# Counter AI Synthesis Paper

Masa Nakura

May 1, 2024

## 1 Introduction

Artificial Intelligence has become increasingly prevalent in our daily lives in various applications ranging from spam mail detection to self-driving cars. As AI becomes more accessible to various users, trust becomes a key factor in whether an AI-powered system is frequently used or not. User trust in AI models is influenced by various factors such as the privacy of personal data, its performance given specific tasks, consistency in varying situations, and even the explainability of AI alogrithms. Trust is especially crucial in safety-critical situations such as assisted driving, where misjudgment involving pedestrians could cause significant damage. In these cases, users will use AI technologies only if they can fully trust them. As such, advancements in trusted AI would promote the further spread of AI applications by allowing users to feel comfortable with AIs that affect their daily lives.

This synthesis paper covers my exploration of counter-AI, a subfield of trusted AI. In a world full of artificial intelligence, one of the biggest threats would be the presence of adversaries, or attackers, who can intentionally cause AI systems to misbehave. Thus, we seek the prevention of all possible attacks so that users can always rely on AI systems. The study of counter-AI, also known as adversarial AI, deals with the methodologies used by adversaries to degrade or manipulate AI systems. Given an understanding of different attacks, our main objective is to build an AI model that is unaffected by these attacks. In particular, my study was motivated by the following questions:

- Can AI be subverted/co-opted to perform sub-optimally or change its goal (from the goal it was supposed to achieve)?
- How can we defend AI systems against being subverted or co-opted?

In this paper, I cover a detailed explanation of counter-AI to the extent to which I have explored and a discussion about how this study has impacted my perspectives on AI.

## 2 Counter AI

In this section, I will explain counter-AI at both high and technical levels. After introducing the various types of attacks that adversaries often employ, I will highlight research efforts for defending against such adversaries. Finally, I introduce methods of standardizing how well AI models defend against known adversaries. These standardization techniques are meant as an encouragement for further efforts to improve upon existing defense strategies.

### 2.1 Types of Attack

#### 2.1.1 Overview

The NIST Report on Adversarial Machine Learning by Alina Oprea and Apostol Vassilev [9] provides a comprehensive overview of adversarial machine learning - especially how attacks are classified. First, attacks are classified as having one of the following objectives: Availability Breakdown, Integrity Violation, and Privacy Compromise. While availability attacks indiscriminately attempt to break the performance of AI models at test time, integrity attacks target specific outputs or behaviors. In contrast to those attacks that decrease a model's performance, privacy attacks attempt to learn more information about an AI model or training data that would otherwise be kept private. Further decomposition of the attack types is given in Figure 1, which includes a detailed classification based on the objectives and the required capabilities of an attacker.

Next, an attack can be classified by the information required to perform the attack. While a white-box attack requires total information of a model such as the weights and training data, a black-box attack simply relies on the model output given testing data. A grey-box attack is somewhere in between, such as simply requiring knowledge about a model architecture but not the weights. A more detailed comparison between white-box and black-box attacks is given in Table 1.

| | White-box | Black-box |
|---|---|---|
| Attacker Access | Attackers have complete knowledge about the target model | Attackers have no prior knowledge about the target model |
| Accessible information | Known Weights, Network Architecture, Hyperparameters, Training Dataset, etc. | Known confident scores, logits, or output labels |
| Attack Types | Optimization methods with choices for distance metrics, universal evasion attacks, physically realizable attacks. | Score-based attacks(scores/logits), Decision-based attacks(labels) |
| Practical use | Easier to perform, more types of attacks. Can be used for attack transferring. | More practical, as essential model information if often confidential. |

Table 1: White-box vs Black-box attack comparison

Attacks that degrade a model's performance can further be separated into Evasion and Poisoning attacks. In an evasion attack, an adversary attempts to generate adversarial examples, which are testing samples whose classification can be controlled by an attacker through minimal perturbations. While evasion attacks are performed under testing time, poisoning attacks happen at the training stage, where training data or models are intentionally changed (or 'poisoned') to break down the availability or integrity of the model.

In my study, I specifically focused on evasion attacks and the methods to mitigate their effect, which I will explain in further detail in the following sections.

