Dimensionality Reduction as a Defense against Evasion Attacks on Machine Learning Classifiers

Arjun Nitin Bhagoji
Princeton University

Daniel Cullina
Princeton University

Prateek Mittal
Princeton University

Abstract

We propose the use of *dimensionality reduction* as a *defense* against evasion attacks on ML classifiers. We present and investigate a strategy for incorporating dimensionality reduction via Principal Component Analysis to enhance the resilience of machine learning, targeting both the classification and the training phase. We empirically evaluate and demonstrate the feasibility of dimensionality reduction of data as a defense mechanism against evasion attacks using multiple real-world datasets. Our key findings are that the defenses are (i) effective against strategic evasion attacks in the literature, increasing the resources required by an adversary for a successful attack by a factor of about two, (ii) applicable across a range of ML classifiers, including Support Vector Machines and Deep Neural Networks, and (iii) generalizable to multiple application domains, including image classification, and human activity classification.

1 Introduction

We are living in an era of ubiquitous machine learning (ML) and artificial intelligence. Machine learning is being used in a number of essential applications such as image recognition [1], natural language processing [2], spam detection [3], autonomous vehicles [4, 5] and even malware detection [6, 7]. In addition, recent advances in deep learning [8, 9] have demonstrated classification accuracy that is close to the accuracy obtained by humans, enabling the widespread deployment of ML systems. Given the ubiquity of ML applications, it is increasingly being deployed in *adversarial* scenarios, where an attacker stands to gain from the failure of a ML system to classify inputs correctly. The question then arises: are ML systems secure in adversarial settings?

Adversarial Machine Learning: Starting in the early 2000s, there has been a considerable body of work [10–12] exposing the vulnerability of machine learning algorithms to strategic adversaries. For example, *poisoning attacks* [13] systematically introduce adversarial data during the *training* phase with the aim of causing the misclassification of data during the test phase. On the other hand, *evasion attacks* [14–16] aim to fool existing ML classifiers trained on benign data by adding *strategic perturbations* to *test inputs*.

Evasion attacks: In this paper we focus on evasion attacks in which the adversary aims to perturb test inputs to ML classifiers in order to cause misclassification. Evasion attacks have been proposed for a variety of machine learning classifiers such as Support Vector Machines [14, 17], tree-based classifiers [17, 18] such as random forests and boosted trees and more recently for neural networks [15, 16, 19–22]. The vulnerability of applications that use machine learning, such as face detection [23], voice command recognition [24] and PDF malware detection [25] has also been demonstrated, highlighting the need for defenses. Surprisingly, it has also been shown that the evasion properties of adversarially modified data (for a particular classifier) persist across different ML classifiers [16], which allows an adversary with limited knowledge of the ML system to attack it. Thus, it is crucial to consider the possibility of adversarial data and evasion attacks while using ML systems in adversarial contexts. However, very few defenses [18, 26] exist against these attacks, and the applicability of each is limited to only certain known attacks and specific types of ML classifiers (see Section 7 for a detailed description of and comparison with previous work).

1.1 Contributions

Through extensive evaluations, we find that our defense mechanism demonstrably reduces the success of evasion attacks. To the best of our knowledge, it is the only defense against evasion attacks with the following properties: (1) applicability across multiple ML classifiers (such as SVMs, DNNs), (2) applicability across multiple application domains (image and activity classification), and (3) mitigation of multiple attack types, including strategic ones. Further, the tunability of our defense allows a system designer to pick appropriate operating points on the utility-security tradeoff curve depending on the application.

1.1.1 Defense

In this paper, we propose the use of *dimensionality reduction* of data as a defense against evasion attacks on ML systems. Dimensionality reduction techniques such as Principal Component Analysis aim to project high-dimensional data to a lower-dimensional space while meeting specific conditions [27, 28].We present and investigate a strategy for incorporating dimensionality reduction to enhance the resilience of machine learning, targeting both the classification and the training phase. We consider an approach in which dimensionality reduction is applied to the training data to *enhance the resilience of the trained classifier*, as well as to the test data for resilient classification.

1.1.2 Empirical Evaluation

We empirically demonstrate the feasibility and effectiveness of our defenses using:

- multiple ML classifiers, such as Support Vector Machines (SVMs) and Deep Neural Networks (DNNs);
- several distinct types of evasion attacks, such as an attack on Linear SVMs from Moosavi-Dezfooli et. al. [29], and on deep neural networks from Goodfellow et. al. [19], as well as strategic attacks targeting our defense ;
- a variety of real-world datasets/applications: the MNIST image dataset [30] and the UCI Human Activity Recognition (HAR) dataset [31].

Our key findings are that even in the face of a powerful adversary with almost complete knowledge of the ML system: i) our defense leads to significant increases of up to 5× in the degree of modification required for a successful attack and equivalently, reduces adversarial success rates by around $2-50\times$ at fixed degrees of modification, ii) the defense can be used for different ML classifiers with minimal modification of the original classifiers, while still being effective at combating adversarial examples, and iii) there is a modest variation of about 1-4% in the classification success on benign samples in most cases. We also provide an analysis of the utility-security tradeoff curve as well as the computational overheads incurred by our defense. Our results may be reproduced using the open source code available at `https://github.com/inspire-group/ml_defense`.

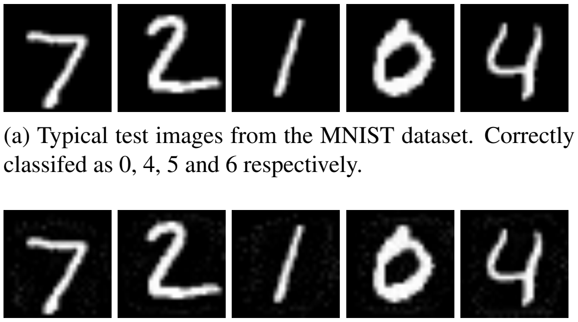
However, our defense does not completely solve the problem of evasion attacks, since it reduces adversarial success rates at fixed budgets, but does not make them negligible in all cases. In Section 4, we discuss the range of budgets available to the adversary in different application scenarios and make explicit the scenarios in which our defense is effective. We hope that our work inspires further research in combating the problem of evasion attacks and securing machine learning based systems.

The rest of the paper is organized as follows: first, in Section 2 we present the necessary background on adversarial machine learning. Then, in Section 3 we describe our defense. We then set up and present our empirical evaluation in Sections 4 and 5 respectively. We discuss our results in Section 6. Finally, we provide a detailed overview of the related work in Section 7 and conclude in Section 8.

2 Adversarial machine learning

In this section, we present the required background on adversarial machine learning, focusing on both (a) ML classifiers such as SVMs and DNNs as well as (b) evasion attacks that induce misclassification by perturbing the test input.

Motivation and Running Examples: We use image data from the MNIST dataset [30] (see Section 4 for details) for our running examples. Figure 1(a) depicts example test images from the MNIST dataset that are correctly classified by a SVM classifier, while Figure 1(b) depicts adversarially crafted test images (perturbed images using the evasion attack of Papernot et. al. [17]), which are misclassified by a SVM classifier.



(a) Typical test images from the MNIST dataset. Correctly classifed as 0, 4, 5 and 6 respectively.

(b) Corresponding adversarial images obtained using the evasion attack on Linear SVMs [29]. Now, **misclassified as 9, 9, 3, 2 and 0.** respectively.

Figure 1: **Comparison of benign and adversarial images taken from the MNIST dataset.**

2.1 Classification using machine learning

In this paper, we focus on *supervised* machine learning, where a classifier is trained on data with pre-existing labels. A trained, supervised machine learning classifier is a function that takes as input a data point $\mathbf{x} \in \mathbb{R}^d$ (or $\{0,1\}^d$ in the case of binary data vectors), and produces a scalar output $\hat{y} \in C$, where C is the set of all possible classification categories. For example, in the case of the MNIST dataset, $\mathbf{x}$ would be a 28×28 pixel grayscale image of a handwritten digit and C would be the finite set $\{0,1,2,3,4,5,6,7,8,9\}$.

2.2 Attacks against machine learning systems

In this subsection we first discuss the adversarial model under consideration, after which we describe general evasion attacks and finally, evasion attacks on specific ML classifiers.

Notation: We denote the complete training data as S_{train}, the complete test data as S_{test}, the machine learning classifier in use as f and the specific parameters chosen for the ML system as $\boldsymbol{\theta}$. The original dimension of the data is denoted as d. We then represent the adversary's attack algorithm as $A(\mathbf{x}_{\text{in}}|K)$, where $\mathbf{x}_{\text{in}}$ denotes the input the adversary starts with and K denotes adversarial knowledge, which may be any subset of $\{S_{\text{train}}, f, \boldsymbol{\theta}\}$. The adversarial sample generated by A is denoted by $\tilde{\mathbf{x}}$.

2.2.1 Adversarial goals and capabilities

In this paper, we focus on attacks where the adversary's objective is either to modify a benign, existing input so that it is *misclassified as any other class* or to cause it to be classified in a *targeted class* different from the original class. Note that these aims are equivalent in the case of binary classifiers.

Our baseline assumptions are that the adversary has the following capabilities.

- The adversary has complete knowledge of the training set the original classifier has been trained on, i.e. she knows the type of feature vectors which the classifier takes as input.
- The adversary knows the classifier structure, hyperparameters, and training procedure.
- Adversarial examples are created offline by the adversary, and submitted to the ML classifier during the test phase.

In summary, $\tilde{\mathbf{x}} = A(\mathbf{x}_{in}|S_{\text{train}}, f, \boldsymbol{\theta}, K_{\text{add}})$, where K_{add} denotes any additional knowledge about the system the adversary may have.

