model still does not imply the access to the model in use.

2.2.2 Outliers

We begin our discussion with the outliers. We consider that outliers are generated by a different distribution than the inputs used during training. An input generated from the different distribution represents a traditional definition of outliers, and to the best of our knowledge, it was first employed by Hawkins [19].[2] The treatment of these inputs is often considered somewhat parallel to the attacker's perspective in the scientific literature. The reason is that they are more vividly related to the question of the proper model generalization, i.e., to the ability of the model to infer the actual parameters of the probability distributions of the general population rather than to the attacking technique. Despite their traditional treatment in the research community, they can also be used to mislead the model behavior by an attacker; the model deployed in the wild should likely be able to deal with the data points coming from a different distribution to the one on which the model has been trained. For a toy example, let us consider the model trained on the MNIST dataset [30]. This dataset contains the handwritten digits. The outliers in such cases could be the inputs from a different dataset, such as FashionMNIST [61] that contains various garments or even randomly generated images. In such a case, the question arises: *How good is the model in dealing with such inputs, and, most importantly, can it distinguish outliers versus inliers?*

To better understand what the outliers mean, it is helpful to pose the following question about a model in plain English: *Does a trained machine learning model know what it does not know?* To answer this question, we first have to understand what it means exactly for the model to deduce if it knows or does not know something. First, recall that DNNs represent universal mapping approximators, i.e., they can learn any function on a compact domain with an arbitrarily close level of precision. However, what type of functions do they learn in most of the applied tasks? *Probability functions!* More precisely, DNNs parameterize either probability density or probability mass functions based on a particular input. This chapter considers attacks on deep learning models based on the supervised discriminative approach, such as DNN classifiers. Hence, such a modeling approach implies parameterizing a conditional distribution over target values y conditioned on the input $\mathbf{x}$: $p_{\boldsymbol{\theta}}(y|\mathbf{x})$. The training of DNNs allows identifying optimum parameters $\boldsymbol{\theta}^*$ based on a stochastic first-order optimization algorithm such as gradient descent. In the case of classification tasks, the standard choice for $p_{\boldsymbol{\theta}}(y|\mathbf{x})$ is a categorical distribution, in the case of regression—a Gaussian distribution (quite often with a constant variance). It may seem only logical to rely on the probability density in answering our question, namely, if the density is low, then the model is uncertain about the input, which can be interpreted as the model not knowing the current input.

Nevertheless, despite the neat theoretical motivation, such an approach drastically fails in practice. The DNNs tend to assign relatively high probability values to the outliers. For example, consider an ImageNet dataset used in Large Scale Vi-

[2] Hawkins distinguishes two categories of outliers: (i) inliers and outliers adhere to the same distribution and (ii) inliers and outliers generated by different distributions. In this chapter, we use the latter.

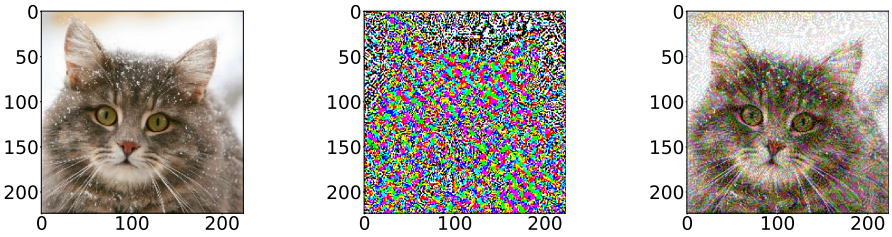
sual Recognition Challenge (ILSVRC) [51]. This dataset contains over a million images for 1000 object classes, including various types of animals, aircrafts, fruits, vegetables, etc. However, despite the vast coverage of potential objects, these 1000 classes do not include images of a microscope or a measuring tape. What would be the probability value for these unknown categories if we train a DNN classifier on ILSVRC based on those 1000 classes? Surprisingly, it turns out to be relatively high! For example, in the case of an image with a microscope, the DNN is 98.18% confident that it is a joystick, and in the case of an image with a measuring tape that it is a chainsaw with a 90.54% probability. Both joystick and chainsaw are present among those 1000 classes, but what is astonishing is the certainty with which the DNN claims to detect objects it has never seen before.

This experiment demonstrates an interesting fact about DNNs: they tend to be *overconfident* with the outliers in their predictions. Such overconfidence has at least two disadvantages. First, it makes it impossible to detect the outliers during the inference stage utilizing the probability values, making such DNNs less suitable for the tasks when such functionality is necessary. Second, it allows an attacker to exploit this overconfidence by allocating particular outliers and providing them as inputs to such DNN, misleading the initially intended behavior, e.g., resulting in misclassification. It is pretty easy to imagine that the impact of such an attack might be life-threatening in many domains, ranging from self-driving cars to medical diagnosis.

2.2.3 Adversarial Examples

Analysis of the input-output mappings of AlexNet Convolutional Neural Network (CNN) [27] from the point of view of continuity reveals unforeseen properties of the DNNs [56]. Namely, the traversal over the manifold learned by the CNN from one category to another with parallel visualizing of the corresponding points in the input space does not result in a smooth morphing of one category to another, e.g., imagine a gradual and substantial transformation of a dog class into a cat class in the resulting images. On the contrary, the results of such a traversal turn out to be quite intriguing; namely, when the CNN classifier indicates that it already observes a different class, the input image still contains a distinct representation of the initial category (e.g., a category of a dog) with a slight amount of the added noise over the image. This means that from the human perspective, the obtained result is almost indistinguishable from the initial image. However, from the perspective of CNN, the new image represents a different category. In parallel, predictably, analysis of the DNNs' robustness against the evasion attacks at test time [6] reveals the same intriguing behavior as in [56]. Two different perspectives on this behavior give rise to two definitions of the adversarial examples: one from the perspective of the generalization properties of the DNN and the other from the attacker's perspective.

