Defending against Adversarial Samples without Security through Obscurity

Wenbo Guo[1], Qinglong Wang[2], Kaixuan Zhang[1], Alexander G. Ororbia II[4], Sui Huang[3], Xue Liu[2], C. Lee Giles[1], Lin Lin[1], Xinyu Xing[1]

[1]Pennsylvania State University, [2]McGill University, [3]Netflix, Inc., [4]Rochester Institute of Technology

{wzg13, kuz22, giles, xxing}@ist.psu.edu, qinglong.wang@mail.mcgill.ca, xueliu@cs.mcgill.ca, shuang@netflix.com, ago@cs.rit.edu, llin@psu.edu

***Abstract*—It has been recently shown that deep neural networks (DNNs) are susceptible to a particular type of attack that exploits a fundamental flaw in their design. This attack consists of generating particular synthetic examples referred to as adversarial samples. These samples are constructed by slightly manipulating real data-points that change "*fool*" the original DNN model, forcing it to misclassify previously correctly classified samples with high confidence. Many believe addressing this flaw is essential for DNNs to be used in critical applications such as cyber security.**

Previous work has shown that learning algorithms that enhance the robustness of DNN models all use the tactic of "security through obscurity". This means that security can be guaranteed only if one can obscure the learning algorithms from adversaries. Once the learning technique is disclosed, DNNs protected by these defense mechanisms are still susceptible to adversarial samples. In this work, we investigate by examining how previous research dealt with this and propose a generic approach to enhance a DNN's resistance to adversarial samples. More specifically, our approach integrates a data transformation module with a DNN, making it robust even if we reveal the underlying learning algorithm. To demonstrate the generality of our proposed approach and its potential for handling cyber security applications, we evaluate our method and several other existing solutions on datasets publicly available, such as a large scale malware dataset and MNIST and IMDB datasets. Our results indicate that our approach typically provides superior classification performance and robustness to attacks compared with state-of-art solutions.

***Index Terms*—Adversarial deep learning, security through obscurity, data transformation, malware detection.**

I. Introduction

Like all other machine learning approaches, deep learning is vulnerable to what is known as adversarial samples [9]. This means that they can be easily deceived by non-obvious and potentially dangerous manipulation [29]. To be specific, an attacker could use the same training algorithm, back-propagation of errors, and a surrogate dataset to construct an auxiliary model. Since this model could provide the attacker with a capability of exploring a DNN's blind spots, one can, with minimal effort, craft an *adversarial sample* – a synthetic example generated by slightly modifying a real example which makes the deep learning system "believe" with high confidence the sample subtly perturbed belongs to an incorrect class.

This work was done while Qinglong Wang and Alexander G. Ororbia II was in Penn State.

According to a recent study [9], adversarial samples occur in a relatively broad subspace, which means it is impractical to build a defense that can rule out all adversarial samples. As such, the design principle followed by existing defense mechanisms is not to harden a DNN to be naturally resistant to any adversarial samples. Rather, the focus is on hiding that subspace, making it difficult for adversaries to find useful adversarial samples. For example, representative defenses – *adversarial training* [9] and *defensive distillation* [23] both increase the complexity of original DNNs with the goal of making adversarial samples – problematic for original DNNs – no longer effective.

However, in this work we show that the defenses proposed are far from ideal and can even be considered dangerous. In particular, we demonstrate that all existing defense mechanisms still follow the approach of "security through obscurity", in which security is achieved by keeping defenses obscured from adversaries. To be clear, defenses following this approach can mitigate the adversarial sample problem. However, when applied to security critical applications such as malware classification, there are problems.

For a while, there has been a significant debate on security through obscurity, and a general consensus has been reached. This seems to be that obscurity is a perfectly valid security tactic but it cannot be trusted for complete security. Once a design or implementation is uncovered, users totally loose all the security gained by obscurity. To regain the security through obscurity, one has to devise a completely new design or implementation. As such, Kerckhoffs' principle [12] suggests obscurity can be used as a layer of defense, but should never be used as the only defense.

Inspired by this, we propose a new mechanism to improve a DNN's resistance to adversarial samples. Different from existing defenses, our proposed approach does not need model obscurity. In other words, even though we reveal the model, it will still be more than burdensome for adversaries to craft adversarial samples.

More specifically, we provide a standard DNN with a data transformation module, which projects the original data input into a new representation before it is passed through a consecutive DNN. This can be used as a defense for the following two reasons. First, the data transformation can stash away the space of adversarial manipulations to a carefully designed hyperspace. This makes it difficult for attackers to find adversarial samples harmful for the newly modified DNN. Second, as we will theoretically prove in Section IV, a data transformation module carefully designed can exponentially increase the computation complexity for an attacker to craft adversarial samples. This means that even though an attacker compromises obscurity and has the full knowledge about the armed DNN model (i.e., the training algorithm, data transformation module, dataset and hyper-parameters), an attack still cannot be launched that is detrimental to DNNs enhanced by other adversary-resistant techniques nor jeopardize model resistance.

Our proposed approach is beneficial for three critical reasons. First, it escalates a DNN's resistance to adversarial samples by having better security assurance. Second, our approach ensures that a DNN maintains desirable classification performance while requiring only minimal modification to its existing architectures. Third, while this work is primarily motivated by the need to safeguard DNN models used in critical security applications, it should be noted that the proposed technique is rather general and can be adopted to other applications where deep learning is applied, such as image recognition and sentiment analysis. We demonstrate this applicability using publicly-available datasets in Section V.

In summary, this work makes the following contributions.

- We propose a generic approach to facilitate the development of adversary-resistant DNNs without following the tactic of security through obscurity.
- Using our approach, we develop an adversary-resistant DNN, and theoretically prove its resistance cannot be jeopardized even if the model is fully disclosed.
- We evaluate the classification performance and robustness of our adversary-resistant DNN and compare it with that of existing defense mechanisms. Our result shows that our DNN exhibits similar – sometimes even better – classification performance but with superior model resistance to adversarial examples.

The rest of this paper is organized as follows. Section II introduces the adversarial sample problem. Section III discusses existing defense mechanisms and defines the problem scope of our research. Section IV presents our generic approach. In Section V, we develop and evaluate DNNs in the context of image recognition, sentiment analysis and malware classification. Finally, we conclude in Section VI.

