**A Game Theoretical Approach to Situating Strategies of Increasing Adversarial Robustness in Image Classification**

Aadi Shah

Pioneer Academics Research Program

Explorations in Algorithmic Game Theory

Dr. Shaddin Dughmi

September 17, 2023

**Abstract**

In recent years, deep learning has become highly effective at addressing classification problems presented in many contexts. Yet, a significant vulnerability of these deep learning classifiers is their susceptibility to adversarial attacks during which an adversary crafts perturbed data to subvert the classifier’s predictions, causing misclassification. To combat this, methods have been developed to improve adversarial robustness for deep models. In this work, a high level game theoretic approach is used to situate these methods into three overarching strategies:

1. Modification of training data such that the resultant classifier is more robust
2. Modification of test data such that the classifier is more robust classifying this new data
3. Modification of the training algorithm such that the resultant classifier is more robust

In experimentation, prominent methods from each strategy were used to train convolutional neural networks tested on clean and adversarial examples to evaluate classification accuracy and provide recommendations regarding optimal use for the strategies. True to the proposed hypothesis, experimentation showed that every overarching strategy had tradeoffs with regard to classification accuracy on clean data and adversarial examples. Strategy A increased adversarial robustness, but decreased accuracy on clean data. Strategy B moderately increased robustness against single-step perturbations, but drastically decreased accuracy on clean data and iteratively perturbed data. Strategy C increased adversarial robustness, but had only a moderate increase against single-step perturbations. Within each strategy, there was little difference in these tradeoffs between individual methods. Code can be found [here](https://github.com/ringholder/Strategies-Adversarial-Robustness).

**A Game Theoretical Approach to Situating Strategies of Increasing Adversarial Robustness in Image Classification**

In today’s world, deep learning classifiers, and in particular convolutional neural networks, are used to automate a variety of tasks across a large number of industries (Biggio & Roli, 2017; Goodfellow et al., 2018; Kurakin et al., 2016b). In many cases, this involves utilizing supervised learning algorithms to train a classifier to classify input data into categories with data collected from relevant real-world scenarios (Biggio & Roli, 2017; Goodfellow et al., 2018). However, a significant vulnerability of these classifiers is their susceptibility to adversarial attacks by malicious entities, who may deliberately alter input data in such a way that a classifier misclassifies the data as a different category than its correct classification (Biggio & Roli, 2017; Goodfellow et al., 2014; Goodfellow et al., 2018; Huang et al., 2011; Kurakin et al., 2016b; Szegedy et al., 2013). This is proven to be particularly effective in two different circumstances:

1. The malicious party crafts adversarial examples that are incorporated into a training dataset, causing a classifier trained on this dataset to learn incorrect decision-making functions (poisoning) (Biggio & Roli, 2017; Goodfellow et al., 2018), or:
2. The malicious party crafts adversarial examples with perturbed or removed features, causing a classifier using them as test data to misclassify the data (evasion) (Biggio & Roli, 2017; Goodfellow et al., 2018). In many cases, the alterations made to this test data remain imperceptible to humans.

While “poisoning” is an effective strategy that adversaries use on classifiers, especially in contexts where personal data (such as from targeted advertising) is gathered and incorporated into deep models (Biggio & Roli, 2017), the basis of this work is on strategies primarily used to subvert convolutional neural networks used for image classification. Thus, this paper focuses on Scenario 2, where the adversary does not have access to and therefore cannot modify the training data, and instead generates adversarial test data that “evades” correct classification. Note that the strategies discussed can also be applied to other classifiers, not simply convolutional neural networks or deep learning classifiers (Goodfellow et al., 2018).

The vulnerability of classifiers to adversarial examples can be highly problematic, especially in prominent fields such as cybersecurity and defense, the autonomous vehicle industry, and the medical field, where misclassifications can lead to grievous injury and/or extreme damages (Biggio & Roli, 2017; Chakraborty et al., 2021).

As a result, research into the development of machine learning classifiers far more robust to adversarial attacks—now known as the field of “Adversarial Machine Learning” (Biggio & Roli, 2017; Chakraborty et al., 2021; Goodfellow et al., 2018; Huang et al., 2011)—began, with the intention of creating better representations of adversarial examples in order to better understand how they can be used to affect machine learning models, while concurrently developing better strategies of defense against misclassification caused by adversarial attacks to implement in the training process of classifiers and make such models more robust to adversarial examples (Biggio & Roli, 2017; Goodfellow et al., 2018; Huang et al., 2011).

However, even now, a decade after research into the field of adversarial machine learning began, classifiers continue to remain readily susceptible to adversarial attacks by malicious parties (Biggio & Roli, 2017; Chakraborty et al., 2021). Many new methods to develop more robust classifiers exist, but none are effective enough to completely eliminate the threat that adversarial examples pose (Biggio & Roli, 2017). As of now, the goal of research into the defense of machine learning classifiers in the field of adversarial machine learning is to modify classifiers such that they are minimally susceptible to adversarial attacks that take advantage of this vulnerability (Biggio & Roli, 2017; Goodfellow et al., 2018).

