Adversarial Attacks and Defences: A Survey

ANIRBAN CHAKRABORTY*, Indian Institute of Technology, Kharagpur, India
MANAAR ALAM, Indian Institute of Technology, Kharagpur, India
VISHAL DEY, The Ohio State University, Columbus, United States
ANUPAM CHATTOPADHYAY, Nanyang Technological University, Singapore
DEBDEEP MUKHOPADHYAY, Indian Institute of Technology, Kharagpur, India

Deep learning has emerged as a strong and efficient framework that can be applied to a broad spectrum of complex learning problems which were difficult to solve using the traditional machine learning techniques in the past. In the last few years, deep learning has advanced radically in such a way that it can surpass human-level performance on a number of tasks. As a consequence, deep learning is being extensively used in most of the recent day-to-day applications. However, security of deep learning systems are vulnerable to crafted adversarial examples, which may be imperceptible to the human eye, but can lead the model to misclassify the output. In recent times, different types of adversaries based on their threat model leverage these vulnerabilities to compromise a deep learning system where adversaries have high incentives. Hence, it is extremely important to provide robustness to deep learning algorithms against these adversaries. However, there are only a few strong countermeasures which can be used in all types of attack scenarios to design a robust deep learning system. In this paper, we attempt to provide a detailed discussion on different types of adversarial attacks with various threat models and also elaborate the efficiency and challenges of recent countermeasures against them.

CCS Concepts: • **Security and privacy → Software and application security**; • **Computing methodologies** → *Computer vision*; *Machine learning*;

ACM Reference Format:
Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. x. Adversarial Attacks and Defences: A Survey. *ACM Comput. Surv.* x, x, Article x (x), 31 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep learning is a branch of machine learning that enables computational models composed of multiple processing layers with high level of abstraction to learn from experience and perceive the world in terms of hierarchy of concepts. It uses backpropagation algorithm to discover intricate

*Corresponding Author

Authors' addresses: Anirban Chakraborty, Indian Institute of Technology, Kharagpur, Department of Computer Science and Engineering, Kharagpur, West Bengal, 721302, India, anirban.chakraborty@iitkgp.ac.in; Manaar Alam, Indian Institute of Technology, Kharagpur, Department of Computer Science and Engineering, Kharagpur, West Bengal, 721302, India, alam.manaar@iitkgp.ac.in; Vishal Dey, The Ohio State University, Columbus, Department of Computer Science and Engineering, Columbus, Ohio, 43210, United States, dey.78@osu.edu; Anupam Chattopadhyay, Nanyang Technological University, School of Computer Science and Engineering, Singapore, 639798, Singapore, anupam@ntu.edu.sg; Debdeep Mukhopadhyay, Indian Institute of Technology, Kharagpur, Department of Computer Science and Engineering, Kharagpur, West Bengal, 721302, India, debdeep@iitkgp.ac.in.

0360-0300/x/0-ARTx $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

details in large datasets in order to compute the representation of data in each layer from the representation in the previous layer [44]. Deep learning has been found to be remarkable in providing solutions to the problems which were not possible using conventional machine learning techniques. With the evolution of deep neural network models and availability of high performance hardware to train complex models, deep learning made a remarkable progress in the traditional fields of image classification, speech recognition, language translation along with more advanced areas like analysing potential of drug molecules [37], reconstruction of brain circuits [49], analysing particle accelerator data [71] [39], effects of mutations in DNA [32]. Deep learning network, with their unparalleled accuracy, have brought in major revolution in AI based services on the Internet, including cloud computing based AI services from commercial players like Google [3], Alibaba [6] and corresponding platform propositions from Intel [7] and Nvidia [5]. Extensive use of deep learning based applications can be seen in safety and security-critical environments, like, self driving cars, malware detection and drones and robotics. With recent advancements in face-recognition systems, ATMs and mobile phones are using biometric authentication as a security feature; Automatic Speech Recognition (ASR) models and Voice Controllable systems (VCS) made it possible to realise products like Apple Siri [1], Amazon Alexa [2] and Microsoft Cortana [4].

As deep neural networks have found their way from labs to real world, security and integrity of the applications pose great concern. Adversaries can craftily manipulate legitimate inputs, which may be imperceptible to human eye, but can force a trained model to produce incorrect outputs. Szegedy et al. [70] first discovered that well-performing deep neural networks are susceptible to adversarial attacks. Speculative explanations suggested it was due to extreme nonlinearity of deep neural networks, combined with insufficient model averaging and insufficient regularization of the purely supervised learning problem. Carlini et al [54] and Zhang et al [28] independently brought forward the vulnerabilities of automatic speech recognition and voice controllable systems. Attacks on autonomous vehicles have been demonstrated by Kurakin et al [43] where the adversary manipulated traffic signs to confuse the learning model. The paper by Goodfellow et al. [31] provides a detailed analysis with supportive experiments of adversarial training of linear models, while Papernot et al. [57] addressed the aspect of generalization of adversarial examples. Abadi et al. [8] introduced the concept of distributed deep learning as a way to protect the privacy of training data. Recently in 2017, Hitaj et al. [34] exploited the real-time nature of the learning models to train a Generative Adversarial Network and showed that the privacy of the collaborative systems can be jeopardised. Since the findings of Szegedy, a lot of attention has been drawn to the context of adversarial learning and the security of deep neural networks. A number of countermeasures have been proposed in recent years to mitigate the effects of adversarial attacks. Kurakin et al. [43] came up with the idea of using adversarial training to protect the learner by augmenting the training set using both original and perturbed data. Hinton et al. [33] introduced the concept of distillation which was used by Papernot et al. [62] to propose a defensive mechanism against adversarial examples. Samangouei et al. [63] proposed a mechanism to use Generative Adversarial Network as a countermeasure for adversarial perturbations. Although each of these proposed defense mechanisms were found to be efficient against particular classes of attacks, none of them could be used as a one-stop solution for all kinds of attacks. Moreover, implementation of these defense strategies can lead to degradation of performance and efficiency of the concerned model.

1.1 Motivation and Contribution

The importance of Deep learning applications is increasing day-by-day in our daily life. However, these deep learning applications are vulnerable to adversarial attacks. To the best of our knowledge, there has been a little exhaustive survey in the field of adversarial learning covering different types of adversarial attacks and their countermeasures. Akhtar et al. [9] presented a comprehensive

survey on adversarial attacks on deep learning but in a restrictive context of computer vision. There have been a handful of surveys on security evaluation related to particular machine learning applications [13], [12], [23], [17]. Kumar et al. [42] provided a comprehensive survey of prior works by categorizing the attacks under four overlapping classes. The primary motivation of this paper is to summarize recent advances in different types of adversarial attacks with their countermeasures by analyzing various threat models and attack scenarios. We follow a similar approach like prior surveys but without restricting ourselves to specific applications and also in a more elaborate manner with practical examples.

Organization

In this paper, we review recent findings on adversarial attacks and present a detailed understanding of the attack models and methodologies. While our major focus is on attacks and defenses on deep neural networks, we have also presented attack scenarios on Support Vector Machines (SVM) keeping in mind their extensive use in real-world applications. In Section 2, we provide a taxonomy of the related terms and keywords and categorize the threat models. This section also explains adversarial capabilities and illustrates potential attack strategies in training (e.g. poisoning attack) and testing (e.g. evasion attack) phases. We discuss in brief the basic notion of black box and white box attacks with relevant applications and further classify black box attack based on how much information is available to the adversary about the system. Section 3 summarizes exploratory attacks that aim to learn algorithms and models of machine learning systems under attack. Since the attack strategies in evasion and poisoning attacks often overlap, we have combined the work focusing on both of them in Section 4. In Section 5 we discuss some of the current defense strategies and we conclude in Section 6.

2 TAXONOMY OF MACHINE LEARNING AND ADVERSARIAL MODEL

Before discussing in details about the attack models and their countermeasures, in this section we will provide a qualitative taxonomy on different terms and key words related to adversarial attacks and categorize the threat models.

2.1 Keywords and Definitions

In this section, we summarize predominantly used approaches with emphasis on neural networks to solve machine learning problems and their respective application.

- **Support Vector Machines** Support vector machines (SVMs) are supervised learning models capable of constructing a hyperplane or a set of hyperplanes in high-dimensional space, which can be used for classification, regression or outliers detection. In other words, a SVM model is a representation of data as points in space with objective of building a maximum-margin hyperplane and splitting the training examples into classes, while maximizing the distance between the split points.
- **Neural Networks:** Artificial Neural networks (ANNs) inspired by the biological neural networks is based on a collection of perceptrons called *neurons*. Each neuron maps a set of inputs to output using an activation function. The learning governs the weights and activation function so as to be able to correctly determine the output. Weights in a multi-layered feed forward are updated by the back-propagation algorithm. Neuron was first introduced by McCulloch-Pitts, followed by Hebb's learning rule, eventually giving rise to multi-layer feed-forward perceptron and backpropagation algorithm. ANNs deal with supervised (CNN, DNN) and unsupervised network models (self organizing maps) and their learning rules. The neural network models used ubiquitously are discussed below.

(1) **DNN:** While single layer neural net or perceptron is a feature-engineering approach, deep neural network (DNN) enables feature learning using raw data as input. Multiple hidden layers and its interconnections extract the features from unprocessed input and thus enhances the performance by finding latent structures in unlabeled, unstructured data. A typical DNN architecture, graphically depicted in Figure. 1, consists of multiple successive layers (at least 2 hidden layers) of neurons. Each processing layer can be viewed as learning a different, more abstract representation of the original multidimensional input distribution. As a whole, a DNN can be viewed as a highly complex function that is capable of nonlinearly mapping original high-dimensional data points to a lower dimensional space.

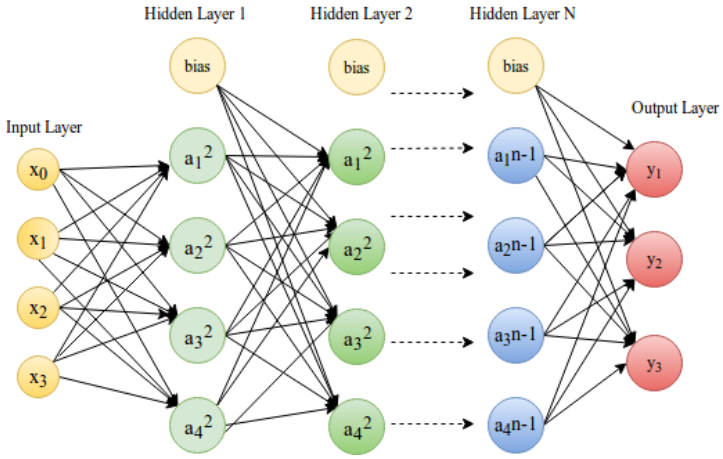


Fig. 1. Deep Neural Network

(2) **CNN:** A Convolutional Neural Network (CNN) consists of one or more convolutional or sub-sampling layers, followed by one or more fully connected layers, to share weights and reduce the number of parameters. The architecture of CNN, shown in Figure. 2, is designed to take advantage of 2D input structure (e.g. input image). Convolution layer creates a feature map; pooling (also called sub-sampling or down-sampling) reduces the dimensionality of each feature map but retains the most important informations to have a model robust to small distortions. For example, to describe a large image, feature values in original matrix can be aggregated at various locations (e.g. max-pooling) to form a matrix of lower dimension. The last fully connected layer use the feature matrix formed from previous layers to classify the data. CNN is mainly used for feature extraction, thus it also finds application in data preprocessing commonly used in image recognition tasks.

