Received: 4 June 2020 | Revised: 18 October 2020 | Accepted: 22 December 2020

CAAI Transactions on Intelligence Technology

DOI: 10.1049/cit2.12028

REVIEW

A survey on adversarial attacks and defences

Anirban Chakraborty[1] | **Manaar Alam**[1] | **Vishal Dey**[2] | **Anupam Chattopadhyay**[3] | **Debdeep Mukhopadhyay**[1]

[1]Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India

[2]Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio, USA

[3]School of Computer Science and Engineering, Nanyang Technological University, Singapore

Correspondence

Manaar Alam, Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India.
Email: alam.manaar@iitkgp.ac.in

Funding information

Swarnajayanti Fellowship program by Department of Science and Technology, Government of India Cyber Security Research in CPS by the Haldia Petrochemicals Ltd. and TCG Foundation

Abstract

Deep learning has evolved 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. The advancement of deep learning has been so radical that today it can surpass human-level performance. As a consequence, deep learning is being extensively used in most of the recent day-to-day applications. However, efficient deep learning systems can be jeopardised by using crafted adversarial samples, 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. Herein, the authors attempt to provide a detailed discussion on different types of adversarial attacks with various threat models and also elaborate on the efficiency and challenges of recent countermeasures against them.

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 details in large datasets in order to compute the representation of data in each layer from the representation in the previous layer [1]. 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 (DNN) 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 and language translation along with more advanced areas like analysing potential of drug molecules [2], reconstruction of brain circuits [3], analysing particle accelerator data [4,5] and effects of mutations in DNA [6]. Deep learning network, with their unparalleled accuracy, has brought in major revolution in artificial intelligence (AI)-based services on the Internet, including cloud-computing-based AI services from commercial players like Google [7], Alibaba [8], and corresponding platform propositions from Intel [9] and Nvidia [10]. Extensive use of deep-learning-based applications can be seen in safety and security-critical environments like malware detection, self-driving cars, and drones and robotics. With recent advancements in face-recognition systems, ATMs and mobile phones are using biometric authentication as a security feature; voice controllable systems (VCS) and automatic speech recognition (ASR) models made it possible to realise products like Amazon Alexa [11], Apple Siri [12] and Microsoft Cortana [13].

As DNNs 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. [14] first showed that well-performing DNNs can also be a victim of adversarial attack. Carlini et al. [15] and Zhang et al. [16]

independently brought forward the vulnerabilities of ASR and VCS. Attacks on autonomous vehicles have been demonstrated by Kurakin et al. [17] where the adversary manipulated traffic signs to confuse the learning model. The paper by Goodfellow et al. [18] provides a detailed analysis with supportive experiments of adversarial training of linear models, while Papernot et al. [19] addressed the aspect of generalization of adversarial examples. Abadi et al. [20] proposed a method to protect the privacy of training data by introducing the concept of distributed deep learning. Recently, in 2017, Hitaj et al. [21] exploited the real-time nature of the learning models to train a generative adversarial network (GAN) and showed that the privacy of the collaborative systems can be jeopardised. Since the findings of Szegedy et al., a lot of attention has been drawn to the context of adversarial learning and its consequences. A number of countermeasures have been proposed in recent years to mitigate the effects of adversarial attacks. Kurakin et al. [17] proposed the idea of using adversarial training to protect the learner by augmenting the training set using both original and perturbed data. Hinton et al. [22] introduced the concept of distillation which was used to propose a defence mechanism against adversarial attacks [23]. Samangouei et al. [24] proposed a mechanism to use GAN as a countermeasure for adversarial perturbations. Although each of these proposed defense mechanisms was 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 defence 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, a number of studies have shown that these applications are vulnerable to adversarial attacks. Akhtar et al. [25] presented a comprehensive outline and summarized adversarial attacks on DNNs, but in a restrictive context of computer vision. There have been a handful of surveys on security evaluation related to particular machine learning applications [26–29]. Kumar et al. [30] 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 analysing 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.

1.1.1 | Organization

Herein, we discuss recent advancements on adversarial attacks and present a detailed understanding of the attack models and methodologies. While our major focus is on attacks and defences on DNNs, we have also presented attack scenarios on support vector machines (SVM) keeping in mind their extensive use in real-world applications. We first provide a taxonomy of related terms and keywords and categorize the threat models in Section 2. 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 keywords 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:** SVMs are supervised learning models mathematically formulated as optimal hyperplanes for linearly and non-linearly separable hyperspace using kernel functions. They are widely used for classification, regression or outlier detection, representing 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 network (ANN) is a framework based on a collection of perceptrons called *neurons*. The concept of ANN is inspired by the biological neural networks. The objective of each neuron is to map a set of inputs to an 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 back-propagation algorithm. ANNs deal with supervised (convolutional neural network [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, DNN enables feature learning using raw data as input. Multiple hidden layers and its interconnections extract the features from unprocessed input and thus enhance the performance by finding latent structures in unlabelled, unstructured data. A typical DNN architecture, graphically depicted in Figure 1, consists of multiple successive layers (at least two hidden layers) of neurons. Each processing layer can be viewed as learning a different, more abstract representation of the original multidimensional input distribution.
2. **CNN:** A 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 design of the architecture of CNN, shown in Figure 2, is done in such a way so that it can take advantage of two-dimensional (2D) input structures (e.g. image). Convolution layer creates a feature map. A process called pooling (also known as sub-sampling or down-sampling) is deployed to reduce the dimensionality of feature maps. However, it ensures to retain the most important information 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 uses the feature matrix formed from the previous layers to classify the data. CNN is mainly used for feature extraction; thus it also finds application in data pre-processing commonly used in image recognition tasks.

2.2 | Adversarial threat model

The security of any machine learning model is evaluated considering the goals of an adversary and his capabilities in accessing the model. In this section, we taxonomize different adversarial threat models possible keeping in mind the strength of an adversary. We first present the identification of threat surface [31] of various real-life applications which are built using machine learning models to identify how and where an adversary may try to compromise the proper working nature of the system.

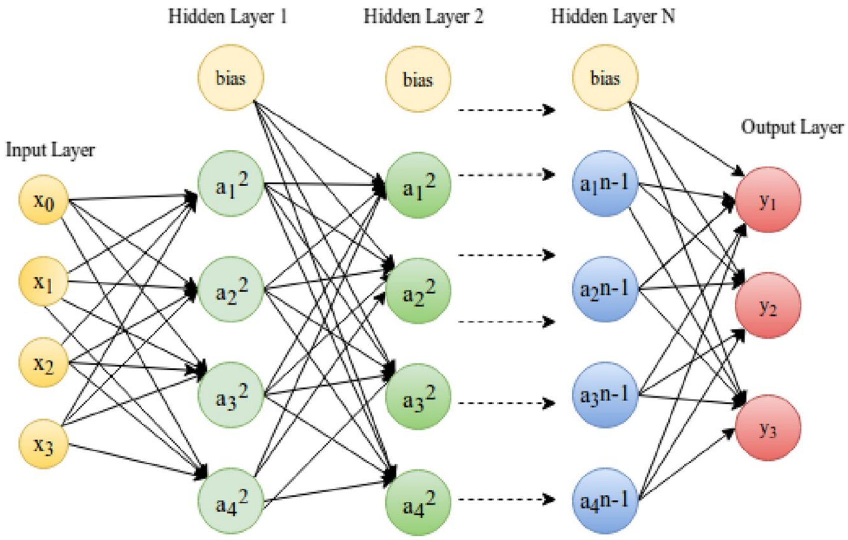


FIGURE 1 Deep neural network

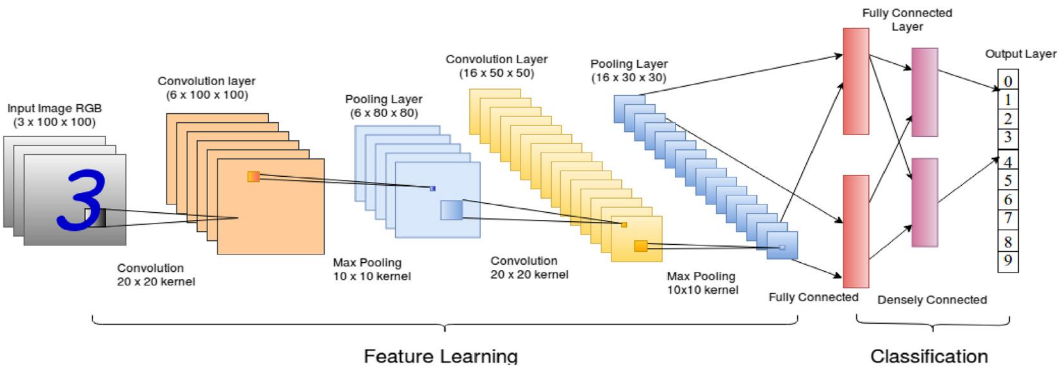

Input Image RGB (3 x 100 x 100)
3
Convolution layer (6 x 100 x 100)
Pooling Layer (6 x 80 x 80)
Convolution Layer (16 x 50 x 50)
Pooling Layer (16 x 30 x 30)
Fully Connected Layer
Output Layer
0
1
2
3
4
5
6
7
8
9
Convolution 20 x 20 kernel
Max Pooling 10 x 10 kernel
Convolution 20 x 20 kernel
Max Pooling 10x10 kernel
Fully Connected
Densely Connected
Feature Learning
Classification

FIGURE 2 Convolutional neural network for MNIST digit recognition

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 be viewed as (a) collection of input data from data repositories or sensors, (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.

