IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 August 2022, accepted 15 September 2022, date of publication 20 September 2022,
date of current version 30 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3208131

SURVEY

Adversarial Deep Learning: A Survey on Adversarial Attacks and Defense Mechanisms on Image Classification

SAMER Y. KHAMAISEH[1], (Member, IEEE), DEREK BAGAGEM[2], ABDULLAH AL-ALAJ[3], (Member, IEEE), MATHEW MANCINO[4], AND HAKAM W. ALOMARI[1], (Member, IEEE)

[1]Department of Computer Science and Software Engineering, Miami University Oxford, OH 45056, USA
[2]Department of Computer Science and Software Engineering, Monmouth University West Long Branch, NJ 07764, USA
[3]Department of Computer Science, Virginia Wesleyan University Virginia Beach, VA 23455, USA
[4]CACI International, Norfolk, VA 23455, USA

Corresponding author: Samer Y. Khamaiseh (khamaisy@miamioh.edu)

ABSTRACT The popularity of adapting deep neural networks (DNNs) in solving hard problems has increased substantially. Specifically, in the field of computer vision, DNNs are becoming a core element in developing many image and video classification and recognition applications. However, DNNs are vulnerable to adversarial attacks, in which, given a well-trained image classification model, a malicious input can be crafted by adding mere perturbations to misclassify the image. This phenomena raise many security concerns in utilizing DNNs in critical life applications which attracts the attention of academic and industry researchers. As a result, multiple studies have proposed discussing novel attacks that can compromise the integrity of state-of-the-art image classification neural networks. The raise of these attacks urges the research community to explore countermeasure methods to mitigate these attacks and increase the reliability of adapting DDNs in different major applications. Hence, various defense strategies have been proposed to protect DNNs against adversarial attacks. In this paper, we thoroughly review the most recent and state-of-the-art adversarial attack methods by providing an in-depth analysis and explanation of the working process of these attacks. In our review, we focus on explaining the mathematical concepts and terminologies of the adversarial attacks, which provide a comprehensive and solid survey to the research community. Additionally, we provide a comprehensive review of the most recent defense mechanisms and discuss their effectiveness in defending DNNs against adversarial attacks. Finally, we highlight the current challenges and open issues in this field as well as future research directions.

INDEX TERMS Deep neural networks, artificial intelligence, adversarial examples, adversarial perturbations.

I. INTRODUCTION

Deep learning makes a significant breakthrough in providing solutions to many hard problems that cannot be solved using traditional machine learning algorithms. Examples include, but are not limited to, image classification, text translation, and speech recognition. Due to the advancement of deep learning neural networks and the availability of powerful computational resources, deep learning is becoming the primary choice for developing a wide spectrum of critical life applications such as DNA analysis [1], autonomous vehicles [2] and other security application such DDoS detection and malware detection [3], [4].

Adapting deep learning models to computer vision tasks was set in motion by Krizhevsky *et al.* [5]. The authors were able to successfully demonstrate the capabilities and performance achievable by utilizing deep neural networks for image recognition [6]. Their work sparked an increased interest in deep learning and computer vision research, giving rise to more complex and more powerful deep learning models

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Tucci.

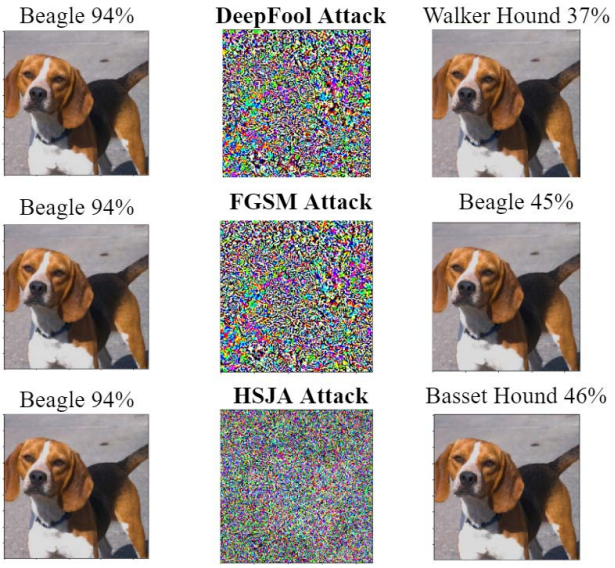

FIGURE 1. Samples of adversarial attacks: DeepFool, FGSM, and HSJA against the pre-trained ResNet-50 deep learning model. The trained deep learning model correctly classifies the original input image with 94% accuracy. After adding small imperceptible perturbations to the image, the trained model misclassified the adversarial image.

such as ResNet [7] and DenseNet [8]. The introduction of more advanced network architectures allowed researchers to explore complex tasks such as X-ray analysis [9], predictive maintenance [10], and crop yield prediction [11].

Despite the evident ability in solving many sophisticated problems with high accuracy, Szegedy *et al.* [12] demonstrated that deep neural networks are susceptible to adversarial attacks. As depicted in Fig. 1, an adversarial example can be generated by adding small perturbations to an image to fool the deep neural networks and reduce their accuracy. Their finding triggered the interest of researchers to study the security of deep neural networks. As a result, several adversarial attacks have been proposed in the literature that show different security vulnerabilities that can be exploited by an adversary to compromise a deep learning system. For example, Su *et al.* [13] showed that changing one pixel on an image can fool a deep learning model. Furthermore, different research works have shown the ability to generate universal perturbations that can fool any neural network [14].

The inherited weaknesses of DNN models against adversarial attacks raise many security concerns especially for critical applications such as the robustness of deep learning algorithms used for autonomous vehicles [15]. Hence, different studies propose various countermeasure methods against adversarial attacks. Examples include the modification to the deep neural network, adding to the neural network, and many others are explained in this literature.

A. MOTIVATION AND CONTRIBUTION

The main motivation of this paper is to provide a comprehensive review of the most recent deep learning adversarial attacks and defense methods to offer easy access to the recent advancement in this field and provide a jump-start to tap on the rapidly growing adversarial deep learning research. Also, we believe there is a need to survey the current and the emerging adversarial deep learning advancement and provide an in-depth study about future research directions. Recently, different articles have reviewed various research works in this field [16], [17], [18], [19], [20], [21]. This survey is different from the existing surveys in several aspects. As compared to other surveys in the literature, this survey provides a comprehensive background review of the mathematical concepts that are vital to understanding the working process of the adversarial attacks and defense mechanisms on image classification, as well as it provides a clear description of the terminologies and technical terms used in this domain based on the most recent research and advancements in this field. The distinctive part of this survey is that it provides a systematic and deep review of the working process of the most recent and state-of-the-art adversarial attacks by focusing on describing their mathematical terminologies and foundation which provide an easy and profound description of these attacks. Besides, we provide a general overview about the effectiveness of these attacks in many terms including but not limited to attack performance. Furthermore, we provide a comprehensive review of the well-known defense mechanisms by highlighting their effectiveness including their limitations and the covered attacks. Not to mention, this article mainly focuses on reviewing adversarial attacks and their defense methods in computer vision. However, we provide a lightweight review of the well-known adversarial attacks in different contexts such as audio, 3-D data, and software which help the interested readers to explore and rapidly tap on the adversarial attacks in different contexts.

The main contributions of this article can be summarized as follows:

- We provide an extensive study of state-of-the-art algorithms for generating deep learning adversarial attacks in computer vision
- We provide an in-depth study of various adversarial attacks defense mechanisms.
- We provide a systematic and comprehensive review of the adversarial threat model that covers the deep learning system attack surface, adversarial knowledge and capabilities, adversarial goals, and attack scenarios.
- We identify and discuss a number of open issues and possible future research directions for adversarial deep learning.

B. ORGANIZATION

The remainder of this paper is organized as follows. Section II provides some technical terms related to adversarial deep learning. In section III we provide an overview of different concepts of deep learning and adversarial attacks. Section IV describes the threat model of deep learning. We dedicate Section V to discuss the adversarial attacks. In Section VI we introduce the defense mechanisms. We discuss the future

research directions of deep learning security in Section VII. Finally, Section VIII concludes this paper.

II. DEFINITIONS OF TERMS

In this section, we describe some of the technical terms used in this survey study.

- **Adversarial example** ($\hat{x}$) is a malicious permutation of a clean image (x) that has been generated by adding small perturbations (*e.g.*, noise) to fool the deep learning model. The added perturbations are usually imperceptible, however, there are a few cases in which the perturbation is perceptible.
- **Adversarial perturbation** is the noise that is added to a clean instance image to create an adversarial example.
- The **Adversary** is the attacker that crafts and carries out adversarial attacks against the ML model.
- **Adversarial training** is a kind of machine learning training that uses the adversarial examples within the training set to increase the robustness of the deep learning model against adversarial attacks.
- **Image-Detector** is a mechanism that is used to reveal adversarial images.
- **Fooling rate/ratio** is a measurement that is used to evaluate the robustness of an ML model and adversarial attack intensity. It refers to the percentage of the adversarial examples that fooled the ML model. A high fooling ratio means that the attack is strong, and the ML model is not robust to the adversarial examples.
- **Targeted attack** is an attack that forces the DNN model to misclassify an adversarial example to a specific target label predefined by the adversary.
- **Untargeted attack** is the most common type of attack that forces the DNN model to misclassify the label of the adversarial example into *any incorrect label*. Such attacks intend to reduce the integrity and availability of the DNN models.

III. BACKGROUND

Adversarial machine learning attacks and defense methods are formed using different mathematical concepts. This section provides a comprehensive overview of several concepts that are crucial for understanding how the attacks and defense methods work.

A. GRADIENT DESCENT

Optimization is the process of either minimizing or maximizing an objective function. In machine learning, the optimization process is used to find the optimal values of the parameters of an objective function that minimize a cost function. Different algorithms can be adapted to perform the optimization process. Gradient descent is one of the most used algorithms to find the optimal parameters for a wide range of machine learning algorithms [22].

Gradient descent is a first-order optimization algorithm that uses the gradient of the function at its current location to find the trajectory used to move through the search space.

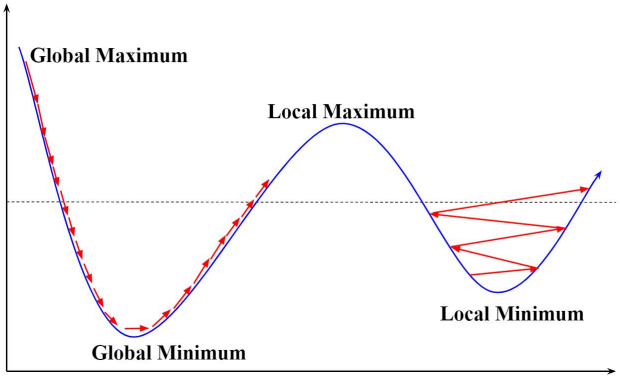

FIGURE 2. A demonstration of the difference between the local and global maximum and minimum values as well as the impact of the learning rate on model training. The learning rate value (shown in red) determines the magnitude of the updates to the model's weights.

As depicted in Fig. 2, the basic gradient descent algorithm consists of: (1) calculating the gradient ∇_f of the objective function $J(\theta)$, (2) moving in the opposite direction of the gradient ∇_f which is the direction of the steepest descent that will lead to an improvement (*i.e.*, finding the global minimum) (see Fig. 2), and (3) selecting the learning rate β that refers to the size of the steps toward the minimum. β is the most important parameter that needs to be tuned carefully to achieve a highly performed DNN model. Generally, a large β allows the ML-model to learn faster. However, it could drastically decrease the model's performance since the algorithm may result on the other side of the valley (missing the optimal minimum). Small β allows the model to converge by finding the local minimum after many iterations which, not surprisingly, requires a long run-time. There is a trade-off between the accuracy of the results and the time required to perform parameter updates.

Gradient descent has three variants: batch gradient descent (BGD), stochastic gradient descent (SGD), and mini-batch gradient descent, which we will discuss in the next sections.

1) BATCH GRADIENT DESCENT (BGD)

As shown in Eq. 1, BGD calculates the gradient of the cost function with regards to the model parameter θ for the whole training dataset. In other words, BGD computes the gradient descent over the full training dataset to perform one parameter update, which explains why it is called *"Batch"* or in some case *"full gradient descent"*. Depending on the size of the training dataset, batch gradient descent is time-consuming and requires a long processing time.

$$\theta = \theta - \beta \cdot \nabla_\theta J(\theta) \tag{1}$$

Despite the long processing time, BGD has several advantages. For example, when the cost function is convex, BGD with a fixed learning rate will converge to the global minimum. When the cost function is not convex, it will converge to the local minimum since it has a straight direction to the minimum value, see Fig. 3. Therefore, BGD guarantees convergence to the minimum value.

2) STOCHASTIC GRADIENT DESCENT (SGD)

Since BGD requires a long processing time to calculate the gradient for large-scale training datasets, SGD [23] was proposed to overcome BGD's limitations. SGD is an iterative technique for optimizing the cost function. As illustrated in Eq. 2, in every iteration SGD randomly selects an example from the training data and calculates the gradient, then it performs a parameter update using the selected example. In contrast to BGD, where the actual gradient is calculated using the entire dataset, the objective of SGD is to calculate an estimation of the gradient using a small number of instances. The cost of updating the stochastic gradient descent is independent of the training dataset size and can reach linear convergence [23]. In some cases, the cost of training some ML models using SGD can reach $\mathcal{O}(1)$ [24]. Thus, SGD has become the most widely used optimization algorithm.

$$\theta = \theta - \beta \cdot \nabla_\theta J(\theta; x^i; y^i) \tag{2}$$

As depicted in Fig. 3, SGD has one main performance issue. Unlike the convergence path of BGD, which gently moves toward the minimum, the convergence path in SGD is irregular and has many fluctuations due to the noise that originates from the random selection of the instances. Therefore, SGD performs high variance parameter updates [25].

3) MINI-BATCH GRADIENT DESCENT

Instead of using the entire training dataset to compute the gradient of the cost function, the training dataset is shuffled and then divided into mini-batches based on a predefined batch size [22]. Subsequently, these small batches are used to calculate the gradient descent of the cost function as shown in Eq. 3.

$$\theta = \theta - \beta \cdot \nabla_\theta J(\theta; x^{i:i+n}; y^{i:i+n}) \tag{3}$$

The advantages of using mini-batch gradient descent over other gradient descent methods are two-fold: First, it is faster than BGD in computing the gradient of the cost function. Second, with a large number of batches, mini-batch gradient descent fluctuates less than stochastic gradient descent while moving towards the minimum, see Fig. 3.

However, mini-batch gradient descent requires tuning the batch size parameter. When the batches are small, extra noise will be added to the training process which helps in reducing the generalization error (*i.e.*, regularization effect). When the batches are large, mini-batch gradient descent converges slowly but has more accurate results concerning the gradient error.

B. DISTANCE METRICS

Distance metrics are used to measure the distance between two points (*i.e.*, vectors). In other words, distance metrics quantify the similarities between two vectors. If the distance is zero, the two vectors are equivalent under that distance metric. To compute the distance between two vectors, the norm of the difference between those two vectors needs to be evaluated using a norm function. A norm L_p is a function that measures the magnitude of a vector. Formally, the norm function L^p of x can be defined as:

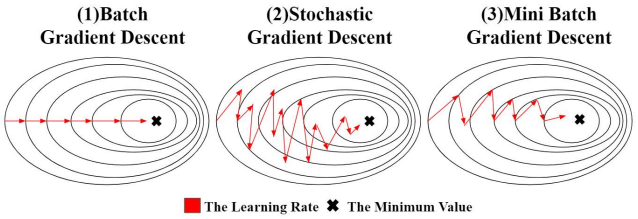

FIGURE 3. An illustration of the differences between training a model using BGD, SGD, and mini-batch gradient with regrading to approaching the minimum value.

$$\|x\|_p = \left(\sum \|x_i\|^p\right)^{1/p}, \exists : p \in \mathbb{R}, p \geq 1 \tag{4}$$

The distance metrics are used within the process of generating the adversarial attacks by quantifying their similarities. L_0, L_1, L_2, and L_∞ distance metrics have been widely adopted by state-of-the-art adversarial attack algorithms [12], [26], [27], [28] which are detailed below.

