SANDIA REPORT
SAND2019-11748
Printed September 2019

Sandia National Laboratories

Defending Against Adversarial Examples

Austin Short, Trevor La Pay, Apurva Gandhi

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods/

ABSTRACT

Adversarial machine learning is an active field of research that seeks to investigate the security of machine learning methods against cyber-attacks. An important branch of this field is *adversarial examples,* which seek to trick machine learning models into misclassifying inputs by maliciously tampering with input data. As a result of the pervasiveness of machine learning models in diverse areas such as computer vision, health care, and national security, this vulnerability is a rapidly growing threat. With the increasing use of AI solutions, threats against AI must be considered before deploying systems in a contested space. Adversarial machine learning is a problem strongly tied to software security, and just like other more common software vulnerabilities, it exploits a weakness in software, like components of machine learning models. During this project, we attempted to survey and replicate several adversarial machine learning techniques with the goal of developing capabilities for Sandia to advise and defend against these threats. To accomplish this, we scanned state of the art research for robust defenses against adversarial examples and applied them to a machine learning problem.

We surveyed 12 peer-reviewed papers on adversarial machine learning, analyzed the results, and applied the most effective attacks and defense sets on our own machine learning testbed. We trained several neural networks against common image recognition problems to a high level of accuracy, and then attempted to degrade their accuracy by passing them data that was perturbed by an attack. We measured attack efficacy by how much it degraded the accuracy of a finely tuned model, and we measured defense efficacy by how well it prevented accuracy degradation using attack data. We also discuss the value of applying software security techniques to machine learning development efforts, such as threat modeling, to understand potential attacks against models in development, and properly address them.

We were not able to find any reliable, robust defense to a wide range of adversarial example attacks on machine learning. Our recommendation is to mitigate such attacks by not deploying machine learning solutions in contested or adversarial spaces; if the model must be deployed to a contested space, adversarial tampering should be added as a valid misuse case, and should be armored using adversarial training to hamper attackers. We also advocate for the use of these techniques demonstrated here to test a model against this sort of attacks, to determine the model's exposure. It is also important to apply general cyber security techniques in development of a machine learning pipeline to make tampering with input data more difficult.

ACKNOWLEDGEMENTS

We would like to thank Kristin Adair for spurring this project, and the guidance and direction she provided throughout the course of the project. We would also like to thank the 10774 organization for supporting the Synapse server at Sandia, which made this work possible.

CONTENTS

1. Adversarial Examples 7
   1.1. Attack Characteristics 7
      1.1.1. Black Box vs White Box 8
      1.1.2. Targeted vs Untargeted Attacks 8
   1.2. Fast Gradient Sign Method 8
   1.3. Basic Iterative Method 10
   1.4. Carlini Wagner Method 10
   1.5. Attack Results 11
2. The Defense 12
   2.1. Defense Characteristics 12
   2.2. Adversarial Training 12
   2.3. Ensembles of Neural Networks 13
   2.4. The GAN Defense 14
      2.4.1. GAN Background 14
      2.4.2. Conditional GANs 15
      2.4.3. GANs as a Defense Mechanism 15
   2.5. Risk Assessment 16
3. Conclusion 18
   3.1. Summary of Results 18
   3.2. Future Work 18
   3.3. Recommendation 18

LIST OF FIGURES

Figure 1-1. Rule for updating the weights to minimize the error function 9
Figure 1-2. FGSM algorithm to perturb input data 9
Figure 1-3. Example implementation of FGSM on CIFAR-10 data 10
Figure 1-4. BIM algorithm 10
Figure 2-1. Ensemble Defense against FGSM 13
Figure 2-2. Ensemble Defense against C&W 13
Figure 2-3. Example APE-GAN Filter 15
Figure 2-4. A typical machine learning input space 16

LIST OF TABLES

Table 1-1. Attack Results 11

ACRONYMS AND DEFINITIONS

Abbreviation	Definition
ML	Machine Learning
AI	Artificial Intelligence
NN	Neural Network
FGSM	Fast Gradient Sign Method
BIM	Basic Iterative Method
C&W	Carlini and Wagner
MNIST	Public dataset of handwritten images
CIFAR-10	Public dataset, 10 classes of simple images
GAN	Generative Adversarial Network
SR-GAN	Super Resolution Generative Adversarial Network

1. ADVERSARIAL EXAMPLES

There are many different types of attacks on machine learning. Some have been studied at Sandia as well, such as tainting training data, known as a poisoning attack, and malicious creation of a model, known as a back-door attack [1]. The Counter Adversarial Data Analytics LDRD [7] also focused on these issues, examining the damage a malicious insider could do with access to training data. For our purposes, we decided to study an evasion attack, involving creating adversarial examples, which are intentionally perturbed data points, passed to a model to create an incorrect classification. The idea is to perturb the data in such a small magnitude that it is imperceptible to a human, but the model sees it differently and misclassifies the data.