The system collects sensor inputs (images using camera) from which model features (tensor of pixel values) are extracted and fed to the models. The model then tries to interpret the meaning of the output (probability that the image is of a stop sign), and takes appropriate action (stopping the car). The *attack surface*, in this case, can be interpreted based on the data processing steps. The objective of an adversary could be to attempt to manipulate either the data *collection* or the data *processing* in order to corrupt the target model, thus tampering the original output. The main attack scenarios identified by the attack surface are sketched as follows [29,32]:

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, is carried out at training phase of the machine learning model. An adversary tries to inject skilfully crafted samples to poison the system in order to compromise the entire learning process.
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 the 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, including the attack vector used 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 attacker is assumed to have much more information and is thus strictly a stronger adversary. We explore the range of attacker capabilities in machine learning systems as they relate to inference and training phases’ [31].

Training phase capabilities

Most of the attacks in the training phase are accomplished by learning, influencing or corrupting the model by direct alteration of the dataset. The attack strategies are broadly classified into the following categories based on the adversarial capabilities:

1. **Data injection:** The adversary neither has access to the training data nor 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 is able to meddle with the learning algorithm. Apparently, it becomes very difficult to design counter strategy against these adversaries who can alter the logic of the learning algorithm, thereby controlling the model itself.

Testing phase capabilities

Adversarial attacks at the testing time do not interfere with the targeted model but rather forces it to produce incorrect outputs. The effectiveness of an attack is determined by the amount of knowledge about the model which is available to the adversary. These attacks can be categorized as *white-box* or *black-box* attack. 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

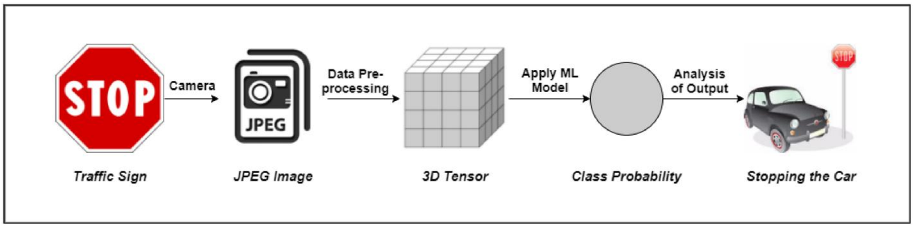


FIGURE 3 Generic pipeline of an automated vehicle system

randomized training procedure *train* having randomness r (e.g. random weight initialization, dropout etc.). The model parameters θ are learnt 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 has been shown in Figure 4

White-box attacks

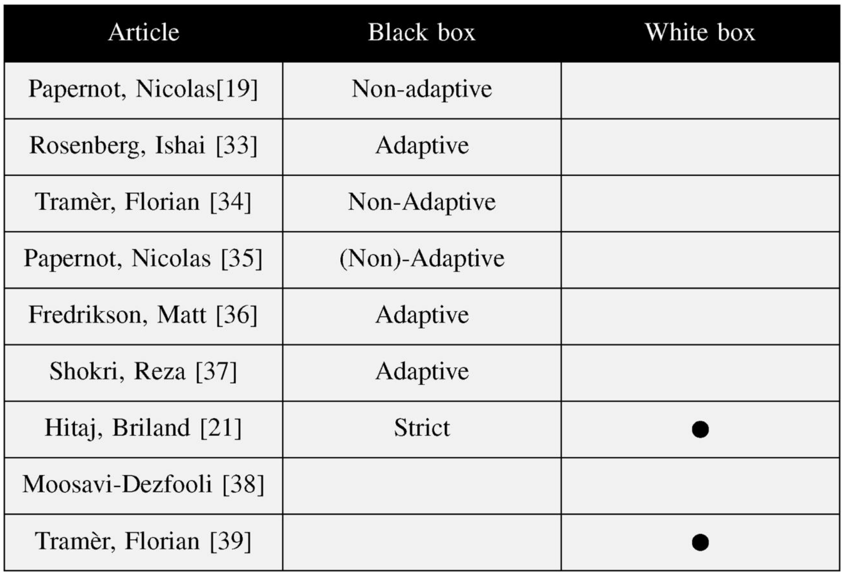
In white-box attack on a machine learning model, an adversary has total knowledge about the model (f) used for classification (e.g. type of neural network along with number of layers). The attacker has information about the algorithm (*train*) used in training (e.g. gradient-descent optimization) and can access the training data distribution (μ). He also knows the parameters (θ) of the fully trained model architecture. The adversary utilizes this information to analyse the feature space where the model might be vulnerable, that is, 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 and prior inputs to exploit the model. 'For example, in an oracle attack, the adversary explores a model by providing a series of carefully crafted inputs and observing outputs' [31]. Black-box attacks are further subdivided into the following categories:

1. **Non-adaptive black-box attack:** For a target model (f), a non-adaptive black-box adversary can only access the 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 misclassifications.
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, that is, 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 cannot 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 black-box attack framework is that an adversary neither tries to learn the randomness r used to train the target model nor the target



Article	Black box	White box
Papernot, Nicolas[19]	Non-adaptive	
Rosenberg, Ishai [33]	Adaptive	
Tramèr, Florian [34]	Non-Adaptive	
Papernot, Nicolas [35]	(Non)-Adaptive	
Fredrikson, Matt [36]	Adaptive	
Shokri, Reza [37]	Adaptive	
Hitaj, Briland [21]	Strict	●
Moosavi-Dezfooli [38]		
Tramèr, Florian [39]		●

FIGURE 4 Overview of threat models in relevant articles [19,21,33–39]

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.

The adversarial threat model 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 such that it produces an incorrect output. 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:

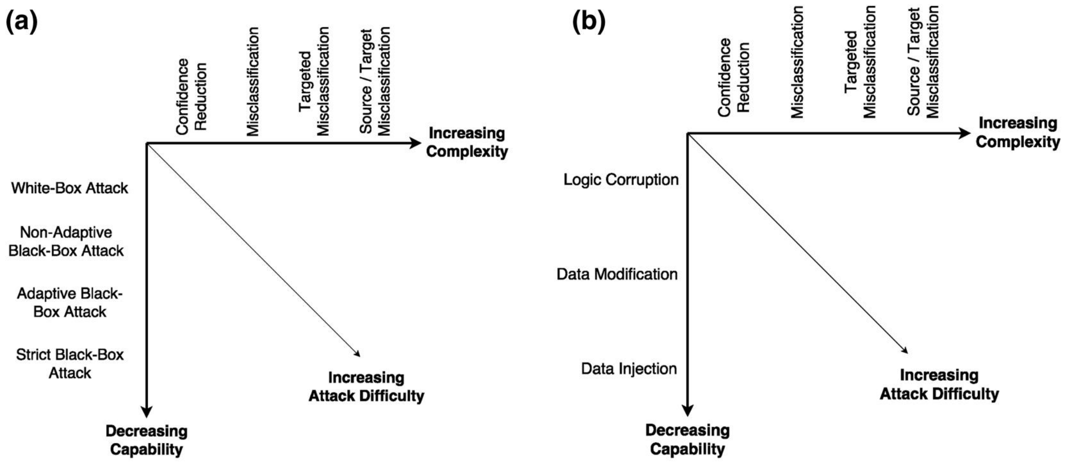
1. **Confidence reduction:** The adversary tries to reduce the confidence of prediction for the target model. For example, a legitimate image of a 'stop' sign can be predicted with a lower confidence having a lesser probability of class belongingness.
2. **Mis-classification:** The adversary tries to alter the output classification of an input example to some other 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 mis-classification:** The adversary tries to craft the inputs in such a way that the model produces the output of a particular 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 mis-classification:** The adversary tries to classify a particular input source to a predefined 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 is represented graphically in 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. The overall architecture of the attack threat model has been categorized in Table 3. Further in Table 4, we categorize those attacks under

TABLE 1 Distinction between black-box and white-box attacks

Description	Black-box attack	White-box attack
Adversary knowledge	Restricted knowledge: capable of only observing the output for some probed inputs.	In-depth knowledge about the underlying architecture and model parameters.
Attack strategy	Based on a greedy search generating, an implicit approximation to the actual gradient with respect to the output by observing changes in the corresponding input.	Based on the gradient of the loss function with respect to the input data.