2.2 Adversarial Threat Model

The security of any machine learning model is measured concerning the adversarial goals and capabilities. In this section, we taxonomize the threat models in machine learning systems keeping in mind the strength of the adversary. We begin with the identification of threat surface [61] of systems built on machine learning models to identify where and how an adversary may attempt to subvert the system under attack.

2.2.1 The Attack Surface. A system built on Machine Learning can be viewed as a generalized data processing pipeline. A primitive sequence of operations of the system at the testing time can

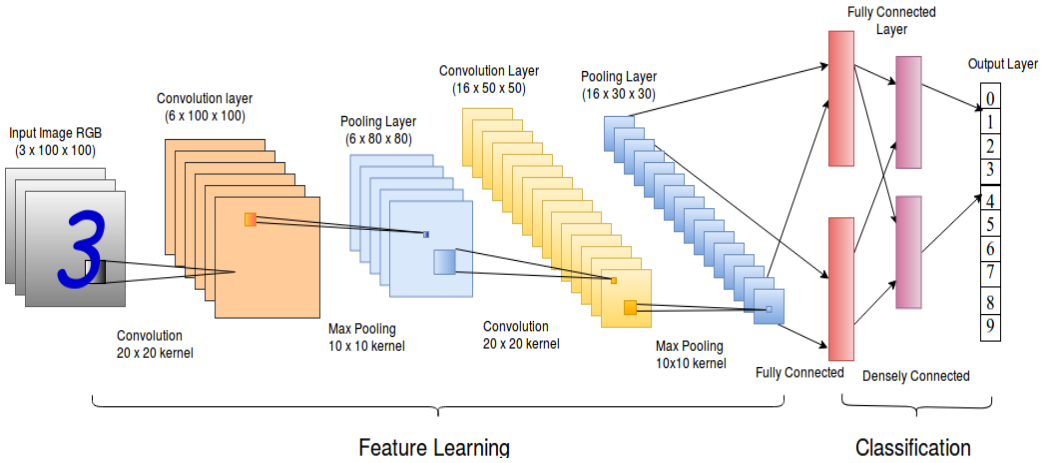


Fig. 2. Convolutional Neural Network for MNIST digit recognition

be viewed as: a) collection of input data from sensors or data repositories, b) transferring the data in the digital domain, c) processing of the transformed data by machine learning model to produce an output, and finally, d) action taken based on the output. For illustration, consider a generic pipeline of an automated vehicle system as shown Figure 3.

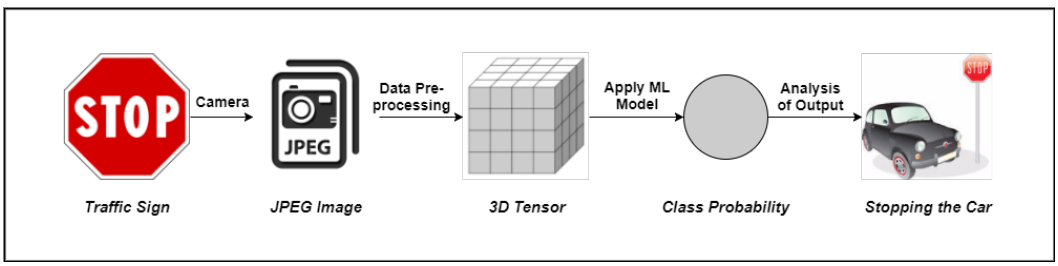


Fig. 3. Generic pipeline of an Automated Vehicle System

The system collects sensor inputs (images using camera) from which model features (tensor of pixel values) are extracted and used within the models. It then interprets the meaning of the output (probability of stop sign), and takes appropriate action (stopping the car). The *attack surface*, in this case, can be defined with respect to the data processing pipeline. An adversary can attempt to manipulate either the *collection* or the *processing* of data to corrupt the target model, thus tampering the original output. The main attack scenarios identified by the attack surface are sketched below [17, 18]:

(1) *Evasion Attack*: This is the most common type of attack in the adversarial setting. The adversary tries to evade the system by adjusting malicious samples during testing phase. This setting does not assume any influence over the training data.
(2) *Poisoning Attack*: This type of attack, known as contamination of the training data, takes place during the training time of the machine learning model. An adversary tries to poison the training data by injecting carefully designed samples to compromise the whole learning process eventually.

(3) *Exploratory Attack*: These attacks do not influence training dataset. Given black box access to the model, they try to gain as much knowledge as possible about the learning algorithm of the underlying system and pattern in training data.

The definition of a threat model depends on the information the adversary has at their disposal. Next, we discuss in details the adversarial capabilities for the threat model.

2.2.2 The Adversarial Capabilities. The term adversarial capabilities refer to the amount of information available to an adversary about the system, which also indicates the attack vector he may use on the threat surface. For illustration, again consider the case of an automated vehicle system as shown in Figure 3 with the attack surface being the testing time (i.e., an Evasion Attack). An internal adversary is one who have access to the model architecture and can use it to distinguish between different images and traffic signs, whereas a weaker adversary is one who have access only to the dump of images fed to the model during testing time. Though both the adversaries are working on the same attack surface, the former adversary is assumed to have much more information and is thus strictly "stronger". We explore the range of adversarial capabilities in machine learning systems as they relate to testing and training phases.

Training Phase Capabilities. Attacks during training time attempt to influence or corrupt the model directly by altering the dataset used for training. The most straightforward and arguably the weakest attack on training phase is by merely accessing a partial or full training data. There are three broad attack strategies for altering the model based on the adversarial capabilities.

(1) ***Data Injection:*** The adversary does not have any access to the training data as well as to the learning algorithm but has ability to augment a new data to the training set. He can corrupt the target model by inserting adversarial samples into the training dataset.
(2) ***Data Modification:*** The adversary does not have access to the learning algorithm but has full access to the training data. He poisons the training data directly by modifying the data before it is used for training the target model.
(3) ***Logic Corruption:*** The adversary has the ability to meddle with the learning algorithm. These attacks are referred as logic corruption. Apparently, it becomes very difficult to design counter strategy against these adversaries who can alter the learning logic, thereby controlling the model itself.

Testing Phase Capabilities. Adversarial attacks at the testing time do not tamper with the targeted model but rather forces it to produce incorrect outputs. The effectiveness of such attacks is determined mainly by the amount of information available to the adversary about the model. Testing phase attacks can be broadly classified into either *White-Box* or *Black-Box* attacks. Before discussing these attacks, we provide a formal definition of a training procedure for a machine learning model.

Let us consider a target machine learning model f is trained over input pair (X, y) from the data distribution μ with a randomized training procedure $train$ having randomness r (e.g., random weight initialization, dropout, etc.). The model parameters θ are learned after the training procedure. More formally, we can write:

$$\theta \leftarrow train(f, X, y, r)$$

Now, let us understand the capabilities of the white-box and black-box adversaries with respect to this definition. An overview of the different threat models have been shown in Figure. 4

White-Box Attacks. In white-box attack on a machine learning model, an adversary has total knowledge about the model (f) used for classification (e.g., type of neural network along with

Article	Black box	White box
Papernot, Nicolas[57]	Non-adaptive	
Rosenberg, Ishai [65]	Adaptive	
Tramèr, Florian [73]	Non-Adaptive	
Papernot, Nicolas [60]	(Non)-Adaptive	
Fredrikson, Matt [26]	Adaptive	
Shokri, Reza [67]	Adaptive	
Hitaj, Briland [34]	Strict	●
Moosavi-Dezfooli [52]		
Tramèr, Florian [72]		●

Fig. 4. Overview of threat models in relevant articles

number of layers). The attacker has information about the algorithm ($train$) used in training (e.g., gradient-descent optimization) and can access the training data distribution (μ). He also knows the parameters (θ) of the fully trained model architecture. The adversary utilizes available information to identify the feature space where the model may be vulnerable, i.e, for which the model has a high error rate. Then the model is exploited by altering an input using adversarial example crafting method, which we discuss later. The access to internal model weights for a white-box attack corresponds to a very strong adversarial attack.

Black-Box Attacks. Black-Box attack, on the contrary, assumes no knowledge about the model and uses information about the settings or past inputs to analyse the vulnerability of the model. For example, in an oracle attack, the adversary exploits a model by providing a series of carefully crafted inputs and observing outputs. Black Box attacks can be further classified into the following categories:

(1) ***Non-Adaptive Black-Box Attack:*** For a target model (f), a non-adaptive black-box adversary only gets access to the target model's training data distribution (μ). The adversary then chooses a procedure $train'$ for a model architecture f' and trains a local model over samples from the data distribution μ to approximate the model learned by the target classifier. The adversary crafts adversarial examples on the local model f' using white-box attack strategies and applies these crafted inputs to the target model to force mis-classifications.

(2) ***Adaptive Black-Box Attack:*** For a target model (f), an adaptive black-box adversary does not have any information regarding the training process but can access the target model as an oracle. This strategy is analogous to chosen-plaintext attack in cryptography. The adversary issues adaptive oracle queries to the target model and labels a carefully selected dataset, i.e., for any arbitrarily chosen x the adversary obtains its label y by querying the target model f. The adversary then chooses a procedure $train'$ and model architecture f' to train a surrogate model over tuples (x, y) obtained from querying the target model. The surrogate model then

produces adversarial samples by following white-box attack technique for forcing the target model to mis-classify malicious data.

(3) ***Strict Black-Box Attack:*** A black-box adversary sometimes may not contain the data distribution μ but has the ability to collect the input-output pairs (x, y) from the target classifier. However, he can not change the inputs to observe the changes in output like an adaptive attack procedure. This strategy is analogous to the known-plaintext attack in cryptography and would most likely to be successful for a large set of input-output pairs.

The point to be remembered in the context of a black-box attack is that an adversary neither tries to learn the randomness r used to train the target model nor the target model's parameters θ. The primary objective of a black-box adversary is to train a local model with the data distribution μ in case of a non-adaptive attack and with carefully selected dataset by querying the target model in case of an adaptive attack. Table 1 shows a brief distinction between black box and white box attacks.

Description	Black box attack	White box attack
Adversary Knowledge	Restricted knowledge from being able to only observe the networks output on some probed inputs.	Detailed knowledge of the network architecture and the parameters resulting from training.
Attack Strategy	Based on a greedy local search generating an implicit approximation to the actual gradient w.r.t the current output by observing changes in input.	Based on the gradient of the network loss function w.r.t to the input.

Table 1. Distinction between black box and white box attacks

The adversarial threat model also depends not only depends on the adversarial capabilities but also on the action taken by the adversary. In the next subsection, we discuss the goal of an adversary while compromising the security of any machine learning system.