1) L_0 DISTANCE

As shown in Eq. 5, L_0 distance is used to calculate the vector size by measuring the total number of the non-zero elements of a given vector. Arguably, sometimes L_0 is referred to as a "*norm*" which is not correct since scaling a vector by constant value a will not change the number of non-zero elements, thus, it is more accurate to be classified as a cardinality function.

$$\|x\|_0 = (i|x_i \neq 0) \tag{5}$$

The L_0 distance was used in [26] and [29] to generate adversarial attacks since it corresponds to the number of altered pixels of an image.

2) L_1 DISTANCE

The L_1 distance, also known as the Manhattan Distance or the Taxicab norm, is used when the difference between non-zero and zero elements is important. Essentially, when an element moves away from the origin (0,0) by a, L_1 increments by a. Therefore, L_1 measures the distance between the origin (0,0) to the point (x,y). Formally, L_1 distance is defined as follows:

$$\|x\|_1 = \sum_{i=1}^{n} x_i \tag{6}$$

The L_1 distance is utilized by the elastic net attack [30] to generate adversarial perturbations. More specifically, the L_1 distance functions as a regularization parameter, representing the perturbation's total variation. The L_1 distance improves the transferability of the elastic net attack by generating distinct adversarial images that fool DNN models.

3) L_2 DISTANCE

The L_2 distance, also known as the Euclidean Distance, is widely used in machine learning, and often denoted as $\|x\|$. As shown in Eq.7, L_2 measures the shortest distance (*i.e.*, length of the straight line) between two vectors.

$$\|x\|_2 = \left(\sum_{i=1}^{n} \|x_i\|^2\right)^{1/2} \tag{7}$$

The L_2 distance was employed by different researchers to generate adversarial attacks [31]. For example, In [27], they used L_2 to measure the distance between the class labels of the original image x and the perturbed image $\hat{x}$. It is used to measure the size of the perturbations in the perturbed image $\hat{x}$.

4) L_∞ DISTANCE

The L_∞ distance, also known as the ''max norm'', returns the maximum magnitude of the difference between each element in a vector. As shown in Eq. 8, the L_∞ norm can be described as the maximum among the absolute values of the differences in a set of numbers (*e.g.*, coordinate pair, n-dimensional vector, etc.) [26].

$$\left\|x - x'\right\|_\infty = max(\left|x_1 - x_1'\right|, \ldots, \left|x_n - x_n'\right|) \tag{8}$$

In adversarial settings, L_∞ can be used as a sufficient constraint over the size of perturbations that could be added to generate the perturbed image [32]. Thus, L_∞ ensures that the perturbations do not modify the true class of the original image.

C. ARTIFICIAL NEURAL NETWORKS

In an attempt to mimic the way the brain learns information and patterns, artificial neural networks (ANNs) were developed to be the generalized mathematical models of biological neural networks [33]. The basic structure of an ANN consists of at least three layers: (1) the input layer, (2) the output layer, and (3) the hidden layer(s). Each layer is a collection of nodes that passes the inputs along to the succeeding layers. Nodes are the basic units of computation in an ANN, taking inputs from other nodes or external sources and producing an output. Depending on the conditions of the problem, different types of neural networks are required to perform different tasks. For example, a text generation problem would employ a recurrent neural network over other ANNs for optimal results. We will further discuss and break down ANNs and their specific components in the ensuing sections.

1) ARTIFICIAL NEURONS

Artificial neurons, or nodes, are mathematical representations of biological neurons. They serve as connection points between layers in an artificial neural network. Nodes take input values (x_1,...,x_n) from the dataset and combine them with a set of weights ($w_1, \ldots, w_n$). The weighted inputs are summed and passed through a non-linear activation or transfer function [33]. This process is shown in Fig. 4, in which a set of inputs are passed through a node. Like their biological

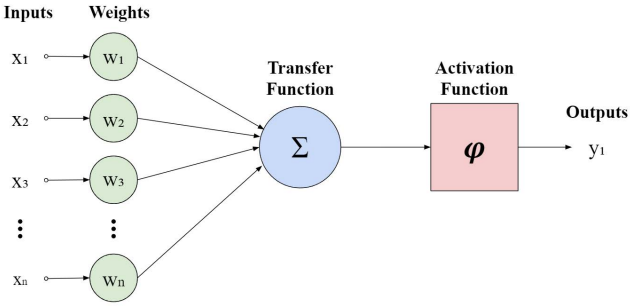

FIGURE 4. A mathematical model of an artificial neuron (Perceptron), with inputs $x_1 \ldots x_n$, weight values $w_1 \ldots w_n$, activation and transformation functions, and output value y. Artificial neurons are the building blocks of a neural network.

counterparts, there exists a hierarchical arrangement in artificial neurons and artificial neural networks. The weighted output of node y is passed into the next layer of the ANN until it reaches the final layer of the network.

2) FEED-FORWARD NEURAL NETWORKS

Artificial neural networks can solve classification problems for non-linear datasets by utilizing the hidden layers within the network. A layer in a network is a row of interconnected nodes, and an ANN can have multiple layers that improve its overall robustness and performance. As demonstrated in Fig. 5, the basic structure of a multi-layer neural network consists of three layers: (1) the input layer, (2) the hidden layer, and (3) the output layer.

The input layer is the initial layer in an ANN, and it is mainly responsible for feeding the data into the network. The inputs are then transferred to the hidden layer(s) for processing. The hidden layer(s) is where the network applies an activation function and weights for the inputs. The hidden layers process the inputs coming from the preceding layer and extract the required information from the data. As shown in Fig. 6, neural networks can have multiple hidden layers, this is known as a deep neural network (DNN). Based on the problem's complexity, multiple hidden layers can be used to increase the prediction accuracy of the network and extract more features from the data. For example, as shown in Fig. 7, a convolutional neural network (CNN) used for facial recognition cannot solely identify a human face with only one hidden layer. One layer that identifies eyes cannot recognize an entire face, but if it is combined with other layers that identify other features, such as noses or mouths, the network becomes stronger and can successfully recognize faces. The output layer is the final layer in an ANN and is responsible for aggregating the information and returning the outputs in the format given by the problem.

Forward propagation is the process of progressively moving through the layers of the ANN and is used in feed-forward neural networks [34]. The hidden layer(s) take the input data, process it, and then pass it onto the next layer. This is a necessary step for feed-forward networks to generate outputs. If the data travels backward at any point, it will form

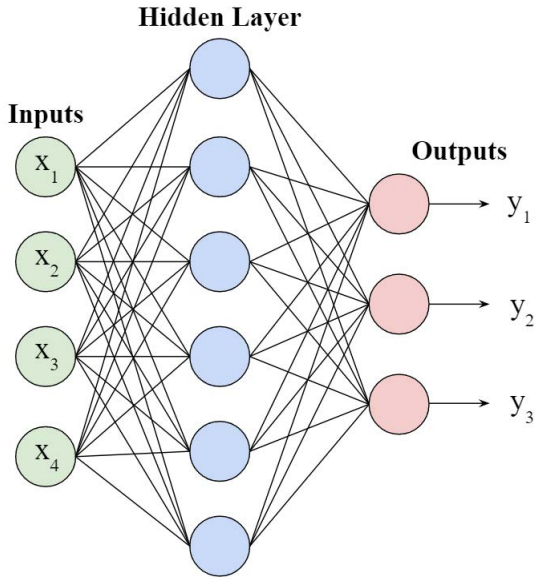

FIGURE 5. The general architecture of a simple neural network and its three layers: the input layer (green), the hidden layer (blue), and the output layer (red).

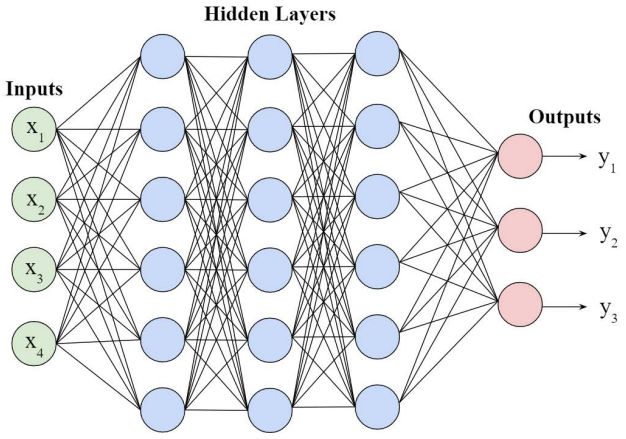

FIGURE 6. Demonstration of a deep neural network with multiple hidden layers. The additional hidden layers allow the neural network to perform complex facial and object recognition operations.

a loop that will block the output generation process. Other ANNs are designed and built using other architectures (*e.g.*, backpropagation) and serve different purposes.

3) BACKPROPAGATION

At the core of most neural networks is the process of backpropagation. Backpropagation is a training algorithm that starts by feeding the input values forward through the ANN, calculating the error, and propagating it back through the network [34]. The main goal of backpropagation is to minimize the cost function by iteratively updating and adjusting the ANN weights and biases. Parameter updates are calculated by taking the gradient of the loss function with respect to the weights of the model. The network will back-propagate until the cost function is minimized.

4) ACTIVATION FUNCTIONS

The activation function determines whether an artificial neuron should be activated. The main objective of the activation function is transforming the values in the node into an output value that can be accepted as input into a function (*e.g.*, vector) while adding non-linearity to the output values. Activation functions map the resulting values from the summation function in the node to lie between [0 to 1] or [−1 to 1]. The result of the activation function forms the input for the next layer. Activation functions fall into two categories: linear and non-linear. The most widely used activation functions are non-linear, and the most popular ones include hyperbolic tangent, sigmoid, and softmax.

a: SIGMOID FUNCTIONS

The sigmoid functions are characterized by an *S*-shaped curve that can be categorized into three different functions: the logistic function, the hyperbolic tangent, and the arctangent. In the context of machine learning, the sigmoid function coincides with the logistic sigmoid function [35]. The logistic sigmoid function as defined in Eq.9, takes any real value x and outputs a value $S(x)$ that lies within the range [0 to 1].

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \tag{9}$$

Sigmoid functions are widely used as activation functions in deep learning because they add non-linearity into the network. Sigmoid functions are also used to convert real numbers into probabilities. A logistic sigmoid function that is placed in the last layer of an ANN converts the output into a probability score.

b: SOFTMAX FUNCTION

The softmax function can be viewed as a generalization of the logistic regression function, sharing similarities to the sigmoid function shown in Eq. 10. The softmax function transforms a vector of values into a single vector, whose values when summed are equal to one. It is normal to see the softmax function implemented as a penultimate layer in a neural network [36] because it can transform the outputs from the hidden layers into a normalized probability.

$$\sigma\vec{z}_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e z_j} \tag{10}$$

where x is a vector of output values, and e is a mathematical constant which is the base for the natural log. The sigmoid and softmax functions are similar in terms of the latter function being a specialized case of the former. The main difference between the two lies in the type of values they can accept as input. The softmax function only accepts vectors as input, whereas the sigmoid function only accepts scalars.

c: HYPERBOLIC TANGENT OR TANH

The hyperbolic tangent or tanh function can be utilized as an alternative to the logistic sigmoid function in an ANN. The tanh function shares similarities with the logistic sigmoid

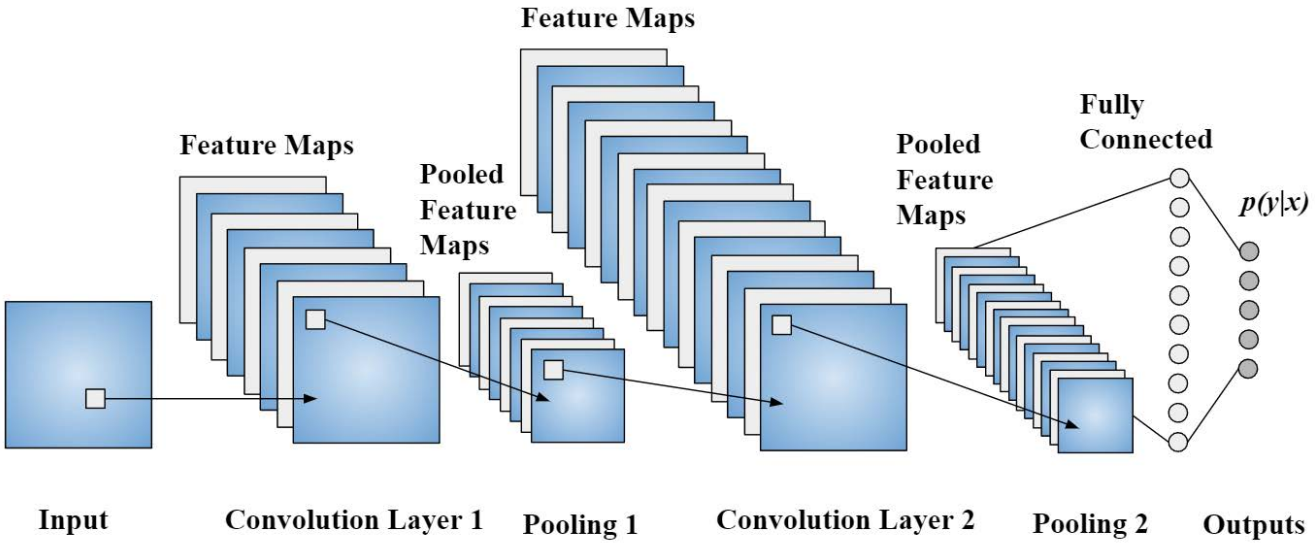

FIGURE 7. An example of a CNN that is used for image classification, recognition, and processing. CNN's differentiate themselves from normal ANNs because of the addition of convolutional layers which utilize convolution to convert the image pixels into values that can be used for classification.

function in that it has a similar S-shaped curve. One difference between the two is the output range. As defined in Eq. 11, the tanh function takes any real value x as an input and generates a value in the range $[-1$ to $1]$ as output.

$$\tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \tag{11}$$

Larger input values (more positive) will result in outputs that are closer to 1, whereas smaller inputs (more negative) will result in outputs closer to -1. tanh is preferable to the logistic sigmoid as it has unrestricted gradients [37], and the outputs function is zero centered. Since the values lie between $[-1$ to $1]$, their mean value is approximately equals zero which helps center the data and allows the next layers to have an easily processing and predicting the data.

IV. ADVERSARIAL THREAT MODEL

This section describes the threat model of machine learning models. As depicted in Fig. 8, the threat model includes the adversarial degree of knowledge, adversarial capabilities, adversarial goals, and the attacking scenarios.

A. ADVERSARIAL DEGREE OF KNOWLEDGE

The adversary knowledge about the targeted machine learning system (*e.g.*, DNN image classification model) may vary and can be classified into three categories:

1) PERFECT-KNOWLEDGE

In this setting, the adversary has *complete-knowledge* of the targeted learning system used for classification/prediction [38]. The adversary has comprehensive knowledge of the training and testing dataset. The adversary has complete knowledge of the ML model which includes: the model architecture, the number of layers, the weights, the inputs and outputs, and the model's features and parameters. The adversary also knows the type of learning algorithm which includes the type of the activation function and loss function. In white-box settings, the adversary has access to the full knowledge of the targeted learning system [39]. Thus, the generated adversarial attacks using this setting are commonly known as white-box attacks [40], [41].