Our assumptions on the adversary's capabilities are conservative since from a security perspective, being able to defend against attacks where the adversary has almost complete knowledge of the system ensures that the defense does not rely on security through obscurity and is robust even in the face of powerful attacks. Further, even adversaries with limited access (such as black-box access) and knowledge of the ML system can infer the classifier well enough to mount evasion attacks, which may be almost as successful as those mounted by an adversary with full access [17, 32–34], justifying our assumptions.

2.2.2 Evasion attacks

In normal operation, when there is no adversary, upon some input $\mathbf{x}_i \in S$, f outputs $\hat{y}$, where S is the set of inputs. The fraction of the training set for which the output class matches the true class y [1] is α, i.e.

$$\alpha(S) = \frac{\#\{(\mathbf{x},y) \in S : f(\mathbf{x}) = y\}}{\#S}, \tag{1}$$

where # gives the cardinality of a set.

The adversary's goal is to design an algorithm A acting on $\mathbf{x} \in S$ which generates adversarial examples, i.e. $A(\mathbf{x}) = \tilde{\mathbf{x}}$. Let

$$S^{\text{adv}} = \{(A(\mathbf{x}), y) : (\mathbf{x}, y) \in S\},$$

the set of adversarially modified examples. These modifications should

- *increase the misclassification percentage* as compared to the normal operation of the classifier, i.e. $\alpha(S^{\text{adv}}) < \alpha(S)$,
- *not be detectable as anomalous* by humans in the case of human-interpretable data such as images and text, and by rule-based detection systems in the case of data like malware samples, network and system logs etc. For example, in the case of malware, the adversary is constrained by the fact that her modifications must ensure the final samples are still malicious.

We next discuss adversarial perturbations and in the case of image data, their perceptibility to humans.

2.2.3 Adversarial perturbations

Modeling human perception of image perturbations is a difficult problem. As a proxy for human perceptibility, we will measure the degree of modification as $\|A(\mathbf{x}) - \mathbf{x}\|$

[1] Assuming there are ground truth labels to compare against for all elements in S

for some norm $\|\cdot\|$. In particular, we will consider perturbations constrained in terms of the ℓ_2 norm, i.e. $\|\tilde{\mathbf{x}}-\mathbf{x}\|_2 \leq \varepsilon$, where ε dictates the strength of the perturbation. A detailed description of the relation between the various norms used to constrain adversarial perturbations and their perception is given in [35].

Now, we define the minimal perturbations required for different adversarial goals to be achieved. In order to cause *misclassification in a specific class* z, the minimum perturbation that has to be added to an input $(\mathbf{x}, y)$, where $z \neq y$, is

$$\Delta(\mathbf{x}, z) = \inf_{\tilde{\mathbf{x}}}\{\|\tilde{\mathbf{x}}-\mathbf{x}\| : f(\tilde{\mathbf{x}}) = z\}.$$

This is the minimum distortion required to cause $\mathbf{x}$ to be classified as z. The mimimum distortion required to cause $\mathbf{x}$ to be *misclassified in any class* is

$$\Delta(\mathbf{x}) = \min_{z \in C \setminus \{y\}} \Delta(\mathbf{x}, z).$$

For image data, the relationship between these quantities and the minimum detectable distortion determines the robustness of the classifier f to adversarial perturbations. In Figure 1, the images correspond to a perturbation value that causes the Linear SVM to classify almost all inputs wrong but the perturbation is scarcely visible to the human eye. This indicates that the standard form of Linear SVMs is not robust to adversarial perturbations. Further discussion of the metrics used to constrain the adversary is given in Section 4.4.

2.3 Evasion attacks on specific classifiers

We now describe existing attacks from the literature on specific ML classifiers and show some adversarial examples from the MNIST dataset for each. A summary of the various attacks is given in Table 1.

2.3.1 Optimal attacks on Linear SVMs

In the multi-class classification setting for Linear SVMs, a classifier g_i is trained for each class $i \in C$, where

$$g_i : \mathbf{x} \mapsto \mathbf{w}_i^T \mathbf{x} + b_i. \tag{2}$$

$\mathbf{x}$ is then assigned to the class $f(\mathbf{x}) = \text{argmax}_{i \in C}\, g_i(\mathbf{x})$. Given that the true class is $t \in C$, the objective of an untargeted attack is to find the closest point $\tilde{\mathbf{x}}$ such that $f(\tilde{\mathbf{x}}) \neq t$.

From [29], we know that for an optimal untargeted attack on affine multi-class classifiers, i.e. if we only care that $f(\tilde{\mathbf{x}}) \neq t$ with the smallest possible $\|\tilde{\mathbf{x}}-\mathbf{x}\|_2$, the optimal choice of $\tilde{\mathbf{x}}$ is $\tilde{\mathbf{x}}_k$, where

$$k = \underset{j}{\text{argmin}} \frac{|g_t(\mathbf{x}) - g_j(\mathbf{x})|}{\|\mathbf{w}_t - \mathbf{w}_j\|}, \tag{3}$$

and thus

$$\tilde{\mathbf{x}}(\varepsilon) = \mathbf{x} + \varepsilon \frac{\mathbf{w}_t - \mathbf{w}_k}{\|\mathbf{w}_t - \mathbf{w}_k\|}, \tag{4}$$

where ε controls the magnitude of the perturbation. The minimum ε required to cause a misclassification is $\varepsilon^* = \frac{|g_t(\mathbf{x})-g_k(\mathbf{x})|}{\|\mathbf{w}_t-\mathbf{w}_k\|}$. It can easily be verified that $f(\tilde{\mathbf{x}}(\varepsilon^*)) = k$. Note that this attack is constrained in terms of the ℓ_2 norm, since $\|\varepsilon \frac{\mathbf{w}_t-\mathbf{w}_k}{\|\mathbf{w}_t-\mathbf{w}_k\|}\| = \varepsilon$.

2.3.2 Gradient based attacks on neural networks

The Fast Gradient Sign (FGS) attack is an efficient attack against neural networks introduced in [19]. In this case, the adversarial examples are generated by adding adversarial noise proportional to the sign of the gradient of the loss function $(\text{sign}(\nabla J_f(\mathbf{x}, y, \theta)))$, where $J_f(\cdot)$ denotes the loss function and θ denotes the hyperparameters used for training. The gradient can be efficiently calculated using backpropagation. Concretely,

$$\tilde{\mathbf{x}} = \mathbf{x} + \eta \, \text{sign}(\nabla J_f(\mathbf{x}, y, \theta)), \tag{5}$$

where η is a parameter that can be varied by the adversary to control the effectiveness of the adversarial examples. As η increases, the adversarial success rate typically increases as well. However, larger perturbations may make identification difficult for humans as well in the case of images (see Appendix for images modified with a range of η). The FSG attack is constrained in terms of the ℓ_∞ norm, since $\|\eta \, \text{sign}(\nabla J_f(\mathbf{x}, y, \theta))\|_\infty = \max_i |\eta \, \text{sign}(\nabla J_f(\mathbf{x}, y, \theta))_i| = \eta$, which controls the strength of the perturbation.

The FGS attack and the attack on Linear SVMs are constrained according to different norms. To facilitate a comparison of the robustness of various classifiers as well as the effectiveness of our defense across them, we propose a modification of the FGS attack which is constrained in terms of the ℓ_2 norm. Denoting this as the Fast Gradient (FG) attack, we define the adversarial examples to be equal to

$$\tilde{\mathbf{x}} = \mathbf{x} + \eta \frac{\nabla J_f(\mathbf{x}, y, \theta)}{\|\nabla J_f(\mathbf{x}, y, \theta)\|}. \tag{6}$$

For the FG attack, η is a constraint on the ℓ_2 norm of the perturbation.

2.4 Dimensionality reduction for machine learning

While dealing with high-dimensional data (in the sense that each sample has a large number of features), it is often difficult to make sense of which features are important. Application constraints may also make it impractical to carry out learning tasks on the data in the original, high-dimensional space. Dimensionality reduction

Table 1: **Summary of attacks on Linear SVMs and neural networks.**

Attack	Classifier	Constraint	Intuition
Optimal attack on Linear SVMs [29]	Linear SVMs	ℓ_2	Move towards classifier boundary
Fast Gradient	Neural Networks	ℓ_2	First order approximation to direction of smallest perturbation
Fast Gradient Sign [19]	Neural Networks	ℓ_∞	Constant scaling for each pixel models perception better

is then a useful preprocessing step for high-dimensional data. It can also help alleviate the problems associated with the 'curse of dimensionality' [28]. In such cases, the data is first projected onto a lower-dimensional space before providing it as input to the ML system. Common dimensionality reduction algorithms are PCA [27], random projections [36] and Kernel PCA [37]. In this paper, we make novel use of dimensionality reduction methods such as PCA to not only aid in interpretability and increase efficiency, but also to increase the robustness of ML systems to adversarial examples.

3 Dimensionality reduction based defense

In the previous section, we have seen that a number of ML classifiers are vulnerable to a variety of different evasion attacks. These vulnerabilities make it easy for adversaries to craft inputs to meet their desired objective, which could range from fooling a self-driving car [4, 5] and making it crash to escaping fraud and malware detection [6, 7]. There is clearly a need for a defense mechanism for ML systems which is effective against a variety of attacks, since a priori, the owner of the system has no knowledge of the range of attacks possible against the system. Further, finding a defense that works across a range of classifiers can direct us to a better understanding of why ML systems are vulnerable in the first place. In this section, we present a defense that is agnostic at least to common attacks from the literature, and remains effective even in the presence of a stronger adversary with knowledge of the defense. Further, the defense makes multiple types of ML classfiers operating in different application scenarios more robust as shown in Section 5. The defense is based on dimensionality reduction, which was introduced briefly in Section 2.