II. Sample adversarial problem

An adversarial sample is a synthetic data sample crafted by introducing slight perturbations to a legitimate input sample. In multi-class classification tasks, such adversarial samples can cause a DNN to classify themselves into a random class other than the correct one (sometimes not even a reasonable alternative). Recent research [5] demonstrates that attackers can uncover such data samples through various approaches (e.g., [1], [9], [14], [22], [24], [26], [29], [36]) which can all be described as solving one of the following optimization problems. It should be noticed that we design our defense targeting the following attacks (optimization problems (1) and (2)), so we do not include work that leverages generative models to attack the target models (*e.g.*, those that adopt a generative adversarial network [34], [37]).

Consider the following:

$$\begin{aligned} \hat{x} &= arg\max_{\hat{x}} L(f(\hat{x};w);y), \\ \text{s.t.} &\quad \|\hat{x}-x\|_p < \varepsilon, \end{aligned} \tag{1}$$

or optimization problem

$$\begin{aligned} \hat{x} &= arg\min_{\hat{x}} L(f(\hat{x};w);\hat{y}), \\ \text{s.t.} &\quad \|\hat{x}-x\|_p < \varepsilon \text{ and } \hat{y} \neq y. \end{aligned} \tag{2}$$

Where the f represents a neural network, $\|\cdot\|_p$ indicates p norm and ε is the selected threshold. Here, optimization problem (1) indicates that an attacker searches for an adversarial sample $\hat{x}$, the prediction of which is as far as its true label, whereas optimization problem (2) indicates an attacker searches for adversarial sample $\hat{x}$ so that its prediction is as close as target label $\hat{y}$ where $\hat{y}$ is not equal to y, the true label of that adversarial sample.

In both optimization problems above, $L(\cdot)$ represents the aforementioned cost function and $f(\cdot)$ denotes the DNN model trained with the traditional learning method discussed above . $\|\cdot\|_p$ is p-norm – sometimes also specified as l_p distance – indicating the dissimilarity between adversarial sample $\hat{x}$ and its corresponding legitimate data sample x. With different values of p – the most popularly selected variable in adversarial learning research – the optimization problems above can be computed in the following manner.

(1) With $p = 2$, p-norm represents the measure of Euclidean distance. The constraint optimization problems above can be specified as unconstrained optimization problems by applying the KKT condition. Then, the unconstrained optimization problems can be solved by following either a first-order optimization method (e.g., stochastic gradient descent [5] and L-BFGS [29]) or a second-order method (e.g., Newton-Raphson method).

(2) With $p = 0$, the p-norm indicates the number of elements in a legitimate data sample that an attacker needs to manipulate in order to turn it into an adversarial sample. Different from the computation method above where the setting of $p = 0$ makes the unconstrained optimization problems not differentiable, the computation for the optimal solution has to follow an approximation method previously introduced by [5] or [22].

(3) With $p = \infty$, p-norm becomes a measure indicating the maximum change to individual features. As such, the optimal solution for (1) and (2) can be approximated by following the fast gradient sign method [9], which computes perturbation $\partial L(f(x;w);y)/\partial x$ (or $\partial L(f(x;w);\hat{y})/\partial x$), multiplies it by distortion scale ε, and then adds the product to the legitimate data sample x.

As is illustrated in the aforementioned optimization problems, in order to generate problematic adversarial samples, an attacker needs to know either a standard DNN model $f(\cdot)$ or know of a way to approximate $f(\cdot)$. A recent study [29] has revealed that an attacker could well approximate a standard DNN model using a traditional DNN training algorithm on an auxiliary training dataset. In this paper, we use the "*cross-model approach*" to refer to those adversarial sample crafting methods that rely upon the approximation of a standard DNN model.

III. Existing Defenses and Problem Scope

To counteract the adversarial learning problem described in the section above, recent research invents various training algorithms [3], [4], [7], [9], [17], [18], [21], [23], [28], [32], [33], [35] to improve the robustness of a DNN model. They indicate by using new training algorithms, one can improve a DNN's resistance to the adversarial samples crafted through the aforementioned cross-model approach. This is due to the fact that their training algorithms smooth a standard DNN's decision boundary, making adversarial samples – problematic to standard DNN models – no longer sufficiently effective. In this section, we summarize these defense mechanisms and discuss their limitations. Following our summary and discussion, we also define the problem scope of our research.

A. Existing Defense Mechanisms

Recently, research in hardening deep learning mainly focuses on two different tactics – data augmentation and model complexity enhancement. Here, we summarize them and discuss their limitations.

Data augmentation. To resolve the issue of "blind spots" (a more informal name given to adversarial samples), many methods that could be considered as sophisticated forms of data augmentation[1] have been proposed (i.e. [7], [9], [17], [21], [28], [32], [33]). In principle, these methods expand their training set by combining known samples with potential blind spots, the process of which has been called adversarial training [9], [17], [21], [28], [33]. Technically speaking, adversarial training can be formally described as adding a regularization term known as *DataGrad* to a DNN's training loss function [21]. The regularization penalizes the directions uncovered by adversarial perturbations (introduced in Section II). Therefore, adversarial training can work to improve the robustness of a standard DNN. Similar to adversarial training, both the layer-wise Parseval regularization [7] and random feature nullification [32] are new regularization terms proposed to defend against adversarial attacks.

Model complexity enhancement. DNN models are already complex with respect to both the nonlinear function that they try to approximate as well as their layered composition of many parameters. However, the underlying architecture is straightforward when it comes to facilitating the flow of information forwards and backwards, greatly alleviating the effort in generating adversarial samples. Therefore, several ideas [3], [4], [18], [23], [35] have been proposed to enhance the complexity of DNN models, aiming to improve the tolerance of complex DNN models with respect to adversarial samples generated from simple DNN models. For example, [23] developed a *defensive distillation* mechanism, which trains a DNN from data samples that are "distilled" from another DNN. By using the knowledge transferred from the other DNN, the learned DNN classifiers become less sensitive to adversarial samples. Similarly, [3] integrated a dimensionality reduction layer via Principal Component Analysis (PCA) before the DNN as a defense mechanism against evasion attacks. [4] proposed a region-based classification approach, which classifies a given testing sample by distributing information in a hypercube centered at that sample. [18] equipped a DNN with a detector and a reformer. With this design, a DNN can detect the adversarial sample and change the detected adversarial samples back to normal samples.