We decided to study this type of attack because of the widespread ease of attack vector and applicability to cyber security. Attackers wishing to employ such an attack would not need insider access to your model or training data; instead, they only need to be able to pass inputs (black-box access) to your system to fool it to gain access or bypass the system in some way. Examples of systems that currently use machine learning for cyber security solutions include network intrusion detection systems, phishing filters, biometric authentication systems [12], malware scanning [16], and image recognition software [3, 4, 6, 8, 9]. All these systems could be vulnerable to adversarial examples attacks and are critical parts of software infrastructure.

Active research is ongoing in all these fields, but the research particularly active in the image recognition field. We decided to apply our effort to image recognition problems, because of the availability of research to draw from, and it is easy to visualize how the attacks affect a given data point.

There are several different techniques to perpetrate adversarial attacks. These attacks are designed for effectiveness against Neural Network (NN) models and are specifically designed for image recognition problems.

1.1. Attack Characteristics

From the perspective of an adversary, an attack is desirable if it is efficient and effective.

- Efficiency: An attack is efficient if it requires minimal computational resources and time to complete.
- Effectiveness: An attack to be effective if it is both successful in lowering the metric of success (generally, accuracy) and the attacked data is practically indistinguishable from the original data.

For our case, we only measure effectiveness of the attacks. To measure effectiveness, we simply determine the accuracy of the attack data compared to the baseline data. The attack process can be somewhat subjective however, because the input perturbation varies, and the idea is for it to be indistinguishable to the human eye, while still causing a misclassification. For our purposes here, we simply reused attack parameters that were implemented in the papers, and visually inspected attack data to ensure that the data points did not create obvious deviations.

Several different algorithms exist to produce adversarial examples, including Fast Gradient Sign method (FGSM), Single Step Least-Likely Class Method (Step-LL), and Iterative Attack (Iter-LL). Both FGSM and Step-LL are single step attacks, meaning they require only one gradient calculation, whereas Iter-LL is an iterative approach that computes multiple gradient updates while generating adversarial examples. The key difference between Iterative and Single Step attacks is that iterative attacks produce higher error rates than single step attacks, but transfer more poorly [17]. These algorithms attempt to exploit gradient descent during backpropagation to misclassify a given input.

1.1.1. Black Box vs White Box

Attacks are considered "white box" attacks when they are informed by intimate details of the system under attack. In our case, the intimate details are the model architecture and parameters of the NN, which include parameter weights. In testing, it is common practice to test our attacks in a white box manner, even though in the real world, it is likely that this type of attack would be enacted from a black box perspective. This not only helps us test defense robustness, but also helps us manage our assumptions. Recent research has shown that even if details of the NN are not provided, often times they can be estimated and attacked effectively through reverse engineering practices [9]. Furthermore, attacks built for models other than NNs, such as Support Vector Machines, Decision Trees, and Logistic Regression, often work effectively on NNs as well. Thus, this attack can be effective from a black box perspective, which makes it more dangerous, and requires less access than other more sophisticated attacks.

1.1.2. Targeted vs Untargeted Attacks

As an attacker, the goal is to make a small perturbation in the input data to force a misclassification on the part of the model.

However, if desired, these attacks can be formulated in a targeted manner, meaning that the input data can be perturbed to resemble a specific class rather than an arbitrary one. For example, researchers at the University of California Berkeley were able to 3D print a turtle that was recognized as a rifle by image recognition software [8]. Untargeted attacks on machine learning are indiscriminate. Unlike the turtle example, untargeted attacks aim to perturb the input data to achieve a misclassification with *any* class rather than one specific class.

All the attacks described in this paper can be calibrated for either targeted or untargeted attacks. Note that when performing a targeted attack, additional perturbations may be needed because untargeted attacks usually perturb data towards the next "nearest" class in the data manifold. For our purposes, all the attacks we implemented were based on untargeted attacks.

1.2. Fast Gradient Sign Method

The simplest attack is the Fast Gradient Sign Method [4]. To understand this method, some background on how neural networks are trained will be helpful.