2.2.3 Adversarial Goals. An adversary attempts to provide an input x_* to a classification system that results in an incorrect output classification. The objective of the adversary is inferred from the incorrectness of the model. Based on the impact on the classifier output integrity the adversarial goals can be broadly classified as follows:

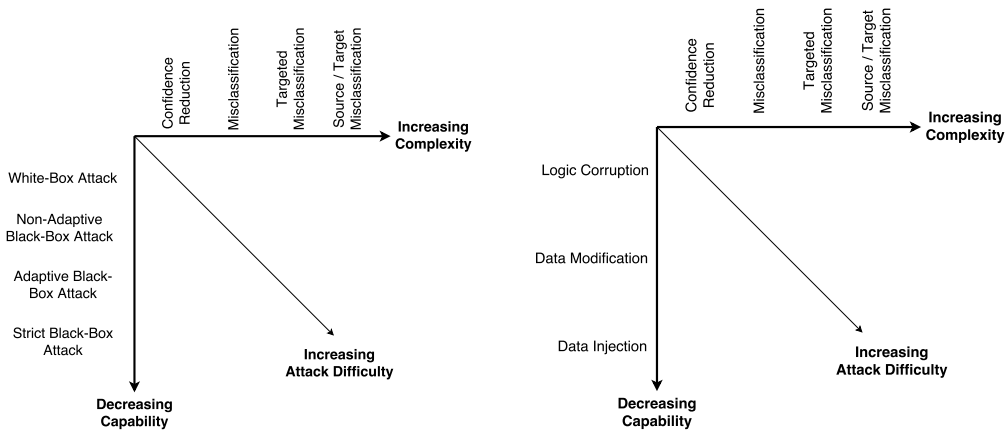
(1) ***Confidence Reduction:*** The adversary tries to reduce the confidence of prediction for the target model. For example, a legitimate image of a 'stop' sign can be predicted with a lower confidence having a lesser probability of class belongingness.

(2) ***Misclassification:*** The adversary tries to alter the output classification of an input example to any class different from the original class. For example, a legitimate image of a 'stop' sign will be predicted as any other class different from the class of stop sign.

(3) ***Targeted Misclassification:*** The adversary tries to produce inputs that force the output of the classification model to be a specific target class. For example, any input image to the classification model will be predicted as a class of images having 'go' sign.

(4) ***Source/Target Misclassification:*** The adversary attempts to force the output of classification for a specific input to be a particular target class. For example, the input image of 'stop' sign will be predicted as 'go' sign by the classification model.

The taxonomy of the adversarial threat model for both the evasion and the poisoning attacks with respect to the adversarial capabilities and adversarial goals are represented graphically in



a) Attack Difficulty with respect to adversarial capabilities and goals for Evasion Attacks

b) Attack Difficulty with respect to adversarial capabilities and goals for Poisoning Attacks

Fig. 5. Taxonomy of Adversarial Model for a) Evasion Attacks and b) Poisoning Attacks with respect to adversarial capabilities and goals

Figure 5. The horizontal axis of both figures represents the complexity of adversarial goals in increasing order, and the vertical axis loosely represents the strength of an adversary in decreasing order. The diagonal axis represents the complexity of a successful attack based on the adversarial capabilities and goals.

Some of the noteworthy attacks along with their target applications is shown in Table. 2. Further in Table. 3, we categorize those attacks under different threat models and discuss in detail about them in the next section.

3 EXPLORATORY ATTACKS

Exploratory attacks do not modify the training set but instead tries to gain information about the state by probing the learner. The adversarial examples are crafted in such a way that the learner passes them as legitimate examples during testing phase.

3.1 Model Inversion(MI) Attack

Fredrikson et al. introduced "model inversion" in [27] where they considered a linear regression model f that predicted drug dosage using patient information, medical history and genetic markers; explored that given white-box access to model f and an instance of data ($\mathbf{X} = \{x_1, x_2, ..., x_n\}, y$), model inversion infers genetic marker x_1. The algorithm produces least-biased maximum a priori (MAP) estimate for x_1 by iterating over all possible values of nominal feature (x_1) for obtaining target value y, thus minimizing adversary's mis-prediction rate. It has serious limitations; for e.g. it cannot handle larger set of unknown features since it is computationally not feasible.

Fredrikson et al. [26] intended to remove limitations of their previous work and showed that for a black-box model, an attacker can predict the patient's genetic markers. This new model inversion attacks through ML APIs that exploit confidence values in a variety of settings and explored countermeasures in both black box and white box settings. The attack infers sensitive features used as inputs to decision tree models for lifestyle surveys, as well as to recover images from API access to facial recognition services. This attack has been successfully experimented in

Articles	Attacks	Applications
Fredrikson et al. [26]	Model Inversion	Biomedical Imaging, biometric identification
Tramèr et al. [73]	Extraction of target machine learning models using APIs	Attacks extend to multiclass classifications & neural networks
Anteniese et al. [11]	Meta-classifier to hack other classifiers	Speech Recognition
Biggio et al. [19], [20]	Poisoning based attacks:	Crafted training data for Support vector Machines
Dalvi et al. [24] Biggio et al. [16], [15]	Adversarial Classification, Pattern recognition	Email Spam detection, fraud detection, intrusion detection, biometric identification
Papernot et al. [60], [57]	Adversarial samples crafting, adversarial sample transferability	digit recognition, black-box attacks against classifiers hosted by Amazon and Google
Hitaj et al. [34]	GAN under collaborative learning	Classification
Goodfellow et al. [30]	Generative Adversarial Network	Classifiers, Malware Detection
Shokri et al. [67]	Membership inference attack	Attack on classification models trained by commercial "ML as a service" providers such as Google and Amazon
Moosavi et al. [52] Carlini et al. [22] Li et al. [45]	Adversarial perturbations: and sample generation: Poisoning based attack	Image classification intrusion detection Collaborative filtering systems

Table 2. Overview of Attacks and Applications

face recognition using neural network models: softmax regression, multilayer perceptron (MLP) and stacked denoising autoencoder network (DAE); given access to the model and person's name, it can recover the facial image. The reconstruction produced by the three algorithms is shown in Figure. 6. Due to the rich structure of deep learning machines, the model inversion attack may recover only prototypical examples that have little resemblance to the actual data that defined the class.

Exploratory Attacks	Model Inversion Membership Inference attack Model Extraction via APIs Information Inference
Evasion Attacks	Adversarial Examples Generation Generative Adversarial Networks (GAN) GAN based attack in collaborative learning Intrusion Detection Systems Adversarial Classification
Poisoning Attacks	Support Vector Machine Poisoning Poisoning on collaborative filtering systems Anomaly Detection Systems

Table 3. Attack Summary

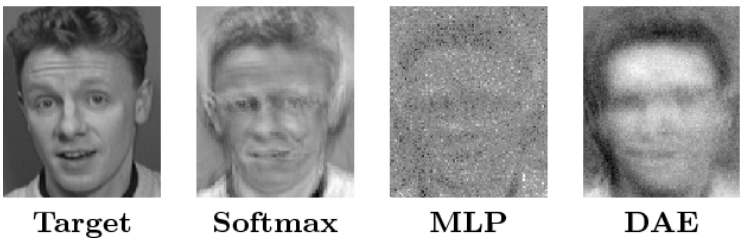


Fig. 6. Reconstruction of the individual on the left by Softmax, MLP, and DAE
(Image Credit: Fredrikson et al. [26])

3.2 Model Extraction using APIs

Tramèr et al. [73] presented simple attacks to extract target machine learning models for popular model classes such as logistic regression, neural networks, and decision trees. Attacks presented are strict black box attacks, but could build models locally that are functionally close to target.The authors demonstrated Model Extraction attack on online ML service providers such as BigML and Amazon Machine Learning. Machine learning APIs provided by ML-as-service providers return precision confidence values along with class labels. Since the attacker do not have any information regarding the model or training data distribution, he can attempt to solve mathematically for unknown parameters or features given the confidence value and equations by quering $d+1$ random d-dimensional inputs for unknown $d+1$ parameters.

3.3 Inference Attack

Ateniese et al. [11] showed it is possible to gather relevant information from machine learning classifiers using a meta-classifier. Given the black box access to a model (e.g., via public APIs) and a training data, an attacker may be interested in knowing whether that data was part of the training set of the model. They experimented with a speech recognition classifier that uses Hidden Markov Models and extracted information such as accent of the users which was not supposed to be explicitly captured.

Another inference attack presented by Shokri et al. [67] is membership inference, i.e., which determines whether a given data point belongs to the same distribution as the training dataset. This attack may fall under the category of non-adaptive or adaptive black box attacks. In a typical black box environment, attacker sends a query to the target model with a data point and obtains model's prediction. The output given by the model is a vector of probabilities which specifies whether the data point belongs to a certain class. For training attack model, a set of shadow models are built. Since the adversary has the knowledge of whether a given record belongs to the training set, supervised learning can be employed and corresponding output labels are then fed to attack model to train it to distinguish shadow model's outputs on members of their training data from those of non-members.

Figure 7 illustrates the end-to-end attack process. The output vectors obtained from shadow model are labeled "in" and added to the attack model's training dataset. A test dataset is also used to query the shadow model and the outputs from this set are labeled "out" and also added to the attack model's training dataset. Thus a collection of target attack models is trained by utilizing the black box behaviour of the shadow models. Authors used membership attacks on classification models trained by commercial "ML as a service" providers such as Google and Amazon.

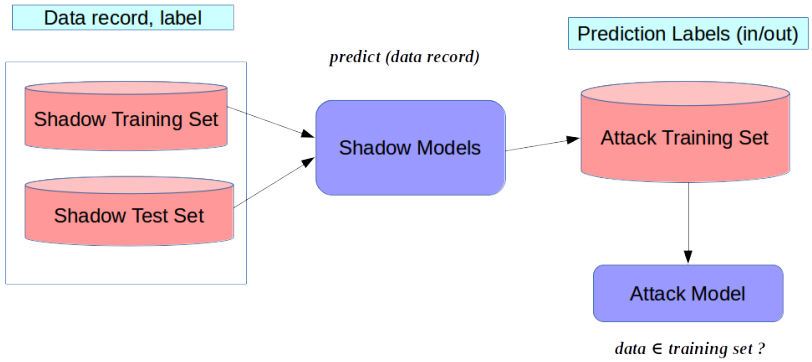


Fig. 7. Overview of Membership inference attack in the black-box setting

4 EVASION & POISONING ATTACKS

Evasion attacks are the most common attacks on machine learning systems. Malicious inputs are craftily modified so as to force the model to make a false prediction and evade detection. Poisoning attack differs in that the inputs are modified during training and model is trained on contaminated inputs to obtain desired output.

4.1 Generative Adversarial Attack(GAN)

Goodfellow et al [30] introduced generative adversarial networks whose aim is to generate samples similar to the training set, having almost identical distribution. The GAN procedure, as depicted in Figure. 8, is composed of a discriminative deep learning network D and a generative deep learning network G. The role of discriminative network is to distinguish between samples taken from the original database and those generated by GAN. The generative network is first initialized with random noise. Its role is to produce samples identical to the training set of the discriminator. Formally, G is trained to maximize the probability of D making a mistake. This competition leads both the models to improve their accuracy. The procedure ends when D fails to distinguish between samples from the training set and those generated by G. The entities and adversaries are in a constant duel where one (generator) tries to fool the other(discriminator), while the other tries to prevent being fooled.

The authors (Goodfellow et al [30]) defined the value function $V(G, D)$ as

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} \log D(x) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D(G(z)))$$

where $p_g(x)$ is the generator's distribution, $p_z(z)$ is a prior on input noise variables. The objective is to train D to maximize the probability of assigning correct label to training and sample examples, while simultaneously training G to minimize it. It was found that optimizing D was computationally intensive and on finite data sets, there was a chance of over fitting. Moreover, during the initial stages of learning when G is poor, D can reject samples with relatively high confidence as it can distinguish them from training data. So, instead of training G to minimize $\log(1 - D(G(z))$, they trained G to maximize $\log(D(G(z))$.