2) LIMITED-KNOWLEDGE

In this setting, the adversary has insubstantial knowledge of the targeted learning system. This includes knowing a similar training dataset to the one that was used in training the targeted DNN model. The adversary can obtain such limited knowledge about the targeted model by building a surrogate model of the same scale [39]. The adversarial attacks generated in this setting are referred to as gray-box attacks [42].

3) ZERO-KNOWLEDGE

In contrast to the perfect knowledge, in this setting, the adversary has no knowledge about the targeted system or access to any surrogate model. The only available option for the adversary is querying the targeted learning system (*i.e.*, oracle). Given the adversary's lack of knowledge, the generated attacks using this setting are referred to as black-box attacks [41].

B. ADVERSARIAL CAPABILITIES

Machine learning threat models can be categorized using the capabilities of the adversary. In cyber-security, the term "*capability*" refers to the adversary's level of access to the system resources (*i.e.*, the learning model and data). Depending on the adversarial attack settings, the adversary capabilities can be categorized as the following:

1) TRAINING DATA

In white-box settings, the adversary has read and write access to the training dataset of the targeted learning system.

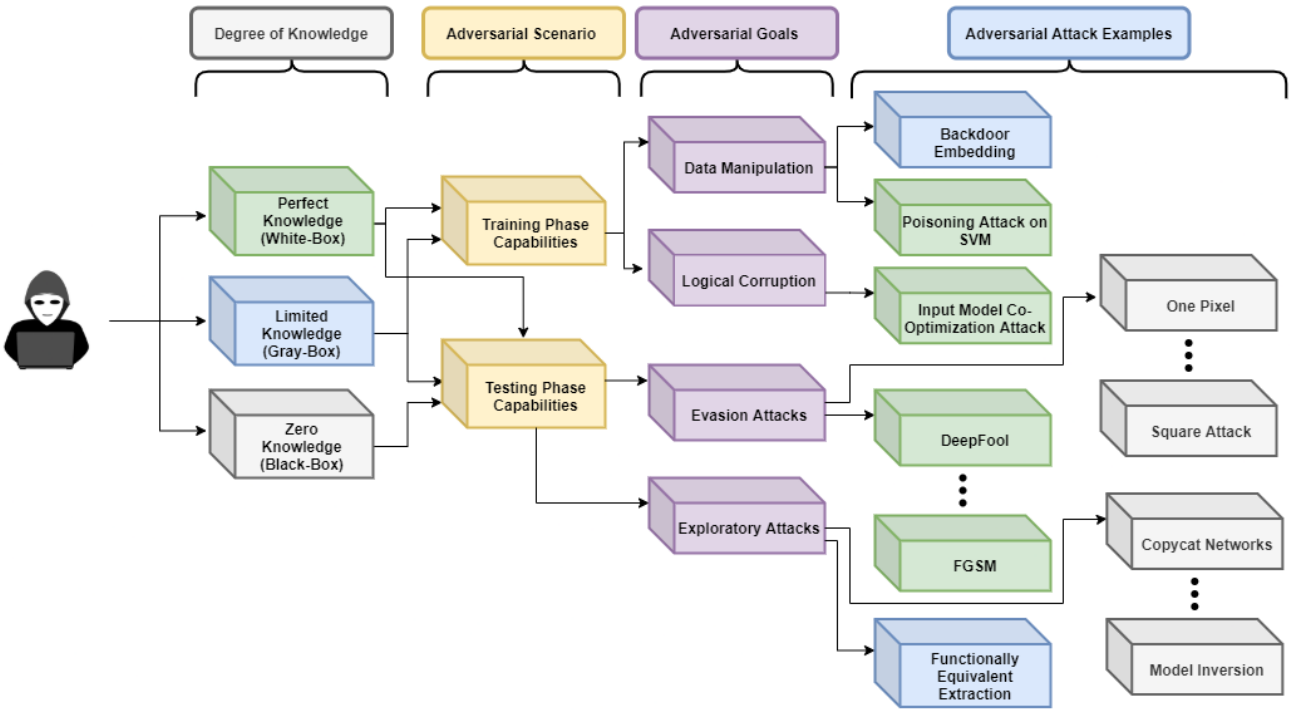

FIGURE 8. The taxonomy of the adversarial threat model, broken down by the degree of knowledge, perfect knowledge (white-box) shown in green, limited knowledge (gray-box) shown in blue, and zero knowledge (black-box) shown in light gray. The threat model further splits into either training or testing phase capabilities, and then into their adversarial goals. The goals for the training phase are data manipulation or logical corruption, whereas the goals for the testing phase capabilities are evasion and extraction attacks. Adversarial attack examples for both the testing and training phase capabilities are also shown. This diagram covers the threat model for most of the adversarial attacks that are discussed in Section V.

The adversary can modify the feature vectors in various ways such as modifying the data of specific feature(s) or adding/removing certain features. In a gray-box attack setting, the adversary can collect surrogate training data that is similar to the original training dataset of the targeted model [42]. In a black-box setting, the adversary does not have access to the training dataset of the targeted system [40].

2) NETWORK ARCHITECTURE

In white-box attacks, the adversary has access to the learning model architecture. It can modify the learning model configurations (*e.g.*, learning rate). In gray-box attacks, the adversary can build a surrogate learning model using the surrogate training data. Conversely, the black box assumes that the adversary has no access to the targeted system architecture.

3) ORACLE

If the adversary can interact with the targeted model or a surrogate model, they can extract vital information that could help in crafting the adversarial attacks [43]. Here, the adversary can query the model multiple times and observe the outputs. This enables the adversary to identify any relationship between the inputs and the outputs. The use of an oracle is common for gray-box and black-box threat models.

C. ADVERSARIAL GOALS

The severity of any threat on a system asset is measured by the potential impact on these three objectives: confidentiality, integrity, and availability [44]. Depending on the business logic of the computer system, the integrity of the output (*i.e.*, predictions and classification) from a machine learning model is indispensable. For instance, an adversary can provide an adversarial example, yielding an incorrect output.

Based on the output incorrectness, the adversarial goals fall into three categories:

- **Untargeted Misclassification.** The adversary tries to increase the misclassification ratio for the DNN model by using the adversarial examples generated by untargeted adversarial attack as an input to produce an incorrect classification. In other words, the adversary tries to force the targeted model to assign any incorrect label to the adversarial examples.
- **Confidence Reduction.** The adversary tries to reduce the prediction confidence by increasing the prediction ambiguity of the targeted model.
- **Source/Target Misclassification.** The adversary tries to craft perturbations that force the classification of an adversarial examples to a specific label (i.e., assign a specific label to an adversarial input) [45]. To achieve this objective, the adversary may use targeted adversarial attacks.

D. ADVERSARIAL ATTACK SCENARIOS

The adversarial attacks against ML-learning systems can be launched either at the training or the testing phase as explained below.

1) ATTACKS DURING TRAINING PHASE

An adversary can compromise the targeted machine learning system by attacking its training dataset during the training phase. This attack scenario is referred to as ***poisoning attack*** in which the adversary tries to corrupt the training dataset to change the statistical characteristics of the training data. This poisoning attack can be launched using the following scenarios:

- **Data Injection & Modification.** The adversary tries to purposefully inject malicious examples into the training dataset and/or maliciously alter the training dataset samples. The malicious examples can be generated using the *label noise* approach [46]. Upon poisoning the training dataset, the trained DNN model generates incorrect outputs [47].
- **Transfer Learning.** The adversary tries to spread malicious examples to other learning methods using transfer learning [48]. This approach has the least impact when attacking a deep learning model because the model is trained using clean data.
- **Logical Corruption.** The adversary intervenes with the learning process of the DNN model to prevent it from learning correctly.

2) ATTACKS DURING TESTING PHASE

In some situations, an adversary can utilize the characteristics of the underlying classes that can be changed without affecting the true classification [48]. Mainly, two attacking scenarios can be carried out at the testing phase:

- **Evasion Attack.** In this attack scenario, the adversary tries to compromise the targeted model by carefully crafting a malicious input sample that is misclassified by the ML model. This kind of attack has been adopted by most of the proposed adversarial works [49].
- **Exploratory Attack.** Like side-channel attacks, the adversary tries to extract information about the learning system at testing time. During this attack, the adversary probes the DNN model to extract information about its parameters, features, architecture, or training datasets by crafting adversarial examples.

V. ATTACK STRATEGIES

In this section, we review state-of-the-art adversarial attacks that compromise the image classification neural networks. We divide the attacks into five categories: white-box attacks, black-box attacks, poisoning attacks, extraction attacks, and inference attacks. In our review, we focus on providing a deep explanation of the working process of these attacks in addition to their performance evaluation. Table 1 provides a quick overview of the reviewed attacks and demonstrates a comparison between them in terms of attack effectiveness, transferability, and other performance metrics such as execution time and the size of the attack perturbation. It should be noted that the majority of the reviewed attacks are focused on crafting perturbations to compromise DNNs. This is because adding perturbations into a clean image appears to be an effective attack method in which the generated adversarial examples are almost identical to the clean images (see Fig. 9) which attracts the research community.

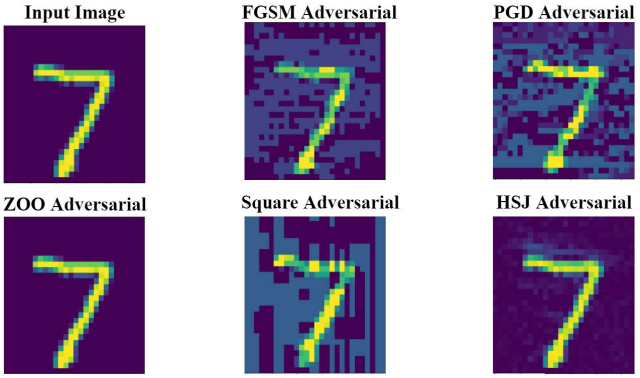

FIGURE 9. A sample of generating different adversarial examples using state-of-the-art attacks. In this example, an input image was selected from the MNIST dataset and had the perturbation generated using the FGSM [28], PGD [50], ZOO [51], Square [52], and HJSA [53] attack methods.

A. WHITE BOX ATTACKS

1) L-BFGS

Adversarial examples were first introduced by Szegedy *et al.* [12] who found that adding a small perturbation ρ to an image x would result in an adversarial image $\hat{x}$ that could successfully fool a deep learning model. To compute the proper size of perturbations, the authors attempted to solve the following optimization problem.

$$\min \|\rho\|_2 \text{ s.t.} f(x+\rho) = l; x+\rho \in [0,1]^m \qquad (12)$$

However, the equation above is difficult to solve, as a result, the Box-Constrained L-BFGS [54] was used to find an estimation for the solution as shown in Eq.13. This is done by finding the minimum value that satisfies the condition $f(x+\rho) = l$ while calculating the loss of the classifier.

$$\min c \cdot |\rho| + L_f(x+\rho, l) \quad \text{s.t.} x+\rho \in [0,1]^m \qquad (13)$$

The authors observed that the adversarial examples generated by the box-constrained L-BFGS appear almost identical to the original images (*i.e.*, imperceptible perturbation). They also noted that the resulting adversarial examples can fool other DNN models (*i.e.*, transferable adversarial examples). The results of their work triggered concerns on the security of deep learning systems and established a wide interest in researching adversarial machine learning.

2) FAST GRADIENT SIGN METHOD

FGSM [28] has been proposed as an efficient algorithm that can generate perturbations for any given image. Compared with L-BFGS [12], FGSM differs in two aspects: (1) it's adversarial examples are measured by L_∞ metric, and (2) it is intended to be a fast method for generating adversarial

examples that can be used in adversarial training to increase the robustness of DNN models against adversarial attacks.

More formally, given an image x, FGSM calculates the perturbation using the formula below,

$$\hat{x} = x - \epsilon \cdot sign(\nabla_x J(\theta, x, y)) \tag{14}$$

where $\nabla_x J(.)$ is the gradient of a cost function (*e.g.*, categorical cross-entropy) of x for a given neural network and ϵ is a small error that is used to constrain the size of the perturbations. In other words, depending on the sign of the gradient, FGSM generates $\hat{x}$ by adding or subtracting ϵ to each pixel of x. FGSM was able to generate effective adversarial examples that can fool different DNN models. However, it can be observed from the reported results in [27] that FGSM requires adding a large size of perturbations to generate powerful adversarial examples which may distort the image resolution.

3) DeepFool ATTACK

Moosavi-Dezfooli *et al*. [27] proposed the DeepFool attack, an iterative algorithm optimized by L_2 distance metric. The DeepFool algorithm was designed under the assumption that deep learning models are linear, with a decision boundaries (i.e., hyperplanes) separating each class. This algorithm centers around an iterative linearization of the classifier f that will produce small perturbations ρ for an input image x. On every iteration, DeepFool linearizes the classifier around the current point x_i and computes the perturbations ρ_i as an orthogonal projection vector that projects x_i onto the closest hyperplane. As depicted in Fig. 10, DeepFool has generated the minimum perturbations that project the ''Cauliflower'' image into the nearest hyperplane ''Broccoli''. DeepFool has been shown to be an effective algorithm for generating the smallest perturbations necessary to fool the targeted deep learning model.

4) CARLINI & WAGNER ATTACKS

Carlini and Wagner [26] introduced a set of three attacks known as the C&W attacks. Their motivation was to build a set of powerful attacks that are capable of bypassing defensive distillation image classification neural networks [29]. The C&W attacks generate adversarial perturbations by solving a norm-restricted constrained optimization problem. The C&W attacks find the adversarial example by solving the following optimization problem:

$$\min \|\rho\|_p + c \cdot f(x + \rho), \text{ s.t. } x + \rho \in [0, 1]^m \tag{15}$$

where x is the input image, ρ is the adversarial perturbation, $\|.\|_\rho$ is a regularization term bounded by an L_p norm, c is a constant value, and $f(x + \rho)$ is an objective function. This attack can be implemented with either the L_0, L_2, or L_∞ distance metrics.

Furthermore, the authors has demonstrated that it is possible for adversarial images generated on un-distilled networks to transfer to distilled networks, effectively bypassing the defensive distillation method. The C&W attacks were able to successfully subvert distilled networks trained on the CIFAR-10 [55] and MNIST [56] datasets with a 100% success rate. Also, C&W attacks were able to generate transferable adversarial examples. Hence, C&W attacks are considered powerful attacks that can be utilized to validate the well-trained DNN models. However, these attacks require huge computational resources on large-scale datasets [51].

5) ITERATIVE FAST GRADIENT SIGN METHOD

Kurakin *et al*. [57] proposed a set of novel attacks that extend the FGSM [28] algorithm to operate iteratively. The authors introduce two methods for generating adversarial perturbations: the basic-iterative method (BIM) and the iterative least-likely class method (ICLM).

Adversarial images generated using BIM are provided by solving the following formula:

$$\hat{x}_{i+1} = Clip_{x,\epsilon}\hat{x}_i + \alpha \cdot sign(\nabla_x J(\hat{x}_i, y)) \tag{16}$$

where $\hat{x}_i$ is the adversarial example at the i^{th} iteration, BIM will find the next image $\hat{x}_{i+1}$ and repeat for the number of iterations, determined heuristically. Therefore, the BIM algorithm minimizes the computational cost while being strong enough to reach the edge of the decision boundary, yielding to misclassifying $\hat{x}$.

The ICLM attack further extends the BIM to generate a targeted attack. The ICLM differentiates itself from BIM by generating a perturbation for the least likely class of x. Adversarial images generated using the ICLM are created using the formula below.

$$\hat{x}_{i+1} = Clip_{x,\epsilon}\hat{x}_i - \alpha \cdot sign(\nabla_x J(\hat{x}_i, y_t)) \tag{17}$$

where y is the class label used in Eq.16 replaced with the target label y_t that corresponds to the least likely class with the lowest confidence score predicted by the model. ICLM uses the same number of iterations and step size as BIM. The adversarial examples generated by the ICLM attack can fool a given model and lower its classification accuracy.