Now, we present our threat model and design goals, followed by our dimensionality reduction based defense mechanism. In what follows, we refer to the original ML classification system with the addition of the defense mechanism as the 'robust classification pipeline'.

3.1 Threat model

Given the adversarial capabilities defined in Section 2.2, we consider two categories of evasion attacks in our threat model when evaluating the effectiveness of the defense mechanism:

1. **Vanilla attacks**: This category considers a range of existing attacks from the literature *without any modification*. Since these attacks were designed for machine learning systems without taking into account any defenses, the adversary is *unaware* of the defense mechanism. We note that the design of defense mechanisms that can successfully mitigate even vanilla/known attacks is a significant challenge facing the security community.

2. **Strategic attacks**: In the second case, the adversary is *aware* of the defense mechanism. This implies that the adversary will take the effects of the defense into account when crafting adversarial examples, and possibly even switch to a different attack strategy optimized for the particular defense in use.

In order to evaluate our defenses thoroughly, we investigate new attack strategies of the second type, optimized for our defense. We show that our defenses are effective even in this case, albeit to a lesser extent.

3.2 Design goals

The primary goal of any supervised machine learning classifier is to achieve the best possible accuracy on the test set. Further, it is desirable that the machine learning classifier is as efficient as possible. In this light, we have to ensure that any defense mechanism does not have an overly large impact on either the efficiency or accuracy of the overall classification process. Thus, the design goals for a defense mechanism are:

1. **High classification accuracy**: Adding the defense mechanism to the overall classification pipeline should maintain high classification accuracy on the benign test set. In fact, an ideal defense mechanism

may even increase the classification accuracy on the test set by reducing overfitting, increasing the capacity of the classifer etc.

2. **High efficiency**: Both time and space complexity play a role in determining the efficiency of an algorithm. In the worst case, the defense mechanism should not more add more than a polynomial overhead to the running time and required space for the original classifier. In an ideal case, the defense mechanism would make the overall pipeline more efficient in both time and space.

3. **Security**: We define the vulnerability of a machine learning classifier as the fraction of adversarially modified input data that is misclassified (subject to the constraint that the original input data was correctly classified[2]). In order for a defense to be effective, we need this misclassification fraction to be lower than for the original classifier. Thus, for a *particular attack*, we can say that the pipeline is more secure if the fraction of adversarial examples that is misclassifed is less than the original classifier, for a comparable attack parameter.

4. **Tunability**: Depending on the application, performance (i.e. classification accuracy), efficiency or security may be the primary concern for the user of the machine learning system. The defense mechanism should give the user the option of navigating different points in the multi-dimensional tradeoff space of performance, utility and security by modifying its parameters.

In Section 5 we quantify how well our defense satisfies the design goals laid out above. Now, we provide an overview of the robust classification pipeline and how it helps defend against existing attacks.

3.3 Overview of defense

In our defense, we leverage dimensionality reduction techniques to project high dimensional data onto a lower dimensional space and to make the classifier more resilient by modifying the training phase. In the first step of our defense, the ML system user chooses a reduced dimension $k < d$. Then, the training data $\mathbf{X}^{\text{train}}$ is transformed to a lower dimensional space using a dimensionality reduction algorithm DR which takes k and $\mathbf{X}^{\text{train}}$ as input. A new classifier f_k is then trained on the reduced dimension *training set*. At test time, all inputs are transformed to a lower dimensional space using the same algorithm DR and the reduced dimension inputs $\mathbf{x}_k$ are provided as input directly to f_k. We now describe a concrete instantiation of the defense using PCA.

3.4 Defense using PCA

3.4.1 PCA in brief

PCA [27] is a linear transformation of the data that identifies so-called 'principal axes' in the space of the data, which are the directions in which the data has maximum variance, and projects the original data along these axes. The dimensionality of the data is reduced by choosing to project it along k principal axes. The choice of k depends on what percentage of the original variance is to be retained. Intuitively, PCA identifies the directions in which the 'signal', or useful information in the data is present, and discards the rest as noise.

Concretely, let the data samples be $\mathbf{x}_i \in \mathbb{R}^d$ for $i \in \{1,\ldots,n\}$. Let $\mathbf{X}'$ be a $d \times n$ matrix such that each column of $\mathbf{X}'$ corresponds to a sample, and let $\mathbf{1} \in \mathbb{R}^n$ be the vector of all ones. Then, $\frac{1}{n}\mathbf{X}'\mathbf{1}$ is the mean of the samples and $\mathbf{X} = \mathbf{X}'(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$ is the matrix of centered samples. The principal components of $\mathbf{X}$ are the normalized eigenvectors of its sample covariance matrix $\mathbf{C} = \mathbf{X}^T\mathbf{X}$. More precisely, because $\mathbf{C}$ is positive semidefinite, there is a decomposition $\mathbf{C} = \mathbf{U}\Lambda\mathbf{U}^T$ where $\mathbf{U}$ is orthonormal, $\Lambda = \text{diag}(\lambda_1,\ldots,\lambda_n)$, and $\lambda_1 \geq \ldots \geq \lambda_n \geq 0$. In particular, $\mathbf{U}$ is the $d \times d$ matrix whose columns are unit eigenvectors of $\mathbf{C}$, i.e. $\mathbf{U} = [\mathbf{u}_1,\ldots,\mathbf{u}_d]$, where $\mathbf{u}_i$ is the i^{th} normalized eigenvector of $\mathbf{C}$. The eigenvalue λ_i is the variance of $\mathbf{X}$ along the i^{th} principal component.

Each column of $\mathbf{U}^T\mathbf{X}$ is a data sample represented in the principal component basis. Let $\mathbf{X}_k$ be the projection of the sample data in the k-dimensional subspace spanned by the k largest principal components. Thus $\mathbf{X}_k = \mathbf{U}\mathbf{I}_k\mathbf{U}^T\mathbf{X} = \mathbf{U}_k\mathbf{U}_k^T\mathbf{X}$, where $\mathbf{I}_k$ is a diagonal matrix with ones in the first k positions and zeros in the last $d-k$ and $\mathbf{U}_k = [\mathbf{u}_1,\ldots,\mathbf{u}_k]$. The amount of variance retained is $\sum_{i=1}^k \lambda_i$, which is the sum of the k largest eigenvalues.

3.4.2 Implementing the defense

In our defense using PCA, the principal components are computed using an appropriately mean-centred and scaled matrix of *training data*. If there are n training samples, then the matrix used to compute the principal components is $\mathbf{X}^{\text{train}}$, which is a $d \times n$ matrix. All subsequent data samples are projected onto these principal components that are initially computed.

[2] See the Appendix for discussion on similar security definitions without this constraint

```
Algorithm 1 DRtrain
Input: $f_k^0$,$\boldsymbol{\theta}_k$, $k$, $\mathbf{X}^{\text{train}}$, $\mathsf{Train}$
Output: $f_k$
  Use PCA to find the matrix $\mathbf{U}_k$ of the top $k$ principal components used to reduce the dimension of $\mathbf{X}^{\text{train}}$
  Compute the reduced dimension training set $\mathbf{U}_k^T \mathbf{X}^{\text{train}}$
  Train the reduced dimension classifier $f_k' = \mathsf{Train}(f_k^0, \boldsymbol{\theta}_k, \mathbf{U}_k^T \mathbf{X}^{\text{train}})$
  Let $f_k = \mathbf{x} \mapsto f_k'(\mathbf{U}_k \mathbf{x})$
```

The resilient lower dimensional classifier f_k is generated using Algorithm 1. It takes a classifier training algorithm Train and adapts it to produce a variant that uses dimensionality reduction. The inputs to Algorithm 1 are:

- f_k^0: This is the initial, untrained classifier of the desired class initialized with appropriate parameters.
- $\boldsymbol{\theta}_k$: These are the parameters used in the training of f_k which are identical to $\boldsymbol{\theta}$ except for any dimension specific parameters which may change due to the reduced dimension training.
- k: This is the reduced dimension to be used. k may be changed to adjust the utility-security trade-off.
- Train: This is the algorithm used to train classifiers of the desired class. An example of a commonly used training algorithm is Stochastic Gradient Descent [38] .

There is a second way to implement dimensionality reduction through principal component analysis.

```
Algorithm 2 Alternative DRtrain
Input: $f_k^0$,$\boldsymbol{\theta}_k$, $k$, $\mathbf{X}^{\text{train}}$, $\mathsf{Train}$
Output: $f_k$
  Use PCA to find the matrix $\mathbf{U}_k$ of the top $k$ principal components used to reduce the dimension of $\mathbf{X}^{\text{train}}$
  Compute the projected training set $\mathbf{U}_k \mathbf{U}_k^T \mathbf{X}^{\text{train}}$
  Let $f_k = \mathsf{Train}(f_k^0, \boldsymbol{\theta}_k, \mathbf{U}_k \mathbf{U}_k^T \mathbf{X}^{\text{train}})$
```

For some choices of Train, Algorithms 1 and 2 are equivalent, i.e. they will output identical classifiers given the same inputs. One important example of this is Linear SVMs trained using ℓ_2 regularization. In this case, Algorithms 1 and 2 will clearly select linear functions with the same behavior on each of the training points. This extends by linearity to the subspace of $\mathbb{R}^n$ containing all of the projected data points.

Among the linear functions that achieve this behavior, regularization will force Algorithm 2 to select the function with the smallest weight vector in the ℓ_2 norm. This will be the linear function that is invariant on all vectors orthogonal to the subspace containing the projected data. In practice, we prefer Algorithm 1 because it operates on lower dimensional representations of the vectors and thus is more computationally efficient.

3.5 Intuition behind the defense

Now, we will give some intuition about why dimensionality reduction should improve resilience for support vector machines. We discuss the two-class case for simplicity, but the ideas generalize to the multiple class case.