Radford et al [10] introduced a new class of CNNs called Deep Convolutional Generative Adversarial Networks (DCGANs) which overcomes these constraints in GANs and make them stable to train. Some of the key insights of DCGAN architecture were:

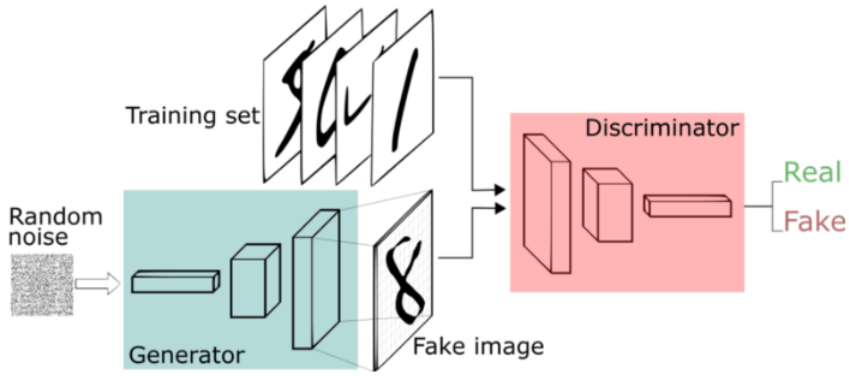


Fig. 8. Generative Adversarial Learning
(Image Credit: Thalles Silva [68])

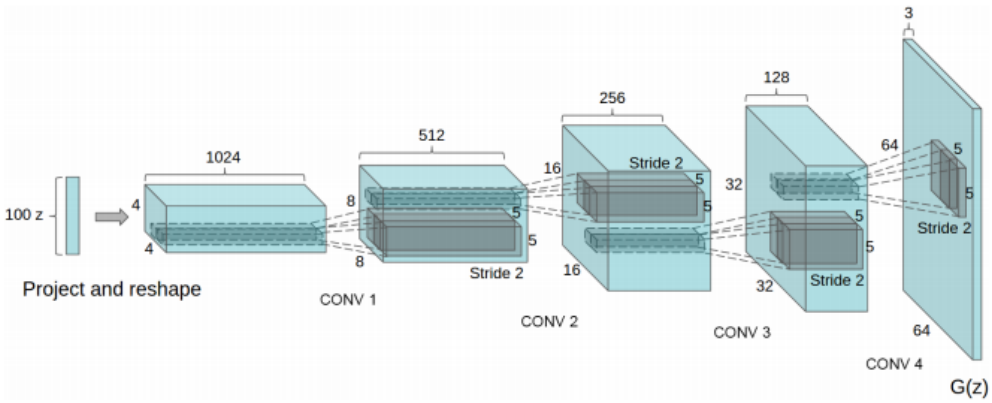


Fig. 9. DCGAN generator used for LSUN scene modeling.
(Image Credit: Radford et al. [10])

- The overall network architecture was based on all convolutional net [69] replacing deterministic pooling functions with strided convolutions.
- The first layer of the GAN, which takes random noise as input, was reshaped into 4-dimensional tensor and used as the start of convolutional stack. The last convolutional layer was flattened and fed into a single sigmoid output (Figure. 9). [53]
- They used batch normalization [36] to stabilize the learning by normalizing the input in order to have zero mean and unit variance. This solved the problem of instability of GAN during training which arise due to poor initialization.
- ReLU activation [55] was used in all layers in generator, except for the output layer which uses Tanh function. Leaky ReLU activation [50], [78] was used for all layers in discriminator.

4.2 Adversarial Examples Generation

In this section, we present a general overview for modifying samples so that a classification model yields an adversarial output. The adversarial sample modification can be performed both in training and testing phase.

4.2.1 Training Phase Modification. A learning process fine-tunes the parameters θ of the hypothesis h by analyzing a training set. This makes the training set susceptible to manipulation by adversaries. Barreno et al.[13] first proposed the term *Poisoning Attacks* which alters the training dataset by inserting, modifying or deleting points keeping the intention of modifying the decision boundaries of the targeted model following the work of Kearns et al. [40], thus challenging the learning system's integrity. The poisoning attack of the training set can be done in two ways: either by direct modification of the labels of the training data or by manipulating the input features depending on the capabilities posed by the adversary. We present a brief overview of both the techniques without much technical details, as the training phase attack needs more powerful adversary and thus is not common in general.

- *Label Manipulation* If the adversary has the capability to modify the training labels only, then he must obtain the most vulnerable label given the full or partial knowledge of the learning model. A basic approach is to randomly perturb the labels, i.e., select new labels for a subset of training data by picking from a random distribution. Biggio et al. [19] presented a study which shows that a random flip of 40% of the training labels is sufficient to degrade the performance of classifiers learned with SVMs.
- *Input Manipulation* In this scenario, the adversary is more powerful and can corrupt the input features of training points analyzed by the learning algorithm, in addition to its labels. This scenario also assumes that the adversary has the knowledge of the learning algorithm.
  Kloft et al. [41] presented a study in which they showed that inserting malicious points in the training dataset could gradually shift the decision boundary of an anomaly detection classifier. The learning algorithm that they used for the study works in an online scenario - new training data are collected at regular intervals, and the parameter values θ are fine-tuned based on a sliding window of that data. Thus, injecting new points in the training dataset is essentially an easy task for the adversary. Poisoning data points can be obtained by solving a linear programming problem which has the objective of maximizing displacement of the mean of the training data.
  In the offline learning settings, Biggio et al. [20] introduced an attack that inserts inputs in the training set, which are crafted using a gradient ascent method. The method identifies the inputs corresponding to local maxima in the test error of the model. They presented a study which shows that by including these inputs into the training set one can result in a degraded classification accuracy for SVM classifier at the testing time. Following their approach, Mei et al. [51] introduced a more general framework for poisoning. Their method finds out an optimal change to the training set whenever the targeted learning model is trained using a convex optimization loss (e.g., SVMs or linear and logistic regression) and its input domain is continuous.

4.2.2 Testing Phase Generation.

- *White-Box Attacks* In this subsection, we precisely discuss how adversaries craft adversarial samples in a white-box setup. Papernot et al. [62] introduced a general framework which builds on the attack approaches discussed in recent literature. The framework is split into two phases: a) *direction sensitivity estimation* and b) *perturbation selection* as shown in Figure 10.

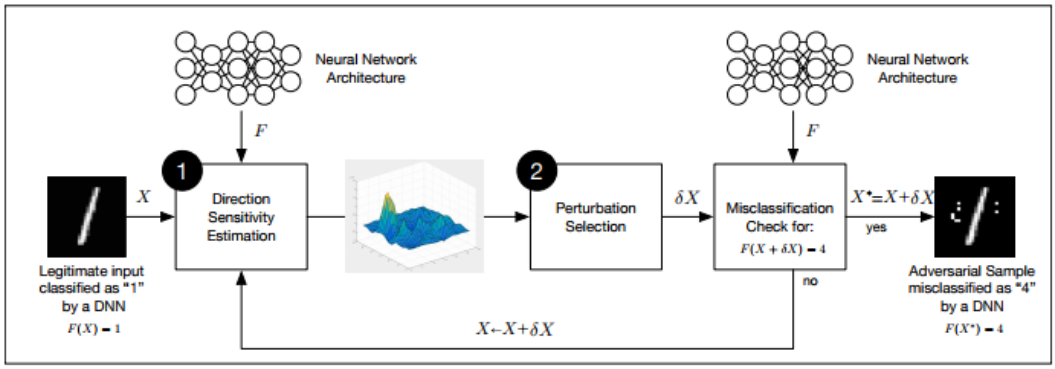


Fig. 10. Adversarial Example Crafting Framework for Evasion Attacks
(Image Credit: Papernot et al. [62])

The figure proposes an adversarial example crafting process for an image classification using DNN, which can be generalized for any supervised learning algorithm.
Suppose X is an input sample, and F is a trained DNN classification model. The objective of an adversary is to craft a malicious example $X_* = X + \delta X$ by adding a perturbation δX with the sample X, so that $F(X_*) = Y_*$ where $Y_* \neq F(X)$ is the target output which depends on the objective of the adversary. An adversary begins with a legitimate sample X. Since the attack setting is a white-box attack, the capability of an adversary is limited to accessing parameters θ of the targeted model F. The adversary employs a two-step process for the adversarial sample crafting, which is discussed below:

(1) ***Direction Sensitivity Estimation:*** The adversary evaluates the sensitivity of a class change to each input feature by identifying directions in the data manifold around sample X in which the model F, learned by the DNN is most sensitive and likely to result in a class change.
(2) ***Perturbation Selection:*** The adversary then exploits the knowledge of sensitive information to select a perturbation δX among the input dimensions in order to obtain an adversarial perturbation which is most efficient.

Both the steps are repeated by replacing X with $X + \delta X$ before the start of each new iteration, until the adversarial goal is satisfied by the perturbed sample. The point to be remembered in this context is that the total perturbation used for crafting the adversarial sample from a valid example needs to be as minimum as possible. This is necessary for the adversarial samples to remain undetected in human eyes.
Adversarial sample crafting using large perturbations is trivial. Thus, if one defines a norm $\|.\|$ to appropriately describe the differences between points in the input domain of the DNN model F, we can formalize the adversarial samples as a solution to the following optimization problem:

$$X_* = X + \arg\min_{\delta X} \{\|\delta X\| : F(X + \delta X) \neq F(X)\} \quad (1)$$

Most DNN models make this formulation non-linear and non-convex, making it hard to find a closed-solution in most of the cases. We now describe in details different techniques to find an approximate solution to this optimization problem using each of the two steps mentioned above.

Direction Sensitivity Estimation. In this step, the adversary considers a legitimate sample X, an n-dimensional input vector. The objective here is to find those dimensions of X which will produce an expected adversarial performance with the smallest selected perturbation. This can be achieved by changing the input components of X and evaluating the sensitivity of the trained DNN model F for these changes. Developing the knowledge of the model sensitivity can be accomplished in several ways. Some of the well-known techniques mentioned in the recent literature are discussed below.

(1) **L-BFGS:** Szegedy et al. [70] first introduced the term adversarial sample by formalizing the following minimization problem as the search for adversarial examples.

$$\arg\min_{r} f(x+r) = l \quad \text{s.t.} \ (x+r) \in D$$

The input example x, which is correctly classified by f, is perturbed with r to obtain the resulting adversarial example $x_* = x + r$. The perturbed sample remains in the input domain D, however, it is assigned the target label $l \neq h(x)$. For non-convex models like DNN, the authors have used the *L-BFGS* [47] optimization algorithm to solve the above equation. Though the method gives good performance, it is computationally expensive while calculating adversarial samples.

(2) **Fast Gradient Sign Method (FGSM):** An efficient solution to equation 1 is introduced by Goodfellow et al. [31]. They proposed a fast gradient sign methodology which calculates the gradient of the cost function with respect to the input of the neural network. The adversarial examples are generated using the following equation:

$$X_* = X + \epsilon * sign(\nabla_x J(X, y_{true}))$$

Here, J is the cost function of the trained model, ∇_x denotes the gradient of the model with respect to a normal sample X with correct label y_{true}, and ϵ denotes the input variation parameter which controls the perturbation's amplitude. Recent literature have used some other variations of FGSM, which are summarised as below:

(a) **Target Class Method:** This variant of FGSM [43] approach maximizes the probability of some specific target class y_{target}, which is unlikely the true class for a given example. The adversarial example is crafted using the following equation:

$$X_* = X - \epsilon * sign(\nabla_x J(X, y_{target}))$$

(b) **Basic Iterative Method:** This is a straightforward extension of the basic FGSM method [43]. This method generates adversarial samples iteratively using small step size.

$$X_*^0 = X; \quad X_*^{n+1} = Clip_{X,e}\{X_*^n + \alpha * sign(\nabla_x J(X_*^n, y_{true}))\}$$

Here, α is the step size and $Clip_{X,e}\{A\}$ denotes the element-wise clipping of X. The range of $A_{i,j}$ after clipping belongs in the interval $[X_{i,j} - \epsilon, X_{i,j} + \epsilon]$. This method does not typically rely on any approximation of the model and produces additional harmful adversarial examples when run for more iterations.