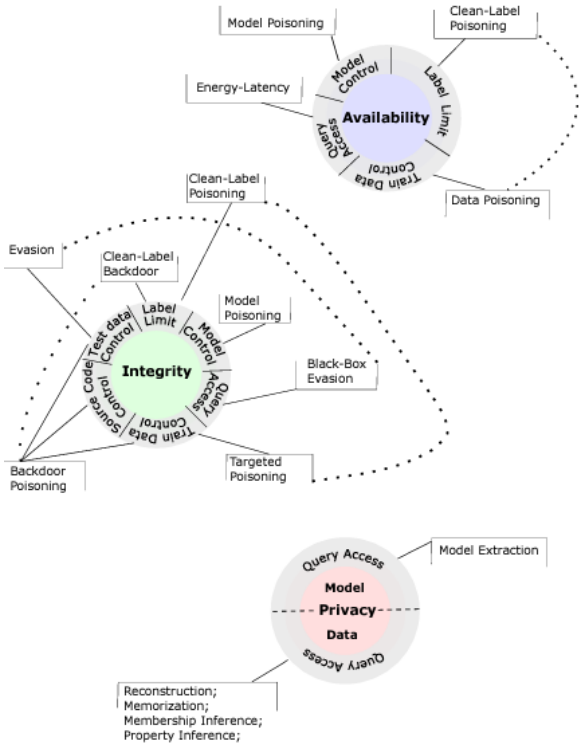


Figure 1: Taxonomy of attacks on AI systems [9]. Each circle represents an attack objective, and the circular outer layer represents the required capabilities an adversary needs. The callout represents the name of a specific attack methodology.

### 2.1.2 Evasion Attacks - Threat Model

A threat model precisely defines the types of attacks an adversary employs, and it is a standard convention to define the threat model at the beginning of any study. Specifically, the two attacks I introduce in the following sections are white-box evasion attacks that utilize first-order gradient information for generating adversarial examples. An evasion attack produces an adversarial example $\tilde{x}$ by perturbing an original data $x$, such that $\|\tilde{x} - x\|_\infty < \epsilon$, where $\epsilon$ is the maximum allowed perturbation distance. For example, in the context of image data in the $\ell_\infty$ distance metric, $\epsilon$ would be the largest change allowed in each pixel. Note that the distance metric is most commonly $\ell_\infty$, but other metrics such as $\ell_2$ can be used. When defining a threat model for evasion attacks, this variable $\epsilon$ is the most important variable. Because $\epsilon$ defines how large a data point can be perturbed, it defines how closely an adversarial example will look semantically compared to the original image. For example, Figure 2 demonstrates the difference of point clouds that were perturbed using different sizes $\epsilon$ (each point in the point cloud was shifted by a maximum distance of $\epsilon$). Since the goal of evasion attacks is to produce an example that is indistinguishable from the original class to a human eye, we would want to find a good $\epsilon$ such that it is still semantically similar to the original data but fools an AI model.

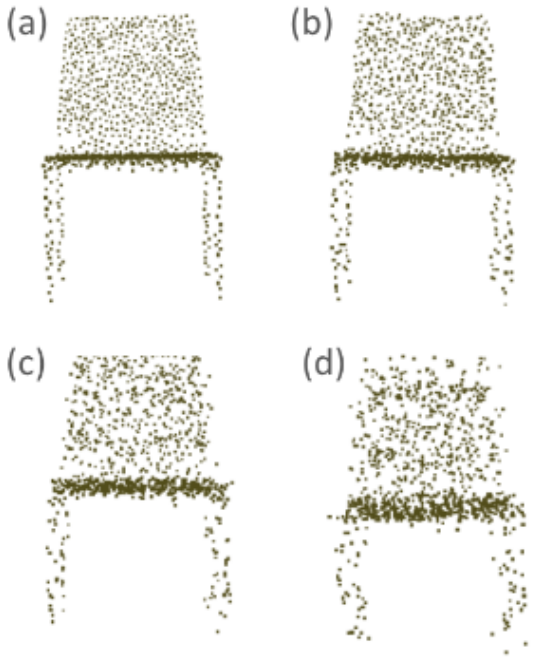


Figure 2: (a) A point cloud of a chair from ModelNet40. This chair was perturbed using APGD (refer to section 2.1.4) with (b) $\epsilon$ = 0.1, (c) $\epsilon$ = 0.25, (d) $\epsilon$ = 0.5. Under a standard DGCNN model for point clouds, they were classified as (a) chair, (b) nightstand, (c) nightstand, and (d) plant