The core of a linear classifier is a function $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$. Both $\mathbf{x}$ and $\mathbf{w}$ can be expressed in the principal component basis as $\mathbf{U}^T \mathbf{x}$ and $\mathbf{U}^T \mathbf{w}$. We expect many of the largest coefficients of $\mathbf{U}^T \mathbf{w}$ to correspond to the smallest eigenvalues of of $\mathbf{C} = \mathbf{U} \Lambda \mathbf{U}^T$.

The reason for this is very simple: in order for different principal components to achieve the same level of influence on the classifier output, the coefficient of $(\mathbf{U}^T \mathbf{w})_i$ must be proportional to $1/\sqrt{\lambda_i}$. To take advantage of the information in a component with small variation, the classifier must use a large coefficient. Of course, the principal components are not equally useful for classification. However, among the components that are useful, we expect a general trend of decreasing coefficients of $(\mathbf{U}^T \mathbf{w})_i$ as $\sqrt{\lambda_i}$ increases.

The top-most plot in Figure 2 validates this prediction. The first principal component[3] is by far the most useful source of classification information. Consequently it does not fit the overall trend and actually has the largest coefficient. However, among the other components the trend holds.

Furthermore, we expect higher variance components to contain better information than lower variance components. By better information, we mean components that generalize better from the training set to the test set, or equivalently components that mostly correspond to underlying features of the classes rather than spurious correlations driven by the specific training examples.

Since the optimal attack perturbation for a linear classifier is a multiple of $\mathbf{w}$, the *principal components with large coefficients are the ones that the attacker takes advantage of.* The projection defense denies this opportunity to the attacker. By removing all variation from the lower components, it causes the classifier to assign no weight to them. This significantly changes the resulting $\mathbf{w}$ that the classifier learns. The classifier loses access to some information, but accessing that information required large weight coefficients, so the attacker is hurt far more.

[3] on the top right in each plot

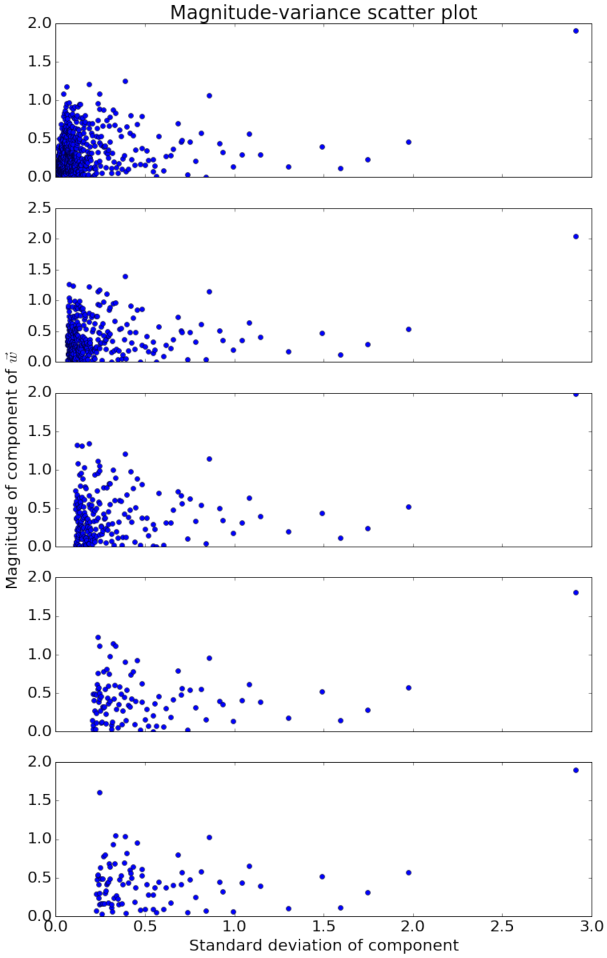


Figure 2: **The magnitudes of the coefficients of the weight vector w in the principal component basis**. On the horizontal axis, we have $\sqrt{\lambda_i}$. On the vertical axis, $|(\mathbf{U}^T\mathbf{w})_i|$. The top-most plot is from the classifier trained on the original MNIST data, with no projection (i.e. $k = 728$). The next four plots are the classifiers trained on the projected data with $k \in \{300, 100, 50, 40\}$.

3.6 Complexity Analysis of Defenses

The defense using PCA adds a one-time $\mathcal{O}(d^2n + d^3)$ overhead for finding the principal components, with the first term arising from the covariance matrix computation and the second term from the eigenvector decomposition. There is also a one-time overhead for training a new classifier on the reduced dimension data. The time needed to train the new classifier will be less than that needed for the original classifier since the dimensionality of the input data has reduced. Each subsequent input will incur a $\mathcal{O}(dk)$ overhead due to the matrix multiplication needed to project it onto the principal components.

4 Experimental setup

In this section we provide brief descriptions and implementation details of the datasets, machine learning and dimensionality reduction algorithms and metrics used in our experiments. We also discuss the reasons for the choice of the adversarial budgets for various attacks and corroborate them with visual data in the case of the MNIST dataset.

4.1 Datasets

In our evaluation, we use two datasets. The first is the MNIST image dataset and the second is the UCI Human Activity Recognition dataset. We now describe each of these in detail.

MNIST dataset: This is a dataset of images of handwritten digits [30]. There are 60,000 training examples and 10,000 test examples. Each image belongs to a single class from 0 to 9. The images have a dimension of 28×28 pixels (total of 784) and are grayscale. The digits are size-normalized and centred. This dataset is used commonly as a 'sanity-check' or first-level benchmark for state-of-the-art classifiers. We use this dataset since it has been extensively studied from the attack perspective by previous work. It is also easy to visualize the effects of our defenses on this dataset.

UCI Human Activity Recognition (HAR) using Smartphones dataset: This is a dataset of measurements obtained from a smartphone's accelerometer and gyroscope [31] while the participants holding it performed one of six activities. Of the 30 participants, 21 were chosen to provide the training data, and the remaining 9 the test data. There are 7352 training samples and 2947 test samples. Each sample has 561 features, which are various signals obtained from the accelerometer and gyroscope. The six classes of activities are Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing and Laying. We used this dataset to demonstrate that our defenses work across a variety of datasets and applications.

4.2 Machine learning algorithms

We have evaluated our defenses across a number of machine learning algorithms including linear Support Vector Machines (SVMs) and a variety of neural networks with different configurations. All experiments were run on a desktop running Ubuntu 14.04, with an 4-core Intel® Core™ i7-6700K CPU running at 4.00GHz, 24 GB of RAM and a NVIDIA® GeForce® GTX 960 GPU.

Linear SVMs: Ease of training and the interpretability of separating hyperplane weights has led to the use of Linear SVMs in a wide range of applications [37, 39]. We use the LinearSVC implementation from the Python package scikit-learn [40] for our experiments, which uses the 'one-versus-rest' method for multi-class classification by default.

In our experiments, we obtained a classification accuracy of 91.5% for the MNIST dataset and 96.7% for the HAR dataset.

Neural networks: Neural networks can be configured by changing the number of layers, the activation functions of the neurons, the number of neurons in each layer etc. We have performed our experiments on a standard neural network used in previous work, for the purpose of comparison. The network we use is a standard one from [16] which we refer to as FC100-100-10. This neural network has an input layer, followed by two hidden layers, each containing 100 neurons, and an output softmax layer containing 10 neurons.

FC100-100-10 is trained with a learning rate of 0.01 and momentum of 0.9 for 500 epochs. The size of each minibatch is 500. On the MNIST test data, we get a classification accuracy of 97.71% for FC100-100-10.

We use Theano [41], a Python library optimized for mathematical operations with multi-dimensional arrays and Lasagne [42], a deep learning library that uses Theano, for our experiments on neural networks.

4.3 Dimensionality reduction techniques

Principal Component Analysis: We use the PCA module from scikit-learn [40]. Depending on the application, either the number of components to be projected onto, or the percentage of variance to be retained can be specified. After performing PCA on the vectorized MNIST training data to retain 99% of the variance, the reduced dimension is 331, which is the first reduced dimension we use in our experiments on PCA based defenses. After performing PCA on the HAR training data to retain 99% of the variance, the reduced dimension is 155, which is the first reduced dimension we use in our experiments on PCA based defenses.

4.4 Metrics

4.4.1 Adversarial success

There is a subtlety in the definition of adversarial success in the case of attacks which just aim to cause a misclassification, as opposed to a targeted misclassification. In this case, there are 3 possible notions of adversarial success that can be reported of which we explain one here. The other, similar notions are explained in the Appendix.

For each benign input $\mathbf{x}$, we check two conditions: if after perturbation, $f(\tilde{\mathbf{x}}) = f(\mathbf{x})$ *and* if initially, $f(\mathbf{x}) = y$, where y is the true label. Thus, the adversary's attempt is successful if the original classification was correct but the new classification on the adversarial sample is incorrect. In a sense, this count represents the number of samples that are truly adversarial, since it is only the adversarial perturbation that is causing misclassification, and not an inherent difficulty for the classifier in classifying this sample. While reporting adversarial success rates, we divide this count by the total number of benign samples correctly classified after they pass through the entire robust classification pipeline.