(3) **Jacobian Based Method:** Papernot et al. [58] introduced a different approach for finding sensitivity direction by using forward derivative, which is the Jacobian of the trained model F. This method directly provides gradients of the output components with respect to each input component. The knowledge thus obtained is used to craft adversarial samples using a complex saliency map approach which we will discuss later. This method is particularly useful for source-target misclassication attacks.

Perturbation Selection. An adversary may use the information about network sensitivity for input differences in order to evaluate the dimensions which are most likely to generate the target misclassification with minimum perturbation. The perturbed input dimensions can be of two types:

(1) **Perturb all the input dimensions:** Goodfellow et al. [31] proposed a way to perturb every input dimensions but with a small quantity in the direction of the sign of the gradient calculated using the FGSM method. This method efficiently minimizes the Euclidian distance between the original and the corresponding adversarial samples.

(2) **Perturb a selected input dimensions:** Papernot et al. [58] choose to follow a more complicated process involving saliency maps to select only a limited number of input dimensions to perturb. The objective of using saliency map is to assign values to the combination of input dimensions which indicates whether the combination if perturbed, will contribute to the adversarial goals. This method effectively reduces the number of input features perturbed while crafting adversarial examples. For choosing the input dimensions which forms the perturbations, all the dimensions are sorted in decreasing order of adversarial saliency value. The saliency value $S(x,t)[i]$ of a component i of a legitimate example x for a target class t is evaluated using the following equation:

$$S(x,t)[i] = \begin{cases} 0, & \text{if } \frac{\partial F_t}{\partial x_i}(x) < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j}{\partial x_i}(x) > 0 \\ \frac{\partial F_t}{\partial x_i}(x) \left| \sum_{j \neq t} \frac{\partial F_j}{\partial x_i}(x) \right|, & \text{otherwise} \end{cases}$$

where $\left[\frac{\partial F_j}{\partial x_i}\right]_{ij}$ could be easily calculated using the Jacobian Matrix J_F of the learned model F. Input components are added to perturbation δxin the decreasing order of the saliency value $S(x,t)[i]$ until the resulting sample $x_* = x + \delta x$ is misclassified by the model F.

Each method has its own advantages and drawbacks. The first method is well fitted for the fast crafting of many adversarial samples but with a relatively large perturbation and thus is potentially easier to detect. The second method reduces the perturbations at the expense of a higher computing cost.

- *Black-Box Attacks* In this subsection, we discuss in details on the generation of adversarial examples in a black-box setting. Crafting adversarial samples in non-adaptive and strict black box scenario is straightforward. In both the cases, the adversary has access to a vast dataset to train a local substitute model which approximates the decision boundary of the target model. Once the local model is trained with high confidence, any of the white-box attack strategies can be applied on the local model to generate adversarial examples, which eventually can be used to fool the target model because of the *Transferability Property of Neural Network* (will be discussed later). However, in the case of an adaptive black-box scenario, the adversary does not have access to a large dataset and thus augments a partial or randomly selected dataset by selectively querying the target model as an oracle. One of the popular method of dataset augmentation presented by Papernot et al. [60] is discussed next.

Jacobian based Data Augmentation. An adversary could potentially make an infinite number of queries to get the Oracle's output $O(x)$ for any input x. This would provide the adversary a copy of the oracle. However, the process is not tractable considering the continuous domain of an input to be queried. Furthermore, making a significant number of queries presents the adversarial behavior easy to detect. The heuristic that can be used to craft synthetic training inputs is based on identifying the directions in which the target model's output is varying, around an initial set of training points. With more input-output pair the direction can be captured easily for a target Oracle O. Hence, the greedy heuristic that an adversary follow

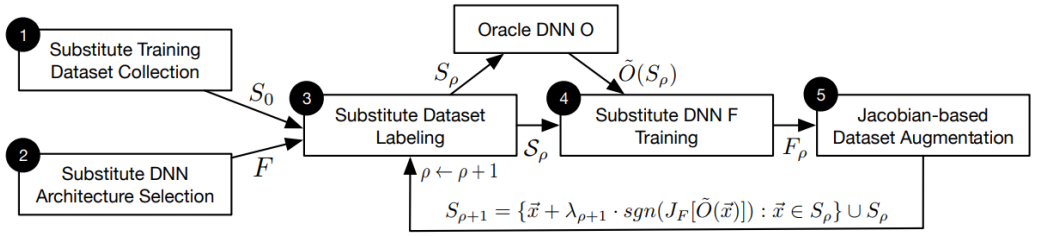


Fig. 11. Substitute Mode Training using Jacobian based dataset augmentation.
(Image Credit: Papernot et al. [60])

is to prioritize the samples while querying the oracle for labels to get a substitute DNN F approximating the decision boundaries of the Oracle. These directions can be identified with the substitute DNN's Jacobian matrix J_F, which is evaluated at several input points x. Precisely, the adversary evaluates the sign of the Jacobian matrix dimension corresponding to the label assigned to input x by the oracle, denoted by $sgn\left(J_F(x)\left[O(x)\right]\right)$. The term $\lambda * sgn\left(J_F(x)\left[O(x)\right]\right)$ is added to the original datapoint, to obtain a new synthetic training point. The iterative data augmentation technique can be summarized using the following equation.

$$S_{n+1} = \{x + \lambda * sgn\left(J_F(x)\left[O(x)\right]\right) : x \in S_n\} \cup S_n$$

where S_n is the dataset at n^{th} step and S_{n+1} is the augmented dataset. The substitute model training following this approach is presented in Figure 11.

4.2.3 Transferability of Adversarial Samples. Adversarial sample transferability [57] is the property that adversarial samples produced by training on a specific model can affect another model, even if they have different architectures - leading to adaptive black box attacks as discussed in Section 2.2.2. Since in case of black-box attack, adversary does not have access to the target model F, an attacker can train a *substitute model* F' locally to generate adversarial example $X + \delta X$ which then can be transfered to the victim model. Formally, if X is the original input, the transferability problem can be represented as an optimization problem (1). It can be broadly classified into two types:

(1) Intra-technique transferability: If models F and F' are both trained using same machine learning technique (e.g. both are NN or SVM)
(2) Cross-technique transferability: If learning technique in F and F' are different, for example, F is a neural network and F' is a SVM.

This substitute model in effect transfers knowledge crafting adversarial inputs to the victim model. The targeted classifier is designated as an oracle because adversaries have the minimal capability of querying it for predictions on inputs of their choice. To train the substitute model, dataset augmentation as discussed above is used to capture more information about the predicted outputs. Authors also introduces reservoir sampling to select a limited number of new inputs while performing Jacobian-based data augmentation, reduces the number of queries made to the oracle. The choice of substitute model architecture has limited impact on transferability.

The attacks have been shown to generalize to non-differentiable target models like decision trees, e.g, deep neural networks (DNNs) and logistic regression (LR) (differentiable models) could both effectively be used to learn a substitute model for many classifiers trained with a

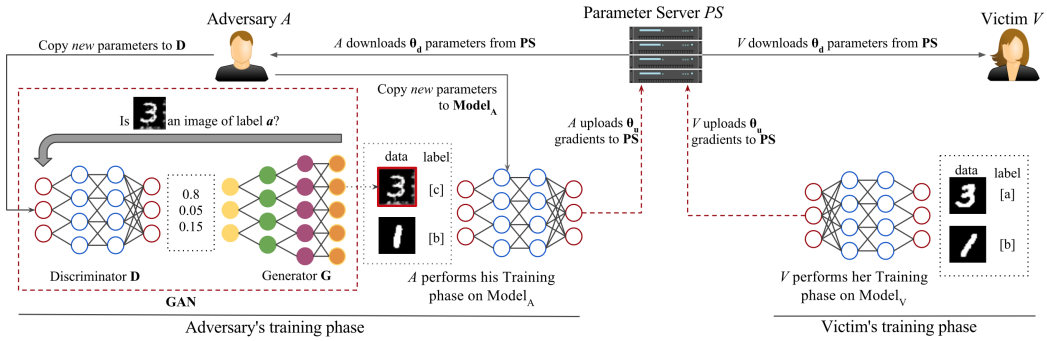


Fig. 12. GAN Attack on collaborative deep learning
(Image Credit: Hitaj et al. [34])

support vector machine, decision tree, and nearest neighbor. Cross-technique transferability reduces the amount of knowledge that adversaries must possess in order to force misclassification by crafted samples. Learning substitute model alleviates the need of attacks to infer architecture, learning model and parameters in a typical black-box based attack.

4.3 GAN based attack in Collaborative Deep Learning

Hitaj et al. [34] presented a GAN based attack to extract information from honest victims in a collaborative deep learning framework. The goal of GAN is to produce samples identical to those in the training set without having access to the original training set. GAN-based method works only during the training phase in collaborative deep learning. The authors showed that the attack can be carried out in Convolutional Neural Networks which are themselves pretty difficult to invert or even when the parameters are hidden using differential privacy. In the white-box setting, the adversary acts as an insider within the privacy-preserving [66] collaborative deep learning protocol. The motive of the adversary is to extract tangible information about the labels that do not belong to his dataset.

Figure 12 shows the proposed attack, which uses GAN to generate similar samples as that of training data in a collaborative learning setting, with limited access to shared parameters of the model. A, V participates in a collaborative learning. V(the victim) chooses labels $[x, y]$. The adversary A also chooses labels $[y, z]$. Thus, while class y is common to both A and V, A does not have any information about the x. The goal of the adversary is to collect maximum information possible about the class x. Using GAN, the adversary generates samples which are identical to class x. The insider injects these fake samples from x, as class z into the distributed learning procedure. The victim unable to distinguish between classes x and z reveals more information about class x than initially intended. Thus, the insider imitates samples from x and uses the victim to improve his knowledge about a class he ignored before training. The GAN attack works as long as A's local model improves its accuracy over time.

Authors [34] proved that GAN attack was successful in all set of experiments conducted juxtaposed with MI attacks, DP based collaborative learning, etc. The GAN will generate good samples as long as the discriminator is learning.

4.4 Adversarial Classification

Dalvi et al [24] defined adversarial classification from cost-sensitive game-theoretical perspective as a game between Classifier, who tries to learn from training set a function $y_c = C(x)$ that will

accurately predict the classes of instances in training set, and Adversary, who tries to make Classifier predict positive instance of the training set as negative by modifying those instances from x to x'= $A(x)$. The classifier and the adversary are constantly trying to defeat each other by maximizing their own payoffs. The classifier uses a cost-sensitive Bayes learner to minimize its expected cost while assuming that the adversary always plays its optimal strategy, whereas the adversary tries to modify feature in order to minimize its own expected cost [12] [79]. The presence of adaptive adversaries in a system can significantly degrade the performance of the classifier, specially when the classifier is unaware of the presence or type of adversary. The goal of the Classifier is to build a classifier C to maximize its expected utility (U_C), whereas the goal of the Adversary is to find a feature change strategy A in order to maximize its own expected utility:

$$U_C = \sum_{(x,y)\epsilon XY} P(x,y) \left[U_C(C(A(x),y) - \sum_{X_i \epsilon X_C(x)} V_i \right]$$

$$U_A = \sum_{(x,y)\epsilon XY} P(x,y) \left[U_A(C(A(x),y) - W(x,A(x)) \right]$$

4.5 Evasion and Poisoning attack on Support Vector Machines

Support Vector Machines (SVM) are one of the most popular classification techniques widely used for malware detection,. intrusion detection systems and spam filtering. SVM follows *stationarity* property, i.e., both the training and test data should come from the same distribution. However, in adversarial learning, an intelligent and adaptive adversary can manipulate data thereby violating stationarity to exploit the vulnerabilities of the learning system.