From the generalization perspective, an adversarial example [56] is a technique in which the input for the DNN image classier is intentionally modified to look almost the same as the original image to the human eye. Yet, it is perceived as something



(a) Persian cat (b) Constrained gradient of the loss (c) Peacock ($\varepsilon = 0.25$)

Fig. 1 (a) the original image; (b) the max-norm constrained gradient of the loss generated by FGSM; (c) the resulting adversarial image.

completely different by DNN. DNNs incorrectly classify such adversarial examples from the human perspective. On the other hand, the attacker perspective does not necessarily demand the part that relates to the imperceptibility of the difference. On the contrary, if the miscreants want their attack's outcome to succeed, they should not constrain themselves to the superfluous imperceptibility demands. In this chapter, nevertheless, we are more inclined to the definition that involves the imperceptibility aspect. The reason for that is the new, often overlooked, category we include in our study: the outliers. These inputs could be considered similar to the perspective of the *adversarial attack to the maximum extent* when there is nothing similar between the original data and the outliers. In particular, we consider three methods of generating adversarial examples: (i) fast gradient sign method [16], (ii) Carlini-Wagner attack [7], and (iii) Jacobian-based Saliency Map attack [45].

Fast Gradient Sign Method. The fast gradient sign method (FGSM) attacks DNNs by leveraging their learning process based on gradients [16]. The following formula describes the process of generating an FGSM example:

$$\mathbf{x}' = \mathbf{x} + \epsilon \cdot \operatorname{sgn}(\nabla_{\mathbf{x}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y_s)) \tag{1}$$

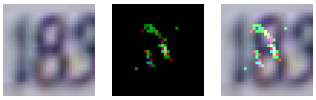
Here $\nabla_{\mathbf{x}} \mathcal{L}$ is the gradient of the loss function with respect to the original input pixel vector $\mathbf{x}$, y_s is the actual or source label for $\mathbf{x}$, and $\boldsymbol{\theta}$ stands for the parameters of the model $f_{\boldsymbol{\theta}}$ that are constant. Gradient w.r.t. $\mathbf{x}$ is easier to calculate with back-propagation than for $\boldsymbol{\theta}$, which allows the fast generation of adversarial examples. FGSM exploits gradient ascent to increase the loss. Subsequently, the sign applies a max-norm constraint on the gradient value, and ϵ represents a small magnitude of the step to increase the loss. This formulation constitutes the untargeted type of adversarial attacks, a particular example depicted in Figure 1.

The FGSM can be converted into a targeted attack by substituting the source label with a target one y_t and doing gradient descent instead of ascent, namely:

$$\mathbf{x}' = \mathbf{x} - \epsilon \cdot \operatorname{sgn}(\nabla_{\mathbf{x}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y_t)) \tag{2}$$



(a) CW applied to the MNIST dataset.



(b) JSMA applied to the SVHN dataset

Fig. 2 (a): *Top row:* MNIST adversarial examples targeted for 0. *Bottom row:* MNIST adversarial examples targeted for 1. (b): JSMA adversarial features visualized.

However, since FGSM is designed to be fast rather than optimal, it is not necessarily guaranteed to produce the targeted adversarial examples of minimal perturbations.

Carlini-Wagner. The Carlini-Wagner (CW) attack [7] aims at optimality in contrast to FGSM, i.e., it attempts to generate as little noise as possible to succeed in the attack. It poses the following optimization objective:

$$\text{minimize } ||\delta||_p \quad \text{subj. to } f_{\boldsymbol{\theta}}(\mathbf{x}+\delta) = y_t, \quad \mathbf{x}+\delta \in [0,1]^n \tag{3}$$

where, $\mathbf{x} \in [0,1]^n$ represents an image, $\delta \in [0,1]^n$ is the added noise to the image, and $f_{\boldsymbol{\theta}}$ is a model that returns a target class label of the image under attack. The noise level is calculated in terms of L_p norms. The authors consider several norms; our example demonstrates the L_2-norm. The CW attack represents a targeted attack with powerful properties. Till the moment, it is one of the strongest known adversarial attacks. The examples of the CW-targeted attack on MNIST digits can be observed in Figure 2.

Jacobian-based Saliency Map Attack. The Jacobian-based Saliency Map Attack (JSMA) [45] leverages the saliency maps to devise an adversarial input. Namely, it computes the forward derivative of the whole DNN (Jacobian) w.r.t. the input, and based on this derivative, it constructs the saliency map. Large absolute values of the saliency map reveal the features significantly impacting the final output. The JSMA takes the maximum absolute value, perturbs it by a hyperparameter θ, and repeats the process. The stopping criteria are either a successful attack with misclassification or reaching the total perturbation threshold of Υ. Figure 2 depicts such a JSMA attack.

2.3 Causative Attacks on Training

This type of attacks implies that the attacker can manipulate some part of the dataset to be subsequently used for training the DNN model under attack. Since this manipulation is always intended as malicious, the term *poisonous* describes this fact rather appropriately and precisely. Generally speaking, there are two possible malicious intents: they aim at intrusion without corrupting the system behavior with

respect to benign inputs or target an overall regular system operation, causing a denial of service. First, we consider the way the training data is modified.

2.3.1 Poisoning Dataset

Construction of a poisonous dataset includes collecting raw samples of the statistical population distribution of interest with their subsequent labeling. Depending on the attackers' capabilities, they may be able to modify raw samples or only their corresponding labels (or both). The first attack is called a clean-label attack, and the second is a dirty or corrupted-label attack. The clean-label attack may seem impossible at first since there is no control over the output category, and it is unclear how the attacker can exploit the training process in such a case. Nevertheless, two known attacking techniques allow us to achieve this goal. Both introduce slight disturbances to the inputs that mislead DNNs during training. These disturbances are identified through the solution of either a bi-level optimization problem or a feature-collision problem.

2.3.2 Data Poisoning to Disrupt Overall Performance

This type incorporates several indiscriminate attacks, i.e., attacks that do not target a specific category or class within the training dataset. First, we describe a corrupted-label attack based on label flipping. After that, we consider a clean-label attack based on bi-level optimization.