Limitation. While the aforementioned defenses have yielded promising results in terms of increasing model resistance, as is discussed in Section I, the scope of the model resistance provided is relatively limited. Consider for example adversarial training and defensive distillation [23]. Once an attacker obtains the knowledge of the new algorithms, instead of using a traditional DNN training algorithm to substitute the algorithm which the target DNN is trained with, he can build his own model with the new training algorithm, and then use it as the cross model to facilitate the crafting of adversarial samples. As we will show in Section V, the adversarial samples crafted through such new cross models sustain their offensiveness to the corresponding DNN models. Consider another example [3] which uses an invertable data transformation PCA as the defense layer. However, once an attacker is able to access the training algorithm, he can generate adversarial samples in the feature space of the PCA and then project these adversarial samples back to input space. This indicates that the effectiveness of existing defense mechanisms is highly dependent upon the obscurity of training algorithms and model parameters.

B. Problem Scope

With the existing defenses and their limitations in mind, we define here the problem scope of our research.

Similar to most previous research, if not all for hardening deep learning, we assume that an attacker crafts adversarial samples by solving the aforementioned optimization problems with derivative calculation (e.g., fast sign gradient descent or Newton-Raphson). We believe this assumption is realistic for the following reason.

Derivative calculation is the most general approach for solving an optimization problem. In the future, while one might be able to derive new forms of approaches in solving the aforementioned optimization problem, one still has to ensure the new approaches are computationally efficient. Without the aid from derivative calculation, this can be relatively difficult. Even if one could computationally efficiently resolve the aforementioned optimization problems, for example perhaps

[1]Data augmentation refers to artificially expanding the datasets. In the case of images, this also can involve deformations and transformations, such as rotation and scaling, of original samples to create new variants.

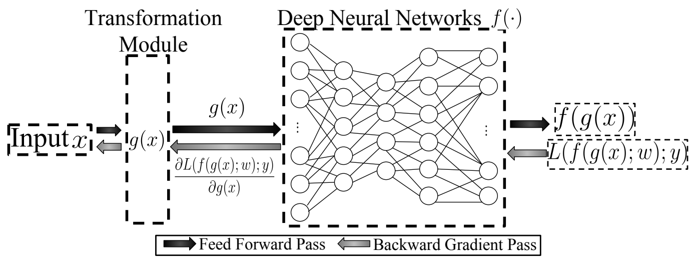


Fig. 1: A deep neural network with a data transformation module, projecting X, an input data sample, to new representation $g(X)$ prior to passing it through the DNN, $f(\cdot)$.

through relaxation, without derivative calculation, one still needs to prove the adversarial samples derived from such an approach are problematic. Given that relaxation reshapes an optimization problem, the "optimal" solution may not even be close to any local optima of that original optimization problem.

Different from prior research, we also assume that an adversary has not only the access to a DNN's structure as well as the dataset(s) used to build the DNN, but more importantly, the algorithm used to train the network. In other words, we assume a target DNN model is no longer obscure to an adversary and rather there exists full knowledge about a DNN model that will be exploited. We believe this assumption is more practical because there is little hope of keeping an adversary-resistant training algorithm completely secret from dedicated attackers. In the long run, any system for which the security relies upon the obscurity of its design is at enormous risk [12].

IV. Our Approach

To address this problem, we propose a new approach to harden DNN models. Technically speaking, it is basically model complexity enhancement, which improves model resistance by increasing model complexity. Different from the existing techniques mentioned above, our approach however goes beyond the scope of robustness they provide. It ensures that an attacker cannot perform the aforementioned attack to generate adversarial samples problematic to our learning model even if we reveal our training algorithm. In other words, our approach enhances a DNN's resistance to adversarial samples without the requirement of obscuring training algorithms.

As is discussed in the previous section, the adversarial learning problem can be viewed as an optimization problem. To resolve that optimization problem, one needs to conduct an analytical computation of gradients with respect to an input data sample and perform backward propagation. Therefore, we enhance a DNN's robustness not only by increasing the model complexity but, more importantly, restricting back-propagation.

More specifically, we integrate with a DNN a data transformation module, $g(\cdot)$ graphically indicated in Figure 1. As is illustrated in the figure, the data transformation module projects X, an input data sample to $g(X)$, a new representation, before passing it through a consecutive DNN. This transformation increases the complexity of a DNN model and augments its resistance to adversarial samples crafted through the aforementioned cross-model scheme. In addition, it blocks the backward flows of gradients. With this block, even if the underlying training algorithm is disclosed, an adversary cannot craft adversarial samples. In the following, we specify the design principle of our data transformation module, followed by its design detail and some necessary discussions.

A. Design Principle

To block the backward flow of gradients, the design of data transformation must satisfy three requirements. Most notably, the data transformation must be non-differentiable. As is discussed above, crafting adversarial samples requires the calculation of gradients as well as the back-propagation of those gradients. By making data transformation module $g(\cdot)$ non-differentiable, we can make gradient calculation intractable and thus obstruct the backward flow of gradients. More formally, we can choose a non-differentiable function $g(x)$, making the derivative difficult to be calculated, i.e.,

$$\frac{\partial^n L(f(g(\hat{x});w);y)}{\partial \hat{x}^n} = \frac{\partial^n L(f(g(\hat{x});w);y)}{\partial g(\hat{x})^n} \cdot \frac{\partial^n g(\hat{x})}{\partial \hat{x}^n}. \quad (3)$$

Here, $f(\cdot)$ represents the DNN model in tandem with the data transformation module, and $L(\cdot)$ denotes the cost function described in Section II. The derivative can be computed using either a first-order optimization method (e.g., gradient descent) or a second-order method (e.g., Newton-Raphson method), in which n is equal to 1 and 2 respectively.

While the non-differentiability feature restricts the crafting of adversarial samples, an adversary might still be able to generate adversarial samples. Since end-to-end gradient flow is blocked at the input layer of the successive DNN, back-propagation can only carry error gradients to the output of the transformation module. Given $g(\cdot)$, an adversary could construct an adversarial sample by inverting transformation module $g(\cdot)$ and passing the manipulated transformation output through the inversion of the transformation. More formally, the adversary can construct an adversarial sample by computing

$$g^{-1}(g(\hat{x}) + \frac{\partial^n L(f(g(\hat{x});w);y)}{\partial g(\hat{x})^n}). \quad (4)$$

In addition to making data transformation non-differentiable, therefore, we must further ensure that the inversion of the data transformation is computationally intractable. In other words, the data transformation $g(\cdot)$ needs to have the property of non-invertibility.