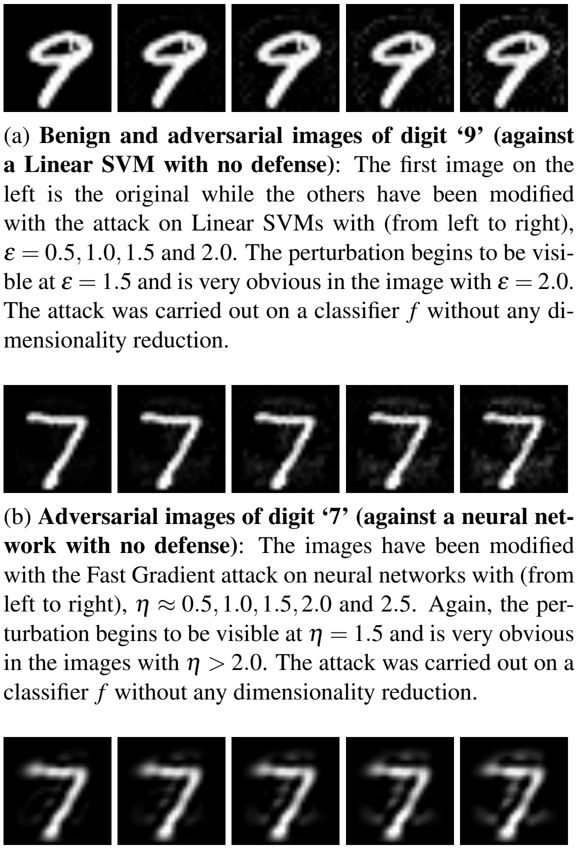
Throughout Section 5, as we evaluate the performance of our defense, we will emphasize the budget that the adversary needs to achieve a specified misclassification rate over the success rate of an attack at a particular budget. There are a couple of reasons for this. First, this allows us to somewhat sidestep the issue of precisely determining the amount of perturbation that is detectable by a human. Additionally, as will be seen in Section 5, the perturbation levels required to modify classifier outputs are often fairly concentrated across examples from the same class. That is, adversarial success rate rises rapidly at some perturbation budget. This means that success rate is somewhat sensitive to the choice of budget, but median required budget is relatively robust.

4.4.2 Perturbation magnitudes

When interpreting the concept of a "perturbation detectable by a human", it is important to distinguish between two variations. The first is a perturbation that will cause a human to misclassify the example. This provides a rough limit on the performance of any classifier: for an adversarial modification that produces an input that belongs to a different class in some "ground truth" sense (the ground truth being a human in this case), that input will not count as a success for the adversary since its ground truth label has changed.

While this is a useful limit to keep in mind, it is not what is usually meant by "detectable". The other level of perturbation is that which a human will recognize as being unlikely to occur naturally. In other words, the modified sample looks like it has been tampered with, even though the original class may be more or less recognizable. We illustrate this in Figure 3 using images from the MNIST dataset modified with a variety of attacks. These images validate the perturbation limits till which our experiments are performed, as we demonstrate the effectiveness of our defense against attacks which are well in the detectable range for humans.

When the ML system owner chooses to either operate in the reduced dimension space, the existence of adversarial perturbations may be checked (for image data) by projecting the input to the ML system back to the original full dimensional space. A key point, highlighted in Figure 3, is that the adversarial perturbations can be detected much more easily when the defense is employed. The perturbations in the image projected back to the original pixel space from the reduced dimension subspace are clearly visible, which indicates that while the ℓ_2 distances of the perturbations are equal, they are further apart perceptually. Thus, evaluating the defense at perturbation levels where the attack begins to be detected in the full dimensional case leads to conservative estimates of its effectiveness.



(a) **Benign and adversarial images of digit '9' (against a Linear SVM with no defense)**: The first image on the left is the original while the others have been modified with the attack on Linear SVMs with (from left to right), $\varepsilon = 0.5, 1.0, 1.5$ and 2.0. The perturbation begins to be visible at $\varepsilon = 1.5$ and is very obvious in the image with $\varepsilon = 2.0$. The attack was carried out on a classifier f without any dimensionality reduction.

(b) **Adversarial images of digit '7' (against a neural network with no defense)**: The images have been modified with the Fast Gradient attack on neural networks with (from left to right), $\eta \approx 0.5, 1.0, 1.5, 2.0$ and 2.5. Again, the perturbation begins to be visible at $\eta = 1.5$ and is very obvious in the images with $\eta > 2.0$. The attack was carried out on a classifier f without any dimensionality reduction.

(c) **Adversarial images of digit '7' (against a neural network with a PCA based defense with** $k = 70$**)**: The images have been modified with the Fast Gradient attack on a neural network taking in reduced dimension inputs with (from left to right), $\eta \approx 0.5, 1.0, 1.5, 2.0$ and 2.5. The reduced dimension vectors were projected back to the image space for visualization. In this case, the perturbation begins to be visible at $\eta = 0.5$ and is very obvious in the images with $\eta > 1.5$. This indicates that our *defense also makes adversarial perturbations more easily detectable*.

Figure 3: **Adversarial images generated to evade Linear SVMs and neural networks**.

5 Experimental results

In this section we present an overview of our empirical results. The main questions we seek to answer with our evaluations are:

i) Is the defense effective against strategic attacks?

ii) Is the defense able to defeat vanilla attacks?

iii) Does the defense work for different classifiers?

iv) Does the defense generalize across different datasets?

Our evaluation results confirm the effectiveness of our defense in a variety of scenarios, each of which has a different combination of datasets, machine learning algorithms, attacks and dimensionality reduction algorithms. For each set of evaluations, we vary a particular step of the classification pipeline and fix the others.

Baseline configuration: We start by considering a classification pipeline with the *MNIST dataset* as input data, a *Linear SVM* as our classification algorithm and *PCA* as the dimensionality reduction algorithm used in our defense. Since we consider the Linear SVM as our classifier, we evaluate its susceptibility to adversarial samples generated using the *Attack on Linear SVMs* described in Section 2.2. We evaluate our defenses on adversarial samples created starting from the *test set* for each dataset. Unless otherwise noted, all security and utility results are for the *complete* test set. To empirically demonstrate that our defense is resilient not only in this baseline case, but also various configurations of it, we systematically investigate its effect as each component of the pipeline, as well as the attacks, are changed.

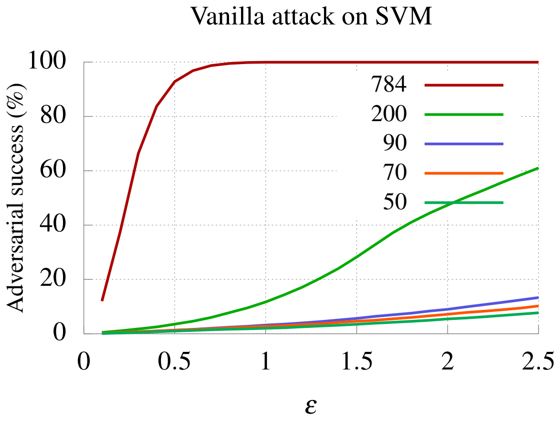


Figure 4: **Effectiveness of the defense for the MNIST dataset against vanilla attacks on Linear SVMs**. The adversarial example success on the MNIST dataset is plotted versus the perturbation magnitude $\varepsilon = \|\tilde{\mathbf{x}} - \mathbf{x}\|$. The attack is performed against the original classifier and the effect of the defense is plotted for each reduced dimension k.

5.1 Effect of defense on Support Vector Machines

In the baseline case, we begin by answering question ii), i.e. 'Are the defenses able to reduce the effectiveness of vanilla attacks?' and i), i.e. 'Are the defenses able to reduce the effectiveness of strategic attacks?' for Linear SVMs.

5.1.1 Defense against vanilla attacks

Figure 4 shows the variation in adversarial success for the defense against vanilla attacks on SVMs. The defense significantly reduces adversarial success rates. For example, at $\varepsilon = 1.0$, the defense using PCA with a reduced dimension of $k = 50$ reduces adversarial success from 99.97% to 1.85%. This is a 98.12% or around 54× decrease in the adversarial success rate. At $\varepsilon = 0.5$, where the adversarial success rate is 92.77% , the defense with $k = 50$ brings the adversarial success rate down to just 0.9%, which is a 103× (two orders of magnitude) decrease. Training with reduced dimension data leads to more robust Linear SVMs, and this can be seen in the added effectiveness of the defense.

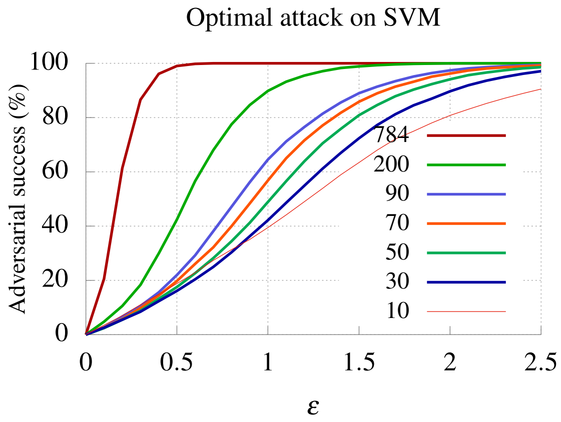


Figure 5: **Effectiveness of the defense for the MNIST dataset against optimal attacks on Linear SVMs**. The adversarial example success on the MNIST dataset is plotted versus the perturbation magnitude $\varepsilon = \|\tilde{\mathbf{x}} - \mathbf{x}\|$. The attack is performed against each reduced dimension classifier and the effect of the defense is plotted.

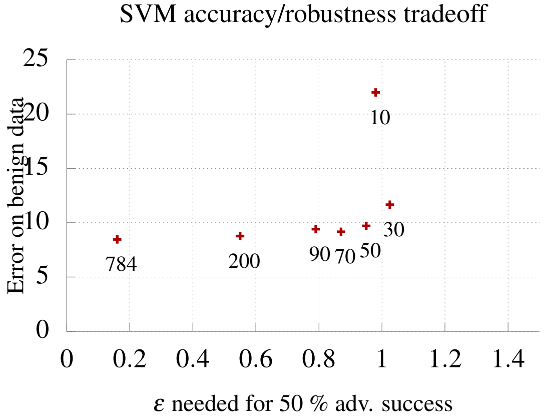


Figure 6: **Tradeoff between SVM classification performance on benign test data, and adversarial performance**. Adversarial robustness is the value of $\|\tilde{\mathbf{x}} - \mathbf{x}\|$ that allows the adversary to achieve a 50% misclassification rate.