Label Flipping Attack. The attacker can significantly reduce the resulting DNN performance only via tampering with the labels of the dataset. It means that it is optional to have access to the model and the raw data. This type of attack can be implemented by mislabelling the data. Labels can be assigned randomly, influencing the system's overall performance; the specific classification category or even several categories can be targeted. The classifier is subsequently trained on the tampered dataset without knowing which labels have been corrupted. The success rate of this type of attack heavily depends on the ratio of the poisoned dataset: the greater the ratio in favor of poisonous examples, the stronger the impact it would have on misclassification.

Attacks via Bi-level Optimization. This type aims to solve the corresponding bi-level optimization problem and represents clean-label attacks. Consider the following scenario: attackers want to attack a particular DNN model that will be trained, validated, and tested on the dataset $\mathcal{D}$. They want to poison this dataset to change the resulting model behavior. For that reason, the attackers split this dataset into training and validating subsets $\mathcal{D} = \mathcal{D}_{\text{tr}} \cup \mathcal{D}_{\text{val}}$. Subsequently, they aim at poisoning the training dataset $\mathcal{D}_{\text{tr}}$ with a specially crafted input $\mathbf{x}_p$. The crafting of the poisonous input is constrained within the space of allowed manipulations imposed by Φ, i.e., $\mathbf{x}'_p \in \Phi(\mathbf{x}_p)$. As a result, they get a poisoned trained dataset $\mathcal{D}_p = \mathcal{D}_{\text{tr}} \cup \{\mathbf{x}'_p\}$. The

goal is to optimize for the optimal poisonous solution allowed within the imposed constraints such that the target input $\mathbf{x}_t$ with the original label y_t from the untampered validation dataset $(\mathbf{x}_t, y_t) \in \mathcal{D}_{\text{val}}$ is misclassified by the model trained on the poisoned dataset.

To achieve this goal, the optimization procedure should be run across two levels. First, the model should be optimized with respect to the DNN parameters on the dataset with the injected poisonous examples, which represents a classical optimization objective of any DNN training, usually referred to as a training loss $\mathcal{L}_{\text{tr}}$. Second, on top of the classical optimization objective, there is an additional adversarial loss $\mathcal{L}_{\text{adv}}$ that should be optimized for the best poisonous input modification in such a way that this adversarial loss is maximized which eventually leads to the sought-after misclassification. Note that the misclassification is untargeted in this particular case. Hence, this formulation of the optimization objectives represents an attack on the availability of the model aiming at disrupting the classification results for the poisonous inputs.

Formally, both of the optimization levels can be formulated as follows:

$$\mathbf{x}_p^* \in \underset{\mathbf{x}'_p \in \Phi(\mathbf{x}_p)}{\arg\max}\ \mathcal{L}_{\text{adv}}(\mathcal{D}_{\text{val}}, \boldsymbol{\theta}^*(\mathbf{x}'_p)) \tag{4}$$

$$\text{s.t.} \qquad \boldsymbol{\theta}^* \in \underset{\boldsymbol{\theta}}{\arg\min}\, \mathbb{E}_{(\mathbf{x}'_p, y) \sim \mathcal{D}_p}\left[\mathcal{L}_{\text{tr}}(f_{\boldsymbol{\theta}}(\mathbf{x}'_p), y))\right] \tag{5}$$

where $\mathbf{x}'_p \in \mathcal{D}_p$, i.e., it is an input from the poisoned training dataset under constrained manipulations allowed by Φ, and $\mathbf{x}_p^*$ is the resulting poisonous input to be added in the original clean dataset.

The process of optimization of this bi-level problem seems straightforward at first glance; the inner minimization procedure obtains the optimal parameters of the DNN with the subsequent maximization of the adversarial loss. This process can be repeated until the convergence. However, in the case of DNNs, this solution is not feasible due to the over-parameterization and impossibility of the exact solution of the inner problem. For that reason, the truncated back-gradient optimization is used [40]. The idea is to first optimize the inner problem for a limited number of iterations, which allows computing the necessary gradients with subsequent backpropagation for the outer objective by tracing the necessary gradients backward. This approach allows the generating of poisonous inputs for DNNs successfully.

2.3.3 Targeted Poisoning Attacks

We consider one corrupted-label targeted attack based on a bi-level optimization objective and one clean-label targeted attack utilizing feature collision.

Attacks via Bi-level Optimization. The indiscriminate bi-level optimization poisonous attack described above can be easily changed into a targeted one, if the outer objective is minimized instead, namely:



(a) Patch (b) Semantical (c) Blending (d) Signal (e) Imperceptible

Fig. 3 Different types of backdoor attacks: (a) artificial patch with a corrupted label, (b) benign semantical trigger with a corrupted label, (c) blending a fake reflection with a clean label, (d) introducing a strong signal with a clean label, and (e) backdoor imperceptible to the human eye with a clean label.

$$\mathbf{x}_p^* \in \underset{\mathbf{x}'_p \in \Phi(\mathbf{x}_p)}{\arg\min}\ \mathcal{L}_{\text{adv}}(\mathcal{D}'_{\text{val}}, \boldsymbol{\theta}^*(\mathbf{x}'_p)) \tag{6}$$

The dataset for validation contains the same raw inputs as $\mathcal{D}_{\text{val}}$ but with the modified labels that the attacker wants to target, hence minimizing the loss towards them.

Attacks on Feature Collision. The intuitive idea behind this type of attacks is to look for the neighbors in the feature space within a relatively close distance from the target class that simultaneously lie close in the input space to the base class, i.e., to the class under attack:

$$\mathbf{x}_p^* \in \underset{\mathbf{x}}{\arg\min}\ \ \|f_{\boldsymbol{\theta}}(\mathbf{x}) - f_{\boldsymbol{\theta}}(\mathbf{t})\|_2^2 + \beta\,\|\mathbf{x} - \mathbf{b}\|_2^2 \tag{7}$$

Given the feature space's usual complexity and high dimensionality, such a search appears feasible. Note that due to the closeness to the target class in the feature space, there is no need to modify the label, making this attack clean-label. Moreover, the closeness of the poisonous sample to the base one in the input space does not raise concerns during the visual inspection since the label is correctly assigned as if it represents a base class. Both of these factors make this attack very powerful.

2.3.4 Data Poisoning for Intrusion

In this category, the attacker introduces a set of poisonous inputs that contains a specifically crafted trigger that serves as a backdoor, allowing the intruder to modify the behavior of the DNN in the desired direction [9].