**Game Theoretic Modeling**

**Players of the Game**

In order to model the interaction between the entities within adversarial machine learning, the use of game theory, a popular framework of modeling decision-making among multiple agents of a game at the intersection of computer science, economics, and mathematics, has been effective (Dasgupta & Collins, 2019). In order to pursue this, one must first model representations of the entities involved, the players—or participants—of the game.

One player within the game is the defender, or the creator(s) of the classifier, who must make the classifier robust to adversarial attacks created by malicious parties, while also ensuring that the model is still robust to clean data (Dasgupta & Collins, 2019; Robey et al., 2023). In doing so, the defender successfully ensures that the classifier correctly classifies data the vast majority of the time.

The other player is an adversary, or the malicious party introducing adversarial examples into test data in an attempt to cause the model to misclassify data (Dasgupta & Collins, 2019; Robey et al., 2023). This is done in order to satisfy an arbitrary purpose, such as to gain information embedded in the model during the model’s training, to extract information regarding the model’s parameters, or simply to cause the model to fail to classify data (Dasgupta & Collins, 2019).

In the vast majority of cases, researchers model the intentions of the defender and the adversary as opposite to each other, and the benefit that each of them get when they obtain their desired result as equal to each other (Dasgupta & Collins, 2019), for a reason that is discussed in the next section.

**Game Design**

In order to model the interaction between a defender and an adversary within a game theoretical framework, one can situate these players in a game, and set the rules of the game following the interaction of the players. In the adversarial machine learning context, the adversary and the learner are typically modeled as players in a two-player, non-cooperative game (Dasgupta & Collins, 2019; Gilmer et al., 2018)—which can be informally defined as a game where the two players are competing over a shared resource (Dasgupta & Collins, 2019).

A prominent model of the game is a zero-sum game, where the utility—or the quantity that each player in a game wants to increase to a maximal amount—gained by one player is equal to the utility lost by the other player (Dasgupta & Collins, 2019; Gilmer et al., 2018; Robey et al., 2023; Zuo et al., 2021). Thus, by modeling the game as zero-sum, the utility the adversary gains in evading correct classification is equal to the utility that the defender loses in failing to correctly classify test data, and vice versa.

The benefit to this approach is that this model allows for a simplistic, easily calculatable solution to the problem proposed by adversarial examples. By modeling the game as a zero-sum game, the Nash Equilibrium—a widely used technique to calculate the strategies of the players of a game based on two assumptions: that all players’ strategies can be represented by probability distributions over their sets of available actions, and all players act rationally, attempting to maximize their utility (Dasgupta & Collins, 2019)—can be calculated using the minimax theorem, which states that the game’s Nash Equilibrium is equivalent to the game’s minimax outcome (Dasgupta & Collins, 2019; Gilmer et al., 2018).

By modeling the two-player zero-sum game such that the defender and adversary respectively seek to minimize and maximize the classification error, a strategy to train classification models to become more adversarially robust appears: directly training them on adversarial examples alongside clean data—a technique known as adversarial training (Robey et al., 2023). In doing so, the trained model learns more robust decision boundaries directly from this adversarial data, thus becoming more robust.

However, this may not be an accurate enough model—in adversarial machine learning, when a defender successfully fends off or fails to avoid misclassification caused by an adversarial attack, its change in utility is not necessarily exactly the negation of the utility change for the adversary attacking, as the adversary’s goals and the defender’s goals are not exact opposites to one another (Robey et al., 2023; Zuo et al., 2021). In the context of adversarial training, the algorithms used to generate adversarial examples may not align with how a true adversary perturbs data, and thus, the measured robustness of adversarial training may not be as high in a real-world scenario (Robey et al., 2023).

An alternative approach to situate the game of adversarial machine learning is a two-player Stackelberg game, where one player (known as the leader) selects a strategy, and the other player (known as a follower), now knowing the leader’s choice of action, implements their own strategy (Zuo et al., 2021). This more closely matches adversarial machine learning, where the defender must train a classifier to address problems proposed by the adversarial examples generated by adversaries. However, unlike two-player zero-sum games, the Nash Equilibrium in two-player Stackelberg games cannot be calculated with the minimax theorem (Robey et al., 2023). This makes isolating a strategy of increasing adversarial robustness without cost to clean data classification accuracy much more difficult.

**Adversarial Strategies**

In the real-world context, adversaries are either white box or black box, referring to their knowledge of the model they are attacking (Chakraborty et al., 2021; Ren et al., 2020; Sun et al., 2018). White box adversaries generally have full access to the model’s architecture and training parameters, while black box adversaries only have access to an interface that allows them to input data into the model and gain an output. In this paper, the assumption is that all adversaries are white box, as it has been shown that in many instances, black box adversaries may be able to reconstruct a working model from a limited access interface (Shokri et al., 2016; Tramèr et al., 2016). In addition, it has also been shown that adversarial attacks designed to be misclassified on one model are often also misclassified by others (Szegedy et al., 2013), which, in many cases, can allow black box adversaries to pursue the same adversarial attack methods as white box adversaries (Chakraborty et al., 2021).