### 2.1.3 White-box Evasion Attack: PGD

Projected Gradient Descent (PGD) is an evasion attack methodology popularized by Madry et. al. [8]. Given an $\epsilon$ from the threat model, we can define the allowed perturbation of a specific data point $x$ by an $\ell_\infty$ (or $\ell_2$) sphere of radius $\epsilon$ around $x$. PGD is a first-order optimization method, where $x$ is iteratively updated in the direction such that the Loss function is maximized. In other words, it is an iterative variant of another evasion attack called the Fast Gradient Sign Method(FGSM) that performs the update steps just once. Specifically, FGSM calculates

$$\tilde{x} = x + \epsilon \text{sgn}(\nabla_x L(\theta, x, y))$$

where the function $L$ returns the loss function given weights $\theta$, data $x$, and label $y$ [8]. On the other hand, PGD is simply an iterative variant such that

$$x^{t+1} = \Pi_{x+S}(x^t + \alpha \text{sgn} \nabla_x L(\theta, x, y))$$

where $x^t$ is the perturbed data after $t$ iterations, $\Pi_{x+S}$ is a function that projects an input to a region in the allowed $\ell_\infty$ ball, and $x^0$ starts at a random point in the $\ell_\infty$ ball[8]. One key distinction between PGD and FGSM is that FGSM perturbs the x by a distance of $\epsilon$ in a single step, while PGD repeatedly takes a step of size $\alpha$ toward a $\tilde{x}$ that maximizes the loss.

One obvious concern for PGD is the possibility of reaching a local maxima within the $\ell_\infty$ ball. However, Madry et.al observed that PGD attacks over multiple runs with random restarts achieved a fairly consistent resulting loss value, which signifies that PGD can almost always find an adversarial example with a 'good enough' loss. From this concentration phenomenon of the loss values, Madry et. al. also propose that a local maxima significantly higher than the ones found by PGD is extremely hard to find using first-order methods; even a large number of random PGD restarts failed to find an adversarial example with a significantly higher loss value. Thus, Madry et. al.[8] claim that PGD is the "ultimate" and "universal" first-order optimization, which becomes a useful proposition when training a model robust against attacks(section 2.2).

### 2.1.4 Another White-box Evasion Attack: APGD

Auto-Projected Gradient Descent(APGD) address three weaknesses of PGD: 1) A fixed step size does not guarantee convergence of loss, and performance is highly dependent on the human-chosen step size $\alpha$, 2) loss plateaus quickly because PGD algorithm is agnostic of the number of iterations, and 3) PGD algorithm is unaware of the current trend of the loss value[6]. Introduced by Croce & Hein, APGD addresses these issues by automatically adjusting the step sizes based on the number of iterations left and on the success rate of finding higher loss values given a specific step size [6]. APGD begins with a high step size to search for a general area within the $\ell_\infty$-ball that has the highest loss value. This is called the exploratory phase so that APGD can locate a good starting point for further optimization. During this exploration phase, the algorithm flags a checkpoint at which the loss value is the highest. When the step sizes decrease, the algorithm restarts at the checkpoint with the highest loss to continue the search for an even higher loss. This phase with lower step sizes is called the exploitation phase, where the algorithm attempts to squeeze out the highest possible loss. Furthermore, APGD has scheduled checkpoint iteration numbers $W = w_0, w_1, ..., w_j$, which are set at

pre-set percentages of the total iteration number. At these checkpoints, the step size is halved if either of the following conditions are met:

**Condition 1:**

$$\sum_{i=w_j-1}^{w_j-1} 1_{f(i+1)>f(i)} < \rho \cdot (w_j - w_{j-1})$$

where $f(i)$ is the highest loss value found in the first $i$ iterations, and $w_j$ are the number of iterations at the jth checkpoint. In other words, the lefthand side of the inequality counts how many times it achieved to find an $x$ that updated the running maximum of the loss value. If this number is less than the fraction $\rho$ is the total update steps between the two most recent checkpoints, we update the step size because the current step size has failed to find a good loss value for enough time.