Backdoor Patches. The first attack from this category exploits virtual or physical visible patches [17]. The training dataset is poisoned with the patched instances. Subsequently, the DNN is trained with the *corrupted-labels* for the samples that contain the corresponding patches. These patches serve as a trigger for the DNN

classifier, allowing the attacker to change the classification result of the model. Since the training dataset contains benign and patched inputs, the normal behavior of the model with benign inputs and classification test errors are not influenced by these patched samples. Examples may represent a patch on the traffic stop sign to misclassify it as a traffic sign for the speed limit (see Figure 3(a)). Please note that such a patch can be imposed with the original image during the construction of the poisonous input without needing a physical photo with the patch. However, sometimes it is reasonable to rely on some physical trigger that carries a distinct semantical meaning but looks benign in contrast to the patching, for example, when a person wears glasses (see Figure 3(b)).

Backdoor as a Strong Signal. Several functional approaches could be applied to implement a *clean-labels* backdoor. First, it is possible to bind a strong signal function that any DNN can easily detect with a particular target class, e.g., a sinusoidal signal with a ramp can be mixed into the small fraction of the target images with subsequent use of this signal as a backdoor for any other image during the inference stage. The sufficient ratio of poisonous examples before training the model for the successful attack constitutes one-third of the target class for the MNIST dataset and one-fifth of the traffic signs dataset [2] (see Figure 3(d)). The problem with the sinusoidal ramp is that it is still quite visible in the resulting image, which can be detected via visual inspection of the dataset.

Another type of signal can be stealthily encapsulated into the target image representing a fake reflection [32]. The benefits of this approach are that it cannot be easily detected as the previous signal and taking into consideration that it is also a clean-label poisonous attack then it provides all the benefits of a nice backdoor in cases when the target images have reflecting surfaces (see Figure 3(c)).

Imperceptible Backdoor. Previous backdooring techniques still require a visible trigger and a relatively high ratio of poisonous examples to be added to the training set. The ultimate backdoor, however, can be forged utilizing the technique that we have already described above, namely feature collision [55]. Such an approach allows the injection of a backdoor even by poisoning one image only. Moreover, thanks to the optimizing within the deep feature representation while minimizing the difference in the input domain, the resulting poisoned image contains imperceptible perturbances to the human eye, making this backdoor one of the most potent poisoning attacks (see Figure 3(e)).

2.4 Transferability

It has been discovered that different architectures of DNNs trained to tackle the same classification problem on similar datasets tend to have similar fairly piece-wise linear decision boundaries that separate categories in the input data domain [16]. This property is called transferability. Transferability is especially dangerous since

it allows devising either an adversarial example or poisonous input that universally targets all DNNs with a similar final objective in a black-box manner [46].

The thorough study of the transferability properties indicates the difference between adversarial and poisonous samples. In particular, the adversarial examples tend to transfer better when forged on a simplified surrogate model. However, for poisonous attacks, the best surrogate models are the ones that match the complexity of the target [11]. Moreover, in the case of the white-box threat model, both adversarial and poisonous inputs favor the more complex models, i.e., the success rate of the attacks is higher for a more complex model than for a simpler one.

2.5 Privacy-oriented Attacks

These attacks aim at stealing any private data. The first type of privacy-oriented attacks involves stealing the pre-trained model's proprietary parameters, including weights and hyperparameters. The model in this scenario is usually queried remotely via the web. The second type relates to identifying if a particular input was in the training dataset, allowing the attacker to deduce the data on which the model under attack was trained. In this section, we describe both of these types.

2.5.1 Model Stealing

Nowadays, the functionality provided by various DNNs is often offered online as a service. Such practice amplifies the significance of securing the privacy-related information related to the models deployed in the cloud since its leakage may induce confidentiality-related infringements and substantial financial costs.

Attacks that aim at model stealing attempt to extract this information somehow and, eventually, to reconstruct the existing target model. It may concern the reconstruction of the entire model along with its weights, the reconstruction of the hyperparameters, or the reconstruction of a functionally equivalent alternative model. The adversary gathers the data about the target model by sending a data sample and receiving the model's prediction. This interaction is termed *query-based*.

Equation-solving Attacks. Equation-Solving Attack (ESA) is based on formulating and solving a system of equations, the solution of which yields desired values for an adversary. It aims at specific values of the target model, e.g., learned parameters or training hyperparameters. Using model outputs $y_1, ..., y_n$ for given data samples $\mathbf{x_1}, ..., \mathbf{x_n}$, it's possible to construct equations $f_{\boldsymbol{\theta}}(\mathbf{x_i}) = y_i, i = 1, ..., n$, revealing parameter values $\boldsymbol{\theta}$ [57]. ESA is quite efficient: depending on the target model type, 1 to 4 queries per parameter is enough. This attack requires the knowledge of the target model's architecture and the data samples to query the model. For black-box access, training hyperparameters need prior architecture and parameter extraction attacks.

Meta-model Training Attacks. The meta-model attack (MMA) is the only query-based attack capable of uncovering target model architecture to date. A meta-model is trained using a set of candidate CNNs with varying architecture, optimization, and data parameters as the meta-model's dataset, predicting a model's structure, training setup, and data volume. The meta-model then links model hyperparameters and performance through specific test samples, revealing target model hyperparameters [42].

MMA's success depends on the hyperparameter's influence and their presence in the training set. For instance, to steal the convolutional layer count, the target model must be a convolutional neural network and possess a corresponding number of layers in the training set for the meta-model. Execution demands considerable computational resources: for MNIST classifiers, 10,000 CNN candidates had to be trained for over 40 days on a GPU. On average, the attack correctly predicted hyperparameters 80.1% of the time, surpassing a 34.9% guessing chance. However, as MMA extracts hyperparameters, an extra parameter-stealing attack is necessary to approximate the whole target model behavior [43].

2.5.2 Membership Inference

Membership inference involves the identification of training data associated with a trained model. When provided with a data instance and granted access to a model, the objective is to detect whether this data instance was a part of the model's training dataset. The access to the model can vary, falling into either the white-box or black-box category.