Satisfying the two requirements above ensures that our proposed approach can prohibit an attacker from crafting adversarial samples directly from the target DNN model, and there is no need to be concerned about the disclosure of training algorithms. However, the data transformation proposed may significantly jeopardize the accuracy of a DNN model if not carefully designed. Consider the following extreme case for example.

Hash functions like MD5 and SHA1 are one-way functions which have the properties of non-differentiability as well as non-invertibility. By simply using them as the transformation module, we can easily prohibit an attacker from crafting adversarial samples even if it is known which hash function we choose and how we integrate it with the DNNs. However, a hash function significantly changes the distribution of input data samples. With it used, a DNN model suffers from significant loss in classification performance. Last but not least, our design must therefore ensure that the data transformation preserves the distribution of the data representation. This can potentially make a DNN robust without sacrificing classification performance.

B. Design Detail

Following this design principle, we choose Locally Linear Embedding (LLE) [25], a non-parametric dimensionality reduction mechanism, to serve as the data transformation module. As we will discuss in the following, this representative non-parametric method is non-differentiable. More importantly, it can be theoretically proven that inverting a LLE is an NP-hard problem. Last but not least, LLE seeks a low-dimensional, neighborhood-preserving map of high-dimensional input samples, and thus is a method that best suited to preserving as much information in the input as possible. In the following, we first describe LLE and then expound upon the fact that, as a non-parametric dimensionality reduction method, LLE is non-differentiable. Furthermore, we theoretically prove LLE is computationally non-invertible.

1) Locally Linear Embedding: LLE is a non-parametric method designed to reduce input data dimensionality and at the same time preserve local properties of high-dimensional input in a lower-dimensional space. To some extent, this can ensure the distribution of high-dimensional data samples is as close as they are in a lower-dimensional space. Technically speaking, this is achieved by representing each high-dimensional data sample via a linear combination of its nearest neighbors. More formally, this can be expressed as $x_i = \sum_{j=1}^{N} w_{ij} \cdot x_j$. Here, x_i and x_j ($x_i, x_j \in \mathbb{R}^{1\times m}$) denote the i^{th} data sample and its j^{th} neighbor ($j = 1, 2..., N$), respectively. w_{ij} represents the weight, indicating the contribution of x_j to data sample x_i. As is described in [25], those weights (a.k.a. reconstruction weights) can be represented as a weight matrix W and computed by solving the following optimization problem:

$$\begin{aligned} \min_{W} \quad & \sum_i \|x_i - \sum_j w_{ij} \cdot x_j\|_2^2 \\ \text{s.t.} \quad & \sum_j w_{ij} = 1. \end{aligned} \tag{5}$$

In weight matrix W, LLE deems $w_{ij} = 0$ if x_j is not considered as a neighbor of x_i, and the total number of neighbors assigned to x_i is a carefully selected hyper-parameter. The neighboring relation between x_i and x_j depends on the value of the l_2 distance between x_i and x_j.

Since the reconstruction weights encode the local properties of the high-dimensional data, they can be used to preserve the data distribution at the time we perform dimensionality reduction. More specifically, LLE imposes the corresponding reconstruction weights on each lower-dimensional data sample via a similar linear combination, and then attempts to find $Y = \{y_1, y_2, \dots, y_N\}$, the lower-dimensional representation of $X = \{x_1, x_2, \dots, x_N\}$ by solving the following optimization problem:

$$\begin{aligned} \min_{Y} \quad & \sum_i \|y_i - \sum_j w_{ij} \cdot y_j\|_2^2 \\ \text{s.t.} \quad & \sum_i y_i = 0 \text{ and } \frac{1}{N}\sum_i y_i^{\mathrm{T}} y_i = I. \end{aligned} \tag{6}$$

where $y_i, y_j \in R^{1\times m_c}$, indicating y_i, y_j consist of m_c of elements.

In order to solve this optimization problem, the Rayleitz-Ritz theorem [10] is typically used. It computes the eigenvectors corresponding to the smallest nonzero eigenvalues of the inner matrix product $(I - W)^{\mathrm{T}} \cdot (I - W)$. For a detailed explication, please see Horn [10]. Here, $I \in \mathbb{R}^{N\times N}$ is an identity matrix, and $W \in \mathbb{R}^{N\times N}$ is the aforementioned reconstruction weight matrix.

LLE is specifically designed to retain the similarity between pairs of high dimensional samples when they are mapped to lower dimensions [25]. This property ensures that using LLE as a data transformation module satisfies the last design principle discussed in Section IV-A, preserving the distribution of the original data. More importantly, this property also helps bound the lower dimensional mapping of adversarial samples to a vicinity which is filled by mappings of original test samples that are very similar to these adversarial samples. As a result, there is a significantly lower chance that an adversarial sample acts as an outlier in the lower dimensional space. In other words, LLE makes a DNN more resistant to adversarial samples. In Section V, we empirically validate this important property.

2) Non-differentiability of LLE: Existing dimensionality reduction methods can be categorized as either parametric or non-parametric [30]. Parametric methods utilize a fixed amount of parameters to specify a direct mapping from high-dimensional data samples to their low-dimensional projections (or vice versa). This direct mapping is characterized by parameters, which are typically optimized to provide the best mapping performance. This is similar to the functionality provided by a standard DNN, which maps high-dimensional data samples to the final decision space through the differentiable function $f(\cdot)$. As such, the derivative of parametric methods typically can be computed in an analytically efficient manner. In other words, parametric methods are generally differentiable, and we argue this nature becomes a disadvantage for blocking the backward gradient flow.

On the contrary, non-parametric methods do not suffer from the issue above. For any non-parametric method, $g(\cdot)$, there is no way to express it in a closed form. Therefore, the derivative of $g(\cdot)$ can be computed only through a numeric not an analytical approach. More formally, this means the calculation of $\partial g(x)/\partial x$ needs to be completed through the calculation of limit $\lim_{h\to 0} \frac{g(x+h)-g(x)}{h}$. Given that a deep neural network takes as input each individual sample, which is discrete in the sample space, it is difficult to define the continuity of $g(\cdot)$ with traditional topologies and thus the differentiability of $g(\cdot)$ cannot be guaranteed. This indicates, as a member of non-parametric methods, that LLE well satisfies the first design principle discussed in Section IV-A (i.e., not capable of performing derivative calculation).