It is noted that in the real world, an adversary has two strategies in order to conduct an adversarial attack on a defender: poison training data, or evade via perturbed test data (Biggio & Roli, 2017; Goodfellow et al., 2018). Both overarching strategies are done for a purpose that may not simply involve the failure of the defender to halt an adversarial attack, but knowing that it is not computationally feasible to calculate a single utility function for the adversary (Chakraborty et al., 2021), the assumption made in this work is that the game is zero-sum and the goal of the adversary is simply to induce misclassification within a classifier. To reiterate, in this paper, the focus is on the second strategy for adversaries, involving creating adversarial data that “evades” correct classification, and comparing the classification accuracies of a variety of potential methods to address the different methods adversaries can utilize in order to pursue this second strategy.

Within the context of image classification, the most effective adversarial attacks involve the fabrication of a perturbation whose magnitude *epsilon* can be measured most commonly by the L0 norm (a measure of magnitude referring to number of features modified), L2 norm (a measure of magnitude referring to the Euclidean distance of the perturbation), or L∞ norm (a measure of magnitude referring to the maximum value of the perturbation) (Carlini & Wagner, 2016b; Ren et al., 2020). The L∞ norm is studied the most of all due to its simplicity and convenience in robust optimization (Carlini & Wagner, 2016b).

**Defender Strategies**

In the real world, defenders have three overarching strategies in order to modify a classifier and make it more robust to adversarial attacks, as follows:

1. Modify Training Data: Training data can be modified in order to generalize the resultant trained model. The most prominent approach, adversarial training, involves the insertion of adversarial examples, obtained by simulating an adversary or from nature, into training data. This aligns the model’s decision boundaries better towards adversarial examples, allowing the model to more accurately classify them (Goodfellow et al., 2018; Ruan et al., 2021; Zhao et al., 2022).
2. Modify Test Data: Test data can be modified to minimize the impact of adversarial perturbations present before invoking the trained model. The most prominent approach involves denoising test data, which is thought to “remove” or “decrease” adversarial perturbations, thus making the model robust to adversarial noise (Sahay et al., 2018; Salman et al., 2020).
3. Modify Classifier: The machine learning model can be modified either during or after training to make it more “smooth”, and hence less susceptible to “small” adversarial perturbations (Carlini & Wagner, 2016a; Carlini & Wagner, 2016b; Papernot et al., 2015). Since “small” perturbations are those that are imperceptible to humans, this is thought to better align the model with what a human would expect.

Each of these strategies has various algorithms created in order to pursue them, though only the most prominent is discussed within this work. It is important to note that each of these strategies has high robustness in various situations, and while the existence of the Nash Equilibrium guarantees the existence of an optimal strategy for robust classification (Chakraborty et al., 2021), each of the strategies mentioned are simply approximations of what such a strategy may involve. Thus, there is no single “best” strategy that always or generally results in higher classification accuracy than the others, nor does there exist a metric to determine an “optimal” strategy.

***More on Strategy A***

Strategy A involves modifying training data such that a resultant classifier is less susceptible to adversarial perturbations within test data. The most prominent method within the category of Strategy A is adversarial training, where classifiers are trained on adversarial samples, allowing such classifiers to incorporate information from perturbed features, thus helping them correctly classify adversarial data alongside clean data (Goodfellow et al., 2018; Ruan et al., 2021; Zhao et al., 2022). Alternative approaches involve making training data more “noisy” in order to generalize the model (Li et al., 2018). However, while attempts at implementing these alternative approaches have been made, the method of using adversarial data to train a model is currently the most popular, as it is thought to be the most accurate of approaches, has been tested with large-scale datasets and deep learning, and has incurred many advancements in order to prevent other vulnerabilities caused by the use of adversarial samples as training data from forming (Goodfellow et al., 2018; Ruan et al., 2021).

***More on Strategy B***

Strategy B involves modifying test data such that adversarial perturbations are less impactful on the classifier the test data is inputted into. It has been pursued via a variety of methods, including denoising test data to “remove” adversarial noise (Sahay et al., 2018; Salman et al., 2020) and transforming test data to “conceal” adversarial noise (Sahay et al., 2018), but none of these methods have displayed the robustness displayed by classifiers trained with the most prominent methods of Strategy A or Strategy C. However, methods involving denoising test data before inputting it into a machine learning classifier at the time of testing have still had a moderate level of classification accuracy (Sahay et al., 2018), as the process removes a large amount of adversarial noise.

***More on Strategy C***

Strategy C involves “smoothing” the classifier in order to generalize its classification and make it less susceptible to misclassification. Past approaches have involved attempts to modify the algorithms used to train classifiers via methods such as Defensive Distillation such that resultant classifiers created from these new algorithms contain decision-making functions that are not easily subverted by minor perturbations that are unnoticeable or not easily noticeable by humans (Carlini & Wagner, 2016a; Carlini & Wagner, 2016b; Papernot et al., 2015). While this strategy has not been pursued as much as Strategy A or Strategy B, results from methods developed thus far have been promising, with adversarial robustness meeting the level of many prominent methods within the categories of Strategy A and Strategy B (Carlini & Wagner, 2016b; Papernot et al., 2015).