Membership inference attacks stem from the fact that a machine learning model might behave differently on the training dataset than the test dataset. It is particularly evident in machine learning, especially within DNNs, which often are over-parameterized when the number of trainable parameters exceeds the number of training instances. It enables a machine learning model to "remember" instances from the training data. Thus, it may assign significantly higher confidence to predictions for training instances than test instances. By exploiting such differences, attackers can deduce whether a given instance is likely to belong to the training dataset.

2.6 CIA triad

The classical information security triad comprises three main components: Confidentiality, Integrity, and Availability (CIA). We can now look at the described attacks through the CIA's prism. The functionality-oriented attacks relate to both Integrity and Availability. The integrity component involves the attacks that maximize the model Type-I errors, i.e., adversarial examples and backdoor attacks. Availability attacks aim at minimizing the utility of the DNN model, e.g., by increasing the

Table 1 Summary of the attacks and links to the CIA triad. Enhanced from Biggio and Roli [5]

.

Attacker's Capability	Attacker's Goal		
	Functionality-oriented		Privacy-oriented
	Integrity	Availability	Confidentiality
Train data	Poisoning for subsequent intrusion	Poisoning to invalidate model	—
Test data	Adversarial examples	Outliers	Model stealing, membership inference

model Type-II errors via poisonous attacks or by flooding the model with outliers, exploiting model overconfidence, and making the model predictions irrelevant in most cases. The privacy-oriented attacks, however, are connected with the Confidentiality component. It includes both model stealing and membership inference. The summary of all of these aspects, including the attacker's capabilities, can be observed in Table 1.

3 Available Defenses

This section covers several available defense strategies against the attacks mentioned above.

3.1 Defense Against Functionality-oriented Attacks

As for the available defenses against functionality-oriented attacks, it should be stressed that the current state-of-the-art represents a constant race between the invention of new defenses against known attacks and the subsequent breaking of this defense with the newly discovered vulnerabilities. Nevertheless, in this section, we introduce the defense methods that demonstrate promising robustness results, and many of them have already passed the test over time.

3.1.1 Adversarial Training

Currently, there is no readily available universal defense mechanism that provides complete protection against adversarial examples. Nevertheless, several techniques proved promising in mitigating the potential damage and consequences, one of them being adversarial training [56, 16]. The transferability not only allows a black-box adversarial attack but also means that it is possible to generate the known adversarial examples in advance automatically and add them to the training set before starting the training. Such an approach forces the DNN model to consider the adversarial perturbations and increases the robustness of DNNs to adversarial attacks.

The advantage of this method is its simplicity. The most significant disadvantage is the requirement to generate adversarial examples in advance, which may defend only against attacks known during the DNN model training. Since adversarial examples can be generated with a vast diversity of different techniques, this approach can be helpful only if the model retraining and redeployment procedures are relatively cheap. Thus, keeping the model up-to-date with developing new attacks is possible.

3.1.2 Enhanced Optimization Objective for Adversarial Training

The idea of this approach is first to enhance the standard optimization objective and then exploit one of the adversarial attacking techniques for approximating one of the stages of the optimizations. Namely, recall the standard loss objective for the classification task:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y))] \tag{8}$$

where $\mathbf{x}$ is the input, y is the corresponding label, $\mathcal{D}$ is the training dataset, $\boldsymbol{\theta}^*$ are the optimal trained parameters of the DNN.

This objective may be improved so that its exact solution would *guarantee* the robustness against the adversarial examples [38]. Specifically, it can be achieved by requiring that every neighbor of the current point within the ϵ-ball adheres to the same class:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\max_{\|\mathbf{x}'-\mathbf{x}\|_p \leq \epsilon} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}'), y))\right] \tag{9}$$

Unfortunately, the exact solution for optimizing this min-max objective is unattainable in a reasonable amount of time since it involves a standard process of first a non-concave inner maximization and then a non-convex outer minimization. The solution is to approximate the worst-case inner approximation problem by optimizing for adversarial examples. In particular, FGSM (see 2.2.3) can be interpreted as a single step for maximizing the inner part of this objective formulation. In order to get a more precise approximation to the solution, one can also apply a multi-step attack such as projected gradient descent (PGD) [7].

3.1.3 Outlier Exposure

Similar to the idea of adversarial training is the outlier exposure approach [20], introduced for DNNs to deal with the outliers. The idea is to enhance the training dataset of the inliers $\mathcal{D}_{\text{in}}$ with the outliers. Since the dataset with the real outliers $\mathcal{D}_{\text{out}}$ is not available, it is though possible to enhance the dataset with the different available datasets for Outlier Exposure $\mathcal{D}_{\text{out}}^{\text{OE}}$. This dataset is different from the inliers. The model f is subsequently trained to learn more conservative signals of the inliers versus provided outliers that enhance robustness against previously

unobserved outliers. It is achieved by introducing an additional term to the loss function, which is responsible for the outlier detector:

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{\text{in}}}[\mathcal{L}(f(\mathbf{x}),y)+\lambda\mathbb{E}_{\mathbf{x}'\sim\mathcal{D}_{\text{out}}^{\text{OE}}}[\mathcal{L}_{\text{OE}}(f(\mathbf{x}'),y)]] \tag{10}$$

The first term of the loss is the standard cross-entropy loss $\mathcal{L}$ used in the classification tasks where $\mathbf{x}$ represents the input, and y stands for the corresponding label. The second term includes the outlier exposure loss $\mathcal{L}_{\text{OE}}$ for the outlier detector. It represents the cross-entropy from $f(\mathbf{x}')$ to the uniform distribution. Unlike the adversarial training approaches described in the previous sections, this method allows us to mitigate and detect the outlier.

3.1.4 Defense via Generative Models

Several defenses based on the Deep Generative Models (DGMs) do not imply any knowledge about the type of adversarial attack in use. Moreover, they can be applied independently of the classifier DNN since they are used as a filter working with the input before passing it further to the DNN model under protection. In addition, some DGMs allow the detection of outliers in a completely unsupervised manner that can be combined with the adversarial filtering approach, enhancing the robustness against outliers and adversarial examples.