6) UNIVERSAL ADVERSARIAL PERTURBATIONS

Popular adversarial machine learning algorithms like FGSM and DeepFool generate perturbations that attack a network on a single image. The universal adversarial perturbations (UAP) [14] method generates universal perturbations that can attack a network with any image. Perturbations are universal if the perturbation ρ satisfies the following constraint:

$$P(f(x + \rho) \neq f(\rho)) \geq 1 - \delta \text{ s.t. } \|\rho\|_p \leq \xi \tag{18}$$

where f is a classification function, $\|\cdot\|_p$ is the L_p norm, δ is the desired fooling rate, and the parameter ξ is responsible for the magnitude of the perturbation ρ.

More specifically, generating a perturbation ρ that can fool most data points in an image set $X = \{ x_1, \ldots, x_n \}$ can be done by iterating over the images in X and gradually building up the UAP. The authors generate universal perturbations in a

FIGURE 10. An adversarial example was generated using DeepFool [27] Attack. The perturbation image in the middle is magnified.

similar way as in the DeepFool [27] algorithm, they gradually push a single data value towards the closest hyperplane. In this case, the UAP method consecutively pushes all the input data towards their respective hyperplanes.

On every iteration, the algorithm calculates the minimum perturbation $\Delta\rho_i$ that will project the current value closer towards the hyperplane. The minimum perturbation is computed by solving a constrained optimization problem that limits the L_p norms. The perturbation ρ is updated by projecting $(\rho + \Delta\rho_i)$ back onto ρ. The algorithm iterates multiple times over the data in the set X to enhance the UAP.

The attack terminates when the fooling rate on the adversarial dataset $X_\rho = \{ x_1 + \rho, \ldots, x_n + \rho \}$ surpasses the target threshold. Multiple random shuffles on the original image set will result in a diverse set of UAPs. Therefore, this algorithm can be used to generate numerous UAPs that are highly effective against deep neural networks.

7) JACOBIAN-BASED SALIENCY MAP ATTACK

JSMA [58] is a first-order adversarial attack that generates adversarial examples optimized by L_0 distance metric. Instead of applying the perturbations to the entire image's pixels, JSMA aims to modify the ''enough pixels'' to fool the model. At a high level, JSMA generates adversarial examples using a well-defined process. It selects the pixels of a clean image x and perturbs them one at a time (*i.e.*, greedy perturbations) and checks the results of the labeling. JSMA calculates the gradient of the output from the network to generate a saliency map. A large value of this map increases the likelihood of the model to label the image as the target label. Upon computing the saliency map, JSMA determines the required pixels that need to be perturbed to fool the model. JSMA repeats this process until the model is fooled or it exceeds the maximum threshold of pixels that can be perturbed without making the perturbations detectable.

8) PROJECTED GRADIENT DESCENT

PGD [50] can be viewed as an extension to FGSM [28] and BIM [57] attacks that generate adversarial examples to maximize the loss of the targeted DNN model. The authors propose two versions of PGD: the L_2 PGD attack, and the L_∞ PGD attack. PGD initializes the attack at a random point in the L_p ball and projects the perturbation back onto the L_p ball after every iteration. The authors also has shown that adversarial robustness of DNN models can be viewed in terms of ''robustness optimization''. As shown in E.q 19, they defined the adversarial training as a formal optimization problem, known as the saddle point problem.

$$\min \rho(\theta), \text{ where } \rho(\theta) = E_{(x,y)\sim D}[\max L(\theta, x + \delta, y)] \quad (19)$$

where $E_D[L(.)]$ is the population risk for a distribution value D into a loss function L. The saddle point optimization problem is an arrangement of an inner maximization problem and an outer minimization problem [50]. Inner maximization finds an adversarial data point that maximizes the loss. The outer minimization finds the model parameters such that the loss generated by the inner function is minimized. Moreover, Eq. 19 also defines a goal for an ideal robust classifier as well as a measurable value of the classifiers robustness. PGD is a powerful first-order attack and has been shown to fool the deep learning models efficiently and effectively [59].

9) NewtonFool ATTACK

The NewtonFool algorithm [60] is used to decrease the probability of the original class label by utilizing Newton's method for solving nonlinear equations. This attack performs gradient descent with step size δ to find a perturbation ρ that will produce an adversarial example $\hat{x}$. The step size is determined adaptively, changing over time according to the change in the perturbation ρ. The step size δ is computed by solving the following equation:

$$\delta = \min \left\{ \eta \|x_0\| \|\nabla F_s^l(x_i)\|, F_s^l(x_i) - \frac{1}{|C|} \right\} \quad (20)$$

where the tuning parameter η controls the size of ρ, x_0 is the input image, and F_s^l represents a neural network with a softmax activation layer. The step size δ is then utilized to calculate the adversarial perturbation ρ as follows,

$$\rho = -\frac{\delta \cdot \nabla F_s^l(x_i)}{\|\nabla F_s^l(x_i)\|^2} \quad (21)$$

where x_i is the current image, δ is the step size calculated in Eq. 20, and ∇F_s^l is the gradient of the classifier. The authors extend the attack to work with multiple class labels. NewtonFool decreases the probability of all labels in a set of clean images L_+ and increases the probability of all labels in a set of perturbed images L_-. NewtonFool produces effective perturbations and significantly reduces the confidence probability of the correct class.

10) ELASTIC NET

The elastic net attack (EAD) [30] is an extension of the C&W attacks [26] and aims to control the L_1 norm of the adversarial perturbations. EAD generates adversarial examples using the iterative shrinkage-thresholding algorithm (ISTA) [61]. ISTA is an optimization algorithm where every iteration includes a matrix-vector multiplication step accompanied by a shrinkage-thresholding step. The shrinkage-thresholding step is responsible for deciding if the algorithm will shrink a pixel value of the perturbed image.

The ISTA algorithm can be seen as an ordinary first-order optimization problem. Specifically, if we let a function $g(x) = c \cdot f(x) + \|x - x_0\|_2^2$ and $\nabla g(x)$ be the gradient of $g(x)$, the adversarial image x^{k+1} for the input image x_0 is determined by:

$$x^{k+1} = S_\beta(x^k - \alpha_k \nabla g(x^k)) \tag{22}$$

where α_k is the step size at the $(k+1)_{th}$ iteration, and S_β is the projected shrinkage-thresholding function. The final adversarial image $\hat{x}$ is chosen from all the successful examples based on distortion metrics. The authors propose two decision rules for selecting $\hat{x}$, the least elastic-net and the L_1 distortions relative to the input image x. The EAD attack, like its predecessor the C&W attack can successfully bypass defensive distillation [29].

11) TARGETED UNIVERSAL ADVERSARIAL PERTURBATIONS

Universal perturbations for targeted attacks proposed by Hirano *et al.* [62] extends the basic iterative algorithm used to generate universal perturbations for untargeted attacks. This algorithm begins with the perturbation $\rho = 0$, and iteratively updates ρ under the constraint that the L_p norm of the perturbation will be less than or equal to a value ϵ.

As compared to the untargeted UAP [14] algorithm which utilizes the DeepFool [27] method for generating perturbations, the attack uses the targeted FGSM [28] method to generate the perturbations. The perturbation ρ is updated additively by generating a perturbation for a randomly selected input x from the set X. On every step, the targeted-FGSM algorithm computes a new perturbation $\psi(x + \rho, y)$ and projects this back onto ρ updating the perturbation.

This process continues until the maximum number of iterations i_{max} is reached or the success rate of the targeted attack is equal to 100%, *i.e.*, all inputs are classified with the correct target class. This attack generates universal perturbations that were shown to be effective at undermining models trained on the CIFAR-10 [55] and ImageNet [6] datasets. This algorithm also has shown that generating a UAP for a targeted attack is easier and less computationally expensive when compared to an untargeted attack [62].

12) BRENDEL & BETHGE ATTACK

The Brendel & Bethge attack [63] utilizes the gradients to estimate the boundary between the perturbed and clean inputs, better known as the adversarial boundary. Unlike other attacks, this attack initiates with an adversarial input $\bar{x}^0$ as well as a clean input x. Both inputs are far from one another within the adversarial boundary. Therefore, $\bar{x}^0$ travels along the adversarial boundary towards x. The attack computes the optimal step by solving a quadratic trust-region optimization problem for every iteration.

The goal of the optimization problem is to discover a step δ^k such that the new perturbation $\bar{x}^k = \bar{x}^{k-1} + \delta^k$ has the minimum L_p distance to the clean input x. The new perturbation $\bar{x}^k$ will stay between the box constraints of a valid input range. The perturbation will be placed on the adversarial boundary. The Brendel & Bethge attack moves along the adversarial boundary to minimize the distance to the clean input. The Brendel & Bethge attack is a proficient algorithm, achieving model accuracies of 69.5% in an untargeted setting on the MNIST [56] dataset, 31.2% on the CIFAR-10 [55] dataset, and 42.5% on ImageNet [6]. Upon utilizing a targeted setting, this attack achieves model accuracies of 56% against MNIST, 37.6% on CIFAR-10, and 37% on ImageNet.

13) WASSERSTEIN ATTACK

Instead of using the L_p distance metrics, the Wasserstein attack [64] generates adversarial examples by using a minimized wasserstein distance [65]. The Wasserstein distance is an optimal transport problem that finds the minimum cost of moving a probability mass. When applied to images, the Wasserstein distance is identified as the cost of moving from one image to another. The cost value is directly proportional to the distance traveled by the pixels. The Wasserstein distance d_W between the two data points x and y is defined as follows:

$$d_W(x, y) = \min \langle \Pi, C \rangle \,; \text{s.t.} \Pi 1 = x, \Pi^T 1 = y \tag{23}$$

where the minimization over transport plans Π, with entries $\Pi_{i,j}$ show how the mass moves from x_i to y_j. With this, the Wasserstein ball with radius ϵ is defined as:

$$B_W(x, \epsilon) = \{x + \Delta : d_W(x, x + \Delta) \leq \epsilon\} \tag{24}$$

The first step in generating adversarial images is to project examples onto a Wasserstein ball. More specifically, projecting w onto the Wasserstein ball around x with a radius ϵ and a cost matrix C.

$$\min \frac{1}{2} \|w - z\|_2^2 \,; \text{s.t.} \Pi 1 = x, \Pi^T 1 = z, \langle \Pi, C \rangle \leq \epsilon \tag{25}$$

Solving the optimization problem for the Wasserstein attack in Eq. 25 is time-consuming and computationally expensive [64]. The authors proposed solving the entropy-regularized projection problem shown in Eq. 26 which efficiently projects the examples onto the Wasserstein ball.

$$\begin{aligned} \min \quad & \frac{1}{2} \|w - z\|_2^2 + \frac{1}{\lambda} \sum_{ij} \Pi_{ij} \log(\Pi_{ij}) \\ \text{subject to} \quad & \Pi 1 = x, \Pi^T 1 = z \, \langle \Pi, C \rangle \leq \epsilon \end{aligned} \tag{26}$$

Although this is only an estimated projection, all feasible solutions are still within the Wasserstein ball's limits. Therefore, adversarial examples generated using projection approximation are still valid as they lie within the boundaries of the attack's threat model. The Wasserstein attack has been shown to generate adversarial images that are capable of adequately fooling deep learning models trained on the MNIST [56] and CIFAR-10 [55] datasets.

14) SHADOW ATTACK

The shadow attack [66] forces neural networks to misclassify images and produce a spoofed certificate by generating and applying large perturbations. This attack is invented to break the certifiable secure neural networks that generate the predicted label with a rigorous guarantee that the input is not maliciously manipulated (i.e, certificate). The shadow attack is a generalization of the projected gradient descent [50] attack. Instead of solving a constrained optimization problem, as done in PGD, the shadow attack solves the following problem with the addition of three penalties that force Eq. 27 to be unconstrained.

$$\max_{\rho} L(\theta, x+\rho) - \lambda_c C(\rho) - \lambda_{tv} TV(\rho) - \lambda_s Dissim(\rho) \quad (27)$$

where $\lambda_c, \lambda_{tv}, \lambda_s$ are the three penalty weights. The penalty $C(\rho)$ limits the perturbation ρ by constraining the change in each color channel c. The penalty $TV(\rho)$ forces ρ to have a small total variation to appear smoother and more natural. The last penalty $Dissim(\rho)$ advances perturbations that share similar values in their color channels. These penalties also allow for larger perturbations in the L_p norm. This algorithm generates adversarial examples for every possible incorrect class y' and chooses the optimal perturbation for the attack.

The shadow attack has been shown to be an effective method for generating adversarial attacks achieving a success rate of 98.5%. However, this attack was tested under very specific constraints, testing only specific target class IDs in the ImageNet [6], and a small sample size from the CIFAR-10 [55] dataset.

B. BLACK BOX ATTACKS

1) BOUNDARY ATTACK

The boundary attack [67] performs a rejection sampling along the decision boundary of the sets of images. This attack seeks to find the minimum perturbation necessary to misclassify the target image by slowly moving towards the hyperplane of the original input. Initially, the boundary attack is set with a large perturbation that is easily detectable and randomly walks along the decision boundary towards the target class, effectively optimizing the original perturbation. This attack has two parameters, the length of the perturbation ρ and the step δ towards the initial image. Both of them adjust to the local geometry of the boundary. If the perturbation is adversarial (*i.e.*, misclassifies the target image) there is a small step towards the original input image. As the algorithm approaches the input image, the decision boundary becomes flatter, and δ must be smaller to continue to make progress. The attack converges when δ converges to zero. The boundary attack is a strong method for targeting deep learning models, outperforming gradient-based white-box attacks such as FGSM [28] and DeepFool [27]. This attack has shown effective in undermining networks trained on the MNIST [56], CIFAR-10 [55], VGG-19 [68], ResNet-50 [7], and ImageNet [6] datasets.

2) ZEROTH ORDER OPTIMIZATION ATTACK

The zeroth-order optimization (ZOO) attack [51] utilizes zeroth order stochastic coordinate descent to generate perturbations. The authors adapted the Carlini & Wagner attacks [26] to the black-box threat model by modifying the loss function and approximating the gradient. The authors propose a new loss function $f(x, t)$ dependent on only the outputs and the target class label. The attack optimizes the new loss function using zeroth-order optimization. Then, ZOO approximates the gradients of the model using stochastic coordinate descent instead of the traditional backpropagation method. Zeroth-order stochastic coordinate descent is utilized to target the model and extract information about the gradients. In special cases, techniques such as dimension reduction, hierarchical attacks, and importance sampling are utilized to optimize the loss function. The ZOO attack is extremely effective, with success rates of 100% in an untargeted setting on both the MNIST [56] and CIFAR-10 [55] datasets. Upon utilizing a targeted setting, the ZOO attack success rate is 98.9% against MNIST and 97% on CIFAR-10.

3) SPATIAL TRANSFORMATION ATTACK

As the name implies, the spatial transformation attack [69] distorts the input image to generate adversarial examples by performing one translation and one rotation of the image. To ensure visual similarity to the clean images, the perturbation space is restricted to only allow 30° max rotation and a 10% max translation in every direction. The optimal perturbation is calculated via hyperparameter optimization, better known as grid search. Grid search is an extensive process in which a subset of the hyperparameter space is searched to find the optimal parameters, in this case, the perturbation, for a given model. The combination of rotation and translation parameters is applied to the entire group of input images. In a sense, the perturbation found by the spatial transformation attack is universal. The spatial transformation attack is able to achieve remarkably high results and fool multiple deep learning models trained on the MNIST [56], CIFAR-10 [55], and ImageNet [6] datasets.