Most commonly, neural networks in a supervised setting are trained by finding some local (and hopefully global) minimum of an error function. The error function defines how much the predictions of our neural network deviate from their true values. By finding the minimum, the weights that minimize the error function will find an optimal fit that allows our neural network to

perform predictions with least error. Due to advances in hardware, this minimum is almost universally found using an iterative optimization technique. The simplest yet most common technique is called gradient descent. The gradient of a function with respect to some parameters can be thought of as a generalization of the slope of the function with respect to those parameters. The idea, then, is to move the weights in the opposite direction of the slope (or negative direction of the gradient) to lower the error function. This is done iteratively until a minimum is found or for a predetermined number of iterations. The update rule for the weights is summarized in the algorithm below, where ∇ is the symbol for the gradient, L is the error/loss function, and m is the number of data points that the gradient is averaged across.

$$\text{Repeat Until Convergence } \{$$

$$\omega \leftarrow \omega - \alpha * \nabla_w \sum_{1}^{m} L_m(w)$$

$$\}$$

Figure 1-1. Rule for updating the weights to minimize the error function

The concept behind FGSM is that to fool the network, we simply need to increase the error of the network with respect to the input. This is a very similar to the training algorithm above, although FGSM aims to *increase* the error function with respect to the input (and importantly not with respect to the weights). Since FGSM moves the error function in the direction of the gradient, this is called gradient ascent of the error function with respect to the inputs. Furthermore, since the direction of the gradient gives us sufficient information to increase the error, the FGSM update rule for the input only considers the sign of the gradient and takes a step of a small predetermined size, allowing for faster computation and better memory efficiency. Unlike the case of training, FGSM is done in one step rather than iteratively. This is because while the attack wants to fool the network, it does not want to perturb the input too much so that it becomes easily distinguishable from the original input. Below is the update rule for FGSM:

$$x^{adv} = x + \varepsilon \cdot \mathtt{sign}\left(\nabla_x J(x, y_{true})\right),$$

where

x is the input (clean) image,

x^{adv} is the perturbed adversarial image,

J is the classification loss function,

y_{true} is true label for the input x.

Figure 1-2. FGSM algorithm to perturb input data

The figure below shows an example of our implementation of the FGSM attack. To the left is an image of a horse from the CIFAR-10 data set. In the first row of the image below, A neural network trained on the CIFAR-10 data set classified this image as a horse with 86.66% probability. To the right is the same image after being attacked using the FGSM attack. Although the images look almost identical to human eyes, the same neural network now classified the adversarial image as a

bird with 100% accuracy. Column (a) in the figure below shows correctly classified input data, while column (b) shows perturbed data, that causes a misclassification by the model.

Figure 1-3. Example implementation of FGSM on CIFAR-10 data

1.3. Basic Iterative Method

The Basic Iterative Method (BIM) is a natural extension of the FGSM method which performs a fixed number of iterative FGSM updates while adding the constraint that the input does not deviate from the original too much. The update rule is summarized below where the clipping is done to satisfy the new constraint. The number of iterations that the update is applied is a parameter chosen by the adversary.

$$x_0^{adv} = x, \quad x_{N+1}^{adv} = Clip_{x,\varepsilon}\{ x_N^{adv} + \alpha.sign(\bigtriangledown_x J(x_N^{adv}, y_{true}))\}$$

where

x – Clean Input Image
x_i^{adv} – Adversarial Image at i^{th} step
J – Loss Function
y_{true} – Model Output for x
ε – Tunable Parameter
α – Step Size

Figure 1-4. BIM algorithm

1.4. Carlini Wagner Method

The Carlini Wagner (C&W) method of generating adversarial examples [4] is considered the state-of-the-art attack for generating adversarial examples, and has been used as an attack for evaluating robustness against these types of attacks [3]. At this point in our research, **we have not discovered**

a single defense for machine learning models that is able to robustly defend against the C&W method. In their paper evaluating their attack, Carlini and Wagner establish a methodology for evaluating robustness of defenses, involving demonstrating their L_2 attack, and showing that their instantiation of the attack is successful on an unsecured model, and then is successful on a secured model. Using this criteria for robustness, we were **not able to find sufficient, robust defenses against this attack**, either in research or implementation.

The L_2 attack is the most commonly referenced attack from C&W. If there is some valid input x, some model that makes a prediction on the input $C(x)$, and some adversarial example corresponding to that input, x', the attack is successful if $C(x) != C(x')$, and x and x' are close. in the L_2 attack, the Euclidean distance, or L_2 norm, is used to determine the distance between the two inputs. For the untargeted attack, the attack is performed against all classes for x', and the smallest L_2 norm between x' and x is then selected as the adversarial example. In other words, the attack attempts to minimize the distance between a valid and perturbed image, while still causing the perturbed image to be misclassified by the model.