**Methodology**

**Experiment Objective**

This study considers the duality between adversarial robustness and robustness in classification of clean data in prominent methods of increasing adversarial robustness in image classification tasks. In particular, the goal is to use the lens of game theory to situate the methods of increasing adversarial robustness within broader, but still unique, strategies that generalize utilized methodologies of increasing classification accuracy, and, at a high level, come to a conclusion about the utility of each strategy with regard to the circumstances the relevant data presents. In other words, this study aims to bridge the gap regarding how adversarial machine learning methods are situated, and presents an alternative procedure for comparing different methods to increase image classifier adversarial robustness involving the categorization of these methods into broad “strategies” at a high level. Then, this proposed structure for comparison is used to evaluate the classification robustness of highly prominent methods within the broad “strategies” created, with each method (and each strategy) being evaluated on the two metrics listed: adversarial robustness, and robustness in classification of clean data. This is done in order to come to a conclusion regarding the utility of the strategies used and the contexts in which some strategies may become better oriented at being used to categorize data (for image classification) than others.

The central questions pertaining to the utility to this study are briefly summarized here: how can a game theoretical approach better situate methods of increasing adversarial robustness within broader strategies and in comparison to one another? Furthermore, in what context does each strategy for increasing adversarial robustness gain more utility than the others? Is there a tradeoff between adversarial robustness and robustness in classification on clean data that must be examined between methods, and if so, how can it be investigated?

The primary hypothesis is that different methods, and, on a larger scale, different strategies, will have distinct efficacies between adversarial robustness and robustness in classification on clean data that may not be similar, as measured by classification accuracy. If such a tradeoff exists, the proposed high level game theoretical approach will be used to describe it. In terms of specific tradeoffs, hypotheses for expected results include the following: both methods of adversarial training are expected to increase adversarial robustness, but also to cause models to generalize less effectively, decreasing classification accuracy on clean data. The denoising encoders are expected to provide some degree of adversarial robustness to models, but may also slightly degrade classification accuracy on clean data. Finally, Defensive Distillation is a “classifier smoothing” method, and thus classifiers trained with the method are expected to generalize well, retaining moderate to high classification accuracy on clean data, but perhaps do not become as adversarially robust as those trained by adversarial training.

**Assessed Classifiers**

***Training Datasets***

The experiments are conducted on three different datasets commonly used as indicators of robustness in image classification. These three datasets are CIFAR-10, containing 60,000 32x32-pixel color images of real-world animals and objects, divided equally into 10 classes and pre-separated into training and test sets (Krizhevsky, 2009); CIFAR-100, containing 60,000 32x32-pixel color images of real-world animals and objects, divided equally into 100 classes and pre-separated into training and test sets (Krizhevsky, 2009); and, Fashion-MNIST containing 70,000 28x28-pixel grayscale images of articles of clothing, divided equally into 10 classes and pre-separated into training and test sets (Xiao et al., 2017).

***Adversarial Robustness Methods***

The strategies discussed previously, involving modification of training data, modification of test data, and the creation of “smoothed” classifiers, each have a variety of algorithms developed in order to pursue them. In this work, prominent methods from each category and models generated using these methods are assessed and compared to one another with regard to their classification accuracy on both clean and adversarial data. The methods utilized and their respective categories are listed, as follows:

**FGSM Adversarial Training.** This method falls in the category of modifying training data. It involves training models using both clean data and adversarial examples generated by the Fast Gradient Sign Method (FGSM). Adversarial examples created with FGSM introduce small perturbations into clean data during training to improve the model's robustness (Goodfellow et al., 2014; Ren et al., 2020). The magnitude *epsilon* of perturbation applied to the training data, as given by the L∞ norm, is 0.1.

**PGD Adversarial Training.** This method falls in the category of modifying training data. In this approach, models are trained using both clean data and adversarial examples crafted by the Projected Gradient Descent (PGD) algorithm. PGD generates more powerful adversarial examples by iteratively applying FGSM, making it a stronger adversary when large numbers of iterations are applied (Kurakin et al., 2016a; Ren et al. 2020). In the experiment, 10 iterative “steps” of perturbation are applied to the training data, each with magnitude *epsilon =* 0.01 as given by the L∞ norm.

**FGSM Adversarial Noise Reduction.** This method falls in the category of modifying test data. To assess the effectiveness of denoising-based defense strategies, an autoencoder trained on FGSM-generated adversarial examples and clean data to denoise adversarial examples is evaluated. This technique aims to reconstruct clean data from adversarial inputs containing FGSM-generated perturbations, effectively removing perturbations and enhancing model robustness by ensuring that test data contains less adversarial perturbations (Sahay et al., 2018).