DGMs as Adversarial Filters. This approach exploits the DGMs' ability to learn a joint distribution of the data that results in a different learned representation compared to the discriminative approach, e.g., in the case of a DNN classifier. The most obvious difference with the discriminative representation is that the generative representation allows the generation of new samples from the learned joint distribution that look similar to those observed in the dataset during the training stage. Based on this representative power, it becomes possible to train a separate DGM model that will work as a filter, i.e., any input is first fed to the DGM filter, which mitigates the ongoing or attempted attack [52, 22]. The mitigation requires normal DGM training on the same dataset as the DNN under protection. As a result, it allows projecting the adversarial input onto the range of the DGMs' generator based on the learned representation by minimizing the reconstruction error. Hence, by default, this method includes filtering out the adversarial perturbations. Finally, the protected DNN classifier will get the clean input without adversarial perturbations.

DGMs as Unsupervised Outlier Detectors. The methods within this category allow the detection of outliers completely unsupervised. They can rely either on the ensemble-based epistemic uncertainty estimation [14] or the DGM latent representation [15]. In both cases, the DGM can be again used as a filter to the DNN under protection. However, this time, it is not only mitigating the attack but also detecting if the current input falls under the category of outlier or not. Moreover, it turns out that the outliers tend to gravitate toward the holes in the latent representation of DGMs (see Figure 4), specifically in the Variational Autoencoders (VAEs), allowing its simple detection. Finally, the same DGM that is used for the outlier detection can

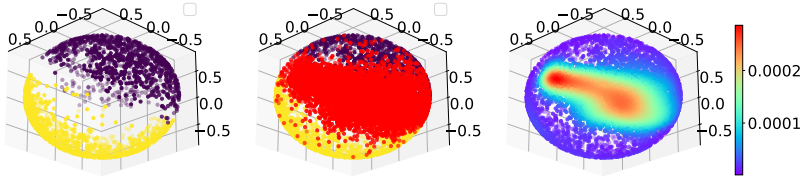


Fig. 4 Compact spherical latent space of VAE trained only on two digits of the MNIST dataset: 0's and 1's. It is a vivid demonstration of the fact that the outliers densely land on the hole in the latent representation. *From left to right*: Yellow depicts means of the estimated posteriors for 1's and purple for 0's; Red represents the mapped means for a held-out outlier: a class of digits 9's; Kernel density estimation of all means with the densest region in the hole packed with the outliers.

be applied in the same logic as the previously described DGMs for the adversarial mitigation, i.e., if the outlier is not detected, then the input is purified from the adversarial perturbations and subsequently fed to the DNN under protection. This approach is auspicious since it targets adversarial examples and outliers utilizing only one filtering model.

3.1.5 Defense Against Poisonous Attacks

Defenses against poisonous attacks can be broadly categorized into one of the following groups:

1. *Data Level Defenses:* Focus on removing poisonous data from the input.
2. *Model Level:* Involve model retraining.
3. *Interaction between Data and Model Levels:* Identify patterns in the interaction between the data and model training.

Data Level Defenses. The first approach to the defense in this category is treating the poisonous samples in the tampered dataset as outliers. It implies that the defenders can access the benign dataset to train an outlier detection model. For example, they can train another model on the benign dataset with a different architecture from the original one but with comparable accuracy metrics [29]. Subsequently, any new input can be forwarded through both models, and if there is disagreement in the predictions between them, then this input is marked as potentially poisonous.

Another approach is based on the intentional input perturbation when the defender modifies the input by blending it with another benign input. It has been noticed that the poisonous backdoor inputs are still successfully classified as the target of the backdoor, whereas the blending of two benign inputs produces a random prediction result [13]. This difference is most likely due to the special optimization procedure used when forging a backdoor input, in which the objective is set so that it should overcome the original trigger "blending", resulting in a successful attack even when additional blending is applied.

Model Level Defenses. The defenses of this type imply that the defender has access at least to the smaller subset of the original benign dataset. In such a case, the defender can run a training procedure based on fine-tuning the poisoned model utilizing this benign subset. The benign data can be simultaneously augmented with the samples by adding random Gaussian noise to make the defended model even more robust to potential poisonous input perturbations. After several iterations of such fine-tuning, the model becomes robust and immune to the previously poisonous inputs [33, 59].

An alternative approach within this type of defense is based on meta-classification when an additional DNN is trained on the features extracted from the defended DNN to classify if this model has been compromised by poisoning. The defender has to construct a set of DNNs evenly divided into poisoned and benign ones. After that, the meta classifier is trained based on the features extracted from these DNNs with a single purpose to distinguish between the benign and tampered models. This approach demonstrated good generalization results even with the before unseen poisonous techniques [63, 26].

Defenses based on the Interaction between Data and Model Training. This type of defense is relatively novel and looks like the most robust and promising one. It does not imply that the defender has access to the benign dataset for outlier detection or a benign subset of the initial dataset for subsequent model retraining using fine-tuning. It does not rely on the meta-classification either, implying that the defender has access to benign and poisonous models. Instead, this defense is working its way out utilizing the provided dataset and the given training objective interchangeably, resulting in the clustering of the samples based on the incompatibility property [24]. The intuition is that the benign inputs should improve the optimization objective during the training or at least not degrade it; however, the poisonous inputs, on the contrary, should reduce the validation accuracy. This idea leads to the clustering method that iteratively builds up clusters based on the incompatibility with respect to the training objective, i.e., benign inputs will eventually generalize to themselves as opposed to the poisonous inputs that would be clustered separately in such cases.

3.2 Defense Against Privacy-oriented Attacks

In this section, we cover both mitigation and detection defensive methods against privacy-oriented attacks. The mitigation methods focus on minimizing the attack's impact, i.e., they do not stop the attacker from acquiring a model; their goal is to reduce the stolen model's quality to a point where it becomes unusable. The detection methods can further be subdivided into ownership verification (usually achieved by unique model identifiers or watermarking that can prove ownership of a stolen model; aims at proving past attacks) and attack detection (monitors whether a model is currently being attacked). Attack detection cannot prevent a model from being stolen, but it can inform the owner about the incident.