The function for the L_2 attack is as follows, where w will be the magnitude of perturbation applied to the input image, x is the original input, and c is some small constant chosen using binary search, based on the mean distance of the adversarial examples generated.

$$\text{minimize} \quad \|\frac{1}{2}(\tanh(w)+1) - x\|_2^2 + c \cdot f(\frac{1}{2}(\tanh(w)+1)$$

where f is an loss function specially designed for this attack.

We implemented this attack on both MNIST and the CIFAR-10 datasets, using the cleverhans library [10].

1.5. Attack Results

We implemented in code the FGSM and BIM attacks found in [4], and called the Carlini and Wagner L_2 attack using the cleverhans library [10]. We tested the attacks on both CIFAR-10 dataset. We also tested attacks agains the MNIST dataset for more complete coverage and had similar results with different parameters. For this paper, we will cover our experiments run using CIFAR-10.

Table 1-1. Attack Results

Baseline Accuracy	FGSM	BIM	C &W
91.91%	14.98%	46.80%	10.60%

Results for CIFAR-10 data (FGSM e=4, i=1)

2. THE DEFENSE

2.1. Defense Characteristics

We will incorporate three different types of threat models into our discussion of adversarial example defenses: Zero Knowledge, Perfect Knowledge, and Limited Knowledge.

A *Zero Knowledge* adversary has no information about the model being attacked or whether the model has a particular defense in place. A *Perfect Knowledge* adversary has total awareness of any defense schemes and the defense model parameters in place and may thus prepare for them in advance. A *Limited Knowledge* adversary is aware that the model is being defended by a particular defense type but does not have any information regarding the inner workings of the defense model or parameters.

2.2. Adversarial Training

Models can be successfully defended in certain contexts using adversarial training; this defense method augments a supervised model's training data to include adversarial examples, allowing the models to better classify them. This approach seeks to minimize the risk posed by adversarial examples by training on both clean and adversarial data.

Adversarial training may be implemented using several different approaches. The most straightforward approach involves simply incorporating adversarial examples along with their "correct" labels, reinforcing the model against specific attacks and hopefully retaining model accuracy. Alternatively, a model attempting to classify N different labels may incorporate label N + 1 and group all adversarial examples into this category [19]. Another tested method involves constructing a binary classifier that attempts to partition adversarial examples and regular examples into different sets [18].

In a *Zero Knowledge* attack scenario using the MNIST training set with single-step adversarial examples, models trained on adversarial data using single step attacks have expressed some resiliency toward subsequent single step attacks [4] ; moreover, single-step attack examples that are created using models that have trained against them are not only less effective against the pre-trained model, but are less effective even against untrained models. This is a result of "reward hacking" behavior that occurs when an agent's maximization of its objective function is at odds with unintended model behavior [17]. This defense is much less robust against the CIFAR training set, suggesting a limited set of use cases, as [4] found that achieving a 70% detection rate required a 40% false positive rate. Against both Limited and Perfect knowledge attacks, these defenses are rendered useless on both MNIST and CIFAR-based training sets with distortion between 5% and 10% and an attack success rate between 98% and 100% Moreover, these defenses have proven ineffective against computationally expensive iterative attacks [4]. This suggests that adversarial training using these approaches is only useful during a zero knowledge attack, which means that these tactics only provide model armoring against unskilled or unmotivated adversaries.

One surprising feature of adversarial training is its ability to make the original model perform better at classification in general by normalizing data and preventing overfitting.

2.3. Ensembles of Neural Networks

The attacks presented in 1.2 and 1.3 are gradient based attacks, meaning they compute the expected gradient of the image data and move the image in the reverse direction of that gradient. One defense presented in [11] and another in [5] suggests creating ensembles of NN's to create encourage diversity in classifiers. We were able to replicate the results of this paper and had success defending against the attacks in 1.2 and 1.3. To do this, we trained 10 classifier with the same network architecture, but with different random initial input, creating a diverse set of 10 neural networks, with differentiating weights at each node. To attack the ensemble, we generated attack data on the average of the 10 gradients, and then passed the attack data to the model. The model then selects a class using a voting system from each of the ensembles, choosing the class that maximizes the average of the output probabilities from each model in the ensemble.