4) UPSET AND ANGRI ATTACKS

Sarkar *et al.* [70] introduced two adversarial attacks: UPSET and ANGRI. The UPSET (Universal Perturbations for Steering to Exact Targets) method seeks to generate n universal perturbations ρ for n target classes such that when ρ is applied to an image, the image is misclassified to the target class. UPSET generates an efficient and robust perturbation with the use of a residual generating network. The UPSET network R accepts the target class as input and generates a perturbation ρ that is applied to the input image x to fool the network. The adversarial image $\hat{x}$ is generated by solving an optimization problem using the UPSET network U, where pixel values of x are normalized to stay within the range $[-1, 1]$. All values outside this range are clipped to guarantee that $\hat{x}$ is

valid. In contrast to the universal perturbations generated by UPSET, ANGRI produces an image-specific targeted perturbation that depends on the input image. The adversarial image is generated using the ANGRI network A. The UPSET and ANGRI networks are built upon a residual generating network such as the ResNet [7] architecture. Both algorithms produce perturbations suitable for targeted fooling, and both attacks have high fooling rates against the MNIST [56] and CIFAR-10 [55] datasets.

5) HOUDINI ATTACK

Common algorithms utilize the gradients of the loss functions to generate perturbations. However, the Houdini attack [71] generates adversarial examples that handle task losses task-specific loss functions, which are combinatorial values that are difficult to optimize. This makes it difficult to use gradient-based solutions to generate perturbations. The normal optimization problem is replaced with a differentiable surrogate loss function $\bar{l}(y_\theta(\tilde{x})), y$, referred to as Houdini.

Houdini is composed of two parts. The first is a stochastic margin that calculates the probability that the difference between the score of the predicted target and the actual target is smaller than a given value. This represents the total model's confidence. The second part to Houdini is the task loss, which is independent of the model and corresponds to the target that will be maximized. Houdini is designed to generate effective adversarial images that can fool a given model, but it has also been shown to be effective against speech recognition systems. Effective targeted and untargeted attacks were able to be generated to attack a DNN that estimated human poses.

6) SIMPLE BLACK-BOX ADVERSARIAL ATTACK

In many cases, white box assumptions of a model are not applicable in real-life attack scenarios. Therefore, an adversary is more likely to utilize a black box threat model. Black box attacks are harder to develop since the adversary has no prior knowledge or access to the targeted DNN model which further intricate finding the optimal perturbations. Furthermore, querying the model for information is time-consuming and reduces attack efficiency. For these reasons, the simple black box attack (SimBA) [72] was introduced an an attempt to tackle the inherent limitations in the black box threat model. SimBA assumes that an input image x is fed through a neural network F that classifies the image using a confidence value or an output probability $p_F(y|x)$. The intention is to find a sufficiently small perturbation ρ that when added to the input will result in an incorrect prediction.

Model information is unavailable to the adversary in a black-box environment. Thus, the output probabilities are used to estimate the adversarial examples. As input, SimBA takes the targeted label, a set of candidate vectors, and a step size $\delta > 0$. Then, the attack randomly picks an orthogonal direction q from the set of candidate vectors. SimBA guarantees query efficiency by assuring that no two previously selected q values undermine each other or amplify any other q disproportionately. Therefore, the orthogonal direction q is selected from a restricted set of vectors and is chosen without replacement.

The SimBa attack algorithm is an effective method for generating adversarial examples. This attack was tested on a deep learning model trained on the ImageNet [6] dataset, and was able to achieve success rates of 98.6% and 100% in an untargeted and targeted attack setting respectively. The SimBa attack is also able to achieve an extremely low number of average queries into the model compared to similar black box algorithms, 1665 queries in an untargeted setting and 7899 queries in a targeted setting.

7) ONE-PIXEL ATTACK

Designed to operate under the strict conditions of only modifying a single pixel from an image to fool the DNN models. One-Pixel attack [13] generates adversarial examples through differential evolution [73] optimization. Differential evolution allows for the one-pixel attack to generate effective adversarial examples while lacking any information about the network or gradient. For a given clean image x, a set of 400 vectors are created such that every vector contains a coordinate pair as well as RGB values for a random pixel. The attack changes the components of the vectors at random, creating child vectors to compete with the parent vectors to stay for the next iteration of the algorithm. The probability labels calculated by the network are used as the filter for the vectors. This will continue until there is only one remaining child vector. The selected vector is then used to modify the pixel in the image.

The reported results demonstrate that, one-Pixel attack can generate adversarial examples to fool DNN models. However, in [74], the authors argued that, One-Pixel attack requires huge computational resources and cannot be applied on large images.

8) FEW-PIXEL AND THRESHOLD ATTACKS

Kotyan and Vargas [75] introduced a set of two adversarial attacks, the few-pixel attack and the threshold attack, which extend the One-Pixel attack [13].

The threshold attack, also known as the L_∞ black-box attack, optimizes a constrained optimization problem using the L_∞ norm. This attack applies the small perturbation ρ slightly to all pixels. The optimization problem is constrained with $\|\rho_x\|_\infty \leq th$, where th is a predefined threshold value. The threshold attack searches for variables in the algorithm search space $\mathbb{R}^k$, which is the same as the input space. The variables can be any variation of the inputs if the threshold is not crossed.

The few-pixel attack attempts to minimize the number of the perturbed pixels by optimizing a constrained optimization problem with the L_0 norm. The search space for the few-pixel attack is smaller than the input space and searches for variables in the search space $\mathbb{R}^{(2+c)*th}$. The fundamental difference between this attack and the threshold attack is the use of a different L_p norm, and a different search space.

Both attacks were tested using the CIFAR-10 [55] dataset with multiple threshold values. In both cases the attack performed at its best with $th = 10$, with the L_0 for ResNet [7] scoring 85% and 79% on adversarial accuracy with the CMA-ES [76] and differential evolution [73] optimization algorithms respectively. Whereas the L_∞ attack scored slightly lower on the CMA-ES optimization algorithm achieving an adversarial accuracy of 83%, the differential evolution optimization algorithm for the threshold attack scored slightly higher at 82%. Overall, both of these attacks work fairly well at fooling deep learning models, albeit they were only tested against two models and one dataset leaving out potential unknown results.

9) HopSkipJump ATTACK

Chen *et al*. [53] developed the HopSkipJump attack (HSJA) to generate adversarial examples by estimating the direction of the gradient using binary information at the decision boundary. HSJA finds the optimal perturbation without the use of hyperparameters while maintaining query efficiency. HSJA can be targeted and untargeted attack. For untargeted attacks, HSJA can be initialized with a sample from the target class, whereas for targeted attacks, it uses a misclassified sample with added noise.

The HSJA will repeat for t iterations or until the optimal adversarial perturbation is generated. Each iteration has three main steps: (1) binary search, (2) gradient-direction estimation, and (3) step size search. First, binary search pushes the value from the previous iteration towards the hyperplane. Then, the direction of the gradient is estimated using the following formula,

$$\widehat{\nabla S}(x_t, \rho) := \frac{1}{B-1} \sum_{b=1}^{B} (\phi_{x^*}(x_t + \rho u_b) - \overline{\phi_{x^*}}) u_b \quad (28)$$

where x is the input image, ρ is the adversarial perturbation, the number of batches B, u_b is a unit vector, and ϕ_{x^*} is a binary function. Lastly, the step size is initialized and updated along the direction of the gradient and decreased through geometric sequencing until the perturbation is successful. The next iteration begins with the adversarial example projected back onto the hyperplane.

The HSJA was evaluated on four datasets, MNIST [56], CIFAR-10 [55], CIFAR-100 [55], and ImageNet [6], and was shown to outperform previous decision based attacks such as the boundary attack [67]. With a limited number the of queries, the HSJA is able to generate adversarial images with a smaller median distance over all the datasets. The HSJA was also shown to minimize the number of queries used, this attack was able to achieve a 70% success rate after 1000 queries, where the boundary attack required 20000 queries to get similar results.

10) ColorFool ATTACK

ColorFool attack [77] aims to generate adversarial examples using natural colors by producing low-frequency perturbations that are highly transferable and robust. The ColorFool attack starts by identifying the most important areas of an image and their specific colors. Then, it classifies those regions as either sensitive or non-sensitive. Sensitive regions are classified into four categories, person, sky, plants, and water, while the non-sensitive regions would be anything that does not fall into those categories. Sensitive regions must stay within a specific range of modification, while non-sensitive regions can be modified more inconsistently and still look normal.

After identifying the different regions, ColorFool splits an image x into k semantic regions using a binary mask that identifies the position of the pixels belonging to the region. The colors of each set are modified in color space, which separates the brightness from the color. Natural color ranges a, b, and L are used to pull apart the color values, where a ranges from red to green, b from blue to yellow, and L from black to white.

The colors of the sensitive regions are then modified and converted from *RGB* to the color space. The adversarial perturbations in the color channels a and b are randomly chosen from the set of natural color ranges. The color ranges are determined by the true colors, the region semantics, and previous information on color perception. The colors are changed iteratively with small intervals until the optimal perturbation fools the classifier. Then, the colors of the non-sensitive regions are modified in the same way as the sensitive ones, but the color values are from the entire range of a and b in order to endure larger changes. Finally, the adversarial image $\hat{x}$ is generated by combining the two modified color regions into one image. The adversarial image is then converted back into *RGB* form from color space form and is multiplied by a function to ensure that the image is in the original range of pixel values.

The ColorFool attack is a strong algorithm that is able to effectively undermine deep learning models trained on the CIFAR-10 [55], ImageNet [6], and P-Places365 [78] datasets. For example, on CIFAR-10 trained models trained with a softmax activation function, ColorFool is able to achieve a success rate of 99.4%. On models trained with the prototype conformity loss (PCL) [79] method as well as PCL with adversarial training, ColorFool was able to get success rates of 100% and 99.9%, respectively. Overall, this algorithm is an impressive and effective method of undermining deep learning models.

11) SQUARE ATTACK

The square attack [52] is modeled on random search, an iterative optimization technique. This attack differentiates itself from other random search-based attacks by iteratively generating perturbations that lie on the L_2 or L_∞ boundaries before projecting them onto the image. As a result, the perturbation can be maximized on every iteration. The attack updates the image at each step modifying a small percentage of neighboring pixels grouped into a square.

The square attack initializes by choosing the side length h^i of the pixel square that will be updated. h^i decreases

according to a fixed schedule. Then, a new perturbation ρ is discovered and subsequently added to the current iteration. The loss value is re-calculated and if the resulting value is smaller than the previous loss value, the perturbation is accepted. The square attack works with both the L_2 and L_∞ distance metrics. This attack focuses on query efficiency to terminate the algorithm as soon as the optimal perturbations is generated. The square attack is a strong algorithm that can efficiently undermine a deep learning model, with a success rate of 99.7% against Inception v3 [80], 100% against ResNet-50 [7], and 100% against VGG-16-BN [68]. Query efficiency of the square attack is outstanding, when tested against Inception v3 the average number of queries is 197, 73 against ResNet-50, and 31 against VGG-16-BN.

C. POISONING ATTACKS

1) POISONING ATTACK ON SVM

If the adversary has full access to the model's learning algorithm and training data, they can effectively create poisoning attacks that can be used to successfully target support vector machines (SVM) [81]. This algorithm can be kernelized but depends entirely on the use of the gradients of points lying within the input space. The adversary utilizes an iterative gradient ascent method to optimize the non-convex objective function of the model. Gradient ascent as opposed to gradient descent takes steps that are proportional to the positive gradient (rather than the negative gradient), such that it approaches the local maximum instead of the local minimum.

The adversary initiates the attack with a vector that replicates a random point from the targeted class and changes its classification label. In practice, about any point deep enough in the adversarial class's margin can be used to initiate the attack. Then, the gradient ascent algorithm is used to find the gradient of the validation error. During the update, it is imperative to preserve the architecture of dataset used to train the SVM classifier. In most cases when gradient ascent is used, a linear search algorithm is used to find the optimal solution. In this case, a large step size is required to secure the training sets, and linear search would be computationally expensive. To circumvent this, the attack fixes the step size to a given constant value. After every update, the optimal solution is re-calculated. This attack terminates once the validation error is less than a certain threshold value. The attack can substantially increase classification error rates, from an initial error rate of 2-5% to 15-20% using a single adversarial data point.

2) ADVERSARIAL BACKDOOR EMBEDDING

Generally speaking, backdoor detection algorithms such as activation clustering [82] are effective against a majority of backdoor attacks. However, they fail to consider more robust adversarial models. The adversarial backdoor embedding attack [83] exploits these weak defenses by including a secondary loss function to the objective training function. The secondary loss function serves as a penalty term that punishes the model when it detects any difference between perturbed and non-perturbed images. Dual objective functions allow the adversary to return high classification accuracy for the model while setting constraints that weaken the model's defenses. As the adversarial training converges, the distribution of backdoor inputs, as well as clean inputs, also converges — minimizing the differences that the defense systems use for detecting poisoning attacks.

3) INPUT MODEL CO-OPTIMIZATION ATTACK

Despite the differences between adversarial examples and poisoned models, both threat models share the same goals of attacking a neural network and misclassifying input data. The input model co-optimization (IMC) attack [84] looks to unify the two threat models. The authors define a unified framework that gives adversaries the freedom to either generate adversarial examples or to poison the model. The attack generates adversarial examples $\hat{x}$ for every input x in a dataset. The perturbed image is then misclassified to a specific target class by a poisoned model. The IMC attack finds the optimal adversarial example and the poisoned model by going back and forth between the model and the input perturbation until the attack converges. It is worth noting that the IMC attack can work in various attack scenarios by adjusting the base algorithm to meet those constraints, one example is the TrojanNN [85] attack.

4) CONVEX POLYTOPE

Algorithms such as the feature collision attack [86] fail when the feature extractor is unknown to the adversary. Thus, the convex polytope attack [87] was introduced to bypass the limitations of such algorithms. This attack creates a set of adversarial examples that contain the target class within the convex hull. The convex polytope attack exploits the association made by the linear classifier of the targeted network between the adversarial examples and the targeted class. Then, the network will classify any point within the convex hull as the targeted class. The attack is highly transferable due to the convex polytope expanding the attack area. The attack will find the optimal adversarial examples by iterating through a specialized non-convex optimization problem 4000 times. The convex polytope attack has several inherent issues, such as scalability, robustness, and generalizability.

Due to its extremely slow execution time, convex polytope attack is considered non-scalable. Notably, it has two time-consuming processes: First, it checks whether the new coefficients have a smaller loss compared to the previous ones, and it checks this on each iteration while optimizing the coefficients. Second, whenever the new coefficients satisfy the previous condition, the convex polytope attack projects onto the probability simplex, a space in which each point represents a probability distribution. The convex polytope also faces other issues, specifically the robustness and generalizability of the attack. Once the target moves through the boundary into the convex polytope, there is no reason to continue the optimization process and move further into

the attack area. For this reason, the target will be close to the boundary of the adversarial polytope.

5) BULLSEYE POLYTOPE

The bullseye polytope attack [88] is a more efficient, transferable, and robust adaptation of the convex polytope attack [87]. This attack fixes the relative position of the target class to the convex hull of the adversarial examples, which allows the bullseye polytope to overcome the implicit issues of the convex polytope attack. Bullseye polytope pre-defines its coefficients as equal values instead of the inefficient way of generating the coefficients through optimization. Pre-assigning the coefficient values effectively alleviates the most time-consuming step of the algorithm. Therefore, bullseye polytope is an order of magnitude faster than the convex polytope attack. The bullseye polytope attack also pushes the targeted class towards the center of the attack zone, improving the overall transferability.

D. EXTRACTION ATTACKS

1) COPYCAT NETWORKS

Correia-Silva *et al.* [89] propose a method with the intent to copy a targeted network into a copy-cat version by only querying the network with clean images. The copycat method goes through two phases: generating fake information and then training the copycat network.