3) Non-invertibility of LLE: We validate the non-invertibility of LLE by theoretically proving that reconstructing original high-dimensional data from low-dimensional representations transformed by LLE is computationally intractable. More formally, we prove that, given a set of low-dimensional data $Y = \{y_1, y_2, \ldots, y_N\}$ ($Y \in \mathbb{R}^{N\times m_c}$) produced by LLE, reconstructing their original high-dimensional representations $X = \{x_1, x_2, \ldots, x_N\}$ ($X \in \mathbb{R}^{N\times m}$) from Y is at least an NP-hard problem.

Recall that LLE computes weight matrix W and utilizes it to project high-dimensional data samples to a lower-dimensional space. As a result, to restore high-dimensional data from its lower-dimensional representations, one has to recover that matrix by following the calculation similar to that shown in (5), except that x_i and x_j are replaced by y_i and y_j.

Once the weight matrix W is restored, the recovery of original high-dimensional data can be viewed as solving the following optimization problem:

$$\min_X \sum_i \left\|x_i - \sum_j w_{ij}\cdot x_j\right\|_2^2. \tag{7}$$

It is not difficult to realize that Equation (7) can be defined in the following quadratic form:

$$\min_X \sum_{i,j} m_{ij}\cdot(x_i\cdot x_j), \tag{8}$$

where $m_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_k w_{ki}w_{kj}$. Note that $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. We can express m_{ij} as a symmetric matrix M, where $M \in R^{N\times N}$.

Now, with the analysis above, the validation of non-invertibility amounts to proving that solving (8) is at least an NP-hard problem. In this work, we conduct this proof by introducing several constraints to this equation. Our basic idea is to use these constraints to relax the optimization problem in (8) to a nearby problem which can be easily proved as an NP-hard problem. More specifically, we introduce the following constraints:

$$\sum_i x_i = \vec{0}, \tag{9}$$

$$-1 \le x_{ij} \le 1, \forall i \in N, j \in m, \tag{10}$$

where $\vec{0}$ denotes a zero vector, and x_{ij} represents the j^{th} element in vector x_i.

With these constraints introduced into Equation 8, we can relax the optimization problem to a quadratic problem with a non-positive semi-definite constraint, which itself is a class of NP-hard problems [8]. In the following, we provide more details as to why the involvement of the aforementioned constraints transforms the optimization problem in (8) to this class of NP-hard problems.

Let $A \in \mathbb{R}^{N_A\times 1}$ denote a column vector which is the concatenation of x_i for $i = 1, \ldots, N$ and $N_A = N\times m$. Then, we have $A = (x_1, x_2, \cdots, x_N)^{\mathrm{T}}$. Let $q \in \mathbb{R}^{1\times N_A}$ denote a row vector in which every element is equal to 1. We further define matrices $P, Q \in \mathbb{R}^{N_A\times N_A}$ as follows:

$$P = \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1N} \\ P_{21} & P_{22} & \cdots & P_{2N} \\ \vdots & \vdots & & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} \end{pmatrix}, \quad Q = -I, \tag{11}$$

where $P_{ij} = m_{ij}\cdot J$. $J \in \mathbb{R}^{m\times m}$ is a matrix of ones where every element is equal to 1.

Given the constraint in (9), it is not difficult to discover $\sum_i x_i^T = \vec{0}$. Since the multiplication of a vector and its transpose derives a non-negative value, we have $\Sigma_i x_i x_i^T \ge 0$ and the constraint in (9) can be expressed as inequation $-\Sigma_i x_i x_i^T + \sum_i x_i^T + \alpha \le 0$, indicating there exists a positive number, α that always holds the inequity. By rewriting the inequation using the notations newly defined above, we can therefore transform the constraint in (9) into the form of $A^{\mathrm{T}}QA + qA + \alpha \le 0$.

Given the constraint in (10), we can easily derive inequation $\Sigma_i x_i x_i^T - N_A \le 0$, which can be further expressed as $\Sigma_i x_i x_i^T - N_A + \gamma \le 0$ indicating there always exists a constant, γ that holds the inequity. By rewriting both the constraint itself and this inequation using newly defined notations, we can derive constraints $\|A\|_\infty \le 1$ as well as $A^{\mathrm{T}}IA - N_A + \gamma \le 0$. As such, we can transform Equation (8) and the aforementioned constraints in (9) and (10) into following form:

$$\begin{aligned} \min\ & A^{\mathrm{T}}PA \\ \text{s.t.}\ & A^{\mathrm{T}}QA + qA + \alpha \le 0. \\ & A^{\mathrm{T}}IA - N_A + \gamma \le 0, \|A\|_\infty \le 1. \end{aligned} \tag{12}$$

Here, Q is negative semi-definite, and thus Equation (12) is a quadratic problem with a non-positive semi-definite constraint. According to [8], [31], Equation (12) belongs to a class of NP-hard problems, which implies the non-invertability of LLE.

C. Discussion

Here, we discuss related issues and possible attacks against our proposed technique.

Approximation of LLE. While the aforementioned discussion and theoretical proof have already indicated the effectiveness of our proposed approach, intuition suggests that an adversary might still come up with an attack. Specifically, one might approximate LLE using a parametric mapping and then substitute LLE accordingly. Since parametric mappings do not have the property of non-differentiability, the adversary can take advantage of the substitute, pass gradients through, and eventually craft adversarial samples. However, as we will show in Section V, even using the state-of-art approximation scheme, an adversary cannot craft problematic adversarial samples.

Other dimensionality reduction methods. As is described above, we choose LLE, a representative non-parametric dimensionality reduction method, to serve as the data transformation module. This is because that it has many properties needed for hardening a DNN, such as non-differentiability, non-invertibility, and the capability of preserving data distribution.

Going beyond LLE, there are other non-parametric dimensionality reduction methods that offer the similar properties, e.g., t-Distributed Stochastic Neighbor Embedding (t-SNE) [16] and Sammon Mapping [27]. However, they cannot be utilized in our problem domain for the following reason.