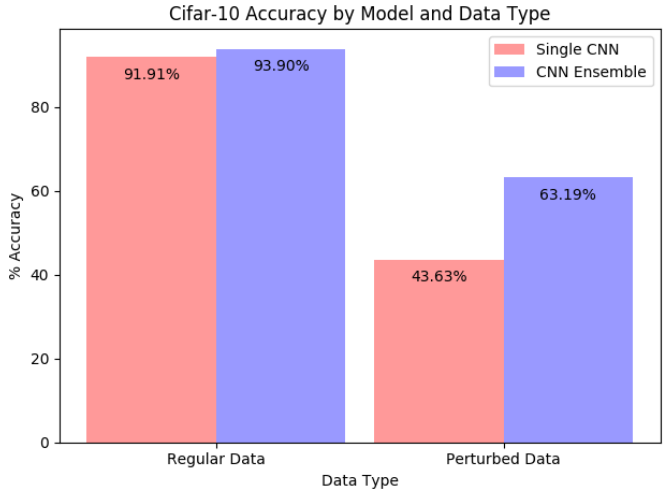

Figure 2-1. Ensemble Defense against FGSM

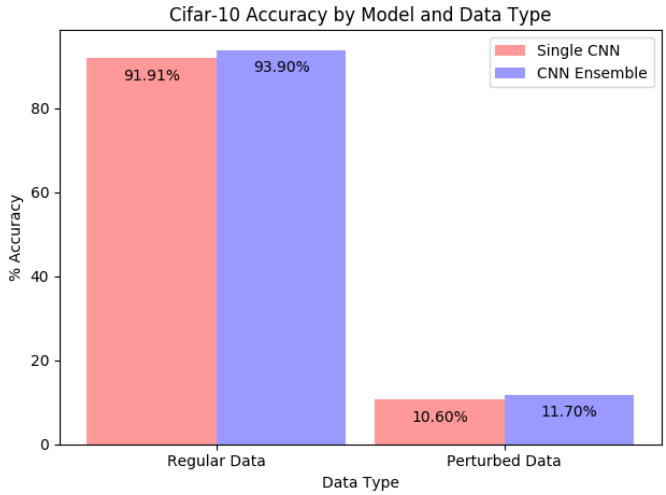

Figure 2-2. Ensemble Defense against C&W

We were able to build a somewhat successful defense against FGSM attack, boosting accuracy on perturbed data by almost 20%. Unfortunately, this technique did not prove to be robust, and did not succeed against the more sophisticated Carlini & Wagner attack.

2.4. The GAN Defense

2.4.1. GAN Background

An interesting approach to defending against attacks is leveraging a recent yet extremely popular model called a *Generative Adversarial Network (GAN)*.

The most common application of neural networks is discriminative, where we have some inputs that we are trying to categorize. However, neural networks can be applied to many other problems as well. One such problem is trying to learn the probability distribution function of some dataset from the data. There are two main approaches to this. One is to explicitly learn the function density function for every possible input. Many density estimation techniques such Parzen Windows and K-Nearest Neighbors take this approach. Another way to *learn* the distribution to which a dataset belongs to is to create a model that can generate new synthetic data points that seem as if they are part of the actual dataset. This is implicit approach is taken by GANs where it learns to generate new data points indistinguishable from the actual dataset. For example, a GAN trained on images of handwritten digits (such as those from the MNIST dataset), will be able to generate new images of digits that look like they were handwritten and are also possibly distinct from the images it was trained on. To make sure that the images generated by the GAN have some variety, we feed a different random seed to it for each new generated image.

The architecture and training process of the GAN is depicted in the figure above. A GAN consists of a generator and a discriminator. The generator takes in some random seed and tries its best to generate a "fake image" that is indistinguishable from the "real images" in the training set. The discriminator is trained to classify whether an inputted image is real or fake. We can model this by creating separate objective function (or cost functions) for the discriminator and generator. The cost function of the discriminator is defined so as when minimized, the discriminator is able to distinguish well between the real and fake images. The generator on the other hand has an objective that is designed to fool the discriminator by making the fake images as indistinguishable to the real images as possible. Note that these two objectives are contradictory – as the generator gets better, the discriminator gets worse and vice versa. The idea is that generator and discriminator compete against each other in a game where the generator tries to fool the discriminator and the discriminator tries not to be fooled. Over time, both the discriminator and generator get better. In the end, the discriminator is generally discarded, and the generator is used as the model to generate new images. When neural networks are used for both the generator and discriminator, gradient descent (described in the FGSM section) is used to train both the networks. In every iteration both the generator and the discriminator have their parameters updated to make sure that they both learn at a similar pace. The exact algorithm and objective function details can be found in the original GAN paper by Goodfellow et al [13].