**PGD Adversarial Noise Reduction.** This method falls in the category of modifying test data. To assess the effectiveness of denoising-based defense strategies, an autoencoder trained on PGD-generated adversarial examples and clean data to denoise adversarial examples is evaluated. This technique aims to reconstruct clean data from adversarial inputs containing PGD-generated perturbations, effectively removing perturbations and enhancing model robustness by ensuring that test data contains less adversarial perturbations (Sahay et al., 2018).

**Defensive Distillation.** This method falls in the category of modifying the classifier. Defensive distillation is a technique where a “student” model is trained on “soft labels” generated by a “teacher” model. The soft labels represent the output probabilities of the teacher model (Carlini & Wagner, 2016a; Carlini & Wagner, 2016b; Papernot et al., 2015). By training on these soft labels, the student model aims to learn a smoother decision boundary, which can mitigate the impact of adversarial attacks.

***Model Architecture***

In this section, the model architecture used to train each of the classifiers during experimentation is discussed. The convolutional neural network architecture, designed for its ability to effectively process and extract features from images, is used to train the classifiers for image classification tasks. Each model comprises the following layers:

**Convolutional Layers.** These layers are responsible for capturing essential patterns and features from the input images. The network uses four convolutional layers with 3x3 filters, each with kernel regularization in the form of L2 regularization to prevent overfitting and followed by a Max Pooling layer for downsampling. Dropout layers are inserted after each Max Pooling layer to further mitigate overfitting.

**Flatten Layer.** After the last Max Pooling layer, the feature maps are flattened into a one-dimensional vector to prepare the data for the fully connected layers.

**Fully Connected Layers.** Two densely connected layers follow the Flatten layer. These layers are intended to further process and transform the extracted features. Kernel regularization in the form of L2 regularization is applied to these layers to prevent overfitting, and Dropout layers are inserted after each one to further mitigate overfitting.

**Output Layer.** The final layer consists of *N* neurons, *N* corresponding to the number of labels in each of the datasets used in our image classification task. All models except those trained with distillation employ a softmax activation function to generate class probabilities. Models trained with distillation use no activation function in their output layers.

The models are compiled using the Adam optimizer with a specified learning rate (0.0003) and a categorical cross-entropy loss function, which is well-suited for multi-class classification tasks. The distillation models in particular use a softmax cross-entropy loss function (with a temperature parameter of 20), as their output layer does not use a softmax activation function. All models are trained on the same training data, or, if the model is adversarially trained, it is also trained on adversarially perturbed data generated from the same training data from the relevant dataset.

**Experimental Procedure**

Per dataset, four models are evaluated on classification accuracy: a model trained on clean data, a model adversarially trained using adversarial examples generated with the FGSM adversarial perturbation technique, a model adversarially trained using adversarial examples generated with the PGD adversarial perturbation technique, and a “student” model trained on the knowledge distilled from a “teacher” model (which, in this case, is a model trained on clean data with the only difference from the training of the unmodified classifier being that both the “student” and “teacher” models also have a temperature parameter of 20). Note that the model trained on clean data is evaluated on classification accuracy for both unmodified test data as well as denoised test data outputted by each of two autoencoders trained for adversarial noise reduction, one trained on FGSM-perturbed adversarial examples and the other trained on PGD-perturbed adversarial examples. It must also be noted that this study did not utilize optimal hyperparameters for training, and instead relied on a general model architecture with all possible parameters shared between models, so it is likely that the robustness of each method can be increased further.

Three iterations of the experiment are conducted, one per dataset in order to evaluate classifier robustness in image classification for that dataset. Each iteration of the experiment involves the evaluation of the “efficacy” of the model at image classification, measured via the classification accuracy of the model on predetermined test datasets of clean data, FGSM-perturbed adversarial examples, and PGD-perturbed adversarial examples. The procedures for each of the three parts of the experiment are as follows:

***Part 1: Clean Data***

In the first part of the experiment, the models’ classification accuracies when classifying images from a predetermined, shared dataset of clean data are evaluated. Note that each dataset used (CIFAR-10, CIFAR-100, Fashion-MNIST) is already separated into a training set and test set, and these sets are used as the training and test sets for the models, respectively.

***Part 2: FGSM-Perturbed Adversarial Examples***

In the second part of the experiment, the models’ efficacies at image classification on a predetermined, shared dataset of adversarial data are evaluated. This adversarial test data is generated by adding perturbations to the clean test data via the adversarial perturbation technique FGSM (magnitude *epsilon* = 0.1, as measured by the L∞ norm), similar to the adversarial examples generated for use in the adversarial training methods discussed previously.

***Part 3: PGD-Perturbed Adversarial Examples***

In the third part of the experiment, the models’ efficacies at image classification on a predetermined, shared dataset of adversarial data are evaluated. This adversarial test data is generated by adding perturbations to the clean test data via the adversarial perturbation technique PGD (10 iterations of perturbations of magnitude *epsilon* = 0.01, as measured by the L∞ norm), similar to the adversarial examples generated for use in the adversarial training methods discussed previously.