Deep neural networks exhibit superior performance when dealing with data in a relatively high dimensionality. Other non-parametric methods are typically designed more for tasks like visualization [16] where it is required that the dimensionality of the mappings be two or three. Using them as our data transformation module, they cannot provide high-dimensional data input for the DNN in tandem with the transformation, and may significantly jeopardize classification performance.

V. Evaluation

As is described in Section I, adversarial training [9] and defensive distillation [23] are the most representative techniques that have been proposed to defend against adversarial samples. Here, we use our proposed approach to train our own adversary resistant DNN (LLE-DNN), and then compare it with those enhanced by these two approaches. We believe that our comparison is representative since it reflects two generic approaches that most adversary-resistant deep learning techniques commonly follow.

A. Dataset

We evaluate our adversary-resistant DNN model by performing multiple experiments on several widely used datasets, including a dataset for malware detection [2], the MNIST dataset for image recognition [13] and the IMDB dataset for sentiment analysis [15].

Malware dataset: This is a collection of window audit logs, each of which ties to either a benign or malicious software sample. The dimensionality of the feature-space for each sample is reduced to 10,000 based on the feature selection metric in [2]. Each feature indicates the occurrence of either a single filesystem access or a sequence of access events, thus taking on the value of 0 or 1. Here, 0 indicates that the sequence of events did not occur while 1 indicates the opposite. For each software sample, it has been labeled with either 1 or 0, indicating malicious and benign software, respectively. The dataset is split into 26,078 training examples, with 14,399 benign and 11,679 malicious software samples, and 6,000 testing samples, with benign and malicious software samples evenly divided.

MNIST dataset: This is a large dataset of handwritten digits that is commonly used for training various image processing systems. It is composed of 70,000 greyscale images (of 28×28, or 784, pixels) of handwritten digits, split into a training set of 60,000 samples and a testing set of 10,000 samples.

IMDB dataset: This dataset consists of 25,000 movie reviews, with one half labeled as "positive" and the other "negative", indicating the sentiment of these reviews. We randomly split the dataset with 70% movie reviews for training and the remaining for testing. Following the procedure introduced in [19], we encoded the words in each movie review using a dictionary carrying 5,000 words most frequently used. Then, we utilized a word embedding technique [19] to convert each word into a vector with a dimensionality of 600. For each movie review, we linearly combined the vectors indicating the words appearing in that review, and then treat the embedding as the representation of that movie review.

B. Experimental Design

For each application described above, we train 4 DNN models using the traditional deep learning training method, adversarial training [9], defensive distillation [23] and our own approach[2] We measure their classification accuracy by applying the models to the corresponding testing datasets. By comparing their classification performance, we evaluate the influence that our proposed approach brings to a DNN. More specifically, we examine if LLE-DNN exhibits similar, if not the same or better, classification accuracy.

Since the goal of this work is to improve the robustness of a DNN model, we also evaluate our DNN models' resistance to adversarial samples[3]. In particular, we derive adversarial samples from the aforementioned testing datasets, test them against our DNN model and compare its model resistance with those of DNNs enhanced by the other two techniques [9], [23].

As is discussed in Section II, an attacker crafts adversarial samples through auxiliary models. In Table II, black-box and white-box indicate the auxiliary models trained through different schemes. More specifically, black-box represents the auxiliary model trained through the standard deep learning training scheme, indicating an attacker does not have sufficient knowledge about the underlying training algorithm and he can use only a standard approach to train a cross model and craft adversarial samples. White-box represents the auxiliary model trained exactly through the learning schemes proposed as a defense. This simulates a situation where a defense mechanism is publicly disclosed and an attacker exploits that mechanism to produce a highly similar – if not the same – model to craft adversarial samples. Note that, for both black-box and white-box tactics, we use the same hyperparameters and training dataset to build auxiliary models. More specifically, our auxiliary model training shares the same hyperparameters and training dataset with the standard DNN shown in Table I.

In addition to the methods described in Section II, the crafting of adversarial samples must ensure a slight perturbation introduced to a data sample in order to not undermine its semantics. In other words, we must make sure that, while

[2]Note that for each dataset, we ensure that 4 techniques share the same DNN architecture and specify the hyperparameters of these DNNs in Table I.

[3]As is introduced in Section II, there exists three different attacks: l_2, l_0, l_∞. Note that in this paper, we use attack methods in [5] for l_2 and l_0 attack, and choose fast gradient sign [9] for l_∞ attack

TABLE I: Hyperparameters of all the investigated DNN models.

Training Algorithms	Datasets	Hyper Parameters						
		DNN Structure	Activation	Optimizer	Learning Rate	Dropout	Batch	Epoch
Standard DNN	MNIST	784-500-300-100	Sigmoid	Adam	0.001	-	100	70
	Malware	3738-3000-1000-100-2	Relu	Adam	0.001	0.25	500	20
	IMDB	600-200-200-100-2	Tanh	Adam	0.001	0.5	100	40
Adversarial Training	MNIST	784-100-100-100-10	Tanh	SGD	0.1	0.25	100	60
	Malware	3738-3000-1000-100-2	Relu	Adam	0.001	0.25	500	20
	IMDB	600-300-100-50-2	Sigmoid	Adam	0.001	0.2	100	100
Distillation (T=20)	MNIST	784-200-50-20-10	Tanh	SGD	0.1	0.25	100	100
	Malware	3738-3000-100-20-2	Relu	SGD	0.1	0.25	100	20
	IMDB	600-100-100-50-2	Sigmoid	SGD	0.1	0.2	100	50
LLE-DNN	MNIST	200-200-100-10	Relu	Adam	0.001	0.5	100	50
	Malware	1000-500-200-100-2	Relu	Adam	0.001	0.5	100	50
	IMDB	500-300-200-100-2	Tanh	Adam	0.001	0.5	100	100

TABLE II: Comparison of model resistance to adversarial samples crafted in different manners. The values in the table represent the classification accuracy that DNN models exhibit when classifying adversarial samples.