The adversary generates a mock dataset by first selecting a large set of images, these images can originate from the targeted network, or they can be from an unrelated image set. The adversary selects the data collection method depending on the amount of access they have over the targeted model. The original image labels are useless to the adversary and are discarded from all image sets. The primary objective for the adversary in this phase is to observe the way images are classified by the targeted network. This is accomplished by feeding the new dataset into the target model and having it classify all the images. The newly generated image labels are known as the ''stolen labels''. The adversary intends to capture the slight imperfections of the classifier. This allows for another network to be trained on the same dataset and produce similar results to the target model.

Upon generating the fake dataset and the stolen labels, the adversary trains a copycat network that mimics the original one in a well defined process: First, the adversary chooses a model architecture for the copycat network without requiring any prior knowledge of the targeted networks architecture. Then the adversary adapts the chosen model architecture to fit the target's problem domain. For example, the adversary can change the number of outputs of their network to match the number of classes in the target model. Ideally, the model is pre-trained and has randomized weights, however, this is not necessary. The final step for generating a copycat network is fine-tuning the generated model using the fake dataset and the stolen labels. This allows the adversary to simulate the original conditions of the target model and to generate the most effective adversarial examples possible to attack the target model.

2) FUNCTIONALLY EQUIVALENT EXTRACTION

Model extraction attacks directly target the most secret parts of a given model, its architecture, and parameters. Model extraction allows the adversary who was previously operating under a black-box threat model to effectively gain access to the model in a white-box threat model, this is achieved by extracting an exact copy of the oracle. Model extraction is one of the most difficult adversarial goals as the adversary is attempting to generate a copy of the model while they only have access to the inputs and outputs. Functionally equivalent extraction [90] looks to construct an oracle O' in such a way that,

$$\forall x \in X, O'(x) = O(x) \tag{29}$$

The functionally equivalent extraction method works on neural networks using the ReLU activation function. The algorithm is split into four steps. First, critical point search determines inputs to the network such that one ReLU unit is at a critical point. This is accomplished by sampling two values and putting them through a linear function. This function computes the slopes and intercepts of the input vectors and then calculates the intersection of the two vectors. If there happen to be more than two linear factors, then it is unlikely that the true values will match the predicted values. Second, the next step in constructing a duplicate oracle is weight recovery. In order to form the weight matrix $A^{(0)}$, they calculate the second derivative of the oracle O in each input direction at the critical points x_i. The second derivative is used to calculate the difference between adjacent linear regions. This is repeated until the entire matrix $A^{(0)}$ is complete. Third, the algorithm determines the sign of every row vector $A_j^{(0)}$, using global information about the matrix. Finally, the least-squares method is used to approximate the architecture of the hidden layer(s) of the neural network. When tested against MNIST [56], the functionally equivalent extraction method produces oracles that have a rate of 100% accuracy and only begins to diminish around 100000 parameters. When tested against CIFAR-10 [55], the accuracy dips below 100% after 200000 parameters. The main issue with this method is that it can't be extended to other deeper neural networks, and only works sufficiently on two-layer models.

E. INFERENCE ATTACKS

1) MODEL INVERSION

This attack is developed to be a general-purpose universal attack. Model inversion [91] works by utilizing the information available to the adversary from the model and using that to estimate the probabilities of a potential target. Rows from a candidate database that share characteristics with the target database are used as input and are processed by the model. The database rows are weighted depending on the accepted priors and the model's output for a given row corresponding

TABLE 1. An overview of the white-box and black-box attacks. The table is organized into: (1) Algorithm name, (2) Attack Type: targeted or untargeted, (3) Scenario: white-box or black-box, (4) Learning: iterative/one-shot (N/A denotes attacks that are neither one-shot or iterative), (5) the number of perturbations (high, low, fair), (6) the perturbation norm (sections with N/A denote attacks that do not utilize the L_p norms to formulate perturbations), (7) the execution time (fast, slow, fair), (8) the transferability of the attack, either universal or model specific, and (9) the attack strength, which is observed from first hand experiments with the specific algorithms or is the perceived strength from the literature. Sections labeled "Unknown" are where the literature did not provide such information, or where further experiments are needed.

Algorithm Name	Attack Type	Scenario	Learning	Performance				
				Num. of Pert.	Norm	Time	Transferability	Strength
L-BFGS [12]	Targeted	White-box	One-Shot	High	L_2	Fair	Model Specific	$***$
C&W [26]	Targeted	White-box	Iterative	Fair	L_0, L_2, L_∞	Fair	Model Specific	$*****$
JSMA [58]	Targeted	White-box	Iterative	Low	L_0	Fair	Model Specific	$***$
Elastic Net [30]	Targeted	White-box	Iterative	Unknown	L_1	Unknown	Model Specific	$*****$
Wasserstein [64]	Targeted	White-box	Iterative	Low	N/A	Unknown	Model Specific	$****$
Targeted UAP [62]	Targeted	White-box	Iterative	Fair	L_1, L_2, L_∞	Unknown	Universal	$*****$
Threshold Attack [75]	Targeted	Black-box	Iterative	High	L_∞	Fair	Model Specific	$***$
ZOO [51]	Targeted	Black-box	Iterative	Fair	L_0, L_2, L_∞	Fair	Model Specific	$*****$
UPSET [70]	Targeted	Black-box	Iterative	High	L_∞	Unknown	Universal	$****$
ANGRI [70]	Targeted	Black-box	Iterative	High	L_∞	Unknown	Model Specific	$****$
Houdini [71]	Targeted	Black-box	Iterative	Unknown	L_2, L_∞	Unknown	Model Specific	$****$
ColorFool [77]	Targeted	Black-box	N/A	Fair	N/A	Unknown	Model Specific	$****$
FGSM [28]	Untargeted	White-box	One-Shot	High	L_∞	Fast	Model Specific	$***$
I-FGSM [57]	Untargeted	White-box	Iterative	High	L_∞	Fast	Model Specific	$****$
PGD [50]	Untargeted	White-box	Iterative	Fair	L_2, L_∞	Fast	Model Specific	$****$
DeepFool [27]	Untargeted	White-box	Iterative	Low	L_2	Fair	Model Specific	$****$
NewtonFool [60]	Untargeted	White-box	Iterative	Low	L_2	Fair	Model Specific	$****$
Shadow Attack [66]	Untargeted	White-box	Iterative	Fair	L_2, L_∞	Fast	Model Specific	$****$
UAP [14]	Untargeted	White-box	Iterative	Fair	L_2, L_∞	Unknown	Universal	$*****$
Square Attack [52]	Untargeted	Black-box	Iterative	Fair	L_2, L_∞	Unknown	Model Specific	$****$
One-Pixel [13]	Untargeted	Black-box	Iterative	Low	L_0	Fast	Model Specific	$**$
Few-Pixel [75]	Untargeted	Black-box	Iterative	Low	L_0	Fast	Model Specific	$**$
Spatial Transform [69]	Untargeted	Black-box	Iterative	Fair	N/A	Unknown	Universal	$***$
Brendel & Bethge [63]	Both	White-box	Iterative	Fair	L_0, L_1, L_2, L_∞	Fair	Model Specific	$***$
SimBA [72]	Both	Black-box	Iterative	Unknown	L_2	Unknown	Model Specific	$****$
Boundary Attack [67]	Both	Black-box	Iterative	Unknown	N/A	Unknown	Model Specific	$****$
HSJA [53]	Both	Black-box	Iterative	Unknown	L_2, L_∞	Unknown	Model Specific	$****$

to the target's previous outputs. The value with the highest weight for the target is returned. Model inversion, while being a universal black-box attack, has limitations such as poor scalability when the targeted features span a large dataset.

2) RECONSTRUCTION ATTACK

As the base model inversion attack [91] is unable to handle large datasets, tasks such as attacking a facial recognition model become increasingly more difficult. Thus, the reconstruction attack [92] was developed to target and attack facial recognition models. The reconstruction attack is an inference attack that exploits its access to the model to gain information about the training data. The reconstruction attack assumes that the adversary has knowledge of at least one output label of the model and will use this information to reconstruct an image of a face corresponding to the label. At the core of this attack is the MI-Face algorithm, which allows inversion attacks against facial recognition systems by utilizing gradient descent to minimize the cost function of the recognition model.

MI-Face first defines a cost function in terms of the recognition model f and a specific function *AuxTerm*, which uses any additional information to advise the cost function. Then MI-Face iteratively performs gradient descent, and after each gradient step, the generated vector is put through a post-processor that does image manipulation such as sharpening and de-noising. If the cost function fails to improve under a given number of iterations or the cost value is close to a given threshold value, then the attack is terminated, and the optimal cost value is returned. The reconstruction attack is an efficient and effective algorithm that was shown to have an increased attack accuracy and precision compared to other attacks, and is able to fool a deep learning image recognition model in both a white box and black box setting.

F. SUPPLEMENTARY ADVERSARIAL ATTACKS

Generally speaking, any system that utilizes a machine learning algorithm can be targeted by adversarial attacks [93]. Hence, the adversarial attacks are not limited to image classifications. In this paper, we focus mainly on adversarial attacks in the context of image classification networks. However, we believe it is important to briefly review some of the well-known adversarial attacks in different contexts such as audio, point clouds, and software.

1) ADVERSARIAL ATTACKS IN AUDIO

Recently, deep learning methods become the primary choice in developing audio systems, specifically, voice recognition and voice-to-text systems. Some researchers have shown that such systems can be compromised by adversarial attacks. Carlini & Wagner [94] demonstrate the existence of targeted audio adversarial examples that can target the automatic speech recognition system such as DeepSpeech [93]. They

introduce a targeted white box attack to compromise the audio generated by DeepSpeech. Given a clean audio wave-form x, an inaudible perturbation ρ is generated such that when added to the original audio wave, $\hat{x}$ is recognized as any phrase. The authors use the same attack methods in [26] to generate the audio adversarial waves. The proposed attack is highly effective with a success rate of 100%.

Qin *et al*. [95] can be viewed as an improvement to Carlini & Wagner's [94] by addressing two shortcomings. First, they noticed that the generated adversarial perturbations can be easily detectable by humans. Secondly, the generated adversarial audio is not effective when it's played over the air. The authors were successfully able to develop effective imperceptible audio perturbations by utilizing psychoacoustics in the form of audio masking while continuing to maintain a high success rate. They also show the possibility to transfer the generated adversarial audio examples to real-world over-the-air situations. The authors used the LibriSpeech dataset [96], which is a corpus of English-speaking audio recordings taken from audiobooks. They were able to achieve a 100% success rate on arbitrary targets. Also, they were able to move forward with research in developing robust audio adversarial examples that can be used in over-the-air situations.

2) ADVERSARIAL ATTACKS IN POINT CLOUDS

Adversarial examples for 2D images and CNNs have been largely studied and researched. However, less amount of research has been put into adversarial attacks within 3D data like point clouds. Point clouds [97] are datasets that represent points in space. Generally, point clouds represent 3D objects, with each point having its own set of coordinates. They are usually produced through 3D scanning or aerial photography and are normally used for 3D CAD modeling, visualization, and animation along with a multitude of other applications. Xiang *et al*. [98] propose two methods for generating adversarial examples to target PointNet which can be defined as a deep neural network that is used for point cloud processing [99]. Their methods, adversarial point perturbation and adversarial point generation, were shown to have success rates of 99% for all targeted attack experiments. Other researchers such as Zhang *et al*. [100] concluded that performing perturbations to point clouds works in a controlled environment, and to reproduce these experiments in a real-world scenario powerful computational resources are required, which may substantially affect the adversarial attack. Hence, they proposed the Mesh Attack, which addresses these shortcomings of other 3D point cloud attacks by directly performing perturbations on the mesh of a 3D object. Mesh Attack is able to achieve an attack success rate of 90% on PointNet, 98% on PointNet++ [101], and 59% on DGCNN [102].

3) ADVERSARIAL ATTACKS IN SOFTWARE

Presently, malwares are widely spread over the internet and becoming a serious and persistent threat. Methods for detecting malware have evolved to become more robust and many of these detection methods utilize machine learning models to improve the detection of malwares. However, these machine learning-based malware detection tools are vulnerable to adversarial examples. Liu *et al*. [103] presents an attack called Adversarial Texture Malware Perturbation Attack (ATMPA). ATMPA can generate adversarial examples to fool ML-based visualization malware detection systems. ATMPA uses FGSM and C&W attacks to generate adversarial examples. The authors have shown that on both gradient descent and L_p norm-based optimization methods, they were able to consistently produce a 100% success rate. However, it is worth noting that they also used fairly high epsilon values $\epsilon = \{0.4, 0.5, 0.6\}$ for FGSM adversarial examples which may have resulted in such high success rates. Nonetheless, their work presents an interesting point of view to utilize adversarial examples to undermine ML-based visualization malware detection systems.

VI. DEFENSE STRATEGIES

As depicted in Table 2, defenses for neural networks against adversarial attacks generally lie within one of four frameworks: (1) modifying the ANN, (2) modifying the training by including the adversarial examples (e.g., adversarial training), (3) transforming the inputs, or (4) having external models that serve as ANN add-ons. Defense methods that change the training or the input data are disconnected from the ANN model itself. However, modified ANNs and ANN add-ons implement more layers, add subnetworks, change the loss function, or use external models to defend against attacks. In this section, we will discuss the various methods used to protect a deep learning model from adversarial attacks.

A. MODIFICATIONS TO THE ANN

1) DEFENSIVE DISTILLATION

Papernot *et al*. [29] introduced defensive distillation as a defense method for deep learning models against adversarial attacks. Defensive distillation builds upon the original distillation algorithm [104], which was originally introduced as a way to reduce the size of a large model into a reduced distilled model. Defensive distillation utilizes the distillation algorithm to increase the robustness of the model. However, instead of reducing the size of the model, defensive distillation modifies the softmax activation function in the last layer of the neural network to include a temperature value T. This temperature value forces the model to make stronger and more confident predictions.

The defensive distillation algorithm operates as follows: First, a large network F is trained by initializing the temperature T of the softmax function during the model's training phase. Then "soft" labels are generated by applying the network to every value in a training set X and recalculating the softmax with the temperature. Next, a new training set is generated using the soft labels. Then using the new training data another deep learning model is trained, with the same model

architecture as the original model F, and the temperature of the softmax function remains T. This new model is known as the distilled model F^d, and when ran at test time the model will classify new input data.

The defensive distillation defense method works effectively against the L-BFGS [12] and DeepFool [27] attacks. However, Carlini & Wagner [26] were able to bypass the defensive distillation defense method. They applied their three attacks to defensively distilled networks trained on MNIST [56] and CIFAR-10 [55], with a model temperature $T = 100$. They showed that their L_0, L_2, and L_∞ attack algorithms were able to successfully attack a model 100% of the time.

2) GRADIENT REGULARIZATION

Input gradient regularization [105], or double backpropagation [106], trains deep neural networks while penalizing any slight variation in the inputs. If any inputs are slightly modified, the relative entropy between the labels and the predictions will be insignificant. This means that a small perturbation would be unlikely to change the output of a trained classifier. Combined with methods such as brute-force training, gradient regularization is known to be highly successful in defending DNNs against attacks such as FGSM [28] and JSMA [58].

3) DeepCloak

The primary goal of DeepCloak [107] is to remove unnecessary features used by an adversary to generate adversarial examples. The authors propose inserting a mask layer in the DNN model right before the classification layer. The mask layer is trained by forward passing both clean and perturbed images, encoding the differences between the outputs of the two in the previous layers. DeepCloak filters out the unnecessary features by setting them to 0, effectively removing them from the adversarial examples. DeepCloak was shown to be effective in filtering out the perturbations in images generated by the FGSM [28] attack.