**Results and Discussion**

To improve result readability, because each of the three parts’ data being analyzed is a single number (classification accuracy) per model, the results for each experiment are placed into one table, rather than the results to each part of each experiment. Note that each measurement is the classification accuracy of each model on either clean data, FGSM-perturbed data, or PGD-perturbed data. The values are not percentages as-is and must be compared as a metric of frequency out of 1 (e.g., if a value in the table is 0.2500, it represents 25.00%).

**Experimental Results**

***Iteration 1: CIFAR-10 Dataset***

| Method to Increase Adversarial Robustness | Accuracy:  Clean Data | Accuracy:  FGSM-Perturbed Data | Accuracy:  PGD-Perturbed Data |
| --- | --- | --- | --- |
| Unmodified Classifier | 0.8142 | 0.0394 | 0.3434 |
| (FGSM) Adversarially Trained Classifier | 0.7367 | 0.8128 | 0.7206 |
| (PGD) Adversarially Trained Classifier | 0.7493 | 0.8175 | 0.7270 |
| (FGSM) Adversarial Noise Reduction | 0.2842 | 0.2502 | 0.2806 |
| (PGD) Adversarial Noise Reduction | 0.2810 | 0.2498 | 0.2776 |
| Defensive Distillation | 0.8053 | 0.1530 | 0.7003 |

***Iteration 2: CIFAR-100 Dataset***

| Method to Increase Adversarial Robustness | Accuracy:  Clean Data | Accuracy:  FGSM-Perturbed Data | Accuracy:  PGD-Perturbed Data |
| --- | --- | --- | --- |
| Unmodified Classifier | 0.5032 | 0.0155 | 0.1714 |
| (FGSM) Adversarially Trained Classifier | 0.4217 | 0.3984 | 0.4179 |
| (PGD) Adversarially Trained Classifier | 0.4103 | 0.3895 | 0.4068 |
| (FGSM) Adversarial Noise Reduction | 0.1012 | 0.0810 | 0.1002 |
| (PGD) Adversarial Noise Reduction | 0.1016 | 0.0818 | 0.1000 |
| Defensive Distillation | 0.4088 | 0.0714 | 0.3505 |

***Iteration 3: Fashion-MNIST Dataset***

| Method to Increase Adversarial Robustness | Accuracy:  Clean Data | Accuracy:  FGSM-Perturbed Data | Accuracy:  PGD-Perturbed Data |
| --- | --- | --- | --- |
| Unmodified Classifier | 0.9259 | 0.2201 | 0.8338 |
| (FGSM) Adversarially Trained Classifier | 0.9259 | 0.9775 | 0.9031 |
| (PGD) Adversarially Trained Classifier | 0.9240 | 0.9779 | 0.9039 |
| (FGSM) Adversarial Noise Reduction | 0.6600 | 0.5903 | 0.6528 |
| (PGD) Adversarial Noise Reduction | 0.6653 | 0.5978 | 0.6593 |
| Defensive Distillation | 0.9179 | 0.5512 | 0.8927 |

**Unprecedented Robustness to PGD-Perturbed Adversarial Examples**

There are notable variances between the expected values and the observed accuracies of the models on PGD-generated adversarial test data during experimentation. All of the classifiers in Iterations 1, 2, and 3—even unmodified classifiers trained on clean data—were surprisingly robust to PGD-perturbed test data in comparison to their robustness to FGSM-perturbed test data, with far higher classification accuracies than expected. In contrast, the classification accuracies for the FGSM-perturbed test data were far lower, especially for unmodified classifiers (e.g., 34.34% accuracy on PGD data vs. 3.94% on FGSM data in Iteration 1, 17.14% accuracy on PGD data vs. 1.55% on FGSM data in Iteration 2, 83.38% accuracy on PGD data vs. 22.01% on FGSM data in Iteration 3).

There are two likely reasons as to why this difference in classification accuracy is occurring. The first becomes clear when looking at the magnitude of perturbation added to FGSM-generated adversarial examples in comparison to PGD-generated adversarial examples. The FGSM-generated adversarial examples each contain a single perturbation of fixed magnitude 0.1 (as per the L∞ norm), while the PGD-generated adversarial samples each contain 10 iterative perturbations of fixed magnitude 0.01 (as per the L∞ norm). This setup *guarantees* that the total magnitude of the perturbation in each PGD-generated adversarial example, as given by the L∞ norm, is less than or equal to 0.1 (10 \* 0.01). Thus, it is likely that for most, if not all PGD-generated adversarial examples, the magnitude of the perturbation added to the base image was less than or equal to the magnitude of the perturbation added to FGSM-generated adversarial examples (as given by the L∞ norm), making the PGD adversarial examples less likely than the FGSM adversarial examples to induce misclassifications in classifiers.