Learning Technology	Black Box						White Box					
	MNIST			MALWARE	IMDB		MNIST			MALWARE	IMDB	
	l_∞	l_2	l_0	l_0	l_∞	l_2	l_∞	l_2	l_0	l_0	l_∞	l_2
Standard DNN	6.86%	6.40%	7.50%	26.19%	28.10%	29.56%	6.86%	6.40%	7.50%	26.19%	28.10%	29.56%
Distillation	87.06%	96.22%	47.36%	79.93%	82.65%	87.31%	34.43%	12.60%	8.43%	40.47%	48.98%	49.12%
Adv. Training	89.09%	96.23%	84.33%	96.70%	87.43%	87.66%	33.94%	14.44%	8.89%	43.09%	50.78%	51.0%
LLE-DNN	95.25%	96.59%	86.12%	95.02%	87.58%	87.69%	97.02%	97.45%	87.49%	94.66%	87.47%	87.53%

misleading a classifier to output the wrong class with high confidence, the perturbation to an input image should be nearly indistinguishable to the human eye, Also, a malicious software sample should not jeopardize software functionality nor break its malevolence, and for a movie review it should not break its semantic meaning. In the following, we describe how we fine-tune adversarial samples to preserve semantics for different applications.

Malware classification. Recall that our malware samples are represented by features, the value of which are binary, indicating the occurrence of an filesystem access or a sequence of access events. When generating adversarial samples, we cannot simply disable filesystem access events since this might jeopardize the functionality of the software sample and even break down its malevolence. With this in mind, some care must be taken.

In this work, our experiment follows the approach introduced in Section II. To be specific, we craft adversarial software samples with the setting of zero norm (i.e., l_0). This indicates that the manipulation to a sample is restricted to flipping binary feature values. Going beyond the adversarial sample crafting approach discussed in Section II, we also restrict that the value change of a feature can be only from 0 to 1 but not the opposite. This amounts to allowing the addition of new filesystem access events only. This manipulation strategy is reasonable since malware mutation techniques (e.g., [20]) can morph a malware sample by stitching together instructions from benign programs, making the malware perform additional filesystem accesses but not undermining its maliciousness nor its functionality. Since malware manipulation is done with the intent of fooling a malware classifier driven by a DNN, it should be noticed that we do not morph a benign software sample, making it malicious.

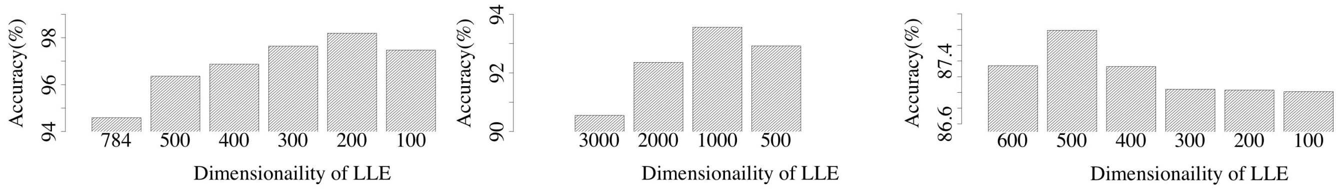
Image recognition. Image data samples contain less strict semantics than the malware data samples above. To preserve image semantics, making a perturbation nearly indistinguishable, we follow the approaches introduced in [5], [9], [29]. More specifically, we selected l_0, l_2 and l_∞ distance to represent the dissimilarity between an image and its corresponding adversarial sample. Especially, we restrict the l_∞ distance in a relative small range (i.e., $\epsilon \leq 0.15$) when crafting adversarial samples.

Sentiment Analysis. To generate adversarial samples for movie reviews, we again followed the approach introduced in Section II. To be specific, we configured p-norm with the setting to l_2 and l_∞. This is due to the fact that each review is encoded in a vector in which each element is a decimal, and l_2 and l_∞ distances represent the best measure for the dissimilarity between a movie review and its corresponding adversarial sample.

As is mentioned above, each vocabulary has been encoded in a vector with a dimensionality of 600, and we embedded a movie review by linearly combining corresponding vectors. When generating an adversarial sample by introducing a slight perturbation to the embedding, we recast the perturbation to only one vector. This ensures that we introduce only one word change to that review with the hope that it preserves the semantic meaning of that review as much as possible. However, one word change does not guarantee the invariance of the semantic meaning. For example, it would be obvious alteration to the semantic meaning if the replacement happens to be the negative word in *"... makes it the biggest disappointment I've experienced from cinema in years ..."*. As such, we manually try to choose the word that incurs minimal semantic change to that movie review.

C. Experimental Setup and Results

On the datasets described above, we first measure the accuracy of all the aforementioned defense techniques. We



(a) MNIST. (b) Malware. (c) IMDB.

Fig. 2: **Variation of classification accuracy vs. the dimensionality of data mappings.**

then measure their resistance to the adversarial samples crafted through the aforementioned tactics.

TABLE III: Classification accuracy on different datasets.

Learning technology	Accuracy		
	MNIST	MALWARE	IMDB
Standard DNN	98.45%	92.97%	87.89%
Distillation	98.46%	92.45%	87.36%
Adv. training	98.77%	91.48%	87.67%
LLE-DNN	98.19%	93.56%	87.79%

1) Classification Accuracy: To identify the optimal dimensionality to which LLE needs to map original data samples, we implemented several LLE-DNNs with different dimensionality settings of LLE mappings. Figure 2 shows the impact of dimensionality mapping upon the classification accuracy obtained by LLE-DNN. Across all three datasets, it we observe that the accuracy first increases when the dimensionality of the LLE mappings rises and then starts to decrease. In our experiments, we choose the highest classification accuracy to represent the performance of our LLE-DNN.

Table III presents the classification accuracy results obtained from all investigated DNNs for the testing datasets. Note that while prior work (e.g, [6]) has already demonstrated that a DNN can be trained with an error rate less than 1% on the MNIST benchmark, their performance improvement does not result from a DNN but from model ensembles or elastic distortions added to the training data. To study the influence of our proposed approach upon a standard DNN, we did not combine models nor augment with artificially distorted versions of the original training samples. The classification accuracy shown in the table has already represented the best performance that a standard DNN can achieve.

Similar to adversarial training and defensive distillation, the LLE-DNN is quite effective in preserving classification accuracy. This implies our proposed approach well preserves data sample distribution. For the malware classification task, it can be observed that LLE-DNN appears to be better at feature learning, achieving the highest classification accuracy among DNNs that we investigated. This is presumably due to the fact that malware data samples are highly sparse carrying a large amount of redundant information, and the data transformation module in LLE-DNN eliminates those redundancy and ameliorates the learning ability of a DNN.