4) PARSEVAL NETWORKS

Cisse *et al*. [108] introduced the idea of using Parseval networks to defend against adversarial attacks. Parseval networks control the Lipschitz constant by utilizing layer-wise regularization of the weight matrices. The Lipschitz constant is the maximum ratio between permutations in the input and output spaces and is used to measure the sensitivity of the classifier. Maintaining a small Lipschitz constant is crucial to remain robust against even the smallest perturbations. The authors realized that they were able to control the spectral norm of the weight matrices, which is the natural norm of the L_2 norm, by parameterizing the network with Parseval tight frames. Parseval networks as a defense mechanism were tested against and shown to be successful in defending against the FGSM [28] attack.

5) SafetyNet

Adversarial examples generate distinctive patterns, notably during the last layers of neural networks of ReLU models rather than what is seen with clean images. SafetyNet [109] proposes an RBF-SVM (radial basis function SVM) classifier should be added to the end of the targeted models. The SVM should use codes distinctly generated by ReLU. The SVM detects adversarial examples by comparing the codes of the testing data with the codes of the training data. SafetyNet was able to effectively detect adversarial perturbations in both DeepFool [27] and L-BFGS [12].

6) DETECTOR SUBNETWORK

Metzen *et al*. [110] use subnetworks that augment the original network to detect adversarial perturbations. Subnetworks work by branching off the main network and producing a probability p_{adv} that weighs the chances of an image being adversarial. This subnetwork is known as the detector and is trained to classify inputs as clean or adversarial.

First, the classification network is trained on the regular data (*i.e*., non-adversarial). Then, the adversarial examples are generated for the entire dataset, using the attack algorithms from which the network is trying to defend itself (*e.g*., FGSM and DeepFool). Once an equal size dataset that has the same amount of clean and perturbed images is generated, the weights of the classification network are frozen and the detector network is trained such that the cross-entropy of the probability p_{adv}, as well as, the labels are minimized. The specifics of the detection subnetworks and how it connects to the classification network are specific to each dataset and classification network. The detector subnetworks defense works in detecting perturbations that are generated using FGSM, DeepFool, and BIM [57].

7) DEEP CONTRACTIVE NETWORKS

Denoising autoencoders (DAE) [111] were introduced to combat adversarial noise. A DAE is trained to detect and remove adversarial noise. However, when coupled alongside the original network, attacks that are targeted at this stacked network are even more powerful than attacks with the original model. Gu and Rigazio propose deep contractive networks (DCN) [111], using a smoothness penalty like the one used in contractive autoencoders, which are a variant of autoencoders that include a minimization penalty. DCNs improve the robustness of the network against adversarial attacks without sacrificing performance. Deep contractive networks are effective in detecting the perturbations generated by the L-BFGS [12] attack algorithm.

B. MODIFICATIONS TO THE TRAINING

1) BRUTE-FORCE TRAINING

Adversarial training is one of the most effective ways to improve overall model robustness against adversarial examples. It corresponds to the process of adding the adversarial examples with their correct labels into the training dataset of the DNN models [32]. This method requires that

the adversarial training is executed with a powerful attack algorithm, an exposed model, and a large dataset. For these reasons, adversarial training is commonly known as brute-force training. It is also shown that adversarial training can provide an added regularization to the network [28] which helps strengthen the DNN models against adversarial attacks. Methods that build on the observations made in [12], [28], and [27] have been proposed, such as stability and virtual adversarial training.

Virtual adversarial training [112] extends the previously mentioned method to work with unlabeled data and in semi-supervised situations. Virtual adversarial training can also work with text classification and sequencing models. Zheng *et al.* [113] proposed the stability training method which stabilizes DNNs and improves robustness against small distortions in input images. Overall, adversarial brute-force training is an effective way of improving the robustness of the model but is also exposed to certain attacks, namely universal perturbations [14].

2) INPUT TRANSFORMATIONS

Another way of protecting neural networks from adversarial attacks is by modifying the inputs in a way to reduce the model's sensitivity to small perturbations. Data compression is one way of minimizing the damage perturbed images can do to the model. Dziugate *et al.* [114] noticed that almost every classification dataset consists of JPG images. They used this observation as a basis for testing the effects of JPG compression on adversarial images generated by FGSM [28]. They found that most of the time JPG compression would reverse the classification of the adversarial image for small perturbations generated by FGSM. Generally speaking, for large perturbations, JPG compression is not an effective method to reverse the effects of the perturbation and only slightly improves the overall classification of the images. Another method proposed by Bhagoji *et al.* [115] seeks to defend against evasion attacks by using dimensionality reduction approaches such as the principal component analysis (PCA) method. They found that by using PCA against evasion attacks, their defense is effective against the L_2 Carlini & Wagner attack [26] and FGSM attacks. Principal component analysis makes it harder for the adversary to perturb an image and perform a successful white-box attack. They found that their defense can work across multiple classifiers including SVMs and DNNs and can be generalized to work in multiple applications.

C. ADDITIONS TO THE ANN

1) TRAPDOORED MODEL

Shan *et al.* [116] propose an approach for defending against adversarial examples by utilizing honeypots to detect perturbed images. These honeypots are decoys that lure adversaries into artificial security vulnerabilities in the network. The authors deliberately implanted trapdoors that attracted adversaries attempting to probe the model for any information to generate adversarial images. The adversaries are drawn towards the trapdoors and generate images designed to target these perceived weaknesses to attack the classifier. The honeypots catch and identify the incoming attacks by measuring and comparing the neuron activation of the inputs to the injected trapdoors. The trapdoored model can protect the ANN against PGD [50], C&W [26], and EAD [30] attacks, with high accuracy and with little impact on normal classifications. This method can also work in multiple classification scenarios like image and facial recognition.

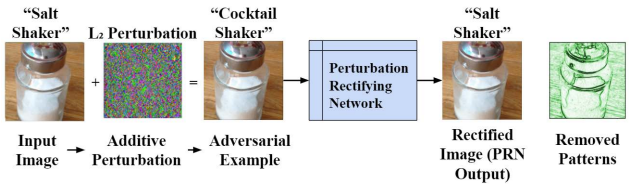

FIGURE 11. Visualization of the framework for defending against UAPs [117]. This method rectifies the images to restore the classifier's predictions. The patterns that are removed during the rectification process are analyzed to determine whether an image is adversarial.

2) DEFENSE AGAINST UAPs

Universal adversarial perturbations (UAP) [14] highlight a critical shortcoming in the security of machine learning models. Therefore, Akhtar *et al.* [117] introduce a framework for defending against UAPs by adding a perturbation rectifying network (PRN) as a pre-input layer to the targeted model to prevent having to alter the model. The PRN catches the perturbed images coming into the network and adjusts them to label these perturbed images with same label of the original image. The perturbation rectifying network is trained using datasets that contain real and artificial UAPs without changing any of the model's parameters. Separately, a perturbation detector is trained on the cosine transform of the differences between inputs and outputs of the PRN. As shown in Fig. 11, the images pass through the PRN and then verified by the detector. When, the perturbations is detected, the output from the PRN is used to predict the labels instead of the actual image. PRN shows promising results in defending DNNs against UAPs with a 97.5% success rate.

3) FEATURE SQUEEZING

Other attempts to defend neural networks from adversarial attacks investigate optimizing the model which can be computationally expensive. Therefore, Xu *et al.* [118] proposed feature squeezing to strengthen DNNs by detecting perturbed images. The feature squeezing process minimizes the search space by consolidating examples that correlate to various feature vectors in the original search space into a single example. Although the feature squeezing process is quite general, the authors specifically explore two methods, spatial smoothing, and the reduction of the color bit depth of every pixel. These techniques are simple, inexpensive, and can be combined

TABLE 2. An outline of the different defense strategies against adversarial machine learning attacks. the "Unknown" indicates lack of information in the literature.

Defense Type	Defense Name	Covered Attacks
Modifications to the ANN	Defensive Distillation [29]	JSMA [58]
	Gradient Regularization [105] [106]	FGSM [28], JSMA [58]
	DeepCloak [107]	FGSM [28]
	Parseval Networks [108]	FGSM [28]
	SafetyNet [109]	L-BFGS [12], DeepFool [27]
	Detector Subnetwork [110]	FGSM [28], DeepFool [27], BIM [57]
	Deep Contractive Networks [111]	L-BFGS [12]
Brute-force Training	Adversarial Training [27] [28] [12]	JSMA [58]
	Virtual Adversarial Training [112]	Unknown
	Stability Adversarial Training [113]	Unknown
Input Transformations	Data Compression [114]	FGSM [28]
	Data Transformation [115]	L_2 C&W [26], FGSM [28]
External Models and Network Add-Ons	Trapdoor Model [116]	PGD [50], C&W [26], EAD [30]
	Defense against UAPs [117]	UAP [14]
	Feature Squeezing [118]	FGSM [28], BIM [57], C&W [26], DeepFool [27], JSMA [58]
	Magnet [119]	FGSM [28], I-FGSM [57], DeepFool [27], C&W [26]

with other defense strategies to have more effective results. Input images first go through an external model that performs feature squeezing over the image. The probability score of the target network using the original input is compared to the score of the squeezed image. If the difference between the probability scores is considerably large, then the image is classified as adversarial and discarded.

4) MagNet

Meng and Chen proposed MagNet [119], a framework used for defending against adversarial attacks on neural network classifiers. MagNet introduces separate detector and reformer networks that classify whether incoming input images are adversarial or not. Detector networks are either based on the reconstruction error or the probability divergence. They estimate the manifold of clean inputs and during the training phase, they learn to separate the clean images from the perturbed ones. In the testing phase, images that are far from the manifold are labeled adversarial and then dropped. For adversarial images with small perturbations, reformer networks, which are based on random noise or an autoencoder, push perturbed images towards the manifold of the unperturbed images, essentially forcing the image to be correctly classified. MagNet is an effective method to defend against black box (zero-knowledge) and gray box attack (limited knowledge) scenarios, and because it does not assume any specific perturbation generation method, MagNet can be generalized to different attacks.

D. DEFENSE AGAINST POISONING ATTACKS

1) DETECTION BASED ON ACTIVATION ANALYSIS

Detecting and defending a deep neural network model against backdoor attacks is extremely difficult because the triggers of the backdoors are only known to the adversary. Backdoors are triggered when specific features are identified by the model that is associated with the source class, which in turn, the backdoor trigger results in network activations that represent a decision made by the model. Chen *et al*. [82] proposed an activation clustering algorithm to detect the poisoned input images that are generated to inject backdoors into deep neural networks. The proposed algorithm analyzes the activations in the neural network to detect backdoors. The working process of their algorithm can be described as follows. First, they train the neural network using an untrusted dataset containing poisoned examples. Second, they query the neural network using the training data and the subsequent activations of the last hidden layer. Third, once the activations of each sample are retained, they are segmented into different segments that are clustered individually, where each segment corresponds to a label. Fourth, by using k-means clustering [120], the clusters are separated into two groups: poisoned and clean data. Finally, the poisoned data is identified either by exclusionary reclassification, relative size comparison, or silhouette score. Once the poisoned data is identified, the authors suggest the fastest way to repair the backdoor by "re-labeling" the poisoned data with its original class, and continue to train the model until convergence. Their method was tested using the LISA [121], MNIST [56], and Rotten Tomatoes [122] datasets. When the authors experimented with 10% poisoned data using MNIST, they were able to achieve accuracy and F1 score of nearly 100% for each class label. Compared to a conventional clustering algorithm, their method outshines in every respect.

2) DETECTION BASED ON SPECTRAL SIGNATURES

Tran *et al*. [123] proposed a new method for defending neural networks against poisoning backdoor attacks by utilizing spectral signatures. The authors noticed that in the aftermath of a backdoor attack, a detectable trace of the attack was left behind in the covariance of a feature representation learned by the model. They refer to the remnant of the attack as a spectral signature. The authors presume that the set of inputs contains both clean and adversarial examples for each label. A backdoor in an adversarial example would yield a strong signal in the representation vector. Signals that are large in magnitude can easily be detected through singular value decomposition and the images that provide that signal can be detected and removed. The authors test their algorithm using CIFAR-10 [55] with 5000 examples for each of the

10 labels. When they use 250 adversarial examples, which is relatively a very small sample, the trained model accuracy drops to around 10%. In most cases by utilizing spectral signals they can remove all traces of corrupted data, minimizing the misclassification rate to around 1%.

VII. DISCUSSION & FUTURE RESEARCH DIRECTION

In the previous sections, we discussed the recent state-of-the-art adversarial attacks against deep neural networks and the defense mechanisms. In this section, we present the current challenges and the broader view of the directions of the future research work based on the literature discussed in this paper.

A. THE EXISTENCE OF ADVERSARIAL ATTACKS

Upon constructing an adversarial example, the adversary needs to craft the perturbations in the way that forces the targeted model to misclassify, as well as, maintain the ''true class'' of the input. Adding large imperceptible perturbations could truly change the identification characteristics of an input. Therefore, few studies present some skepticisms about the existence of adversarial attacks and consider their existence as normal behavior in deep neural networks. In other words, yielding a different class of the original input after adding the perturbations by the DNN model is a normal behavior [32]. However, in Sections V and VI we has demonstrated how adversarial attacks are real and pose a real threat to deep neural networks.

The reason why deep neural networks are vulnerable to adversarial attacks is unknown. Many viewpoints have been proposed attempting to explain this phenomenon. For instance, Goodfellow [28] argued that the deep neural networks are ''too linear'' in a high dimensional space which makes them susceptible to any subtle changes that occur to any input. Fawzi [124] suggested that the ''low flexibility'' of the classifiers in certain tasks can be the reason. Despite the recent advances, we believe that identifying the adversarial vulnerabilities of deep learning is still unclear and requires further investigation.

B. THE TRANSFERABILITY OF ADVERSARIAL ATTACKS

The term ''transferability'' refers to the degree of generalizing the adversarial attack of a specific neural network to fool other neural networks of different architectures. Most of the adversarial attacks are transferable. This is especially true for neural networks that have a similar architecture [74]. The attack transferability can be categorized into three levels:

- **Low transferability.** The adversarial attack can fool similar deep neural networks when trained with a different dataset. For example, the DeepFool attack is an example of low transferability. The adversarial images generated using DeepFool on any given architecture are hardly able to fool other neural networks.
- **Medium transferability.** The adversarial attack can fool different types of neural networks when trained using the same dataset (*i.e.*, performing the same task). FGSM is an example of medium transferability.
- **High transferability.** At this level, the adversarial attack can fool different neural networks of different architectures performing different tasks.

Currently, most of the existing adversarial attack research is focused on image classification. Very limited studies have focused on different applications [41]. Therefore, further research is required to focus on adversarial deep neural networks in different applications. In addition, further investigation is required to evaluate the applicability, efficiency, and practical use of the current adversarial attacks in different applications.

C. ADVERSARIAL ATTACK DEFENSE METHODS

Multiple defense methods have been proposed to countermeasure the adversarial attacks. However, oftentimes showed that a defended model has been successfully attacked by an existing attack or a zero-day attack. For example, the distilled neural network defense mechanism [29] has been defeated against C&W attacks [26]. Furthermore, the adversarial training defense technique has been proved to be ineffective [125]. Thus, further research is required to focus on developing a universal adversarial defense method that covers the various aspects of adversarial attacks.

D. EVALUATION METHODOLOGY OF THE ROBUSTNESS OF DEEP NEURAL NETWORKS

Currently, most of the adversarial attacks and defense mechanisms have been simulated in limited environments. Also, in many cases, the source code and the configuration parameters of the work environment are not available to the research community to further evaluate the robustness of the defense method as well as the adversarial attack. Hence, having a deep learning robustness methodology is crucial. Different works have conducted initial studies emphasizing the importance of evaluating the robustness of neural networks [126], [127], [128]. However, different questions arose such as (1) how to stress-test neural networks on different business domains? and (2) what are the general robustness parameters and acceptance score of a neural network application?. To answer these questions, further research is required.