Again, we also notice that as we decrease the reduced dimension k used in the projection step of the defense, adversarial success decreases. At $k = 331$, the adversarial success rate is 48.75% at $\varepsilon = 1.0$, which drops to 5.53% when $k = 100$. At $k = 30$, at the same ε, the adversarial success rate drops to just 2.63% with a modest decrease to 2.52% with $k = 10$.

In the case of vanilla attacks, the defense also acts like a noise removal process, removing adversarial perturbations and leaving behind the clean input data. This accounts for the added robustness we see for vanilla attacks as compared to strategic attacks.

5.1.2 Effect of defenses on optimal attacks

Figure 5 shows the variation in adversarial success for the defense against the *optimal, strategic attack* on Linear SVMs. This plot corresponds to the case where the adversary is aware of the dimensionality reduction defense and inputs a sample to the pipeline which is designed to optimally evade the reduced dimension classifier. At a perturbation magnitude of 0.5, where the classifier with no defenses has a misclassification rate of 99.04%, the reduced dimension classifier with $k = 70$ has a misclassification rate of just 19.75%, which represents a 80.25% or 5.01× decrease in the adversarial success rates. However, since 0.5 is a small adversarial budget, the perturbation will simply not be visible even when the reduced dimension image is projected back to the pixel space. At an adversarial budget of 1.3, where the perturbations begin to be clearly visible (see Section 4.4), the misclassification rate for the classifier with no defenses is 100%, while it is about 77.11% for the classifier with a reduced dimension of 70, which is almost a 23% drop in adversarial success rates. Recall that the perturbations required to evade the lower dimensional classifier are more clearly visible to the human eye, making these figures conservative.

We can also study the *effect of our defense on the adversarial budget required to achieve a certain adversarial success rate*. A budget of 0.3 is required to achieve a 86.6% misclassification without the defense, while the required budget for a classifier with a defense with $k = 70$ is 1.6, which is a 5.33× increase. The corresponding numbers to achieve a 90% misclassification rate are 0.4 without the defense and 1.7 with, which represents a 4× increase. Thus, our defense clearly reduces the effectiveness of an attack carried out by a very powerful adversary with full knowledge of the defense and the classifier as well as the ability to carry out optimal attacks.

5.1.3 Utility-security tradeoff for defense

Figure 6 shows the tradeoff between performance under ordinary and adversarial conditions. The optimal number of dimensions for this dataset is clearly between 50 and 30, where the kink in the tradeoff occurs. There is very little benefit in classification performance by using more dimensions, and essentially no benefit in robustness by using fewer. At $k = 50$, we see a drop in classification success on the test set from 91.5% without any defenses, to 90.29% with the defense. Thus, there is a modest utility loss of about 1.2% at this value of k, as compared to a security gain of 5.9×, since the perturbation needed to cause 50 % of the test set to be misclassified increases from 0.16 to 0.95.

With these results, we can conclude that our defense is effective at least in the baseline case, for both vanilla and optimal attacks against Linear SVMs. Now, we investigate the performance of our defenses on neural networks, to substantiate our claim of the applicability of our defenses across machine learning classifiers.

5.2 Effect of defense on neural networks

We now modify the baseline configuration by changing the classifier used to FC100-100-10. We evaluate our defenses on both gradient-ased attacks for FC100-100-10: the Fast Gradient (FG) and Fast Gradient Sign (FGS) attacks. We continue to use the MNIST dataset and PCA for dimensionality reduction. With these experiments, we answer question iii), i.e. ‘Do the defenses work for different classifiers?’ and further strengthen our claim that our defense is effective against strategic attacks.

5.2.1 Defense for strategic Fast Gradient attack

Figure 7 shows the variation in adversarial success for the defense as $\eta = \|\tilde{\mathbf{x}} - \mathbf{x}\|_2$, the parameter governing the strength of the strategic FG attack, increases. The defense also reduces adversarial success rates for this attack. At $\eta = 1.44$, the defense using PCA with a reduced dimension of $k = 30$ reduces adversarial success from 91.95% to 43.41%. This is a 48.54% or around 2.1× decrease in the adversarial success rate. Again, at $\eta = 2.05$, while the adversarial success rate is 98.02% without any defense, the defense with $k = 30$ brings the adversarial success rate down to 82.25%, which is a 15.77% decrease. Thus, even for neural networks, the defense causes significant reductions in adversarial success.

Again, we can study the effect of our defense on the adversarial budget required to achieve a certain misclassification percentage. A budget of 0.82 is required to achieve 60% misclassification without our defense, while the required budget for a classifier with a defense with $k = 30$ is 1.43, which is a roughly 1.6× increase. The

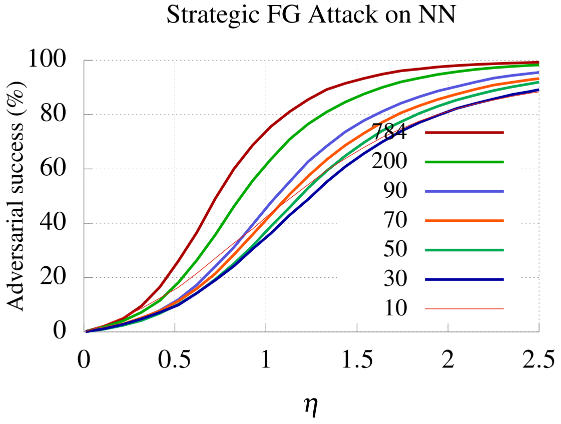


Figure 7: **Effectiveness of the defense for the MNIST dataset against strategic FG attacks on FC100-100-10**. The adversarial example success on the MNIST dataset is plotted versus the perturbation magnitude $\eta = \|\tilde{\mathbf{x}} - \mathbf{x}\|$. The attack is performed against each reduced dimension classifier and the effect of the defense is plotted.

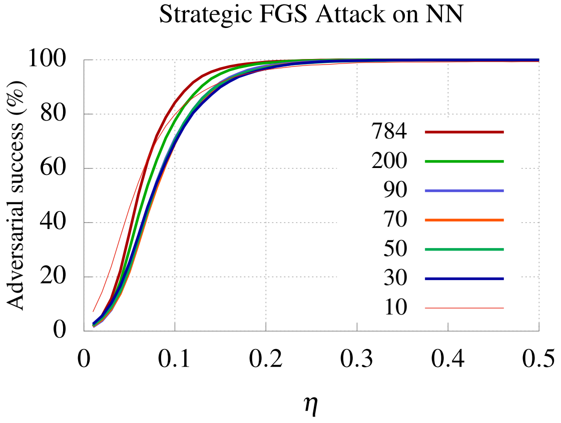


Figure 8: **Effectiveness of the defense for the MNIST dataset against strategic FGS attacks on FC100-100-10**. The adversarial example success on the MNIST dataset is plotted versus the perturbation magnitude η. The attack is performed against each subspace classifier (obtained from Algorithm 2) and the effect of the defense is plotted.

corresponding numbers to achieve a 90% misclassification rate are 1.43 without the defense and 2.45 with, which represents a roughly 2× increase.

5.2.2 Defense for strategic Fast Gradient Sign attack

The FGS attack is constrained in terms of the ℓ_∞ norm, so all features with non-zero gradient are perturbed by either η or $-\eta$. The MNIST dataset has pixel values normalized to lie in $[0,1]$. Thus, we restrict η to be less than 0.5, which is a conservative upper bound on the adversary's capability, since the perturbations begin to be clearly visible at $\eta = 0.25$, which is also the value used in previous work [19].

At $\eta = 1.0$, the adversarial success rate falls from 84.21% to 69.20% for the defense with $k = 50$ which is a 15% reduction. Further, the perturbation needed to cause 90% misclassification is about 0.12 without the defense but increases to 0.15 for the defense with $k = 50$, which is a 25% increase.

With these results for neural networks, we conclude that our defenses are effective against a variety of different attacks, in each of which the nature of the adversarial perturbation is very different. Also, these results demonstrate that our defenses can be used in conjunction with different types of classifiers, providing a general method for defending against adversarial inputs.

5.3 Applicability for different datasets

Next, we modify the baseline configuration by changing the datasets used. We show results with Linear SVMs as the classifier and PCA as the dimensionality reduction algorithm. We present results for the Human Activity Recognition dataset.

5.3.1 Defense for the HAR dataset

In Figure 9, the reduction in adversarial success due to the defense is shown. At $\varepsilon = 1.0$, the adversarial success rate drops from 99.56% without the defense to 91.75% with $k = 70$ and to 76.21% with $k = 30$. In order to achieve a misclassification rate of 90%, the amount of perturbation needed is 0.65 without the defense, which increases to 0.876 with $k = 70$ and to 1.26 with $k = 30$. Thus, the adversarial budget increases 2× to achieve the same adversarial success rate. The impact on utility is modest, with a drop of 2.3% for $k = 70$ and 5.4% for $k = 30$, which are small in comparison to the gain in security achieved.

	MNIST data			HAR data
	FC100-100-10	Linear SVM		Linear SVM
k (MNIST)			*k* (HAR)	
No D.R.	97.47	91.52	No D.R.	96.67
784	97.32	91.54	561	96.57
331	97.35	91.37	200	96.61
200	97.04	91.28	100	92.43
100	97.36	90.89	90	94.60
90	97.14	90.58	80	94.54
80	97.25	90.64	70	94.37
70	97.52	90.76	60	93.72
60	97.38	90.47	50	92.47
50	97.26	90.18	40	92.06
40	96.71	89.03	30	91.11
30	96.56	88.37	20	88.63
20	96.67	86.69	10	86.67
10	93.22	77.79		

Table 2: **Utility values for the dimensionality reduction defense**. For the MNIST and HAR datasets, the classification accuracy on the benign test set is provided for various values of reduced dimension k used for the PCA based defense, as well as the accuracy without the defense.