2) Model Resistance: Table II illustrates the DNNs that we investigated as well as their accuracy in classifying adversarial samples. It can be observed that black-box adversarial samples can cut down the accuracy of the standard DNN to 6.86%, 6.40% and 7.50% under the attacks of l_∞, l_2 and l_0, respectively. In contrast, all of the defense mechanisms investigated demonstrate strong resistance to these black-box adversarial samples. This indicates, without sufficient knowledge on the underlying defense mechanisms, that it is difficult for an attacker to craft problematic adversarial samples. In other words, existing defense mechanisms can significantly enhance a DNN's resistance to adversarial samples if one can obscure the design of the defenses.

Despite the improvement in model robustness, we also observe that our LLE-DNN generally exhibits the best resistance to black-box adversarial samples, whereas the defensive distillation approach typically yields the least resistance. This is presumably due to the fact that the dimensionality reduction residing in LLE-DNN transforms adversarial samples into a subspace which no longer acts as outliers, while defensive distillation smooths only a classification decision boundary which does not significantly reduce the subspace of adversarial samples.

With regard to the white-box setting, we discover both adversarial training and defensive distillation suffer from these adversarial samples. Their resistance to white-box adversarial samples is significantly worse than those created under the black-box. This observation is consistent with that reported in [5]. The reason is that both techniques stash away the adversarial sample subspace, but the disclosure of defense mechanisms uncovers the paths in finding that subspace.

Different from adversarial training and defensive distillation, our LLE-DNN is naturally resistant to white-box adversarial samples. As is discussed in Section IV, our proposed approach stashes away the adversarial sample subspace and at the same time restricts derivative calculation. Even if our defense mechanism is revealed, therefore, it is still computationally difficult to find adversarial samples.

To perform quantitative comparison with the other two approaches, however, we approximate the data transformation in the LLE-DNN – non-parametric dimensionality reduction component – using a parametric model. To be specific, we choose a DNN to approximate LLE in that a DNN has a large amount of parameters which is typically viewed as the best approximation for non-parametric learning models [11]. With the support from this approximation, we treated the LLE-DNN as a white box and generated adversarial samples accordingly. We show its model resistance in Table II. It can be observed that our LLE-DNN still demonstrates strong resistance to white-box adversarial samples even if we substituted LLE to its best approximation. This implies that there might be a theoretical lower bound between a non-parametric model and its parametric approximation, which could naturally serve as a defense against white-box adversarial samples.

VI. Conclusion

It is well known that DNNs are vulnerable to adversarial samples. Existing defenses improve a DNN's resistance to adversarial samples by using the tactic of security through obscurity. Once the design of the defense is disclosed, the robustness they provide decreases. Motivated by this, this work introduces a new approach to increase the robustness of a DNN model by using a DNN model LLE, a non-parametric dimensionality reduction method. With this approach, we show that one can develop a DNN model resistant to adversarial samples even if its design details (i.e., the underlying training algorithm) are known. By demonstrating the performance of our enhanced DNNs across various applications, we argue that our proposed approach introduces nearly no degradation in classification performance. And for some applications, it even exhibits performance improvement. Future work will explore our approach on a wider variety of applications.

VII. Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPU. This project was supported in part by NSF grant CNS-1718459.

References

[1] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi, "Measuring neural net robustness with constraints," in *Proceedings of the 29th Conference on Neural Information Processing Systems (NIPS)*, 2016.

[2] K. Berlin, D. Slater, and J. Saxe, "Malicious behavior detection using windows audit logs," in *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2015.

[3] A. N. Bhagoji, D. Cullina, and P. Mittal, "Dimensionality reduction as a defense against evasion attacks on machine learning classifiers," *arXiv preprint arXiv:1704.02654*, 2017.

[4] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification," in *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC)*, 2017.

[5] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the 38th IEEE Symposium on Security and Privacy (S&P)*, 2017.

[6] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition," *CoRR*, 2010.

[7] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[8] A. d'Aspremont and S. Boyd, "Relaxations and randomized methods for nonconvex qcqps," *Stanford University*, 2003.

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[10] R. A. Horn and C. R. Johnson, "Matrix analysis. corrected reprint of the 1985 original," 1990.

[11] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, 1991.

[12] A. Kerckhoffs, "La cryptographie militaire," *Journal des Sciences Militaires*, 1883.

[13] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," 1998.

[14] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[15] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2011.

[16] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, 2008.

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

[18] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS)*, 2017.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th Conference on Neural Information Processing Systems (NIPS)*, 2013.

[20] V. Mohan and K. W. Hamlen, "Frankenstein: Stitching malware from benign binaries." in *6th USENIX Workshop on Offensive Technologies (WOOT)*, 2012.

[21] I. Ororbia, G. Alexander, C. L. Giles, and D. Kifer, "Unifying adversarial training algorithms with flexible deep data gradient regularization," *Neural Computation*, 2016.

[22] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the 1st IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.

[23] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proceedings of the 37th IEEE Symposium on Security and Privacy (S&P)*, 2016.

[24] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*, 2017.

[25] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, 2000.

[26] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial manipulation of deep representations," *arXiv preprint arXiv:1511.05122*, 2015.

[27] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on computers*, 1969.

[28] A. Sinha, H. Namkoong, and J. Duchi, "Certifiable distributional robustness with principled adversarial training," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[29] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[30] L. van der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality reduction: A comparative review," 2008.

[31] S. A. Vavasis, *Nonlinear Optimization: Complexity Issues*, 1991.

[32] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia II, X. Xing, X. Liu, and C. L. Giles, "Adversary resistant deep neural networks with an application to malware detection," in *Proceedings of the 23rd International Conference on Knowledge Discovery and Data Mining (KDD)*, 2017.

[33] X. Wu, U. Jang, L. Chen, and S. Jha, "Reinforcing adversarial robustness using model confidence induced by adversarial training," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[34] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[35] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Proceedings of the 24th Network and Distributed Systems Security Symposium (NDSS)*, 2018.

[36] X. Xu, X. Chen, C. Liu, A. Rohrbach, T. Darell, and D. Song, "Fooling vision and language models despite localization and attention mechanism," in *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[37] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.