VIII. CONCLUSION

In this article, we provide a comprehensive review of the state-of-the-art adversarial attack methods. In our review, we focused on adversarial attacks against the image classification neural networks. In addition, we have provided a detailed discussion of the most widely used defense strategies against adversarial attacks by focusing on their usage in real-life applications. Also, we have provided a detailed overview of the attacking scenarios that can be employed by the adversary to compromise DNNs. Furthermore, in this paper, we analyzed and discussed the current open issues and challenges that require further investigation.

REFERENCES

[1] H. Y. Xiong, B. Alipanahi, L. J. Lee, H. Bretschneider, D. Merico, R. K. C. Yuen, Y. Hua, S. Gueroussov, H. S. Najafabadi, T. R. Hughes, Q. Morris, Y. Barash, A. R. Krainer, N. Jojic, S. W. Scherer, B. J. Blencowe, and B. J. Frey, "The human splicing code reveals new insights into the genetic determinants of disease," *Science*, vol. 347, no. 6218, Jan. 2015, Art. no. 1254806.

[2] E. Ackerman, "How drive.AI is mastering autonomous driving with deep learning," Binus Univ., Palmerah Jakarta, Indonesia, Tech. Rep. 1, 2021.

[3] S. Khamaiseh, E. Serra, and D. Xu, "VSwitchGuard: Defending openflow switches against saturation attacks," in *Proc. IEEE 44th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2020, pp. 851–860.

[4] Z. Li, W. Xing, S. Khamaiseh, and D. Xu, "Detecting saturation attacks based on self-similarity of openflow traffic," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 1, pp. 607–621, Mar. 2020.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 84–90.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[8] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[9] D. Mery, *Computer Vision for X-Ray Testing*, vol. 10. Cham, Switzerland: Springer, 2015, p. 978.

[10] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni, and J. Loncarski, "Machine learning approach for predictive maintenance in industry 4.0," in *Proc. 14th IEEE/ASME Int. Conf. Mech. Embedded Syst. Appl. (MESA)*, Jul. 2018, pp. 1–6.

[11] S. Khaki and L. Wang, "Crop yield prediction using deep neural networks," *Frontiers Plant Sci.*, vol. 10, p. 621, May 2019.

[12] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[13] J. Su, D. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.

[14] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1765–1773.

[15] H. Wu, S. Yunas, S. Rowlands, W. Ruan, and J. Wahlstrom, "Adversarial driving: Attacking end-to-end autonomous driving," 2021, *arXiv:2103.09151*.

[16] Y. Hu, W. Kuang, Z. Qin, K. Li, J. Zhang, Y. Gao, W. Li, and K. Li, "Artificial intelligence security: Threats and countermeasures," *ACM Comput. Surveys*, vol. 55, no. 1, pp. 1–36, Jan. 2023.

[17] N. Akhtar, A. Mian, N. Kardan, and M. Shah, "Advances in adversarial attacks and defenses in computer vision: A survey," *IEEE Access*, vol. 9, pp. 155161–155196, 2021.

[18] J. Zhang and C. Li, "Adversarial examples: Opportunities and challenges," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2578–2593, Jul. 2020.

[19] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.

[20] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," *Int. J. Autom. Comput.*, vol. 17, no. 2, pp. 151–178, Apr. 2020.

[21] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.

[22] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3668–3681, Nov. 2019.

[23] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 1951.

[24] J. Heaton, *Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep Learning*. Cambridge, MA, USA: MIT Press, 2018.

[25] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*.

[26] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.

[27] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.

[28] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[29] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 582–597.

[30] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "EAD: Elastic-net attacks to deep neural networks via adversarial examples," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 1–8.

[31] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," 2018, *arXiv:1810.00069*.

[32] T. Hazan, G. Papandreou, and D. Tarlow, *Perturbations, Optimization, and Statistics*. Cambridge, MA, USA: MIT Press, 2016.

[33] A. Abraham, "Artificial neural networks," in *Handbook of Measuring System Design*. Hoboken, NJ, USA: Wiley, 2005.

[34] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics Intell. Lab. Syst.*, vol. 39, no. 1, pp. 43–62, 1997.

[35] *Sigmoid Function*, DeepAI, Sep. 2020. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function

[36] *Softmax Function*, DeepAI, May 2019. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

[37] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci.*, vol. 6, no. 12, pp. 310–316, 2017.

[38] K. T. Co, L. Muñoz-González, S. de Maupeou, and E. C. Lupu, "Procedural noise adversarial examples for black-box attacks on deep convolutional networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Nov. 2019, pp. 275–289.

[39] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 506–519.

[40] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Berlin, Germany: Springer, 2013, pp. 387–402.

[41] S. Y. Khamaiseh, I. Alsmadi, and A. Al-Alaj, "Deceiving machine learning-based saturation attack detection systems in SDN," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2020, pp. 44–50.

[42] Y. Du, M. Fang, J. Yi, J. Cheng, and D. Tao, "Towards query efficient black-box attacks: An input-free perspective," in *Proc. 11th ACM Workshop Artif. Intell. Secur.*, 2018, pp. 13–24.

[43] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Query-efficient black-box adversarial examples (superceded)," 2017, *arXiv:1712.07113*.

[44] P. Laskov and M. Kloft, "A framework for quantitative security analysis of machine learning," in *Proc. 2nd ACM Workshop Secur. Artif. Intell.*, 2009, pp. 1–4.

[45] P. Rathore, A. Basak, S. Harsha Nistala, and V. Runkana, "Untargeted, targeted and universal adversarial attacks and defenses on time series," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[46] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Proc. Asian Conf. Mach. Learn.*, 2011, pp. 97–112.

[47] R. Lopez-Valcarce and D. Romero, "Design of data-injection adversarial attacks against spatial field detectors," in *Proc. IEEE Stat. Signal Process. Workshop (SSP)*, Jun. 2016, pp. 1–5.

[48] H. Chacon, S. Silva, and P. Rad, "Deep learning poison data attack detection," in *Proc. IEEE 31st Int. Conf. Tools with Artif. Intell. (ICTAI)*, Nov. 2019, pp. 971–978.

[49] W. Jiang, H. Li, S. Liu, X. Luo, and R. Lu, "Poisoning and evasion attacks against deep learning algorithms in autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4439–4449, Apr. 2020.

[50] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[51] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 15–26.

[52] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2020, pp. 484–501.

[53] J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A query-efficient decision-based attack," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 1277–1294.

[54] R. Fletcher, *Practical Methods of Optimization*. Hoboken, NJ, USA: Wiley, 2013.

[55] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[56] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[57] A. Kurakin, *Adversarial Examples in the Physical World*. Ithaca, NY, USA: Arxiv, 2016.

[58] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 372–387.

[59] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 2206–2216.

[60] U. Jang, X. Wu, and S. Jha, "Objective metrics and gradient descent algorithms for adversarial examples in machine learning," in *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, Dec. 2017, pp. 262–277.

[61] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[62] H. Hirano and K. Takemoto, "Simple iterative method for generating targeted universal adversarial perturbations," *Algorithms*, vol. 13, no. 11, p. 268, Oct. 2020.

[63] W. Brendel, J. Rauber, M. Kümmerer, I. Ustyuzhaninov, and M. Bethge, "Accurate, reliable and fast robustness evaluation," 2019, *arXiv:1907.01003*.

[64] E. Wong, F. Schmidt, and Z. Kolter, "Wasserstein adversarial examples via projected Sinkhorn iterations," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6808–6817.

[65] L. V. Kantorovich and S. G. Rubinshtein, "On a space of totally additive functions," *Vestnik St. Petersburg Univ., Math.*, vol. 13, no. 7, pp. 52–59, 1958.

[66] A. Ghiasi, A. Shafahi, and T. Goldstein, "Breaking certified defenses: Semantic adversarial examples with spoofed robustness certificates," 2020, *arXiv:2003.08937*.

[67] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2017, *arXiv:1712.04248*.

[68] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[69] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 1802–1811.

[70] S. Sarkar, A. Bansal, U. Mahbub, and R. Chellappa, "UPSET and ANGRI: Breaking high performance image classifiers," 2017, *arXiv:1707.01159*.

[71] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured prediction models," 2017, *arXiv:1707.05373*.

[72] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 2484–2493.

[73] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[74] A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard, "SparseFool: A few pixels make a big difference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9087–9096.

[75] D. V. Vargas and S. Kotyan, "Robustness assessment for adversarial machine learning: Problems, solutions and a survey of current neural networks and defenses," 2019, *arXiv:1906.06026*.

[76] N. Hansen, "The CMA evolution strategy: A tutorial," 2016, *arXiv:1604.00772*.

[77] A. Shahin Shamsabadi, R. Sanchez-Matilla, and A. Cavallaro, "ColorFool: Semantic adversarial colorization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1151–1160.

[78] A. López-Cifuentes, M. Escudero-Viñolo, J. Bescós, and Á. García-Martín, "Semantic-aware scene recognition," *Pattern Recognit.*, vol. 102, Jun. 2020, Art. no. 107256.

[79] A. Mustafa, S. Khan, M. Hayat, R. Goecke, J. Shen, and L. Shao, "Adversarial defense by restricting the hidden space of deep neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3385–3394.

[80] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[81] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," 2012, *arXiv:1206.6389*.

[82] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," 2018, *arXiv:1811.03728*.

[83] T. Juin Lester Tan and R. Shokri, "Bypassing backdoor detection algorithms in deep learning," 2019, *arXiv:1905.13409*.

[84] R. Pang, H. Shen, X. Zhang, S. Ji, Y. Vorobeychik, X. Luo, A. Liu, and T. Wang, "A tale of evil twins: Adversarial inputs versus poisoned models," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 85–99.

[85] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Proc. NDSS*, 2018, pp. 1–17.

[86] A. Shafahi, W. Ronny Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! Targeted clean-label poisoning attacks on neural networks," 2018, *arXiv:1804.00792*.

[87] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 7614–7623.

[88] H. Aghakhani, D. Meng, Y.-X. Wang, C. Kruegel, and G. Vigna, "Bullseye polytope: A scalable clean-label poisoning attack with improved transferability," 2020, *arXiv:2005.00191*.

[89] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, "Copycat CNN: Stealing knowledge by persuading confession with random non-labeled data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.

[90] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *Proc. 29th USENIX Secur. Symp. (USENIX Secur.)*, 2020, pp. 1345–1362.

[91] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *Proc. 23rd USENIX Secur. Symp. (USENIX Secur.)*, 2014, pp. 17–32.

[92] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 1322–1333.

[93] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep speech: Scaling up end-to-end speech recognition," 2014, *arXiv:1412.5567*.

[94] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2018, pp. 1–7.

[95] Y. Qin, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 5231–5240.

[96] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 5206–5210.

[97] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-semantic data for indoor scene understanding," 2017, *arXiv:1702.01105*.

[98] C. Xiang, C. R. Qi, and B. Li, "Generating 3D adversarial point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9136–9144.

[99] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.
[100] J. Zhang, L. Chen, B. Liu, B. Ouyang, Q. Xie, J. Zhu, W. Li, and Y. Meng, "3D adversarial attacks beyond point cloud," 2021, *arXiv:2104.12146*.
[101] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.
[102] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Nov. 2019.
[103] X. Liu, J. Zhang, Y. Lin, and H. Li, "ATMPA: Attacking machine learning-based malware visualization detection methods via adversarial examples," in *Proc. IEEE/ACM 27th Int. Symp. Quality Service (IWQoS)*, Jun. 2019, pp. 1–10.
[104] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[105] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–10.
[106] H. Drucker and Y. Le Cun, "Improving generalization performance using double backpropagation," *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 991–997, Nov. 1992.
[107] J. Gao, B. Wang, Z. Lin, W. Xu, and Y. Qi, "DeepCloak: Masking deep neural network models for robustness against adversarial samples," 2017, *arXiv:1702.06763*.
[108] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 854–863.
[109] J. Lu, T. Issaranon, and D. Forsyth, "SafetyNet: Detecting and rejecting adversarial examples robustly," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 446–454.
[110] J. Hendrik Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," 2017, *arXiv:1702.04267*.
[111] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," 2014, *arXiv:1412.5068*.
[112] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," 2016, *arXiv:1605.07725*.
[113] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4480–4488.
[114] G. Karolina Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of JPG compression on adversarial images," 2016, *arXiv:1608.00853*.
[115] A. N. Bhagoji, D. Cullina, C. Sitawarin, and P. Mittal, "Enhancing robustness of machine learning systems via data transformations," in *Proc. 52nd Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2018, pp. 1–5.
[116] S. Shan, E. Wenger, B. Wang, B. Li, H. Zheng, and B. Y. Zhao, "Gotta Catch'Em all: Using honeypots to catch adversarial attacks on neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 67–83.
[117] N. Akhtar, J. Liu, and A. Mian, "Defense against universal adversarial perturbations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3389–3398.
[118] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," 2017, *arXiv:1704.01155*.
[119] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 135–147.
[120] J. MacQueen, "Classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, 1967, pp. 281–297.
[121] M. B. Jensen, M. P. Philipsen, M. Trivedi, T. Møgelmose, and T. Moeslund, "Vision for looking at traffic lights: Issues, survey, and perspectives," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1800–1815, Jul. 2016.
[122] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proc. ACL*, 2005, pp. 1–11.
[123] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.
[124] A. Fawzi, H. Fawzi, and O. Fawzi, "Adversarial vulnerability for any classifier," 2018, *arXiv:1802.08686*.
[125] B. Wu, H. Pan, L. Shen, J. Gu, S. Zhao, Z. Li, D. Cai, X. He, and W. Liu, "Attacking adversarial attacks as a defense," 2021, *arXiv:2106.04938*.
[126] N. Carlini, G. Katz, C. Barrett, and D. L. Dill, "Provably minimally-distorted adversarial examples," 2017, *arXiv:1709.10207*.
[127] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Proc. Int. Conf. Comput. Aided Verification*. Berlin, Germany: Springer, 2017, pp. 97–117.
[128] A. Fawzi, O. Fawzi, and P. Frossard, "Analysis of classifiers' robustness to adversarial perturbations," *Mach. Learn.*, vol. 107, no. 3, pp. 481–508, 2018.

SAMER Y. KHAMAISEH (Member, IEEE) received the Ph.D. and M.S. degrees in computing with an emphasis on cyber security from Boise State University. He is currently an Assistant Professor and the Director of the Network and Software Security Research (NSSR) Laboratory, Miami University, Oxford, OH, USA. His research interests include cyber security with the integration of artificial intelligence (AI), specifically, adversarial machine learning, software-defined networking (SDN), software security, machine learning security, and network security and conferences.

DEREK BAGAGEM received the B.Sc. and M.Sc. degrees in computer science from Monmouth University. He is a Research Scientist at the NSSR Laboratory. His research interests include applying high-level concepts such as machine learning, deep learning, and natural language processing to solve security problems.

ABDULLAH AL-ALAJ (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from the Jordan University of Science and Technology (JUST) and the Ph.D. degree in computer science from the University of Texas at San Antonio (UTSA). He is currently an Assistant Professor in computer science with Virginia Wesleyan University, Virginia Beach, VA, USA. His research interests include cyber security, access control models, software-defined networking, and applied machine learning. He is a reviewer and a technical committee member for several journals and conferences.

MATHEW MANCINO received the M.Sc. and B.Sc. degrees in computer science from Monmouth University. He is currently working as a Cyber Security Research Engineer with CACI International. His research interests include deep-learning security, machine learning security, and code security.

HAKAM. W. ALOMARI (Member, IEEE) received the bachelor's degree from Yarmouk University, in 2004, the master's degree from the Jordan University of Science and Technology, Jordan, in 2006, and the Ph.D. degree from Kent State University, OH, USA, in 2012, all in computer science. He is currently an Assistant Professor with the Department of Computer Science and Software Engineering, Miami University, from 2015. His research interest includes developing and constructing methods for lightweight static program analysis.