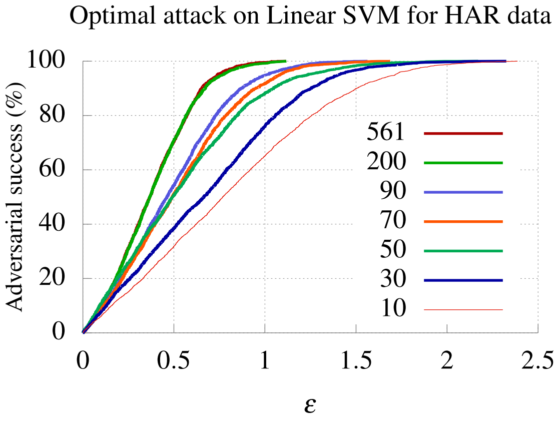


Figure 9: Adversarial example success on the HAR dataset versus perturbation magnitude ε for the Linear SVM attack (against original classifier). Plotted for each reduced dimension k used in the defense.

5.4 Effect of defense on utility

Table 2 presents the effect of our defense on the classification accuracy of benign data. The key takeaways are that the decrease in accuracy for both neural networks and Linear SVMs for reduced dimensions down to $k = 50$ is at most 4%. Further, we notice that dimensionality reduction using PCA can actually *increase* classification accuracy as when $k = 70$, the accuracy on the MNIST dataset increases from 97.47% to 97.52%. More aggressive dimensionality reduction however, can lead to steep drops in classification accuracy, which is to be expected since much of the information used for classification is lost.

These results highlight the broad applicability of our defense across application domains. It is clear that the effectiveness of our defenses is not an artifact of the particular structure of data from the MNIST dataset, and that the intuition for their effect holds across different kinds of data.

6 Discussion and limitations

6.1 Meeting the design goals

In Section 3.2, we laid out desirable goals that any defense should possess. First, the defense should maintain *high classification accuracy*. From Table 2, it is clear that for both datasets and classifiers, there is a range of reduced dimensions which have a minimal impact on classification accuracy, and in certain cases may even im-

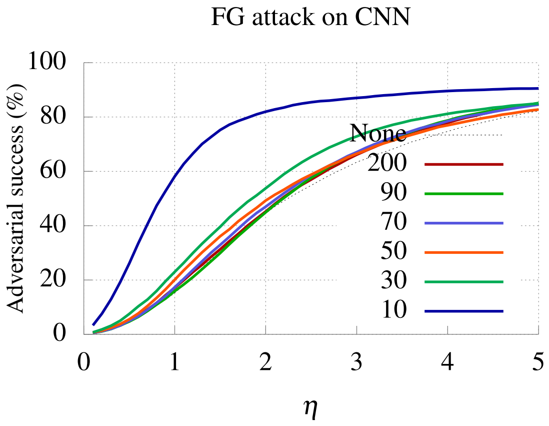


Figure 10: **Effectiveness of the defense for the MNIST dataset against strategic FG attacks on Papernot-CNN** . The adversarial example success on the MNIST dataset is plotted versus the perturbation magnitude $\eta = \|\tilde{\mathbf{x}} - \mathbf{x}\|$. The attack is performed against each subspace classifier (obtained from Algorithm 2) and the effect of the defense is plotted.

prove it. Secondly, the version of the defense based on PCA adds an overhead which is polynomial in the number of samples n and the number of dimensions d during both test and . The time and space required to train the reduced dimension classifier is at most that for the classifier in the original high-dimensional space, thus our defense maintains *high efficiency* for the training and test phases. The added *security* conferred by our defense has been illustrated for a variety of settings in the previous section. From figure 6, it is clear that changing dimensions allow the ML system owner to navigate different points in the utility-security space. Thus, our defense creates a *tunable* system.

In summary, it is clear that our defense achieves all of the design goals that may be desired of a defense against evasion attacks on ML systems. However, our defense is not effective in all plausible scenarios when a system may be under attack, which leads to the limitations we discuss below.

6.2 Limitations

Even though our defense reduces adversarial success rates in a number of cases, there are two main areas where it falls short of being a comprehensive defense mechanism against evasion attacks:

1. **Insufficient on its own**: While our defense causes significant reductions in adversarial success rates in a variety of settings, there are cases where the adversarial success rate is still too high. In such cases, it is likely our defense would have to be combined with other defenses such as adversarial training [19] and ensemble methods [43] for detection in order to create a ML system secure against evasion attacks. Our defense has the advantage that it can be used in conjunction with a variety of ML classifiers and it will not interfere with the operation of other defense mechanisms. Further, as demonstrated in Section 4, our defense leads to the creation of adversarial perturbations that have greater visual perceptibility. This may aid defenses which aim to detect adversarial perturbations.

2. **Lack of universality**: In certain settings, our defense has limited applicability. For example, in Figure 10, we see that the PCA based defense offers little to no security improvements for the Papernot-CNN (See Section 9.3 for details). It is likely that this effect stems from the fact that CNNs already incorporate domain-specific knowledge in their convolutional layers, and an additional layer of preprocessing using PCA does not confer any additional robustness. Further, PCA may reduce the amount of local information that the convolutional layers in the CNN are able to use for the purposes of classification.

A key step in addressing these limitations of our defense is to use other dimensionality reduction techniques which could reduce adversarial success to negligible levels and work better when combined with classifiers such as CNNs. In future work we plan to explore techniques for reducing dimension such as autoencoders, kernel PCA and various compression schemes to better understand the relationship between dimensionality reduction and the robustness of classifiers.

7 Related work

7.1 Adversarial machine learning

The lack of robustness of machine learning algorithms to adversarially modified data was first pointed out in [44, 45] in the context of spam classifiers. A clear taxonomy for possible attacks and adversarial knowledge was laid out in [10–12], with the main adversarial objectives being *integrity*, availability and privacy violations. The objective most relevant to the current work is the integrity violation, where the adversary attempts to cause the misclassification of data for her own benefit. Integrity violations can be broadly divided into two categories, poisoning attacks and *evasion* attacks, and this paper concentrates on combating the latter. In poisoning attacks [13, 46–48], the adversary modifies the training data before or during the training phase, in order to achieve objectives such as evasion at test time.

7.2 Evasion attacks

On the other hand, in evasion attacks, the adversary's objective is to construct samples that are misclassified by the ML system, which she does using varying degrees of knowledge about the system and only at test time. These attacks have been proposed for a variety of machine learning classifiers such as Support Vector Machines [14, 17], tree-based classifiers [17, 18] such as random forests and boosted trees and more recently for neural networks [15, 16, 19–22]. The vulnerability of applications that use machine learning, such as face detection [23], voice command recognition [24] and PDF malware detection [25] has also been demonstrated, highlighting the need for defenses.

7.3 Analyses of evasion attacks

There have been several theoretical attempts to understand the reason for vulnerabilities in machine learning systems. Nelson et al. [33] obtain bounds on the number of classification queries to a convex-inducing classifier in order to find adversarial samples. Fawzi et al. [35, 49] study the relation between random noise and adversarial perturbations in order to understand why classifiers are robust to random noise but not to adversarial perturbations.

Tanay et al. [50] provide a geometric perspective on adversarial examples, in terms of the distance between the data submanifold and the classifier boundary, for linear classifiers.

7.4 Comparison with previous defenses

Previous work on defenses against adversarial examples has largely focused on specific classifier families or application domains. Further, the existing defenses provide improved security only against existing attacks in the literature, and it is unclear if the defense mechanisms will be effective against adversaries with knowledge of their existence, i.e. strategic attacks exploiting weaknesses in the defenses. As a case in point, Papernot et al. [51] demonstrated a defense using distillation of neural networks against the Jacobian-based saliency map attack [15]. However, Carlini et al. [21] showed that a modified attack negated the effects of distillation and made the neural network vulnerable again. Now, we give an overview of the existing defenses in the literature.

7.4.1 Classifier-specific

Russu et al. [26] propose defenses for SVMs by adding various kinds of regularization. Kantchelian et al. [18] propose defenses against optimal attacks designed specifically for tree-based classifiers. Existing defenses for neural networks [52–56] make a variety of structural modifications to improve resilience to adversarial examples. These defenses do not readily generalize across classifiers and may still be vulnerable to adversarial examples in spite of the modifications, as shown by Gu and Rigazio [52].

7.4.2 Application-specific

Hendrycks and Gimpel [57] study transforming images from the RGB space to YUV space to enable better detection by humans and decrease misclassification rates. They also use whitening to make adversarial perturbations in RGB images more visible to the human eye. The effect of JPG compression on adversarial images has also been studied [58]. Their conclusion was that it has a small beneficial effect when the perturbations are small. These approaches are restricted to combating evasion attacks on image data, and by their nature, do not generalize across applications.

7.4.3 General defenses

An ensemble of classifiers was used by Smutz and Stavrou [43] to *detect* evasion attacks, by checking for disagreement between various classifiers. However, an ensemble of classifiers may still be vulnerable to adversarial examples since they generalize across classfiers. Further, Goodfellow et. al. [19] show that ensemble methods have limited effectiveness for evasion attacks against neural networks. Goodfellow et. al. [19] re-train on adversarial samples to improve the resilience of neural networks. They find that this method reduces adversarial success somewhat, but still leads to high confidence predictions on the adversarial samples. In our experiments, we find that re-training on adversarial samples has an extremely limited effect on increasing the robustness of linear SVMs, thus this defense may not be applicable across classifiers. In [59], random feature nullification is used to reduce adversarial success rates for evasion attacks on neural networks. The applicability of this idea across classifiers is not studied. [60] use adversarial feature selection to increase the robustness of SVMs. They find and retain features that decrease adversarial success rates. This defense may be generalized across other classifiers and is an interesting direction for future work.