Biggio et al [21] using gradient-descent based approach demonstrated the evasion attack on kernel-based classifiers [29]. Even if the adversary is unaware of the classifier's decision function, he can learn a *surrogate* classifier to evade the classifier. The authors considered two approaches: in the first, the adversary had knowledge about feature space and classifier's discriminant function, while in the second, the adversary did not know the learned classifier. In the second scenario, the adversary is assumed to learn a *surrogate* classifier on a surrogate training set. So, even if the adversary can learn a classifier's copy on surrogate data, SVMs can be evaded with relatively high probability. So, the optimal strategy to find an attack sample x that will minimize the value of the classifier's discriminant function $g(x)$ can be given as:

$$x^* = \arg\min_x \hat{g}(x) \qquad s.t \quad d(x, x^0) \leq d_{max}$$

However, if ĝ(x) is not convex, the gradient descent may lead to a local minima outside of sample's support. To overcome this problem, additional components were introduced into the equation yielding the following modified equation:

$$\arg\min_x E(x) = \hat{g}(x) - \frac{\lambda}{n} \sum_{i|f_i=-1} k(\frac{x - x_i}{h}) \qquad s.t \quad d(x, x^0) \leq d_{max}$$

Biggio et al [20] also demonstrated poisoning attacks against SVMs in which the adversary manipulates training data to force the SVM to misclassify test samples. Although the training data was tampered to include well-crafted attack samples, it was assumed that there was no manipulation of test data. For the attack, the adversary's goal is to find a set of points whose addition to the training dataset will maximally decrease the SVM's classification accuracy. Although having complete knowledge of the training dataset is difficult in real-world scenarios, but collecting a *surrogate* dataset having the same distribution as the actual training dataset may not be difficult for

the attacker. Under these assumptions, the optimal attack strategy can be given as:

$$x^* = \arg\max_x P(x) = \sum_{k=1}^{m}(1 - y_k f_x(x_k))_+ = \sum_{k=1}^{m}(-g_k)_+ \qquad s.t \quad x_{lb} \leq x \leq x_{ub}$$

Another class of attacks on SVM called privacy attack, presented by Rubinstein et al [14], is based on the attempt to breach of training data's confidentiality. The ultimate goal of the adversary is to determine features and/or the label of an individual training instance by trying to inspect test-time classifications made by the classifier or inspecting the classifier directly.

4.6 Poisoning attacks on Collaborative Systems

Recommendation and collaborating filtering systems play an important part in the business strategies of modern e-commerce systems. The performance of these systems can seriously affect the business, both in a positive and negative way, thereby making them attractive targets for the adversaries. Bo Li et al [45] demonstrated poisoning attacks on collaborative filtering systems where an attacker, having complete knowledge of the learner, can generate malicious data to degrade the effectiveness of the system.

The two most popular algorithms used for factorization-based collaborative filtering are alternating minimization [64] and nuclear norm minimization [38]. For the former one, the data matrix M can be determined by the following optimization problem:

$$\min_{U \epsilon \mathbb{R}^{m\times k}, V \epsilon \mathbb{R}^{n\times k}} \left\{ \left\| \mathcal{R}_\Omega(M - UV^T) \right\|_F^2 + 2\lambda_U \|U\|_F^2 + 2\lambda_V \|V\|_F^2 \right\} \tag{1}$$

Alternatively, the latter one can be solved by the following problem:

$$\min_{X \epsilon \mathbb{R}^{m\times k}} \left\{ \|\mathcal{R}_\Omega(M - X)\|_F^2 + 2\lambda \|X\|_* \right\} \tag{2}$$

where $\lambda > 0$ is a regularization parameter and $\|X\|_* = \sum_{i=1}^{rank(X)} |\sigma_i(X)|$ is the nuclear form of X .

Based on these problems, the authors Bo Li et al [45] listed the three types of attacks and their utility functions:

- *Availability Attack :* In this attack model, the goal of the attacker is to maximize the error of the collaborative filtering system thereby making it unreliable and useless. The utility function can be defined as the total number of perturbations of predictions between $\overline{M}$ (prediction without data poisoning) and $\widehat{M}$ (prediction after poisoning) on unseen entries Ω^C

$$R^{av}(\widehat{M}, M) = \left\| \mathcal{R}_{\Omega_C}(\widehat{M} - \overline{M}) \right\|_F^2$$

- *Integrity Attack :* In this model, the attacker tries to manipulate the popularity of a subset of items. Thus, if $J_0 \subseteq [n]$ is the subset of items and $w : J_0 \rightarrow \mathbb{R}$ is a pre-specified weight vector, then the utility function can be defined by

$$R^{in}_{J_0,w}(\widehat{M}, M) = \sum_{i=1}^{m} \sum_{j \epsilon J_0} w(j) \widehat{M}_{ij}$$

- *Hybrid Attack :* It is a hybrid of the above two models, defined by the function

$$R^{hybrid}_{J_0,w,\mu}(\widehat{M}, M) = \mu_1 R^{av}(\widehat{M}, M) + \mu_2 R^{in}_{J_0,w}(\widehat{M}, M)$$

where $\mu = (\mu_1, \mu_2)$ are the coefficients that provides the trade off between availability and integrity attacks.

4.7 Adversarial attacks on Anomaly Detection Systems

Anomaly detection is referred to the detection of events that do not conform to an expected pattern or behaviour. Kloft et al [41] analyzed the behaviour of online centroid anomaly detection systems under poisoning attack. The purpose of anomaly detection system is to determine whether a test example x is likely to originate from the same distribution as the given dataset X. It flags x as outlier if it lies in a region of low density compared with the probability density function of sample space X. The authors used *finite sliding window* of training data where, with the arrival of every new data point, the centre of mass changes by

$$c' = c + \frac{1}{n}(x - x_i)$$

As illustrated in Figure. 13, the adversary will try to force the anomaly detection algorithm to accept an attack point A which lies outside the normal ball, i.e. $||A - c|| > r$.

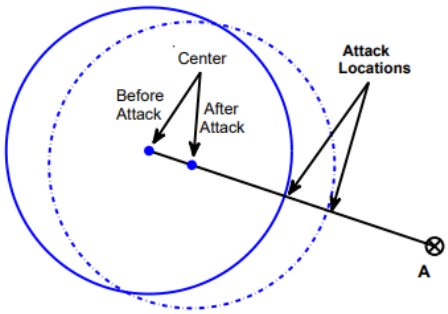


Fig. 13. Illustration of poisoning attack
(Image Credit: Kloft et al. [41])

Although the attacker is assumed to have knowledge about the algorithm and the training data, he cannot modify existing data other than adding new points.

5 ADVANCES IN DEFENSE STRATEGIES

In this section, we provide a brief discussion about the recent advancements in defense strategies. Adversarial examples demonstrate that many modern machine learning algorithms can be broken easily in surprising ways. A substantial amount of research to provide a practical defense against these adversarial examples can be found in recent literature. These adversarial examples are hard to defend because of the following reasons [75]:

(1) *A theoretical model of the adversarial example crafting process is very difficult to construct.* The adversarial sample crafting is a complex optimization process which is non-linear and non-convex for most Machine Learning models. The lacking of proper theoretical tools to describe the solution to these complex optimization problems make it even harder to make any theoretical argument that a particular defense will rule out a set of adversarial examples.
(2) *Machine Learning models are required to provide proper outputs for every possible input.* A considerable modification of the model to incorporate robustness against the adversarial examples may change the elementary objective of the model.

Most of the current defense strategies are not adaptive to all types of adversarial attack as one method may block one kind of attack but leaves another vulnerability open to an attacker who

knows the underlying defense mechanism. Moreover, implementation of such defense strategies may incur performance overhead, and can also degrade the prediction accuracy of the actual model. In this section, we discuss the most recent defense mechanisms used to block these adversarial attacks along with their respective possible drawbacks of implementation.

The existing defense mechanisms can be categorized among the following types based on their methods of implementation.

5.1 Adversarial Training

The primary objective of the adversarial training is to increase model robustness by injecting adversarial examples into the training set [31, 48, 66, 70]. Adversarial training is a standard brute force approach where the defender simply generates a lot of adversarial examples and augments these perturbed data while training the targeted model. The augmentation can be done either by feeding the model with both the original data and the crafted data, presented in [43] or by learning with a modified objective function given by [31] -

$$\widetilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha) J(\theta, x + \epsilon sign(\nabla_x J(\theta, x, y)), y)$$

with J being the original loss function. The central idea behind this strategy is to increase model's robustness by ensuring that it will predict the same class for legitimate as well as perturbed examples in the same direction. Traditionally, the additional instances are crafted using one or multiple attack strategies mentioned in Section 4.2.2.

Adversarial training of a model is useful only on adversarial examples which are crafted on the original model. The defense is not robust for black-box attacks [56, 60] where an adversary generates malicious examples on a locally trained substitute model. Moreover, Tramèr et al. [72] have already proved that the adversarial training can be easily bypassed through a two-step attack, where random perturbations are applied to an instance first and then any classical attack technique is performed on it, as mentioned in Section 4.2.2.

5.2 Gradient Hiding

A natural defense against gradient-based attacks presented in [72] and attacks using adversarial crafting method such as FGSM, could consist in hiding information about the model's gradient from the adversary. For instance, if the model is non-differentiable (e.g, a Decision Tree, a Nearest Neighbor Classifier, or a Random Forest), gradient-based attacks are rendered ineffective. However, this defense are easily fooled by learning a surrogate Black-Box model having gradient and crafting examples using it [60].

5.3 Defensive Distillation

The term distillation was originally proposed by Hinton et al. [33] as a way to transfer knowledge from a large neural networks to a smaller one. Papernot et al. [59, 62] recently proposed using it as a defensive mechanism against adversarial example crafting methods.

The two-step working of the distillation method is described as below. Let us assume that we already have a neural network F which classifies a training dataset X into the target classes Y. The final softmax layer in the produces a probability distribution over Y. Further, assume that we want to train a second neural network F' on the same dataset X achieving the same performance. Now, instead of using the target class labels of the training dataset, we use the output of the network F as the label to train another neural network F' with the same architecture having the same input dataset X. The new labels, therefore, contain more information about the membership of X to the different target classes, compared to a simple label that just chooses the most likely class.

The final softmax layer in the distillation method is modified according to the following equation:

$$F_i(X) = \frac{e^{\frac{z_i(X)}{T}}}{\sum_{i=1}^{|Y|} e^{\frac{z_i(X)}{T}}}$$

where T is the distillation parameter called temperature. Papernot et al. [62] showed experimentally, a high empirical value of T gives a better distillation performance. The advantage of training the second model using this approach is to provide a smoother loss function, which is more generalized for an unknown dataset and have high classification accuracy even for adversarial examples.

Papernot et al. [62] used defensive distillation, learned by the DNN architecture, to smooth the model. The technique is known as label smoothing which involves converting class labels into soft targets. A value close to 1 is assigned for the target class and the rest of the weight is distributed to the other classes. These new values are used as labels for training the model instead of the true labels. As an outcome, the need to train an additional model as for defensive distillation is relaxed. The advantage of using soft targets lies in the fact that it uses probability vectors instead of hard class labels. For example, given an image X of handwritten digits, the probability of similar looking digits will be close to one another.