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

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

FIGURE 5 Taxonomy of adversarial model for (a) evasion attacks and (b) poisoning attacks with respect to adversarial capabilities and goals

TABLE 2 Overview of attacks and applications

Articles	Attacks	Applications
Fredrikson et al. [36]	Model inversion	Biomedical imaging, biometric identification
Tramèr et al. [34]	Extraction of target machine learning models using APIs	Attacks extend to multiclass classifications and neural networks
Anteniese et al. [40]	Meta-classifier to hack other classifiers	Speech recognition
Biggio et al. [41,42]	Poisoning-based attacks:	Crafted training data for support vector machines
Dalvi et al. [43]	Adversarial classification, pattern recognition	Email spam detection, fraud detection, intrusion detection, biometric identification
Biggio et al. [44,45]		
Papernot et al. [19,35]	Adversarial samples crafting, adversarial sample transferability	Digit recognition, black-box attacks against classifiers
Hitaj et al. [21]	GAN under collaborative learning	Classification
Goodfellow et al. [46]	Generative adversarial network	Classifiers, malware detection
Shokri et al. [37]	Membership inference attack	Attack on classifiers trained on commercial “ML as a service” platforms
Moosavi et al. [38]	Adversarial perturbations:Sample generation: Poisoning-based attack	Image classificationIntrusion detectionCollaborative filtering systems
Carlini et al. [47]		
Li et al. [48]		

TABLE 3 Classification of different attacks based on the attack threat model

Adversary assumptions		Details		
Attack threat models	Attack surface	Evasion attacks		
		Poisoning attacks		
		Exploratory attacks		
	Adversarial capabilities	Training phase	Data injection	
			Data modification	
			Logic corruption	
		Testing phase	White-box attack	
			Black-box attack	Non-adaptive
				Adaptive
				Strict
	Adversarial goals	Confidence reduction		
		Mis-classification		
		Targeted mis-classification		
		Source/target mis-classification		

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 try 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 the testing phase.

3.1 | Model inversion attack

Fredrikson et al. introduced ‘model inversion’ (MI) in [49] where they used a linear regression model f for predicting drug dosage using patient information, medical history and genetic markers; explored the model as a white box and an instance of data $(\mathbf{X} = \{x_1, x_2, \ldots, x_n\}, y)$, and try to infer genetic marker x_1. The algorithm produces ‘least-biased maximum a posteriori (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

TABLE 4 Attack summary

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

rate. It has serious limitations; for example, it cannot handle larger set of unknown features since it is computationally not feasible.

Fredrikson et al. [36] intended to remove limitations of their previous work and showed that patient's genetic markers can be predicted by an attacker in a black-box environment. This new MI attack through Machine Learning (ML) APIs that exploit confidence values in a variety of settings and explored countermeasures in both black-box and white-box settings. This attack has been successfully experimented in face recognition using neural network models: multilayer perceptron (MLP), softmax regression and stacked denoising autoencoder network (DAE); given access to the model and person's name, it can recover the facial image. Figure 6 illustrates the reconstruction produced by the three algorithms. Due to the convoluted structure of DNN model, MI attacks cannot retrieve more than some prototypical samples which might not have great similarity with the original data that was used to define that class.

3.2 | Model extraction using APIs

Tramèr et al. [34] presented simple attack scenario using which an adversary can extract information about target machine learning models such as decision trees, regression and neural networks. These attacks are strict black-box attacks which could develop local models having close functionality to target model. The authors demonstrated model extraction attack on online ML service providers such as Amazon Machine Learning and BigM. Machine learning APIs provided by ML-as-service providers return probability values along with class labels. Since the attacker does 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.

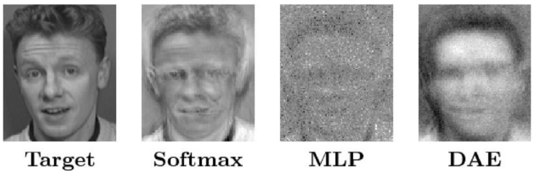


FIGURE 6 Reconstruction of the individual on the left by Softmax, multilayer perceptron and denoising autoencoder network (Image credit: Fredrikson et al. [36])

3.3 | Inference attack

Ateniese et al. [40] elaborated on gathering 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. [37] is membership inference attack, which determines if a 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, the attacker uses a datapoint to query the target model and obtains its output which is a vector of probabilities that specifies whether the data point belongs to a certain class. For training the attack model, a set of shadow models is built. Since the adversary has the knowledge of whether a given record belongs to the training set or not, supervised learning can be employed and the corresponding output labels are then fed to attack model to train it to distinguish shadow model's outputs.

Figure 7 illustrates the end-to-end attack process. The output vectors obtained from shadow model are added to the training dataset of the attack model. The shadow model is also queried using a test dataset and added to the training dataset of the attack model. Thus a collection of target attack models is trained by utilizing the black-box behaviour of the shadow models. The authors used membership attacks on classification models trained by commercial players like Amazon and Google who provide 'ML as a service'.

4 | EVASION AND 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 the model is trained on contaminated inputs to obtain the desired output.

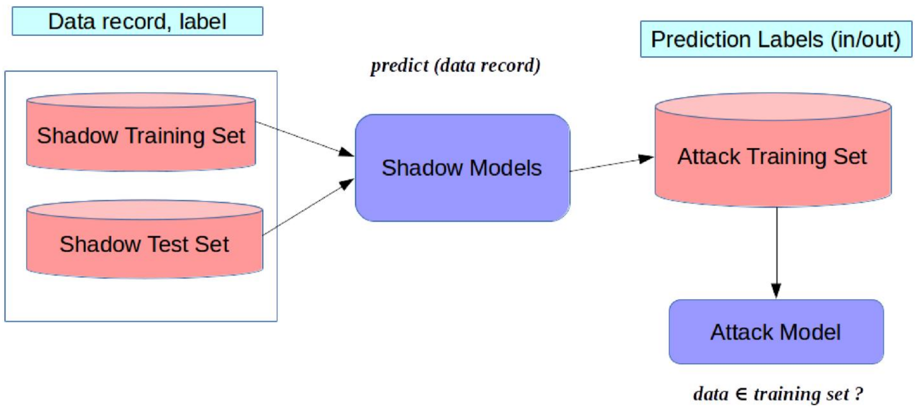


FIGURE 7 Overview of membership inference attack

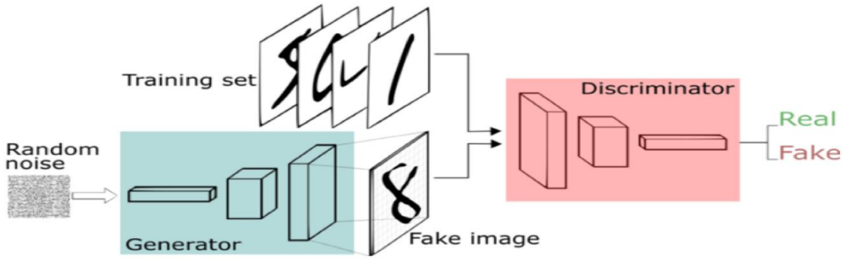


FIGURE 8 Generative adversarial learning (Image credit: Thalles Silva [50])

4.1 | Generative adversarial attack

Goodfellow et al. portrayed GAN [46] as a game between two learning networks: one generates samples similar to the training set, having almost identical distribution, and the other discriminates between original and crafted training sample. The GAN procedure, as depicted in Figure 8, is composed of a DNN G, known as Generator and another DNN D, known as 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 original training samples 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. [46]) 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 and $p_z(z)$ is a prior on input noise [46]. The objective is to train D such that it maximizes the probability to assign correct labels to sample examples, while simultaneously training G to minimize it. It was found that optimizing D was computationally intensive and on finite datasets, there was a chance of overfitting. Moreover, during the initial stages of learning for G, D can discriminate crafted samples by G with relatively high confidence and rejects them. So, instead of training G to minimize $\log(1 - D(G(z)))$, they trained G to maximize $\log(D(G(z)))$.