The second cause for the difference in classification accuracy between FGSM-generated adversarial examples and PGD-generated adversarial examples involves the datasets used. The CIFAR-10 and CIFAR-100 datasets contain images of pixel dimensions 32x32, while the Fashion-MNIST dataset contains images of pixel dimensions 28x28. Naturally, when the training dataset is as low resolution as the datasets used in this experimentation, the FGSM-generated adversarial examples, each containing a single large perturbation, have much higher visibility (and likelihood of misclassification) than the PGD-generated adversarial samples, which contain multiple small iterative perturbations.

***High Accuracy of PGD Adversarial Training***

Interestingly, despite the PGD-generated adversarial perturbations’ likelihood of having far smaller L∞ magnitude than the FGSM-generated adversarial perturbations, the classifiers adversarially trained on PGD-generated adversarial examples had very similar adversarial robustness compared to the classifiers trained on FGSM-generated adversarial examples, for all three forms of test data (clean data, FGSM-generated adversarial examples, and PGD-generated adversarial examples) and in all Iterations. Unlike the autoencoders, the adversarially trained models did not appear to become dramatically less accurate when classifying clean data, but still had small decreases in classification accuracy, which were about the same between models trained on PGD-perturbed data and FGSM-perturbed data. This indicates that adversarial training shifted classifier decision boundaries in such a way that they were better able to generalize over adversarial examples, making them adversarially robust but less adept at classifying clean data.

**Evaluating the Situated Strategies**

***Strategy A: Modifying Training Data***

From a cursory view, it is clear that on a broad scale, the strategy of modifying training data—in this case, via the technique of adversarial training—vastly increases adversarial robustness, to a degree unparalleled by any other method tested during experimentation. At the same time, it fails to generalize as well as another method tested, Defensive Distillation. While adversarial training maintains much of the classification accuracy on clean data, it does not do so to the same degree as the original classifier. Even this small dip in classification accuracy indicates that the adversarial examples caused the models to overfit to features present in adversarial examples, warping the decision boundaries of the model in an unwanted direction, only increasing the adversarial robustness of the model rather than the overall robustness in classification and causing clean data classification accuracy to fall. As such, the strategy of adversarial training is not optimal when the keeping the classification accuracy on clean data high is paramount. However, when the most important condition of a classifier is that the adversarial examples it takes as input need to be classified correctly, adversarial training, and more broadly, the strategy of modifying data before training a model, is the most suited of those discussed.

***Strategy B: Modifying Test Data***

From the experimental results, it can be safely assumed that though the strategy of modifying test data (in this case, denoising test data) has potential to further increase adversarial robustness of models, it poorly generalizes other aspects of the features that must be recognized. Surprisingly, all autoencoders in the experimentation, even those trained on PGD-generated adversarial data, increased robustness on FGSM-generated data and decreased robustness on PGD-generated data. A possible explanation is that while the autoencoders were likely able to identify and denoise (to some degree) the large magnitude of perturbation present in FGSM-perturbed data, they were likely unable to identify all of the smaller-magnitude perturbations in PGD-perturbed data, only some of them, which is why they failed to increase adversarial robustness for PGD-perturbed data. Moreover, given that the robustness on both clean data and PGD-perturbed data decreased rather than staying the same, it is likely that the autoencoders denoised relevant features, not simply adversarial noise. Though the strategy of modifying input data (in this case, via autoencoders trained to denoise test data to “remove adversarial noise”) do moderately benefit classification of single-step (FGSM, not PGD) adversarial data, they also fail to generalize classification robustness to the degree necessary to reliably and correctly denoise input data.

***Strategy C: Modifying Classifier***

The final strategy that was evaluated during the experiment was modification of classifiers via “smoothing”—in this experiment, the technique of Defensive Distillation. Defensive Distillation, as seen in the experimentation, generalized far better than most other techniques. It retained the unmodified model’s classification accuracy on clean data and had high adversarial robustness to PGD-generated adversarial data. However, although it did moderately increase robustness to FGSM-generated adversarial data, it did not do so to the extent of adversarial training, or even as much as the denoising strategy. This suggests that Defensive Distillation can provide high adversarial robustness to classifiers that must regularly classify data with minute perturbations in features, but will not be as robust for adversarial examples with large perturbations.

**Conclusion**

In this study, a comprehensive analysis regarding various methods aimed at enhancing the adversarial robustness of image classifiers was conducted. The experiments covered three iterations, each run on different low-resolution image datasets to test classification accuracy on clean, single-step perturbed, and iteratively perturbed data. Using the high level game theoretical approach discussed previously, each method to increase adversarial robustness was situated in the context of one of three overarching strategies: modifying training data to increase adversarial robustness (in this case, via adversarial training), modifying test data to decrease adversarial noise in data (in this case, via denoising with the use of autoencoders), or modifying the algorithm used to train the classifier (in this case, via classifier “smoothing”).

The experimentation revealed that modifying training data showed unparalleled effectiveness in enhancing adversarial robustness, but at the cost of reduced generalization to clean data. In contrast, modifying test data, specifically through autoencoders trained to denoise test data, demonstrated moderate benefits for single-step adversarial data but lacked robustness in generalizing to different types of adversarial examples and even decreased accuracy when classifying clean data. Finally, the modification of classifiers using classifier “smoothing” exhibited strong adversarial robustness, particularly against iteratively-perturbed adversarial examples, while maintaining high classification accuracy on clean data.