3.2.1 Basic Defenses

We begin with a listing of several simple basic defenses that can be applied to mitigate attacks that aim at stealing the DNN model. These defenses demonstrate good protection results despite their simplicity.

Input Modification Defenses. The first type involves input modification, i.e., a defender can modify the input provided to the DNN so that only insignificant parts of this input are modified. For example, in the case of CNNs and image classification tasks, these parts would represent insignificant pixels for the resulting classification. If the defender adds random noise to these pixels, it will make it very complicated for the attacker to steal the DNN's parameters [18]. The technique that identifies such insignificant pixels in the first place is based on the Gradient-based CAM [54] that was initially developed for the visual interpretability of the decisions made by CNN in the input feature space.

Ouput Modification Defenses. The second type implies output modification. It is based on rounding of the predicted values [57]. This rounding prevents the attacker from stealing the model's parameters due to the impossibility of solving the corresponding system of equations. Another alternative is to utilize the so-called adaptive misinformation [25]. It is a special technique to return wrong predictions for user queries but in an adaptive way. The intuition behind this approach is to notice that model attackers quite frequently use queries that lie out-of-the-distribution. Based on this observation, it is logical to adapt a model so that it will assign wrong predictions to such queries, making it much more complicated for the attacker to steal the model.

Model Modification Defenses. In addition to the perturbation of inputs and outputs, modifying the model architecture and parameters is possible. A CNN feature extractor can be simulated to train a simpler model, namely a shallow and sequential convolutional block, via Knowledge Distillation (KD) [62]. Such an approach is akin to the source code obfuscation technique obstructing the attacker from stealing the initial proprietary model parameters and architecture. Moreover, it is possible to choose an opposite way by adding redundant layers that do not change the functionality, which makes theft of the model much more complicated [8].

3.2.2 Unique Identifiers and Watermarking

This section addresses several ownership verification defensive techniques.

Unique Model Identifier. Unique Model Identifier (UMI) is a detection method that identifies a distinctive model property that transfers to a substitute model during theft. By revealing this property, a model owner can prove the model was stolen. This technique does not require an active embedding of this unique, distinctive property in contrast to watermarking, as it is inherent to the model. One example of this technique is a Dataset Inference (DI) defense. DI detects if a model was trained on a specific dataset [39]. This method is based on the idea that training

samples lie farther from the decision boundary than other samples. A subset of the original training data measures the distance of samples from the boundary in the substitute model. If they are distant, the substitute model contains the target model's identifier, indicating a potential model theft. However, DI is ineffective when the original dataset is public, misclassifying independent models trained on it as stolen [31]. Alternatively, conferrable adversarial examples can form unique fingerprints for substitute models [36]. These samples represent a unique set of adversarial examples that transfer to the substitute models but not to the other independently trained models.

Model Watermarking. Another approach is based on the active intervention in the model training process so that the owner changes the model behavior on some or all inputs so that only the owner can identify these changes. Model Watermarking (WM) actively embeds concealed data to establish ownership. Unlike UMIs, it involves secret backdoors that make models predict predefined values for some data containing outliers, revealing watermarks. However, embedded watermarks also have to be persistent to model stealing, i.e., they have to be identifiable in the model after it has been stolen. This property can be achieved by training a model that extracts common features from problem-domain inputs and watermarking samples. Such an approach ensures that watermarks would also be extracted during the model theft [23].

3.2.3 Monitor-based Defenses

Monitor-based detection defense involves query analysis to identify malicious users. For example, a defender can train a dedicated discriminative DNN to classify adversarial versus benign samples, using each hidden layer outputs of a protected DNN as features [68]. Alternatively, a DGM such as VAE can also differentiate benign from malicious queries [44].

4 Conclusion

In this chapter, we have covered the broad range of attacks on discriminative DNNs. We have touched upon all potentially vulnerable spots, including the influence on the model's behavior and the potential confidential data leakage. In addition, we detailed the most prominent examples of adversarial and poisonous attacks, including the potential adversarial usage of the outliers and various backdoor techniques. Furthermore, we delved into two powerful privacy-related attacks: model stealing and membership inference. Lastly, we highlighted the most promising and robust defense mechanisms that are currently available in the arsenal of the DNN developer to either mitigate or detect the undergoing attack.

References

[1] Kartik Ahuja, Ethan Caballero, Dinghuai Zhang, Jean-Christophe Gagnon-Audet, Yoshua Bengio, Ioannis Mitliagkas, and Irina Rish. Invariance Principle Meets Information Bottleneck for Out-of-Distribution Generalization. In *Advances in Neural Information Processing Systems (NIPS)*, 2021.

[2] M. Barni, K. Kallas, and B. Tondi. New Backdoor Attack in CNNs by Training Set Corruption Without Label Poisoning. In *IEEE International Conference on Image Processing (ICIP)*, 2019.

[3] Andrew R. Barron. Approximation and Estimation Bounds for Artificial Neural Networks. *Machine Learning*, 14(1):115–133, 1994.

[4] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[5] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84, 2018.

[6] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks Against Machine Learning at Test Time. In *Machine Learning and Knowledge Discovery in Databases*, 2013.

[7] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks, 2016.

[8] Hervé Chabanne, Vincent Despiegel, and Linda Guiga. A Protection against the Extraction of Neural Network Models, 2020.

[9] Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Sebastiano Vascon, Werner Zellinger, Bernhard A. Moser, Alina Oprea, Battista Biggio, Marcello Pelillo, and Fabio Roli. Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Comput. Surv.*, 55(13s), jul 2023. ISSN 0360-0300. doi: 10.1145/3585385.

[10] G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Math. Control Signals Syst.*, 2(4):303–314, 1989.

[11] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. In *USENIX Security Symposium*, 2019.

[12] Gintare Karolina Dziugaite. *Revisiting Generalization for Deep Learning: PAC-Bayes, Flat Minima, and Generative Models*. PhD thesis, University of Cambridge, 2020.

[13] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Annual Computer Security Applications Conference (ACSAC)*, 2019.

[14] Misha Glazunov and Apostolis Zarras. Do Bayesian Variational Autoencoders Know What They Don't Know? In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2022.