However with the recent advancement in the black-box attack, the defensive distillation method as well as the label smoothing method can easily be avoided [22, 60]. The main reason behind the success of these attacks is often the strong transferability of adversarial examples across neural network models.

5.4 Feature Squeezing

Feature squeezing is another model hardening technique [76, 77]. The main idea behind this defense is that it reduces the complexity of representing the data so that the adversarial perturbations disappear because of low sensitivity. There are mainly two heuristics behind the approach considering an image dataset.

(1) Reduce the color depth on a pixel level, i.e., encoding the colors with fewer values.
(2) Use of a smoothing filter over the images. As a result, multiple inputs are mapped into a single value which makes the model safe against noise and adversarial attacks.

Though these techniques work well in preventing adversarial attacks, these have the collateral effect of worsening the accuracy of the model on true examples

5.5 Blocking the Transferability

The main reason behind defeat of most of the well-known defense mechanism is due to the strong transferability property in the neural networks, i.e., adversarial examples generated on one classifier are expected to cause another classifier to perform the same mistake. The transferability property holds true even if the classifiers have different architectures or trained on disjoint datasets. Hence, the key for protecting against a black-box attack is to block the transferability of the adversarial examples.

Hosseini et al. [35] recently proposed a three-step *NULL Labeling* method to prevent the adversarial examples to transfer from one network to another. The main idea behind the proposed approach is to augment a new NULL label in the dataset and train the classifier to reject the adversarial examples by classifying them as NULL. The basic working of the approach is shown in Figure 14. The figure illustrates the method taking as an example image from MNIST dataset, and three adversarial examples with different perturbations. The classifier assigns a probability vector to each image. The NULL labeling method assigns a higher probability to the NULL label with higher

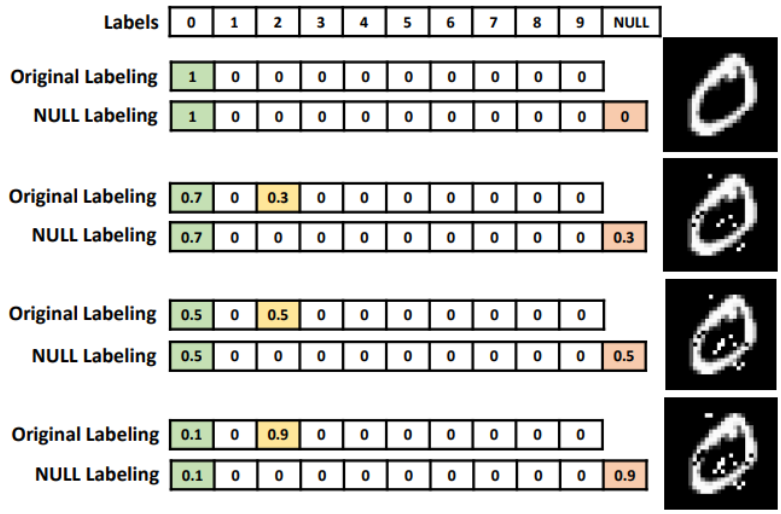


Fig. 14. Illustration of NULL Labeling Method
(Image Credit: Hosseini et al. [35])

perturbation, while the original labeling without the defense increases the probabilities of other labels.

The NULL Labeling method is composed of three major steps:

(1) **Initial Training of the target classifier:** Initial training is performed on the clean dataset to derive the decision boundaries for the classification task.

(2) **Computing the NULL probabilities:** The probability of belonging to the NULL class is then calculated using a function f for the adversarial examples generated with different amount of perturbations.

$$p_{NULL} = f\left(\frac{\|\delta X\|_0}{\|X\|}\right)$$

where, δX is the perturbation and $\|\delta X\|_0 \sim U[1, N_{max}]$. N_{max} is the minimum number for which $f(\frac{N_{max}}{\|X\|}) = 1$.

(3) **Adversarial Training:** Each clean sample is then re-trained with the original classifier along with different perturbed inputs for the sample. The label for the training data is decided based on the NULL probabilities obtained in the previous step.

The advantage of this method is the labeling of the perturbed inputs to NULL label instead of classifying them into their original label. To the best of our knowledge, this method is the most effective defense against the adversarial attacks to date. This method is accurate to reject an adversarial example while not compromising the accuracy of the clean data.

5.6 Defense-GAN

Samangouei et al. [63] proposed a mechanism to leverage the power of Generative Adversarial Networks [30] to reduce the efficiency of adversarial perturbations, which works both for white box and black box attacks. In a typical GAN, a generative model which emulated the data distribution, and a discriminative model that differentiates between original input and perturbed input, are

trained simultaneously. The central idea is to "project" input images onto the range of the generator G by minimizing the reconstruction error $\|G(z) - x\|_2^2$, prior to feeding the image x to the classifier. Due to this, the legitimate samples will be close to the range of G than the adversarial samples, resulting in substantial reduction of potential adversarial perturbations. An overview of the Defense-GAN mechanism is shown in Figure. 15.

Although Defense-GAN showed to be quite effective against adversarial attacks, its success relies on the expressiveness and generative power of the GAN. Moreover, training of GAN can be challenging and if not properly trained, the performance of the Defense-GAN can significantly degrade.

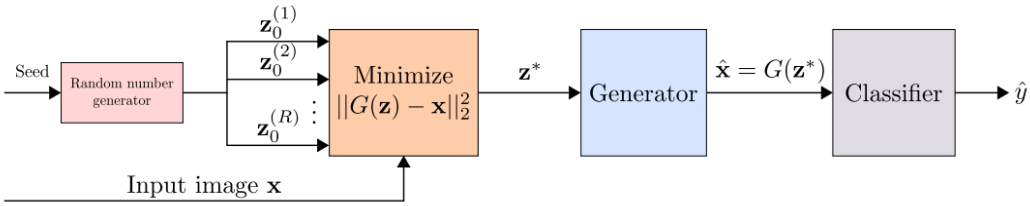


Fig. 15. Overview of Defense-GAN algorithm
(Image Credit: Samangouei et al. [63])

5.7 MagNet

Meng et al. [25] proposed a framework, named MagNet, which uses classifier as a black box to read the output of the classifier's last layer without reading the data on any internal layer or modifying the classifier. MagNet uses *detectors* to distinguish between a normal and adversarial example. The detector measures the distance between the given test example and the manifold and rejects it if the distance exceeds a threshold. It also uses a *reformer* to reform adversarial example to a similar legitimate example using autoencoders. Although MagNet was successful in thwarting a range of black-box attacks, its performance degraded significantly in case of white-box attacks where the attackers are supposed to be aware of the parameters of MagNet. So, the authors came up with the idea of using varieties of autoencoders and randomly pick one at a time to make it difficult for the adversary to predict which autoencoder was used.

5.8 Using High-Level Representation Guided Denoiser

While standard denoisers like pixel-level reconstruction loss function, suffer from error amplification, high-level representation guided denoiser(HGD) can effectively overcome this problem by using a loss function which compares the target model's output produced by clean image and denoised image. Liao et al [46] introduced HGD to devise a robust target model immune against white-box and black-box adversarial attacks. Another advantage of using HGD is that it can be trained on a relatively small dataset and can be used to protect models other than the one guiding it.

In the HGD model, the loss function is defined as the L_1 norm of the difference between the l-th layer representation of the neural network, activated by x and $\hat{x}$

$$L = \|f_l(\hat{x} - f_l(x)\|$$

The authors [46] proposed three different training methods for HGD, illustrated in Figure. 17. In FGD (Feature Guided Denoiser), they defined $l = -2$ as the index for the topmost convolutional

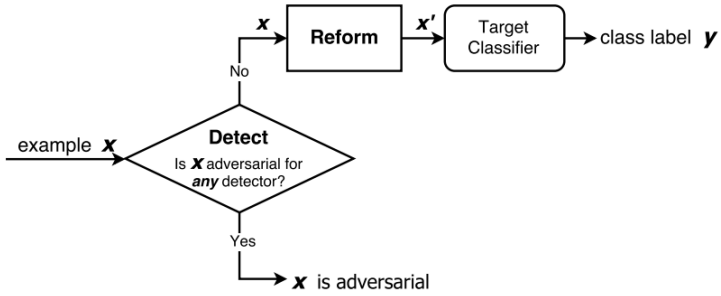


Fig. 16. MagNet workflow in test phase.
(Image Credit: Meng et al. [25])

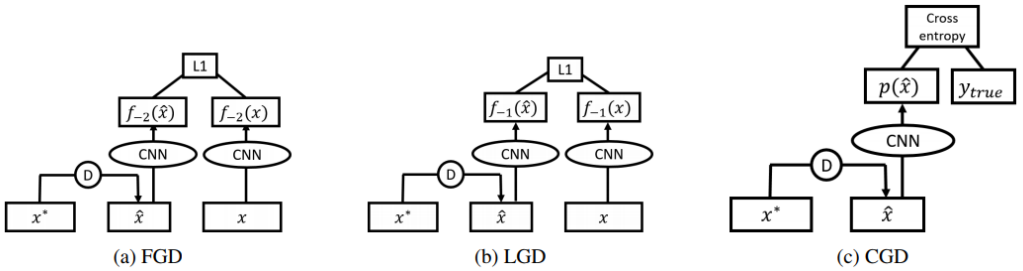


Fig. 17. Three different training methods for HGD. D stands for denoiser, CNN is the model to the defended.
(Image Credit: Liao et al. [46])

layer and fed the activations of that layer to the linear classification layer after global average pooling. In LGD (Logits Guided Denoiser), they defined $l = -1$ as the index for the logits, i.e., the layer before the softmax layer. While both of these variants were unsupervised models, they proposed a supervised variant, CGD (Class label Guided Denoiser), which uses classification loss of the target model as the loss function.

5.9 Using Basis Function Transformations

Shaham et al. [74] investigated various defense mechanisms like PCA, low-pass filtering, JPEG compression, soft thresholding, etc. by manipulations based on basis function representation of images. All the mechanisms were applied as a pre-processing step on both adversarial and legitimate images and evaluated the efficiency of each technique by their success at distinguishing between the two sets of images. The authors showed that JPEG compression performs better than all the other defence mechanisms under consideration across all types of adversarial attacks in black-box, grey-box and white-box settings.

As described, the existing defense mechanisms have their limitations in the sense that they can provide robustness against specific attacks in specific settings. The design of a robust machine learning model against all types of adversarial examples is still an open research problem.

6 CONCLUSION

Despite their high accuracy and performance, machine learning algorithms have been found to be vulnerable to subtle perturbations that can have catastrophic consequences in security related environments. The threat becomes more grave when the applications operate in adversarial environment. So, it has become immediate necessity to devise robust learning techniques resilient to adversarial attacks. A number of research papers on adversarial attacks as well as their countermeasures has surfaced since Szegedy et al [70] demonstrated the vulnerability of machine learning algorithms. In this paper we have tried to explore some of the well known attacks and proposed defense strategies. We have also tried to provide a taxonomy on topics related to adversarial learning. After the review we can conclude that adversarial learning is a real threat to application of machine learning in physical world. Although there exists certain countermeasures, but none of them can act as a panacea for all challenges. It remains as an open problem for the machine learning community to come up with a considerably robust design against these adversarial attacks.