The choice of strategy for increasing classification robustness should be guided by the requirements of the application of such a classifier. In the context of image classification, applications are quite diverse, with classifiers used in a variety of fields, so this choice becomes exceptionally significant. Adversarial training is ideal when the primary goal is to correctly classify adversarial data, even at the expense of some accuracy on clean data. Classifier smoothing, contrarily, maintains high accuracy on clean data and generalizes well in terms of adversarial robustness (though not to the degree of adversarial training), but remains vulnerable to large-magnitude single-step perturbations, making it ideal for circumstances requiring high-accuracy clean data classification where the risk of highly perturbed data entering the test dataset is unlikely. Denoising strategies, at the moment, do not provide the same level of adversarial robustness as adversarial training and also decrease robustness in clean data classification, so they may not yet be suited for application. Further study must be done on improving generalization and the capability of denoisers to preserve clean data. Other studies to further classification robustness could experiment with more methods within each overarching strategy, studying the orthogonality of each technique in relation to one another and pursuing a more equal balance of high adversarial robustness and clean data classification accuracy that could be gained using multiple techniques at once, if they are truly orthogonal.

**References**

Biggio, B., & Roli, F. (2017). Wild patterns: Ten years after the rise of adversarial machine learning. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security.*

Carlini, N., & Wagner, D. A. (2016a). Defensive distillation is not robust to adversarial examples. *ArXiv*, *abs/1607.04311*.

Carlini, N., & Wagner, D. A. (2016b). Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., & Mukhopadhyay, D. (2021). A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, *6*(1), 25–45.

Dasgupta, P., & Collins, J. B. (2019). A survey of game theoretic approaches for adversarial machine learning in cybersecurity tasks. *AI Mag.*, *40*, 31–43.

Gilmer, J., Adams, R. P., Goodfellow, I. J., Andersen, D. G., & Dahl, G. E. (2018). Motivating the rules of the game for adversarial example research. *ArXiv*, *abs/1807.06732*.

Goodfellow, I. J., Mcdaniel, P., & Papernot, N. (2018). Making machine learning robust against adversarial inputs. *Communications of the ACM*, *61*, 56–66.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *CoRR*, *abs/1412.6572*.

Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I. P., & Tygar, J. D. (2011). Adversarial machine learning. *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, 43–58.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

Kurakin, A., Goodfellow, I. J., & Bengio, S. (2016a). Adversarial examples in the physical world. *ArXiv*, *abs/1607.02533*.

Kurakin, A., Goodfellow, I. J., & Bengio, S. (2016b). Adversarial machine learning at scale. *ArXiv*, *abs/1611.01236*.

Li, B., Chen, C., Wang, W., & Carin, L. (2018). Certified adversarial robustness with additive noise. *Neural Information Processing Systems*.

Papernot, N., Mcdaniel, P., Wu, X., Jha, S., & Swami, A. (2015). Distillation as a defense to adversarial perturbations against deep neural networks. *2016 IEEE Symposium on Security and Privacy (SP)*, 582–597.

Ren, K., Zheng, T., Qin, Z., & Liu, X. (2020). Adversarial attacks and defenses in deep learning. *Engineering*.

Robey, A., Latorre, F., Pappas, G., Hassani, H., & Cevher, V. (2023). Adversarial training should be cast as a non-zero-sum game. *ArXiv*, *abs/2306.11035*.

Ruan, W., Yi, X., & Huang, X. (2021). Adversarial robustness of deep learning: Theory, algorithms, and applications. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*.

Sahay, R., Mahfuz, R., & Gamal, A. E. (2018). Combatting adversarial attacks through denoising and dimensionality reduction: A cascaded autoencoder approach. *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, 1–6.

Salman, H., Sun, M., Yang, G., Kapoor, A., & Kolter, J. Z. (2020). Denoised smoothing: A provable defense for pretrained classifiers. *arXiv: Learning*.

Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2016). Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18.

Sun, L., Tan, M., & Zhou, Z. (2018). A survey of practical adversarial example attacks. *Cybersecurity*, *1*, 1–9.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., & Fergus, R. (2013). Intriguing properties of neural networks. *CoRR*, *abs/1312.6199*.

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., & Ristenpart, T. (2016). Stealing machine learning models via prediction apis. *Proceedings of the 25th USENIX Conference on Security Symposium*, 601–618.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *ArXiv*, *abs/1708.07747*.

Zhao, W., Alwidian, S. A., & Mahmoud, Q. H. (2022). Adversarial training methods for deep learning: A systematic review. *Algorithms*, *15*, 283.

Zuo, S., Liang, C., Jiang, H., Liu, X., He, P., Gao, J., Chen, W., & Zhao, T. (2021). Adversarial regularization as stackelberg game: An unrolled optimization approach. *Conference on Empirical Methods in Natural Language Processing*.