**Condition 2:**

$$\nu^{(w_{j-1})} \equiv \nu^{w_j} \text{ and } f_{\max}^{(w_{j-w})} \equiv f_{\max}^{(w_j)}$$

where $f^i$ is still the highest loss value found in the first $i$ iterations, and $\nu^{w_j}$ is the step size at iteration $w_j$. In other words, the step size updates if there has been no update in the running max of the loss function and the step size since the previous checkpoint.

Thus, APGD addresses limitations 2 and 3 of PGD by taking into account the trend of loss values by updating the step sizes using conditions 1 and 2 and by considering the total number of iterations through the usage of checkpoints. Results have shown that APGD performs better than PGD in obtaining a higher Loss value as well as obtaining a lower accuracy in different models[6]. Figure 3 shows a comparison between the performances of PGD and APGD in image classification, where APGD almost always performs better than PGD in both loss(higher) and accuracy(lower).

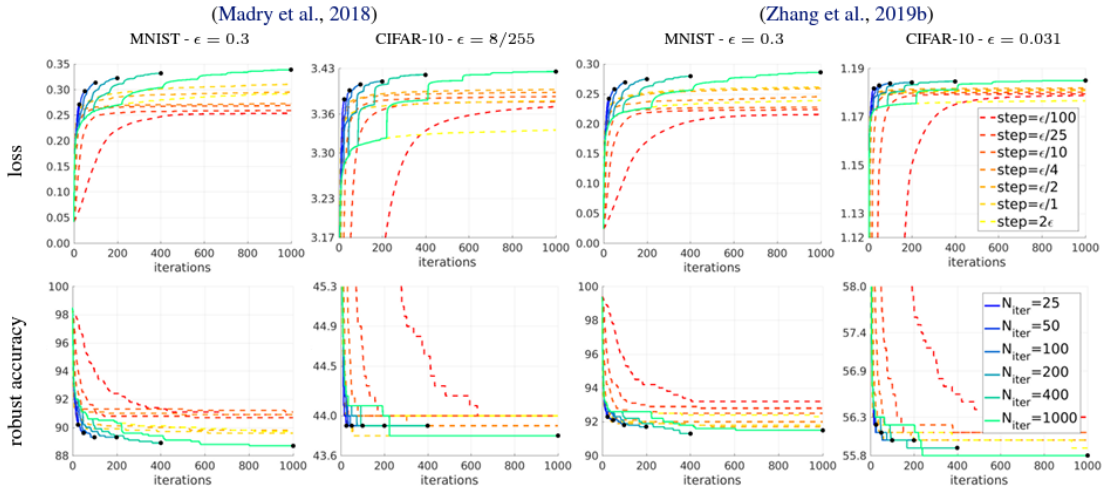


Figure 3: PGD with Momentum vs APGD[6]:best cross-entropy loss (top) and robust accuracy(bottom) obtained as function iterations. Tested on two models by Madry et. al. (left)[8] and Zhang et. al. (right)[10] for PGD with a momentum term $\alpha$ = 0.75 (dashed lines) with different fixed step sizes and APGD(solid lines) with a different total number of iterations. APGD outperforms PGD with Momentum for every budget of iterations in terms of loss.

Beyond the observation that APGD performs better than PGD in terms of both loss and accuracy, this graph provides us an explanation about the behavior of APGD. In Figure 3, the APGD loss lines appear to have occasional jumps that significantly increases the loss value. This can be explained by the transition to a smaller step size $\alpha$. As explained before, APGD keeps track of a location in the $\ell_\infty$ ball with the highest loss until a specified checkpoint $w_j$. If the step size $\alpha$ decreases at checkpoint $w_j$ (if conditions 1 and 2 are satisfied), the algorithm backtracks to the location with the highest loss and continues to explore with smaller step sizes. This causes a sudden spike in the best loss value found, as observed in the graph. This behavior can be characterized as the transition from the exploration phase to the exploitation phase, where a more specific neighborhood of perturbations with high loss is exploited every time the step size decreases. I attempted to verify this explanation by replicating the loss graph from Figure 3. To do so, I edited the AutoAttack source code[6], which included an implementation of APGD. I succeeded in creating a graph that displayed the expected loss of the data batch for every iteration of applying a step of APGD. The result can be seen in Figure 4. As expected, we observed jumps in the loss value when the step sizes decreased.