Radford et al. [51] introduced deep convolutional generative adversarial networks (DCGANs) which overcome these constraints in GANs and make them stable to train. Some of the key insights of DCGAN architecture were as follows:

- The overall network architecture was based on all convolutional net [52] replacing deterministic pooling functions with strided convolutions.
- A random noise z was introduced as input to the first layer. The result thus obtained was transformed into a 4D tensor. The last layer was transformed into a vector and fed into a single sigmoid output (Figure 9). [53]
- They used batch normalization [54] to stabilize the learning. They normalized the input to achieve zero mean and unit variance. This solved the problem of instability of GAN during training which arise due to poor initialization.
- For generator, ReLU activation [55] was used in all layers for the output layer, whereas Leaky ReLU activation [56,57] was used for all layers in discriminator.

4.2 | Adversarial examples generation

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

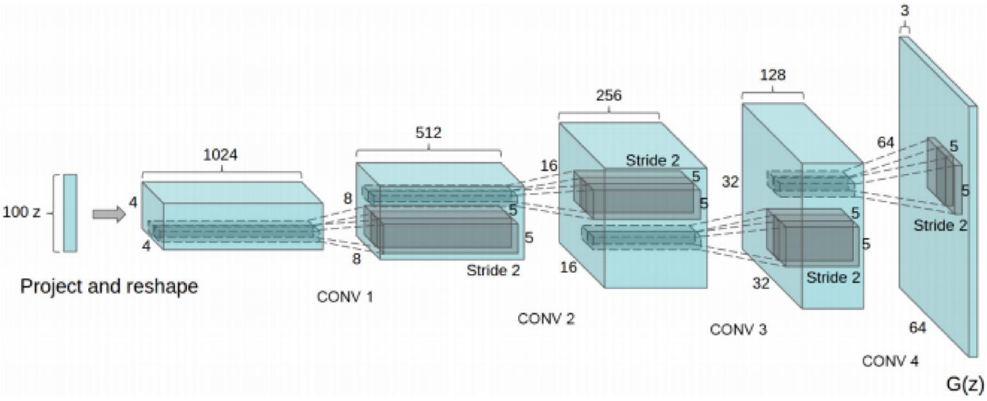


FIGURE 9 Deep convolutional generative adversarial network generator used for LSUN scene modelling (Image credit: Radford et al. [51])

4.2.1 | Training phase modification

A learning process fine-tunes the parameters θ of the hypothesis h by analysing a training set. This makes the training set susceptible to manipulation by adversaries. Barreno et al. [26] first coined the term *Poisoning Attacks*; training dataset is manipulated ('poisoned') with the intention of altering decision boundary characteristics of the target model following the work of Kearns et al. [58], 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 do perturbations of the labels randomly, that is, selecting new labels drawn from a random distribution for a subset of training data. It was shown that a random flip of 40% of the training labels can significantly reduce the performance of the SVM classifiers [41].
- **Input manipulation:** In this scenario, the adversary is more powerful and can modify both the labels and input features of the training data with knowledge of the learning algorithm to effectively train the model on poisoned data.

Kloft et al. [59] presented a study in which they showed that the decision boundary of an anomaly detection classifier could be gradually shifted by inserting malicious points in the training dataset. The learning algorithm they used works in an online fashion. The parameter values of the learning algorithm are fine-tuned based on a sample of that training data which are collected at regular intervals in online learning scenario. 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 with the objective of maximizing displacement of the training data's mean.

In offline learning scenario, Biggio et al. [42] presented an attack scenario in which an adversary uses gradient ascent algorithm to craft inputs into training dataset that corresponds to local maxima error. The study shows that insertion of these crafted inputs into the training set degrades inference accuracy for SVM classifier on test data. Following their approach, a generic framework for poisoning was proposed [60] as an optimization problem to find optimal change to the training set sufficient for adversarial classification if the loss function used in targeted learning model is convex and it's 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. [23] 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. 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 selectively add a perturbation δX with the sample X, thereby generating a malicious example $X_* = X + \delta X$, 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, an adversary can access parameters θ_F of the target 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 and estimates the variability of output on input feature

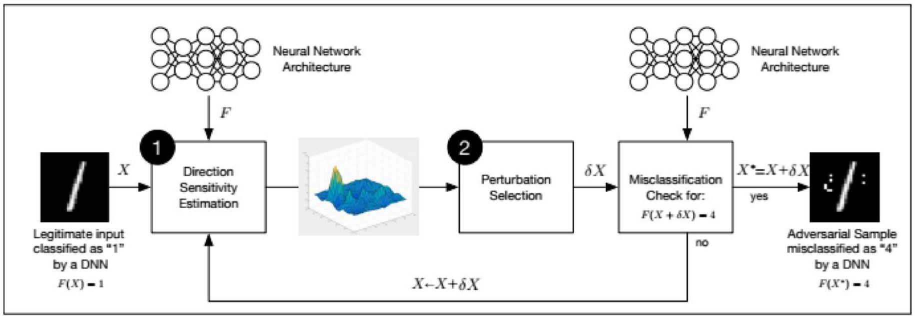

Neural Network Architecture
F
Neural Network Architecture
F
1
2
X
Direction Sensitivity Estimation
Perturbation Selection
δX
Misclassification Check for: $F(X+\delta X)=4$
$X^*=X+\delta X$
yes
no
Legitimate input classified as "1" by a DNN
$F(X)=1$
Adversarial Sample misclassified as "4" by a DNN
$F(X^*)=4$
$X \leftarrow X+\delta X$

FIGURE 10 Adversarial example crafting framework for evasion attacks (Image credit: Papernot et al. [23])

vectors on how a model responds to change in input feature by identifying directions in input space around vicinity of sample X.

2. **Perturbation selection:** The adversary then exploits the knowledge of sensitive information to choose a perturbation δX in order to obtain an adversarial perturbation which is most efficient.

The adversary may repeat both the steps by substituting X by $X + \delta X$ at each iteration until his goal is satisfied such that the sum of perturbation used for crafting X_* from a valid example needs to be as minimum as possible. This helps the adversarial samples to remain imperceptible to human eye.

It becomes insignificant to use large perturbations to craft adversarial samples. Thus, it is better to define appropriate norm $\|.\|$ to denote the difference between two input points, so as to formalize adversarial samples 'as a solution to the following optimization problem' [23]:

$$X_* = X + \underset{\delta X}{\arg\min}\{\|\delta X\| : F(X+\delta X) \neq F(X)\} \qquad (1)$$

In most cases, it becomes hard to find a closed solution to the above problem because of its non-linear and non-convex nature. The authors have described different techniques to devise an attack approximating the solution using both the steps.

Direction sensitivity estimation

In this step, the adversary takes a sample X, an n-dimensional input vector, and tries to find out the dimensions for which the model behaves as expected by the adversary, with as little perturbation as possible. This can be achieved by changing the input components of X and evaluating the sensitivity of the model for these changes. There are a number of techniques to evaluate the sensitivity of a model, some of which are discussed below.

1. **L-BFGS:** In 2014, Szegedy et al. [14] were the first to formalize the following minimization problem as the search for adversarial examples.

$$\underset{r}{\arg\min} f(X+\delta X) = l \text{s.t.} (X+\delta X) \in D \qquad (2)$$

Formally, Equation (2) finds δX such that the input example X although accurately labelled by f, if perturbed with δX results adversarial example $X_* = X + \delta X$ and X_* still belongs to the input domain D, it is assigned the target label $l \neq h(X)$. For non-convex models like DNN, the authors used the *L-BFGS* [61] optimization to solve Equation (2). 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. [18]. They proposed an FGSM which calculates perturbation to input data in a single step using the gradient of the loss (cost) function with respect to the input data with the objective to maximize the loss. The adversarial examples are produced 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 has used some other variations of FGSM, which are summarized as follows:

(a) **Target class method:** This variant of FGSM [17] 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 [17]. This method generates adversarial samples iteratively using small step size:

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

Here, α is the step size and $Clip_{X,\epsilon}\{A\}$ 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. [62] introduced a different approach for finding sensitivity direction by using forward derivative, by using Jacobian of the trained model. This method provides gradients of the output features corresponding to each input feature. 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 misclassification attacks.

Perturbation selection

After gaining knowledge about the network sensitivity, the adversary will try to determine the dimensions for which the target model will generate misclassification with smallest of perturbations. The perturbed input dimensions can be of two types:

1. **Perturb all the input dimensions:** Goodfellow et al. [18] proposed using FGSM to compute cost function gradient with respect to input dimenisons and perturb every input dimensions but with a minuscule amount in the direction of the sign of the gradient. FGSM method efficiently minimizes the loss computed as distance between the adversarial and the original training samples.
2. **Perturb a selected input dimensions:** Papernot et al. [62] used saliency map to perturb a finite number of input dimension. 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. Given all the dimensions sorted in decreasing order of adversarial saliency value, the perturbations can be crafted by choosing input dimension in order. 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. In summary, the input components i having high saliency value $S(x, t)[i]$ will lead to misclassification of the samples into the target class.