2.4.2. Conditional GANs

Conditional GANs are a natural extension of the GAN model described above. Conditional GANs learn conditional distribution functions. Instead of just feeding in a random noise, some other features can be inputted to the GAN to condition outputs.

The figure above demonstrates this (Which figure? -Trev). An example of a conditional GAN where we still train the GAN on images of handwritten digits, but this time we input to the Generator not only noise but also a number. For example, if we feed in the number '7' and noise to the Generator we would expect an image of the number '7' as the output. The features to condition on do not have to be as simple as numbers. For example, the Super-resolution GAN (SRGAN) takes in a low-resolution image and outputs a high resolution one [14].

2.4.3. GANs as a Defense Mechanism

Finally, we can use the idea of conditional GANs and SRGAN to defend against adversarial examples. The SRGAN is essentially a transformation of a low-resolution image to a high-resolution one. In a similar manner, we can transform or clean an adversarial example to make it non-adversarial. This is the idea behind APE-GAN, and is shown in the figure below (taken from the APE-GAN paper: [15].

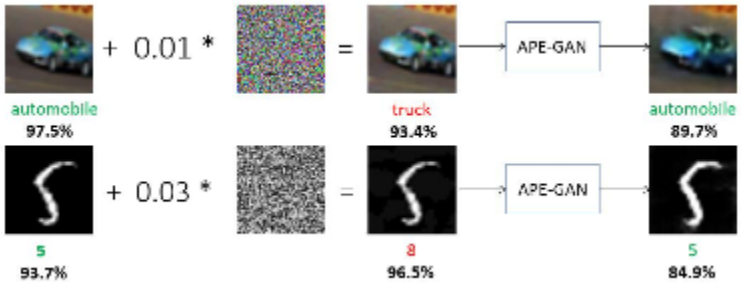

Figure 2-3. Example APE-GAN Filter

APE-GANs supply the Generator with adversarial images as input. The Generator attempts to take these adversarial images and transform them into normal images (non-adversarial). The role of the discriminator, then, is to determine whether an image is adversarial or not. The training of the APE-GAN is similar to the original GAN, except that while training the Generator, we also minimize the mean square error (MSE) between the generated image and the corresponding non-adversarial image in addition to trying to fool the discriminator. This minimization makes sure that the generated images resemble the original non-adversarial dataset.

Once the generator is trained, we simply use it as a denoiser or cleaner. This is shown on the right in the figure above. (I take it we have a task to include figures in this paper -Trev) If we train a target model to classify digits, the input (regardless of whether it is adversarial or not) is passed through the generator to create a "cleaner" non-adversarial image. This image is then passed through the target model for classification.

While this technique has promise, we were not able to replicate the success of the APE-GAN paper in which they were able to successfully defend against several attacks. We found the GAN we trained to be very dependent on specific ratios of input data, and were unable to successfully train a GAN that could defend against sophisticated attacks.

This technique for defense is still quite new, and we will continue to watch to see if the defense progresses to the point where it can be an acceptable and applicable defense to these types of attacks.

2.5. Risk Assessment

A more general defense against adversarial attacks involves developing high level risk assessment and planning a holistic cybersecurity approach based on that assessment. Most crucially, if the model will be deployed into a high risk input space, all stakeholders must be aware of the possible threats to the model. A "high risk input space" represents an environment where there exists a trust boundary around the set of inputs, or where these inputs are otherwise insecure.