REFERENCES

[1] 2011. iOS - Siri - Apple. https://www.apple.com/ios/siri/

[2] 2014. Alexa - Amazon. https://developer.amazon.com/alexa

[3] 2014. Google Cloud AI. https://cloud.google.com/products/machine-learning/

[4] 2015. Cortana | Your Intelligent Virtual & Personal Assistant | Microsoft. https://www.microsoft.com/en-us/windows/cortana

[5] 2015. nVIDIA GPU Cloud Computing. http://www.nvidia.com/object/gpu-cloud-computing.html

[6] 2017. Alibaba Cloud. https://www.alibabacloud.com/

[7] 2019. Intel Nervana Platform. https://www.intelnervana.com/intel-nervana-platform/

[8] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 308–318.

[9] Naveed Akhtar and Ajmal Mian. 2018. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *arXiv preprint arXiv:1801.00553* (2018).

[10] Luke Metz Alec Radford and Soumith Chintala. 2018. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434* (2018).

[11] Giuseppe Ateniese, Giovanni Felici, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, and Domenico Vitali. 2013. Hacking Smart Machines with Smarter Ones: How to Extract Meaningful Data from Machine Learning Classifiers. *CoRR* abs/1306.4447 (2013). arXiv:1306.4447 http://arxiv.org/abs/1306.4447

[12] Marco Barreno, Blaine Nelson, Anthony D Joseph, and JD Tygar. 2010. The security of machine learning. *Machine Learning* 81, 2 (2010), 121–148.

[13] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. 2006. Can machine learning be secure?. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM, 16–25.

[14] Ling Huang Nina Taft Benjamin I. P. Rubinstein, Peter L. Bartlett. 2009. Learning in a Large Function Space: Privacy-Preserving Mechanisms for SVM Learning. *arXiv preprint arXiv:0911.5708, 2009* (2009).

[15] Battista Biggio, Igino Corona, Blaine Nelson, Benjamin I. P. Rubinstein, Davide Maiorca, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. 2014. *Security Evaluation of Support Vector Machines in Adversarial Environments*. Springer International Publishing, Cham, 105–153. https://doi.org/10.1007/978-3-319-02300-7_4

[16] Battista Biggio, Giorgio Fumera, and Fabio Roli. 2008. Adversarial pattern classification using multiple classifiers and randomisation. *Structural, Syntactic, and Statistical Pattern Recognition* (2008), 500–509.

[17] Battista Biggio, Giorgio Fumera, and Fabio Roli. 2014. Security Evaluation of Pattern Classifiers under Attack. *IEEE Trans. Knowl. Data Eng.* 26, 4 (2014), 984–996. https://doi.org/10.1109/TKDE.2013.57

[18] Battista Biggio, Giorgio Fumera, and Fabio Roli. 2014. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering* 26, 4 (2014), 984–‘–996.

[19] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2011. Support Vector Machines Under Adversarial Label Noise. In *Proceedings of the 3rd Asian Conference on Machine Learning, ACML 2011, Taoyuan, Taiwan, November 13-15, 2011*. 97–112. http://www.jmlr.org/proceedings/papers/v20/biggio11/biggio11.pdf

[20] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning Attacks against Support Vector Machines. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 -*

July 1, 2012.
[21] Corona I. Nelson B. Rubinstein B. I. Maiorca D. Fumera G. Giacinto G. Biggio, B. and F Roli. 2014. Security evaluation of support vector machines in adversarial environments. (2014).
[22] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 39–57.
[23] Igino Corona, Giorgio Giacinto, and Fabio Roli. 2013. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences* 239 (2013), 201–225.
[24] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. 2004. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 99–108.
[25] Hao Chen Dongyu Meng. 2017. MagNet: a Two-Pronged Defense against Adversarial Examples. (2017).
[26] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1322–1333.
[27] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. [n. d.]. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing.
[28] X. Ji T. Zhang T. Zhang G. Zhang, C. Yan and W. Xu. 2017. Dolphinatack: Inaudible voice commands. *arXiv preprint arXiv:1708.09537* (2017).
[29] Polina Golland. 2001. Discriminative Direction for Kernel Classifiers. (2001).
[30] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. *CoRR* abs/1406.2661 (2014). arXiv:1406.2661 http://arxiv.org/abs/1406.2661
[31] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *CoRR* abs/1412.6572 (2014). http://arxiv.org/abs/1412.6572
[32] J. L. Lee H. Bretschneider D. Merico R.K. Yuen H. Y. Xiong, B. Alipanahi and Q. Morris. 2015. The human splicing code reveals new insights into the genetic determinants of disease. 347, no. 6218 (2015).
[33] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015). arXiv:1503.02531 http://arxiv.org/abs/1503.02531
[34] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. *arXiv preprint arXiv:1702.07464* (2017).
[35] Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Blocking Transferability of Adversarial Examples in Black-Box Learning Systems. *CoRR* abs/1703.04318 (2017). arXiv:1703.04318 http://arxiv.org/abs/1703.04318
[36] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167, 2015* (2015).
[37] A. Liaw G. E. Dahl J. Ma, R. P. Sheridan and V. Svetnik. 2015. Deep neural nets as a method for quantitative structure-activity relationships. *Journal of chemical information and modeling* 55, no. 2 (2015), 263–274.
[38] Emmanuel J. CandÃĺs Jian-Feng Cai and Zuowei Shen. 2010. A singular value thresholding algorithm for matrix completion. 20 (2010), 1956 – 1982. Issue 4.
[39] Kaggle. 2014. Higgs boson machine learning challenge. (2014). https://www.kaggle.com/c/higgs-boson,2014
[40] Michael Kearns and Ming Li. 1993. Learning in the Presence of Malicious Errors. *SIAM J. Comput.* 22, 4 (aug 1993), 807–837. https://doi.org/10.1137/0222052
[41] Marius Kloft and Pavel Laskov. 2010. Online Anomaly Detection under Adversarial Impact. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Yee Whye Teh and Mike Titterington (Eds.), Vol. 9. PMLR, Chia Laguna Resort, Sardinia, Italy, 405–412. http://proceedings.mlr.press/v9/kloft10a.html
[42] Atul Kumar and Sameep Mehta. 2017. A Survey on Resilient Machine Learning. *CoRR* abs/1707.03184 (2017). arXiv:1707.03184 http://arxiv.org/abs/1707.03184
[43] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial Machine Learning at Scale. *CoRR* abs/1611.01236 (2016). arXiv:1611.01236 http://arxiv.org/abs/1611.01236
[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.
[45] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data Poisoning Attacks on Factorization-Based Collaborative Filtering. *CoRR* abs/1608.08182 (2016). arXiv:1608.08182 http://arxiv.org/abs/1608.08182
[46] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Jun Zhu, and Xiaolin Hu. 2017. Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser. *arXiv preprint arXiv:1705.09064* (2017).
[47] Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming* 45, 1 (1989), 503–528.

[48] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. 2015. A Unified Gradient Regularization Family for Adversarial Examples. In *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*. 301–309. https://doi.org/10.1109/ICDM.2015.84

[49] S. C. Turaga V. Jain H. S. Seung M. Helmstaedter, K. L. Briggman and W. Denk. 2013. Connectomic reconstruction of the inner plexiform layer in the mouse retina. 500, no. 7461 (2013), 168–147.

[50] Hannun Awni Y Maas, Andrew L and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. 30 (2013).

[51] Shike Mei and Xiaojin Zhu. 2015. Using Machine Teaching to Identify Optimal Training-set Attacks on Machine Learners. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, 2871–2877. http://dl.acm.org/citation.cfm?id=2886521.2886721

[52] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2574–2582.

[53] Olah-Christopher Mordvintsev, Alexander and Mike Tyka. 2015. Inceptionism : Going deeper into neural networks. (2015). https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

[54] T. Vaidya Y. Zhang M. Sherr C. Shields D. Wagner N. Carlini, P. Mishra and W. Zhou. 2016. Hidden voice commands. (2016), 513–530.

[55] Vinod Nair and Geoffrey Hinton. 2010. Rectified linear units improve restricted boltzmann machines. (2010), 807–814.

[56] Nina Narodytska and Shiva Prasad Kasiviswanathan. 2017. Simple Black-Box Adversarial Attacks on Deep Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops, Honolulu, HI, USA, July 21-26, 2017*. 1310–1318. https://doi.org/10.1109/CVPRW.2017.172

[57] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).

[58] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 372–387.

[59] Nicolas Papernot and Patrick D. McDaniel. 2017. Extending Defensive Distillation. *CoRR* abs/1705.05264 (2017). http://arxiv.org/abs/1705.05264

[60] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*. 506–519. https://doi.org/10.1145/3052973.3053009

[61] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. 2016. Towards the Science of Security and Privacy in Machine Learning. *CoRR* abs/1611.03814 (2016). http://arxiv.org/abs/1611.03814

[62] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. 582–597. https://doi.org/10.1109/SP.2016.41

[63] Maya Kabkab Pouya Samangouei and Rama Chellappa. 2018. Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. *arXiv preprint arXiv:1805.06605* (2018).

[64] Praneeth Netrapalli Prateek Jain and Sujay Sanghavi. 2012. Low-rank matrix completion using alternating minimization. *arXiv preprint arXiv:1212.0467, 2012* (2012).

[65] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. 2017. Generic Black-Box End-to-End Attack against RNNs and Other API Calls Based Malware Classifiers. *arXiv preprint arXiv:1707.05970* (2017).

[66] Uri Shaham, Yutaro Yamada, and Sahand Negahban. 2015. Understanding Adversarial Training: Increasing Local Stability of Neural Nets through Robust Optimization. *CoRR* abs/1511.05432 (2015). http://arxiv.org/abs/1511.05432

[67] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 3–18.

[68] Thalles Silva. 2018. An intuitive introduction to Generative Adversarial Networks. (2018). https://medium.freecodecamp.org/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394

[69] Dosovitskiy Alexey Brox Thomas Springenberg, Jost Tobias and Martin Riedmiller. 2014. Striving for Simplicity: The All Convolutional Net. *arXiv preprint arXiv:1412.6806, 2014* (2014).

[70] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *CoRR* abs/1312.6199 (2013). http://arxiv.org/abs/1312.6199

[71] J. de Seixas T. Ciodaro, D. Deva and D. Damazio. 2012. Online particle detection with neural networks based on topological calorimetry information. *Journal of physics: conference series* 238, no. 1 (2012).

[72] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick D. McDaniel. 2017. Ensemble Adversarial Training: Attacks and Defenses. *CoRR* abs/1705.07204 (2017). arXiv:1705.07204 http://arxiv.org/abs/1705.07204

[73] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing Machine Learning Models via Prediction APIs.. In *USENIX Security Symposium*. 601–618.
[74] Yutaro Yamada Ethan Weinberger Alex Cloninger Xiuyuan Cheng Kelly Stanton Uri Shaham, James Garritano and Yuval Kluger. 2018. Defending against Adversarial Images using Basis Functions Transformations. *arXiv preprint arXiv:1803.10840* (2018).
[75] OpenAI: Attacking Machine Learning with Adversarial Examples. 2017.
[76] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. *CoRR* abs/1704.01155 (2017). arXiv:1704.01155 http://arxiv.org/abs/1704.01155
[77] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. *CoRR* abs/1704.01155 (2017). arXiv:1704.01155 http://arxiv.org/abs/1704.01155
[78] Wang Naiyan Chen Tianqi Xu, Bing and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853, 2015* (2015).
[79] Bowei Xi Yan Zhou, Murat Kantarcioglu. 2018. A survey of game theoretic approach for adversarial machine learning. *Wires Data Mining and Knowledge Discovery* (2018).