Each method has its own advantages and drawbacks. The first method can generate adversarial samples relatively faster but with high perturbations making them easier to detect. The second method actually reduces the perturbations but it is computationally intensive.

- **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 deceive the target model because of the "Transferability Property" [19] (will be discussed later). However, in an adaptive black-box setting, the adversary cannot access 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 methods of dataset augmentation presented by Papernot et al. [35] is discussed next.

Jacobian-based data augmentation

Although an adversary can make unlimited queries to an Oracle to obtain its output for any given input, the process is not amenable considering the continuous domain of an input to be queried, and the system might also detect the abnormal behaviour of the adversary. An alternative can be to heuristically generate adversarial inputs in the direction along which the model's output is varying, given the original training set. With more input–output pairs, the direction can be captured easily for a target Oracle O. Hence, the greedy heuristic that an adversary follows is to prioritize the samples while querying the oracle for labels to get a substitute DNN F having decision boundary identical to that of the Oracle. The decision boundary can be determined by evaluating the sign the Jacobian matrix J_f of the substitute DNN: $sgn(J_F(x)[O(x)])$, where x is the input label. The original datapoint x is augmented with the term $\lambda^* sgn(J_F(x)[O(x)])$ to generate the 'new synthetic training point' [35]. The iterative data augmentation technique can be summarized using the following equation:

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

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

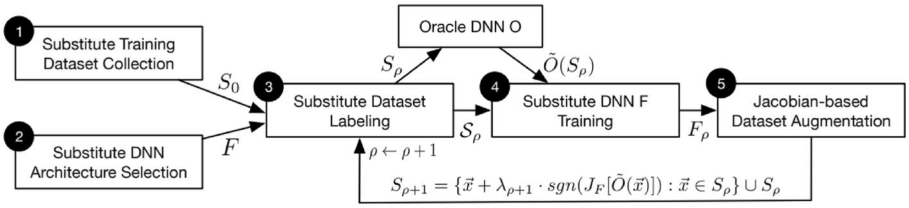

1 Substitute Training Dataset Collection
2 Substitute DNN Architecture Selection
S_0
F
3 Substitute Dataset Labeling
Oracle DNN O
S_ρ
$\tilde{O}(S_\rho)$
4 Substitute DNN F Training
S_ρ
F_ρ
5 Jacobian-based Dataset Augmentation
$\rho \leftarrow \rho + 1$
$S_{\rho+1} = \{\vec{x} + \lambda_{\rho+1} \cdot sgn(J_F[\tilde{O}(\vec{x})]) : \vec{x} \in S_\rho\} \cup S_\rho$

FIGURE 11 Substitute mode training using Jacobian-based data augmentation (Image credit: Papernot et al. [35])

4.2.3 | Transferability of adversarial samples

Adversarial samples have a property where the samples produced to deceive a particular model can be used to deceive other models, irrespective of their underlying architectures—leading to adaptive black-box attacks as discussed in Section 2.2.2. This property is known as 'Transferability of Adversarial Samples' [19]. Since in the case of black-box attack, an adversary cannot access the target model F, he 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:** Both the models F and F' are trained using similar 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, one is an NN and the other one SVM.

This substitute model in effect transfers knowledge-crafting adversarial inputs to the victim model. Owing to the limited capability of the adversary to query the target model for outputs given a number of datapoints, the model can be considered as an oracle. To train the substitute model, dataset augmentation as discussed above is used to capture more information about the predicted outputs. The authors also introduced 'reservoir sampling to select a limited number of new inputs K when performing Jacobian-based dataset augmentation' [19]. It reduces the number of queries made to the oracle. However, the choice of substitution model architecture does not affect the transferability property.

The attacks have been shown to generalize to non-differentiable target models, for example, logistic regression (LR) and DNNs (differentiable models) 'could both effectively be used to learn a substitute model for many classifiers' trained with a SVM, decision tree, DNN and nearest neighbour. The amount of knowledge of the classifier required by the adversary to force a mis-classification of skilfully crafted inputs is greatly reduced by the cross-technique transferability. 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. [21] presented a GAN-based attack to extract information during training phase from 'honest victims in a collaborative deep learning framework'. GAN tends to produce samples identical to those in the training set without having access to the original training set. In a typical white-box setting, the motive of the adversary is to extract tangible information about the labels that do not belong to his own dataset without compromising privacy.

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 x and thus tries to collect maximum information possible about the class x. A uses GAN to produce instances that look similar to class x and injects them into the classifier as class z. V finds it hard to differentiate between the two classes (x and z) and as a result reveals a lot of information about class x. Thus by mimicking the samples, the adversary gains additional information about the classes with the help of the victim himself.

The authors [21] proved that GAN attack was successful in all sets of experiments conducted juxtaposed with MI attacks, DP-based collaborative learning, and so on. 'The GAN will generate good samples as long as the discriminator is learning'.

4.4 | Adversarial classification

Dalvi et al. [43] defined adversarial classification from cost-sensitive game-theoretical perspective as a game between two players: Model (classifier) and an Adversary, each competing to maximize its utility. The classifier tries to learn a function $y_c = C(x)$ from the training set which predicts the instances in the training set accurately. The adversary modifies the instances in the training data from x to $x' = A(x)$ and tries to force the classifier to misclassify the instances in the training data. The classifier and the adversary constantly try to defeat each other by maximizing their own pay-offs. 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

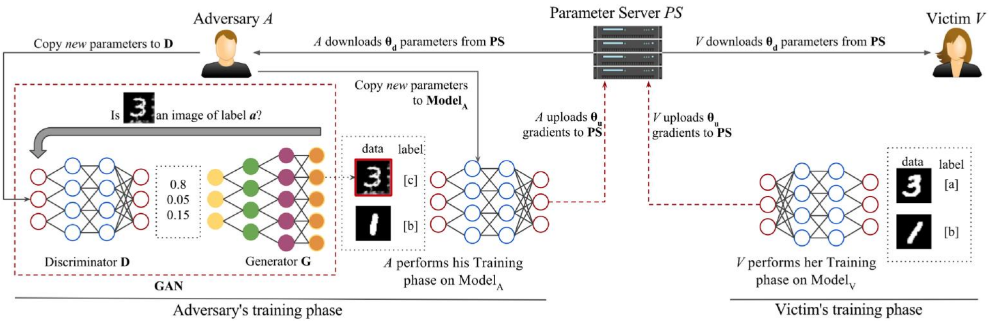

Adversary *A*
Parameter Server *PS*
Victim *V*
Copy *new* parameters to **D**
A downloads θ_d parameters from **PS**
V downloads θ_d parameters from **PS**
Copy *new* parameters to **Model**$_A$
Is 3 an image of label ***a***?
A uploads θ_u gradients to **PS**
V uploads θ_u gradients to **PS**
0.8
0.05
0.15
data
label
[c]
[b]
[a]
[b]
Discriminator **D**
Generator **G**
GAN
A performs his Training phase on Model$_A$
V performs her Training phase on Model$_V$
Adversary's training phase
Victim's training phase

FIGURE 12 Generative adversarial network attack on collaborative deep learning (Image credit: Hitaj et al. [21])

adversary tries to modify feature in order to minimize its own expected cost [27,63]. The presence of adaptive adversaries in a system can significantly degrade the performance of the classifier, especially when the presence and type of adversary are unknown to the classifier. The objective of the classifier is to maximize its expected utility (U_C), whereas the adversary would try to find an optimal feature modification strategy in order to maximize its expected utility (U_A):

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

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

4.5 | Evasion and poisoning attack on support vector machines

SVMs are popular classifiers for detecting malwares and intrusion in a system. Owing to the *stationarity* property in SVM, that is, the same distibution should be used to sample both the training and test samples. However, in adversarial learning, an intelligent and adaptive adversary can modify data and thereby violates the stationarity in order to exploit vulnerabilities present in the learning system.