[15] Misha Glazunov and Apostolis Zarras. Vacant Holes for Unsupervised Detection of the Outliers in Compact Latent Representation. In *Uncertainty in Artificial Intelligence (UAI)*, 2023.

[16] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples, 2014.

[17] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*, 7: 47230–47244, 2019.

[18] L. Guiga and A. W. Roscoe. Neural network security: Hiding CNN parameters with guided grad-Cam. In *International Conference on Information Systems Security and Privacy (ICISSP)*, 2020.

[19] D. M. Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.

[20] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep Anomaly Detection With Outlier Exposure, 2018.

[21] K. Hornik. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Netw.*, 4(2):251–257, 1991.

[22] Uiwon Hwang, Jaewoo Park, Hyemi Jang, Sungroh Yoon, and Nam Ik Cho. PuVAE: A Variational Autoencoder to Purify Adversarial Examples. *IEEE Access*, 7:126582–126593, 2019.

[23] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled Watermarks as a Defense against Model Extraction. In *USENIX Security Symposium*, 2021.

[24] Charles Jin, Melinda Sun, and Martin Rinard. Incompatibility Clustering as a Defense Against Backdoor Poisoning Attacks. In *International Conference on Learning Representations (ICLR)*, 2023.

[25] Sanjay Kariyappa and Moinuddin K. Qureshi. Defending Against Model Stealing Attacks With Adaptive Misinformation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal Litmus Patterns: Revealing Backdoor Attacks in CNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification With Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[28] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-Distribution Generalization via Risk Extrapolation (REx). In *International Conference on Machine Learning (ICML)*, 2021.

[29] Hyun Kwon. Detecting Backdoor Attacks via Class Difference in Deep Neural Networks. *IEEE Access*, 8:191049–191056, 2020.

[30] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[31] Yiming Li, Linghui Zhu, Xiaojun Jia, Yong Jiang, Shu-Tao Xia, and Xiaochun Cao. Defending against Model Stealing via Verifying Embedded External Features. In *AAAI Conference on Artificial Intelligence*, 2022.

[32] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. *arXiv preprint arXiv:2007.02343*, 2020.

[33] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural Trojans. In *IEEE International Conference on Computer Design (ICCD)*, 2017.

[34] Sanae Lotfi, Marc Finzi, Sanyam Kapoor, Andres Potapczynski, Micah Goldblum, and Andrew G Wilson. PAC-Bayes Compression Bounds So Tight That They Can Explain Generalization. In *Advances in Neural Information Processing Systems (NIPS)*, 2022.

[35] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The Expressive Power of Neural Networks: A View From the Width. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[36] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep Neural Network Fingerprinting by Conferrable Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, 2021.

[37] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[38] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[39] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset Inference: Ownership Resolution in Machine Learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[40] Luis Muñoz González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. Towards Poisoning of Deep Learning Algorithms With Back-Gradient Optimization. In *ACM Workshop on Artificial Intelligence and Security*, 2017.

[41] Vaishnavh Nagarajan and J. Zico Kolter. Uniform Convergence May Be Unable to Explain Generalization in Deep Learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[42] Seong Joon Oh, M. Augustin, M. Fritz, and B. Schiele. Towards Reverse-Engineering Black-Box Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[43] Daryna Oliynyk, Rudolf Mayer, and Andreas Rauber. I Know What You Trained Last Summer: A Survey on Stealing Machine Learning Models and Defences. *ACM Computing Surveys*, 55(14s):1–41, 2023.

[44] Soham Pal, Yash Gupta, Aditya Kanade, and Shirish Shevade. Stateful Detection of Model Extraction Attacks, 2021.

[45] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.

[46] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks Against Machine Learning, 2016.

[47] Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey Hinton. Regularizing Neural Networks by Penalizing Confident Output Distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[48] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized Input Sampling for Explanation of Black-Box Models. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[49] A. Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numer.*, 8:143–195, 1999.

[50] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets, 2022.

[51] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[52] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-Gan: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations (ICLR)*, 2018.

[53] Patrick Schwab and Walter Karlen. CXPlain: Causal Explanations for Model Interpretation Under Uncertainty. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[54] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-Cam: Visual Explanations From Deep Networks via Gradient-Based Localization. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[55] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[56] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks, 2013.

[57] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium*, 2016.

[58] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971. doi: 10.1137/1116025.

[59] Akshaj Kumar Veldanda, Kang Liu, Benjamin Tan, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, Brendan Dolan-Gavitt, and Siddharth Garg. NNoculation: Broad spectrum and targeted treatment of backdoored DNNs. *arXiv preprint arXiv:2002.08313*, 2020.

[60] Deng-Bao Wang, Lei Feng, and Min-Ling Zhang. Rethinking Calibration of Deep Neural Networks: Do Not Be Afraid of Overconfidence. In *Advances in Neural Information Processing Systems (NIPS)*, 2021.

[61] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-Mnist: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[62] Hui Xu, Yuxin Su, Zirui Zhao, Yangfan Zhou, Michael R. Lyu, and Irwin King. DeepObfuscation: Securing the Structure of Convolutional Neural Networks via Knowledge Distillation, 2018.

[63] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting AI Trojans Using Meta Neural Analysis. In *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[64] Yuesheng Xu and Haizhang Zhang. Convergence of Deep Convolutional Neural Networks, 2022.

[65] Yuesheng Xu and Haizhang Zhang. Uniform Convergence of Deep Neural Networks With Lipschitz Continuous Activation Functions and Variable Widths, 2023.

[66] Haotian Ye, Chuanlong Xie, Tianle Cai, Ruichen Li, Zhenguo Li, and Liwei Wang. Towards a Theoretical Framework of Out-of-Distribution Generalization. In *Advances in Neural Information Processing Systems (NIPS)*, 2021.

[67] Tsuyoshi Yoneda. Pointwise Convergence Theorem of Gradient Descent in Sparse Deep Neural Network, 2023.

[68] Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. DeepEM: Deep Neural Networks Model Recovery through EM Side-Channel Information Leakage. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2020.

[69] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.