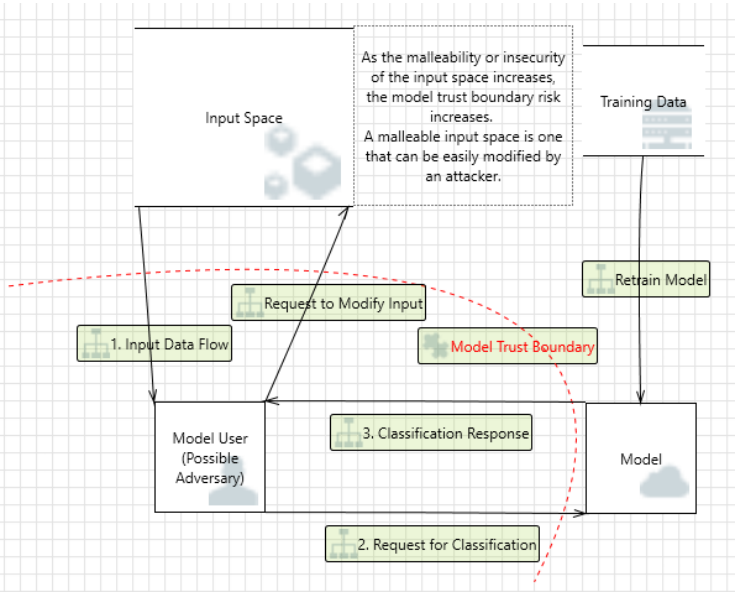

Figure 2-4. A typical machine learning input space

The request/response interaction between the model users and the input space may or may not be a digital interaction. For example, using stickers to tamper with traffic signs - which are considered the 'inputs' to a convolutional neural network - is purely a real-world interaction. Examples of high-risk input spaces include any real-time imaging system such as those used by facial recognition, or any other un-curated set of dynamic inputs from an external device. Examples of low-risk input spaces include static sets of inputs (like the MNIST training set), inputs comprising values that are difficult for an adversary to spoof or predict (keys or passwords), or inputs that are not shared between users and are part of a curated classification process, such as classifying MRI images in a secure

environment. In general, as the malleability of the input space increases, so does the risk associated with the model trust boundary; if the input space can be defended in some way using either physical or cybersecurity controls, this can help mitigate the trust boundary risk. A malleable input space is one that can be modified by an attacker.

When developing a machine learning tool, it will be helpful to threat model its vulnerabilities as well as its deployment environment. We have identified three security categories for machine learning models that we hope will be useful during its design: defensible, semi-defensible, and indefensible. Defensible models include those with low-risk input spaces. These models and their deployment environments still need to be designed securely, but with the right controls in place, system users can expect that these inputs have not been tampered with. Semi-defensible models include models that have at least one high-risk trust boundary that can be mitigated in some way. For example, an image detection model that allows authenticated users to modify shared inputs may be vulnerable to tampering if the set of authenticated users contains potential adversaries. By managing these users to detect adversarial behavior, or by employing basic security constraints like rate-limiting or partitioning input data, at least some of this vulnerability can be mitigated. Indefensible models are those models that allow unfettered access to the shared input space and output data sets. A computer vision model that provides instant classification on any given input image is classified as indefensible. Furthermore, any model that exposes training data to any potential adversary is indefensible.

3. CONCLUSION

3.1. Summary of Results

Machine learning is going to be vulnerable to these types of attacks as the field progresses. We were not able to find a legitimate defense that could not be bypassed by a more specialized, advanced attack. It seems that the field has currently taken on a "cat and mouse" effect that a lot of cyber security tends to evolve to, where a new defense is created, only to be defeated in another iteration of research by a new more sophisticated attack. While adversarial training has a low barrier of entry, it can be defeated by robustness of an attack. Ensembles seem to work very well (Microsoft Defender), but also have drawbacks in amount of time to develop, real time forecasting delays, and applicability of model variety (not all models will work for all types of problems and input data). Ensembles have not been tested thoroughly against the most sophisticated attacks. The GAN defense seems new and promising, as it has not been yet defeated on an academic level, but is tricky to develop and implement properly, as we found in our study.

3.2. Future Work

We have proposed an effort that would provide more automation around the cleverhans library, providing ML engineers at Sandia a resource to easily test their models for vulnerability. While this may not inform ML engineers on how to defend against these attacks, it would at least be a starting point in assessing and controlling vulnerabilities.

We also plan to continue to scan academic literature for defenses to adversarial examples.

3.3. Recommendation

Given our conclusion that no truly effective methods exist to defend machine learning models from adversarial attacks, we recommend a holistic, cybersecurity-focused approach to ensure model and data integrity when deploying these tools into high-risk input spaces.

Once a model has a risk classification, a more comprehensive threat model can be created. If the model is defensible or semi-defensible, all relevant vulnerabilities must be discovered and validated across all trust boundaries. These should include any general cybersecurity vulnerabilities that may exist in the deployment environment, such as risks associated with spoofing, tampering, non-repudiation, denial of service, elevation of privilege, and especially disclosure of information. Special care must be taken to avoid data leakage concerning model behavior and training data. Given practitioners' reliance on third party libraries, attention must be paid to any vulnerabilities that exist in these libraries.

If the model has been classified as indefensible and the risk of adversarial attack has been accepted, general cybersecurity principles should still be applied, but the risk should nonetheless be understood by all stakeholders. If the indefensible model's risk cannot be accepted, then the only way to mitigate that risk is to only deploy the model to a space that is perfectly non-adversarial.