Biggio et al. [64], using gradient-descent based approach, demonstrated the evasion attack on kernel-based classifiers [65]. Even if the adversary does not have any knowledge on the classifier's decision function, it can evade the classifier by learning a *surrogate* classifier. The authors considered two settings where the adversary had complete or no knowledge about feature space and classifier's discriminant function; the adversary can 'learn a *surrogate* classifier on a surrogate training set'. SVMs can be deceived with relatively high probability even if there is a little possibility that an adversary can imitate the actual model with the surrogate data. Formally, the optimal strategy to find a sample x can be stated as an optimization problem with the objective of minimizing the value of classifier's discriminant function $g(x)$ as in Equation (3):

$$x^* = \arg\min_x \widehat{g}(x) \qquad s.t \quad d(x, x^0) \leq d_{max} \tag{3}$$

However, if $\hat{g}(x)$ is non-convex, the gradient descent algorithm may lead to a local minimum outside of sample's support. To overcome this problem, additional components were introduced into Equation (4):

$$\arg\min_x E(x) = \widehat{g}(x) - \frac{\lambda}{n}\sum\nolimits_{i|f_i=-1} k\left(\frac{x - x_i}{h}\right) \; s.t \; d(x, x^0) \leq d_{max} \tag{4}$$

Biggio et al. [42] also demonstrated, in the form of a poisoning attack, that an adversary can force the SVM to result in a mis-classified output on test samples by manipulating training data. Although the training data was tampered to include well-crafted attack samples, it was assumed that there was no manipulation of test data. In order the attack to happen, the adversary intends to find a set of points and adding those to the training dataset which decreases SVM's classification accuracy at most. It is difficult in real-world scenarios to have a complete knowledge of the training dataset, but obtaining a *surrogate* dataset having a distribution which is the same as the actual training dataset may not be difficult for the attacker. Under these assumptions, the optimal attack strategy is as follows:

$$\begin{aligned} x^* = \arg\max_x P(x) &= \sum\nolimits_{k=1}^{m} \left(1 - y_k f_x(x_k)\right)_+ \\ &= \sum\nolimits_{k=1}^{m} \left(-g_k\right)_+ \; s.t \; x_{lb} \leq x \leq x_{ub} \end{aligned}$$

Another class of attacks on SVM, called privacy attack, presented by Rubinstein et al [66], is based on the attempt to breach the confidentiality of training data. The ultimate goal of the adversary is to obtain features and labels of each training instance individually by trying to inspect classification results obtained from the classifier for test data or by directly inspecting the classifier.

4.6 | Poisoning attacks on collaborative systems

Recommendation and collaborating filtering systems play a crucial role in the business strategies of modern e-commerce systems. Bo Li et al. [48] demonstrated poisoning attacks on collaborative filtering systems where an attacker can generate malicious data to degrade the effectiveness of the system if he has the complete knowledge of the learner. The two most popular algorithms used for factorization-based collaborative filtering are alternating minimization [67] and nuclear norm minimization [68]. The data matrix M for the former one can be determined by solving the optimization problem as mentioned below:

$$\min_{U\epsilon\mathbb{R}^{m\times k},V\epsilon\mathbb{R}^{n\times k}}\left\{\left\|\mathscr{R}_{\Omega}(M-UV^{T})\right\|_{F}^{2}+2\lambda_{U}\|U\|_{F}^{2}+2\lambda_{V}\|V\|_{F}^{2}\right\}$$

Alternatively, the latter one can be solved as follows:

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

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

Based on these problems, Bo Li et al. [48] listed the three types of attacks and their utility functions:

- **Availability attack:** In this attack model, the attacker tries to maximize error obtained from the collaborative filtering system thereby making it unreliable and useless. The utility function can be characterized as a quantity of perturbations of predicted value between $\overline{M}$ (prediction without data poisoning) and $\widehat{M}$ (prediction after poisoning) on unseen entries Ω^{C}

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

- **Integrity attack:** The objective of the adversary is 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-defined weight vector, then we define the utility function 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 combination 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 provide 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. [59] analysed the behaviour of 'online centroid anomaly detection systems' as data poisoning attack. To detect anomaly for a test example x and a 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, as every new data point arrives, the centre of mass changes by

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

As illustrated in Figure 13, with prior knowledge of algorithm and training dataset, the adversary will try to force the anomaly detection algorithm to accept an attack point A which lies outside the solid circle, that is, $\|A - c\| > r$.

5 | ADVANCES 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. In this section, we will briefly discuss about the recent advancements and the challenges. These adversarial examples are difficult to defend because of the following reasons [69]:

1. *A theoretical model of the adversarial example crafting process is very difficult to construct.* The adversarial sample generation is a complex optimization process due to its

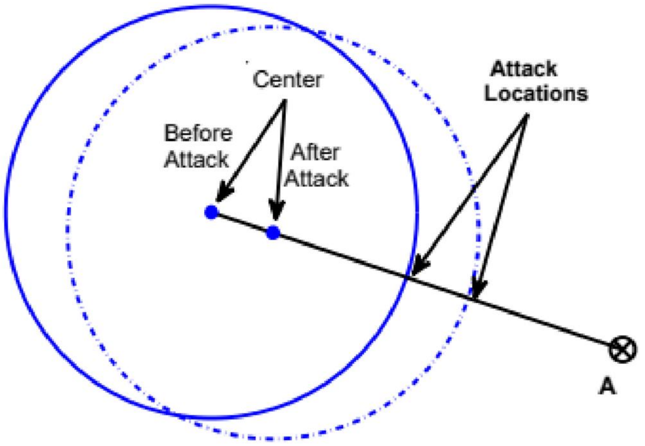


FIGURE 13 Illustration of poisoning attack (Image credit: Kloft et al. [59])

non-linearity and non-convex properties for most machine learning models. The lack of proper theoretical tools to describe the solution to these complex optimization problems makes 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.

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

5.1 | Adversarial training

The primary objective of the adversarial training is to increase model robustness by injecting adversarial examples into the training set [14,18,70,71]. 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 legitimate data and the crafted data, presented in [17] or by learning with a modified objective function with J being the original loss function, given by [18]:

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

The model will predict the same class for both the legitimate as well as perturbed examples in the same direction thereby increasing the model robustness. Traditionally, the additional instances are crafted using one or multiple attack methodologies mentioned in Section 4.2.2.

Adversarial training implies training on adversarial samples which are crafted on the original model. The defense is not robust for black-box attacks [35,72] where an adversary generates malicious examples on a locally trained substitute model. Moreover, Tramèr et al. [39] 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 traditional attack is performed on it, as mentioned in Section 4.2.2.

5.2 | Gradient hiding

A natural defense against gradient-based attacks presented in [39] 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 neighbour classifier or a random forest), gradient-based attacks are rendered ineffective. However, this defense is easily fooled by learning a surrogate black-box model having gradient and crafting examples using it [35].

5.3 | Defensive distillation

Papernot et al. [23,73] showed that distillation [22] can be used as a adversarial training technique so that the model is less susceptible to adversarial inputs.

The two-step working model and the distillation method are described as below. A neural network F is trained to classify input samples X to 'hard labels' and 'soft labels' Y, that is, the final softmax layer produces a probability distribution over Y. Then, 'soft labels' output of F is fed as input to second neural network F' with the same architecture as F on the same dataset X to achieve the same accuracy. The second 'distilled' model makes the output more smooth and robust to adversarial perturbations.

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. [23] showed experimentally that a high empirical value of T gives a better distillation performance. The reason for 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.

Defensive distillation hence smoothens the model, *label smoothing* which converts class labels into soft targets. Target class label is assigned a value close to 1 while the remaining weights are distributed among other classes. These modified values are then used for training in place of true labels. As a result, training an additional model for 'defensive distillation' is no longer required. 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 [35,47]. 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 [74]. The main idea behind this defense is that it reduces the complexity of representing the data which makes adversarial perturbations to disappear because of low sensitivity. There are mainly two heuristics behind the approach considering an image dataset:

1. Reduce the colour depth on a pixel level, that is, use fewer values to encode the colours.
2. Use of a smoothing filter over the images to map multiple inputs into same value. This makes the model resistant against noise and adversarial attacks.

Although these techniques provide a strong countermeasure against adversarial attacks, they are known to significantly worsen the accuracy of the model.

5.5 | Blocking the transferability

The main reason behind the defeat of most of the well-known defense mechanisms is due to the strong transferability property in the neural networks, that is, adversarial examples generated on one classifier are expected to cause another classifier to perform the same mistake. The transferability property holds true irrespective of the underlying architecture of the classifiers or the datasets they were trained on. Hence, the key for protecting against a black-box attack is to block the transferability of the adversarial examples.

Hosseini et al. [75] recently proposed a three-step *NULL labelling* 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 labelling method assigns a higher probability to the NULL label with higher perturbation, while the original labelling without the defense increases the probabilities of other labels.