8 Conclusion

In this paper, we considered the novel use of dimensionality reduction as a defense mechanism against evasion attacks on ML classifiers. Our defenses rely on the insight that (a) that dimensionality reduction of data can serve as noise reduction process, helping reduce the magnitude of adversarial perturbations, and (b) training classifiers on reduced dimension data leads to enhanced resilience of ML classifiers.

Using empirical evaluation on multiple real-world datasets, we demonstrated a 2x reduction in adversarial success rates across a range of attack strategies (including strategic ones), ML classifiers, and applications. Our defenses have a modest impact on the utility of the classifiers (1-2% reduction), and are computationally efficient.

Our work thus provides an attractive foundation for countering the threat of evasion attacks.

Acknowledgements

We would like to thank Chawin Sitawarin for help with experiments and useful discussions. Arjun Nitin Bhagoji is supported by NSF and DARPA. Daniel Cullina is supported by DARPA.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[3] G. V. Cormack, "Email spam filtering: A systematic review," *Foundations and Trends in Information Retrieval*, vol. 1, no. 4, pp. 335–455, 2007.

[4] D. CireşAn, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks*, vol. 32, pp. 333–338, 2012.

[5] NVIDIA, "Self driving vehicles development platform."

[6] N. Šrndić and P. Laskov, "Hidost: a static machine-learning-based detector of malicious files," *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 22, 2016.

[7] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 3422–3426.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[9] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[10] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and Artificial Intelligence*. ACM, 2011, pp. 43–58.

[11] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM, 2006, pp. 16–25.

[12] P. Laskov and M. Kloft, "A framework for quantitative security analysis of machine learning," in *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*. ACM, 2009, pp. 1–4.

[13] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 2012, pp. 1807–1814.

[14] 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 *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 387–402.

[15] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.

[16] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

[17] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[18] A. Kantchelian, J. Tygar, and A. D. Joseph, "Evasion and hardening of tree ensemble classifiers," in *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*, 2016.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[20] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[21] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy, 2017*.

[22] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 427–436.

[23] M. McCoyd and D. Wagner, "Spoofing 2d face detection: Machines see people who aren't there," *arXiv preprint arXiv:1608.02128*, 2016.

[24] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *25th USENIX Security Symposium (USENIX Security 16), Austin, TX*, 2016.

[25] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proceedings of the 2016 Network and Distributed Systems Symposium*, 2016.

[26] P. Russu, A. Demontis, B. Biggio, G. Fumera, and F. Roli, "Secure kernel machines against evasion attacks," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, ser. AISec '16. New York, NY, USA: ACM, 2016, pp. 59–69. [Online]. Available: http://doi.acm.org/10.1145/2996758.2996771

[27] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.

[28] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative review," *J Mach Learn Res*, vol. 10, pp. 66–71, 2009.

[29] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," *arXiv preprint arXiv:1511.04599*, 2015.

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

[31] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *ESANN*, 2013.

[32] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Berkay Celik, and A. Swami, "Practical black-box attacks against deep learning systems using adversarial examples," in *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*.

[33] B. Nelson, B. I. Rubinstein, L. Huang, A. D. Joseph, S.-h. Lau, S. J. Lee, S. Rao, A. Tran, and J. D. Tygar, "Near-optimal evasion of convex-inducing classifiers." in *AISTATS*, 2010, pp. 549–556.

[34] B. Biggio, I. Corona, B. Nelson, B. I. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, and F. Roli, "Security evaluation of support vector machines in adversarial environments," in *Support Vector Machines Applications*. Springer, 2014, pp. 105–153.

[35] A. Fawzi, O. Fawzi, and P. Frossard, "Analysis of classifiers' robustness to adversarial perturbations," *arXiv preprint arXiv:1502.02590*, 2015.

[36] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[37] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[39] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "Drebin: Effective and explainable detection of android malware in your pocket." in *NDSS*, 2014.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[41] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-print arXiv:1605.02688*.

[42] S. Dieleman and J. S. et.al., "Lasagne: First release." Aug. 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.27878

[43] C. Smutz and A. Stavrou, "When a tree falls: Using diversity in ensemble classifiers to identify evasion in malware detectors," in *23nd Annual Network and Distributed System Security Symposium, NDSS 2016*.

[44] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 641–647.

[45] N. Dalvi, P. Domingos, S. Sanghai, D. Verma *et al.*, "Adversarial classification," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 99–108.

[46] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. Tygar, "Antidote: understanding and defending against poisoning of anomaly detectors," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 1–14.

[47] ——, "Stealthy poisoning attacks on pca-based anomaly detectors," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 2, pp. 73–74, 2009.

[48] M. Kloft and P. Laskov, "Online anomaly detection under adversarial impact." in *AISTATS*, 2010, pp. 405–412.

[49] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard, "Robustness of classifiers: from adversarial to random noise," *arXiv preprint arXiv:1608.08967*, 2016.

[50] T. Tanay and L. Griffin, "A boundary tilting persepective on the phenomenon of adversarial examples," *arXiv preprint arXiv:1608.07690*, 2016.

[51] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE Symposium on Security and Privacy, SP 2016*, 2016, pp. 582–597.

[52] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[53] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of neural nets through robust optimization," *arXiv preprint arXiv:1511.05432*, 2015.

[54] Q. Zhao and L. D. Griffin, "Suppressing the unusual: towards robust cnns using symmetric activation functions," *arXiv preprint arXiv:1603.05145*, 2016.

[55] Y. Luo, X. Boix, G. Roig, T. Poggio, and Q. Zhao, "Foveation-based mechanisms alleviate adversarial examples," *arXiv preprint arXiv:1511.06292*, 2015.

[56] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary," *CoRR, abs/1511.03034*, 2015.

[57] D. Hendrycks and K. Gimpel, "Visible progress on adversarial images and a new saliency map," *arXiv preprint arXiv:1608.00530*, 2016.

[58] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of jpg compression on adversarial images," *arXiv preprint arXiv:1608.00853*, 2016.

[59] Q. Wang, W. Guo, K. Zhang, X. Xing, C. L. Giles, and X. Liu, "Random feature nullification for adversary resistant deep architecture," *arXiv preprint arXiv:1610.01239*, 2016.

[60] F. Zhang, P. P. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 766–777, 2016.

9 Appendix

9.1 Measuring adversarial success

Recall that we used a particular notion of adversarial success in Section 4. There are two other related notions which may be used:

- For each $\mathbf{x}$, we check if $y_{adv}(= f(\mathbf{x}_{adv})) = f(\mathbf{x})$ or not. This counts the total number of adversarial samples for which the perturbation causes a *change from the class assigned by the classifier* for the clean sample $\mathbf{x}$. It may be the case that neither the clean nor the adversarial sample are assigned the true class y, since the classifier does not have 100% accuracy on the test set (possibly on the training set as well). However, it may also be the case that adding the perturbation causes the classifier to correctly classify a previously erroneous input, i.e.

we may have the case that $f(\mathbf{x}) \neq y$, but $y_{adv} = y$. This is an unlikely but possible occurrence.

- For each $\mathbf{x}$, we check if $y_{adv} = y$ or not. This counts the total number of adversarial samples for which the class after perturbation is *not equal to the true class*. However, this number will also include those adversarial samples for which the clean starting samples were already wrongly classified. In this case, regardless of the efficacy of the attack or defense, the percentage of incorrectly classified adversarial samples cannot be less than the baseline classifier inaccuracy on the benign test samples, which represents a lower limit on the effectiveness of any attack.

It is unclear which of these 3 counts have been reported as adversarial success in previous work. We computed all 3 counts in our experiments and found they were similar.

9.2 Intuition for use of test data for defense evaluation

Data used: Evasion attacks typically involve the modification of existing samples so that they are classified incorrectly. An attack can be deemed to be effective if it leads to high misclassification rates on adversarial examples crafted from the training set. The classifier may have poor accuracy on the test set due to various reasons, and isolating the adversarial modification as the reason for misclassification may be problematic. Further, since classifiers are typically trained till they have very high accuracy on the training set, and their decision boundaries reflect the distribution of the training data, an attack involving minimal modification of the training data is a successful one.

We evaluate the defense on adversarially modified samples from the *test set*. The main reason for this is that overfitting on the training set could be a possible reason for the effectiveness of the defense on it, so an accurate evaluation of the defense should involve adversarial samples crafted from the test set. So, a defense mechanism makes a classifier more secure if the classification accuracy of the pipeline on adversarially modified samples from the test set is higher than that of the original classifier. The higher the accuracy, the more accurate the defense is.

9.3 CNNs

We also run our experiments on a Convolutional Neural Network [38] whose architecture we obtain from Papernot et al. [51]. This CNNs architecture is as follows: it has 2 convolutional layers of 32 filters each, followed by a max pooling layer, then another 2 convolutional layers of 64 filters each, followed by a max pooling layer. Finally, we have two fully connected layers with 200 neurons each, followed by a softmax output with 10 neurons (for the 10 classes in MNIST). All neurons in the hidden layers are ReLUs. We call this network Papernot-CNN. It is trained with a learning rate of 0.1 (adjusted to 0.01 for the last 10 epochs) and momentum of 0.9 for 50 epochs. The batchsize is 500 samples for MNIST and on the MNIST test data we get a classification accuracy of 98.91% with the Papernot-CNN network.