Finally, all stakeholders must understand that even the most defensible input spaces are vulnerable to tampering from malicious users, especially insiders. High-risk models must take into account that their models will never be completely immune to adversarial attacks.

REFERENCES

[1] Dr. Philip Kegelmeyer, Sandia National Laboratories, "Adversarial Issues in Machine Learning," SAND2017-12451J, 2017.

[2] B. Biggio, I. Pillai, S. Bulo, D. Ariu, M. Pelillo and F. Roli, "Is Data Clustering in Adversarial Settings Secure?," in *ACM*, Berlin, Germany, 2013.

[3] N. Carlini and D. Wagner, "Adversarial Examples are not Easily Detected: Bypassing Ten Detection Methods," in *AISec* , Dallas, 2017.

[4] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," 2016.

[5] R. Treit, H. Stewart and J. Parikh, "Protecting the protector: Hardening machine learning defenses agains adversarial attacks," 9 8 2018. [Online]. Available: https://cloudblogs.microsoft.com. [Accessed 7 9 2018].

[6] I. J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *ICLR 2015*, 2015.

[7] P. Kegelmeyer, T. Shead, J. Crussell, K. Rodhouse, D. Robinson, C. Johnson, D. Zage, W. Davis, J. Wendt, J. Doak, T. Cayton, R. Colbaugh, K. Glass, B. Jones and J. Shelburg, "Counter Adversarial Data Analytics," *Sandia National Laboratories,* Vols. SAND2015-3711, 2015.

[8] A. Athalye, L. Engstrom, A. Ilyas and K. Kwok, "Synthesizing Robust Adversarial Examples," in *International Conference on Machine Learning*, Stockholm, Sweden, 2018.

[9] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, B. Celik and A. Swami, "Practical Black-Box Attacks agains Machine Learning," in *ACM Asia Conference on Computer and Communications Security*, Abu Dhabi, UAE, 2017.

[10] I. Goodfellow, N. Papernot and N. Carlini, "cleverhans," [Online]. Available: https://github.com/tensorflow/cleverhans.

[11] T. Strauss, M. Hanselmann, A. Jurnginger and H. Ulmer, "Ensemble Methods as a Defense to AdversarialPerturbations Against Deep Neural Networks," February 9, 2018.

[12] A. Brebisson, K. Kumar and J. Sotelo, "Lyrebird," [Online]. Available: https://lyrebird.ai/.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative Adversarial Nets," *Universit´e de Montŕeal,* 2014.

[14] C. Ledig, "Photo-Realistic Single Image Super-Resolution Using a Generative AdversarialNetwork," *Twitter,* 2017.

[15] S. Shen, G. Jin, K. Gao and Y. Zhang, "APE-GAN: Adversarial Perturbation Elimination with GAN," in *Institute of Computing Technology Chinese Academy of Sciences*, Beijing, China, 2017.

[16] H. Anderson, A. Kharkar, B. Filar and P. Roth, "Evading Machine Learning Malware Detection," in *Black Hat USA*, as Vegas, NV, USA, 2017.

[17] A. Kurakin, I. Goodfellow and S. Bengio, "Adversarial Machine Learning at Scale," in *International Conference on Learning Representations*, Toulon, France, 2017.

[18] T. Pang, C. Du, Y. Dong and J. Zhu, "Towards Robust Detection of Adversarial Examples," in *32nd Conference on Neural Information Processing Systems*, Montréal, Canada, 2018.

[19] "On the (Statistical) Detection of Adversarial Examples," *CoRR,* vol. abs/1702.06280, 2017.

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Kristin Adair	9370	kladair@sandia.gov
Joselyne Gallegos	9370	jogalle@sandia.gov
Austin Short	9371	ashort@sandia.gov
Trevor La Pay	9371	tlapay@sandia.gov
Gary Huang	9371	ghuang@sandia.gov
Marc Sanchez	10757	msanch7@sandia.gov
Joshua Mitchell	10757	josmitc@sandia.gov
Philip Kegelmeyer	8700	wpk@sandia.gov
Jeremy Wendt	5854	jdwendt@sandia.gov
Joe Ingram	9365	jbingra@sandia.gov
Han Lin	9315	hwlin@sandia.gov
Matthew Smith	9323	mdsmith@sandia.gov
Technical Library	01177	libref@sandia.gov

Email—External [redacted]

Name	Company Email Address	Company Name
Apurva Gandhi	apurvaga@usc.edu	University of Southern California

This page left blank

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.