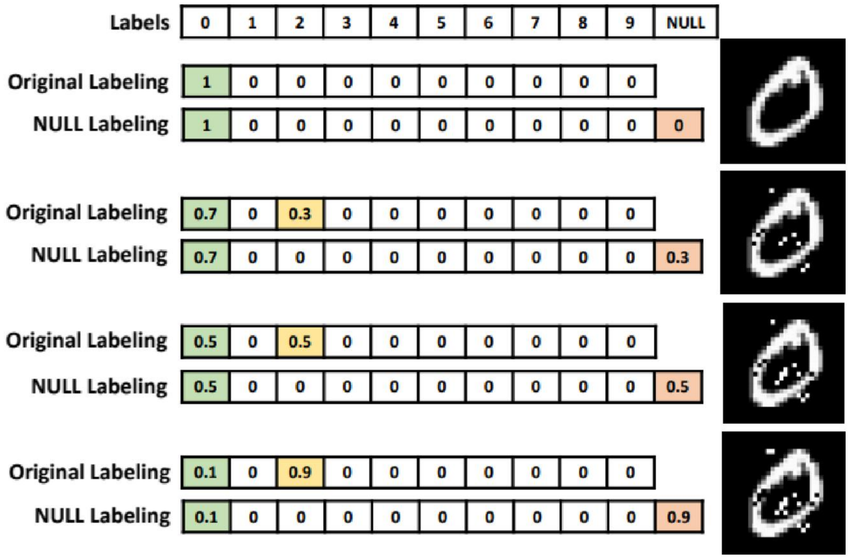
The NULL labelling method is composed of three major steps:

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.



Labels	0	1	2	3	4	5	6	7	8	9	NULL
Original Labeling	1	0	0	0	0	0	0	0	0	0	
NULL Labeling	1	0	0	0	0	0	0	0	0	0	0
Original Labeling	0.7	0	0.3	0	0	0	0	0	0	0	
NULL Labeling	0.7	0	0	0	0	0	0	0	0	0	0.3
Original Labeling	0.5	0	0.5	0	0	0	0	0	0	0	
NULL Labeling	0.5	0	0	0	0	0	0	0	0	0	0.5
Original Labeling	0.1	0	0.9	0	0	0	0	0	0	0	
NULL Labeling	0.1	0	0	0	0	0	0	0	0	0	0.9

FIGURE 14 Illustration of NULL labelling method (Image credit: Hosseini et al. [75])

The advantage of this method is the labelling of the perturbed inputs to NULL label instead of classifying them into their original label. This method can be regarded as the most effective defense mechanism against adversarial attacks till 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. [24] proposed a mechanism to leverage the power of GAN [46] 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, the training of GAN can be challenging, and if not properly trained, the performance of the Defense-GAN can significantly degrade.

5.7 | MagNet

Meng et al. [76] proposed a framework, named MagNet, which uses classifier as a black box to read the output of the classifier's last layer only without modifying the classifier and uses *detectors* to differentiate a normal and adversarial example. The detector checks if the distance between the given test example and the manifold exceeds a threshold. It also uses a *reformer* to reform adversarial example to a similar legitimate example using autoencoders (Figure 16). 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 HGD

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 removing noise from input samples. It uses a loss function which compares the output produced by unperturbed image of target model and denoised image. Liao et al. [77] 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 lth layer representation of the neural network, activated by x and $\hat{x}$:

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

The authors [77] proposed three training methods for high-level denoisers, illustrated in Figure 17. In feature-guided denoiser (FGD), the activations of the last layer of CNN are

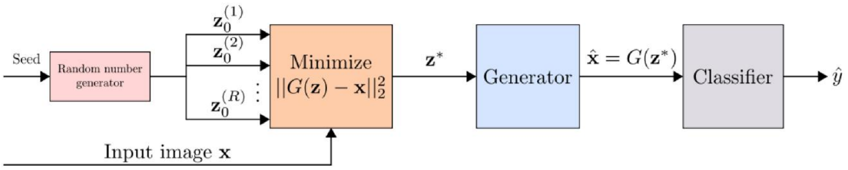


FIGURE 15 Overview of Defense-GAN algorithm (Image Credit: Samangouei et al. [24])

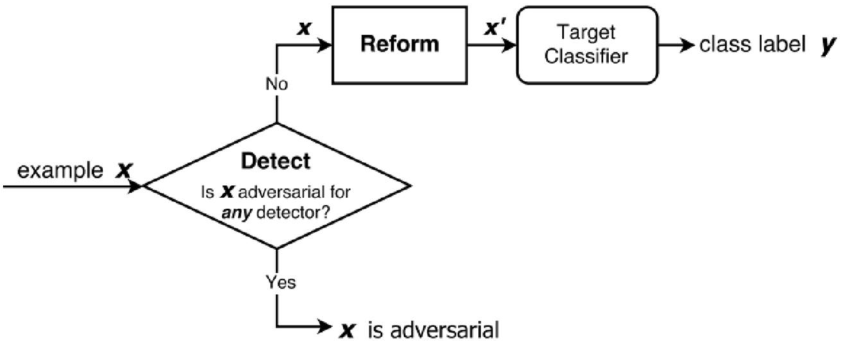


FIGURE 16 MagNet workflow in test phase (Image credit: Meng et al. [76])

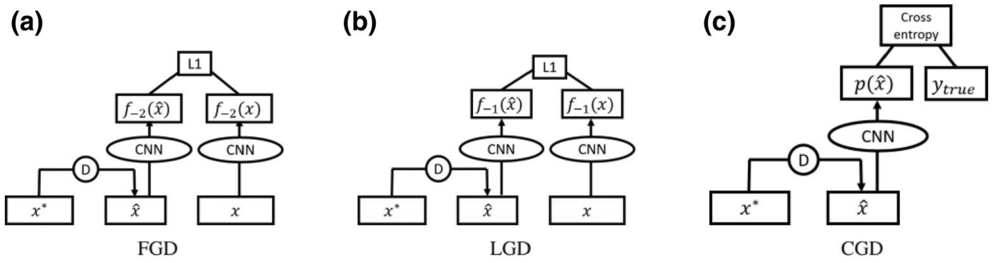


FIGURE 17 Training methods: feature-guided denoiser, logits-guided denoiser and class label guided denoiser. D denotes denoiser (Image credit: Liao et al. [77])

fed to the linear classification unit after global average pooling. In logits-guided denoiser (LGD), they defined $l = -1$ as the index for the logits, that is, the layer before the softmax layer. While both of these variants were unsupervised models, they proposed a supervised variant, class label guided denoiser (CGD), which uses classification loss of the target model as the loss function.

5.9 | Using basis function transformations

Shaham et al. [78] investigated various defense mechanisms like PCA, low-pass filtering, JPEG compression, soft thresholding and so on 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, recent applications based on machine learning algorithms are 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. [14] demonstrated the vulnerability of machine learning algorithms. In this paper, we try to present some of the well-known adversarial attacks and some of the recent defense strategies against them. 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 exist 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.

ACKNOWLEDGEMENT

We want to acknowledge the Department of Science and Technology, Government of India, to partially support the research through the Swarnajayanti Fellowship program. We would also like to acknowledge the Haldia Petrochemicals Ltd. and TCG Foundation for the research grant entitled Cyber Security Research in CPS.

REFERENCES

1. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature. 521(7553), 436 (2015)
2. Ma, J., et al.: Deep neural nets as a method for quantitative structure-activity relationships. J. Chem. Infor. Model. 55(2), 263–274 (2015)
3. Helmstaedter, M., et al.: Connectomic reconstruction of the inner plexiform layer in the mouse retina. Nature. 500(7461), 168–147 (2013)
4. Ciodaro, T., et al.: Online particle detection with neural networks based on topological calorimetry information. J. Phys. Conf. Series. 238(1) (2012)
5. Kaggle: Higgs Boson Machine Learning Challenge. (2014) https://www.kaggle.com/c/higgs-boson,2014
6. Xiong, H.Y., et al.: The human splicing code reveals new insights into the genetic determinants of disease. Science. 347(6218) (2015)
7. Google cloud ai. (2014) https://cloud.google.com/products/machine-learning/
8. Alibaba cloud (2017) https://www.alibabacloud.com/
9. Intel nervana platform (2019) https://www.intelnervana.com/intel-nervana-platform/
10. nvidia gpu cloud computing (2015) http://www.nvidia.com/object/gpu-cloud-computing.html
11. Alexa - amazon (2014) https://developer.amazon.com/alexa
12. ios - siri - apple (2011) https://www.apple.com/ios/siri/
13. Cortana — your intelligent virtual & personal assistant — microsoft. (2015) https://www.microsoft.com/en-us/windows/cortana
14. Szegedy, C., et al.: Intriguing properties of neural networks. CoRR. abs/1312.6199 (2013) http://arxiv.org/abs/1312.6199
15. Carlini, N., et al.: Hidden voice commands, pp. 513–530 (2016)
16. Zhang, G., et al.: Dolphinatack: Inaudible voice commands arXiv preprint arXiv:170809537 (2017)
17. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial machine learning at scale. CoRR. abs/1611.01236 (2016) http://arxiv.org/abs/1611.01236

18. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. CoRR. abs/1412.6572 (2014) http://arxiv.org/abs/1412.6572
19. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples arXiv preprint arXiv:160507277 (2016)
20. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., et al.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 308–318. ACM (2016)
21. Hitaj, B., Ateniese, G., Perez.Cruz, F.: Deep models under the gan: Information leakage from collaborative deep learning arXiv preprint arXiv:170207464 (2017)
22. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. CoRR. abs/1503.02531 (2015) http://arxiv.org/abs/1503.02531
23. Papernot, N., et al.: 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, pp. 582–597 (2016) https://doi.org/10.1109/SP.2016.41
24. Pouya.Samangouei, M.K., Chellappa, R.: Defense-gan: Protecting classifiers against adversarial attacks using generative models arXiv preprint arXiv:180506605 (2018)
25. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: A survey arXiv preprint arXiv:180100553 (2018)
26. Barreno, M., et al.: Can machine learning be secure? In: Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, pp. 16–25. ACM (2006)
27. Barreno, M., et al.: The security of machine learning. Mach. Learn. 81(2), 121–148 (2010)
28. Corona, I., Giacinto, G., Roli, F.: Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. Infor. Sci. 239, 201–225 (2013)
29. Biggio, B., Fumera, G., Roli, F.: Security evaluation of pattern classifiers under attack. IEEE Trans. Knowl. Data Eng. 26(4), 984–996 (2014) https://doi.org/10.1109/TKDE.2013.57
30. Kumar, A., Mehta, S.: A survey on resilient machine learning. CoRR. abs/1707.03184 (2017) http://arxiv.org/abs/1707.03184
31. Papernot, N., et al.: Towards the science of security and privacy in machine learning. CoRR. abs/1611.03814 (2016) http://arxiv.org/abs/1611.03814
32. Biggio, B., Fumera, G., Roli, F.: Security evaluation of pattern classifiers under attack. IEEE Trans. Knowl. Data Eng. 26(4), 984–996 (2014)
33. Rosenberg, I., et al.: Generic black-box end-to-end attack against rnns and other api calls based malware classifiers arXiv preprint arXiv:170705970 (2017)
34. Tramèr, F., et al.: Stealing machine learning models via prediction apis. In: USENIX Security Symposium, pp. 601–618 (2016)
35. Papernot, N., et al.: 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, pp. 506–519 (2017) http://doi.acm.org/10.1145/3052973.3053009
36. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1322–1333. ACM (2015)
37. Shokri, R., et al.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 3–18. IEEE (2017)
38. Moosavi.Dezfooli, S.M., Fawzi, A., Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2574–2582 (2016)
39. Tramèr, F., et al.: Ensemble adversarial training: Attacks and defenses. CoRR. abs/1705.07204 (2017) http://arxiv.org/abs/1705.07204
40. Ateniese, G., et al.: Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. CoRR. abs/1306.4447 (2013) http://arxiv.org/abs/1306.4447
41. Biggio, B., Nelson, B., Laskov, P.: 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, pp. 97–112 (2011) http://www.jmlr.org/proceedings/papers/v20/biggio11/biggio11.pdf
42. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26–July 1 (2012)
43. Dalvi, N., et al.: Adversarial classification. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 99–108. ACM (2004)
44. Biggio, B., Fumera, G., Roli, F.: Adversarial pattern classification using multiple classifiers and randomisation. Struct. Syntact. Statist. Pattern Recogn, 500–509 (2008)
45. Biggio, B., et al. Security Evaluation of Support Vector Machines in Adversarial Environments. Support Vector Machines Applications. In: Ma, Y., et al. (eds.). pp. 105–153. Springer International Publishing, Cham (2014) https://doi.org/10.1007/978-3-319-02300-7_4
46. Goodfellow, I.J., et al.: Generative adversarial networks. CoRR. abs/1406.2661 (2014) http://arxiv.org/abs/1406.2661
47. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57. IEEE (2017)
48. Li, B., et al.: Data poisoning attacks on factorization-based collaborative filtering. CoRR. abs/1608.08182 (2016) http://arxiv.org/abs/1608.08182
49. Fredrikson, M., et al.: Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing.
50. Silva, T.: Medium. An intuitive introduction to generative adversarial networks (2018) https://medium.freecodecamp.org/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394
51. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks arXiv preprint arXiv:151106434 (2018)
52. Springenberg, J. T., et al.: Striving for simplicity: The all convolutional net arXiv preprint arXiv:14126806 (2014)
53. Mordvintsev, A., Olah, C., Tyka, M.: Inceptionism: Going deeper into neural networks (2015) https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
54. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift arXiv preprint arXiv:150203167 (2015)
55. Nair, V., Hinton, G.: ICML-10: Rectified Linear Units Improve Restricted Boltzmann Machines, pp. 807–814 (2010)
56. Maas, A.L., Hannum, A.Y., Ng, A.Y.: Rectifier Nonlinearities Improve Neural Network Acoustic Models, p. 30 (2013)
57. Xu, B., et al.: Empirical evaluation of rectified activations in convolutional network arXiv preprint arXiv:150500853 (2015)
58. Kearns, M., Li, M.: Learning in the presence of malicious errors. SIAM J. Comput. 22(4), 807–837 (1993) Available from. http://dx.doi.org/10.1137/0222052
59. Kloft, M., Laskov, P.: Online anomaly detection under adversarial impact. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Vol. 9 of Proceedings of Machine Learning Research, pp. 405–412. PMLR, Sardinia, Italy (2010) http://proceedings.mlr.press/v9/kloft10a.html
60. Mei, S., Zhu, X.: 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, pp. 2871–2877. AAAI Press (2015) http://dl.acm.org/citation.cfm?id=2886521.2886721
61. Liu, D.C., Nocedal, J.: On the limited memory bfgs method for large scale optimization. Math. Program. 45(1), 503–528 (1989)

62. Papernot, N., et al.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 372–387. IEEE (2016)
63. Zhou, Y., Murat, K., Xi, B.: A survey of game theoretic approach for adversarial machine learning. Wires Data Mining Knowl. Discov. (2018)
64. Biggio, B., et al.: Security Evaluation of Support Vector Machines in Adversarial Environments. Springer (2014)
65. Golland, P., NIPS: Discriminative Direction for Kernel Classifiers (2001)
66. Benjamin, I., et al.: Learning in a large function space: Privacy-preserving mechanisms for SVM learning arXiv preprint arXiv:09115708 (2009)
67. Prateek.Jain, P.N., Sanghavi, S.: Low-rank matrix completion using alternating minimization arXiv preprint arXiv:12120467 (2012)
68. Jian-Feng, C., Candes, E. J., Shen, Z.: A singular value thresholding algorithm for matrix completion, 20, 1956–1982 (2010)
69. Attacking Machine Learning with Adversarial Examples (2017) https://openai.com/blog/adversarial-example-research/
70. Shaham, U., Yamada, Y., Negahban, S.: Understanding adversarial training: Increasing local stability of neural nets through robust optimization. CoRR. abs/1511.05432 (2015) http://arxiv.org/abs/1511.05432
71. Lyu, C., Huang, K., Liang, H.: 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, pp. 301–309 (2015) https://doi.org/10.1109/ICDM.2015.84
72. Narodytska, N., Kasiviswanathan, S.P.: 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, pp. 1310–1318 (2017) https://doi.org/10.1109/CVPRW.2017.172
73. Papernot, N., McDaniel, P.D.: Extending defensive distillation. CoRR. abs/1705.05264 (2017) http://arxiv.org/abs/1705.05264
74. Xu, W., Evans, D., Qi, Y.: Feature squeezing: Detecting adversarial examples in deep neural networks. CoRR. abs/1704.01155 (2017) http://arxiv.org/abs/1704.01155
75. Hosseini, H., et al.: Blocking transferability of adversarial examples in black-box learning systems. CoRR. abs/1703.04318 (2017) http://arxiv.org/abs/1703.04318
76. Dongyu.Meng, H.C., Magnet: a two-pronged defense against adversarial examples. ACMCCS (2017)
77. Liao, F., et al.: Defense against adversarial attacks using high-level representation guided denoiser arXiv preprint arXiv:170509064 (2017)
78. Shaham, Uri., et al.: Defending against adversarial images using basis functions transformations arXiv preprint arXiv:180310840. (2018)

How to cite this article: Chakraborty A, Alam M, Dey V, Chattopadhyay A, Mukhopadhyay D. A survey on adversarial attacks and defences. *CAAI Trans. Intell. Technol*. 2021;6:25–45. https://doi.org/10.1049/cit2.12028