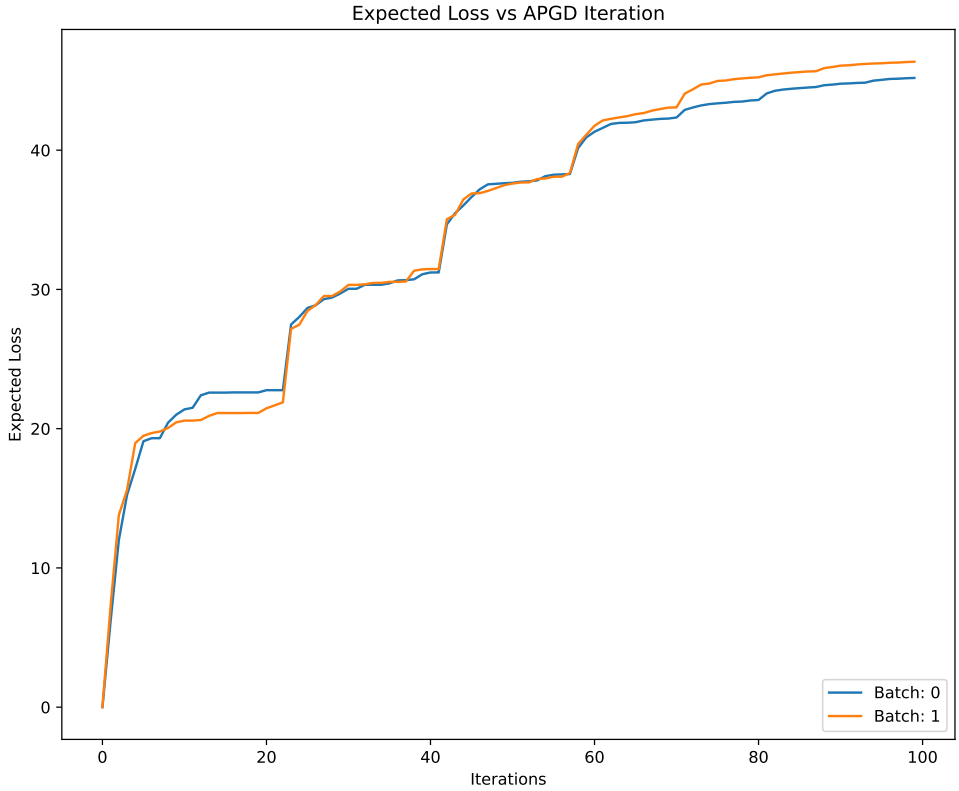


Figure 4: Expected Adversarial Loss 2.3 of APGD-perturbed Cifar images on ResNet model. Expected loss for different batches are shown on different lines. APGD was performed for 100 iterations, and the batch size was 5. These numbers are significantly lower than the experiments from Figure 3, but this experiment was modified to perform quickly on the CPU. The observations made will generally be consistent with higher batch sizes and APGD iterations.

## 2.2 Towards Robustness Against Adversarial Attacks

### 2.2.1 Adversarial training overview

The main purpose of gaining a deeper understanding of the above attacks is to design methodologies to mitigate their risks. One such methodology is adversarial training, proposed by Madry et. al. [8], where the model learns to be robust against adversarial examples during the training phase. As above, we define our specific threat model to be the $\ell_\infty$-ball around any data point $x$. Now, let $S$ be the set of all allowed perturbations $\delta$, such that $x+\delta$ is included within the $\ell_\infty$-ball. Given this threat model, the objective function for adversarial training is as below[8].

$$\min_\theta \rho(\theta), \text{ where } \rho(\theta) = \mathbb{E}_{(x,y)\sim D}[\max_{\delta \in S} L(\theta, x+\delta, y)]$$

This formulation of the objective function allows us to view the problem as a saddle point problem separated into an inner maximization and an outer minimization. The inner maximization aligns with the goals of the objective functions of PGD and APGD, as it aims to find a perturbation $\delta$ such that the loss value of the adversarial example is maximized. On the other hand, the outer minimization attempts to find the model parameters, or weights, such that the expected loss is minimized for the strongest adversarial examples. The outer minimization is similar to traditional training objectives as below:

$$\min_\theta \rho(\theta), \text{ where } \rho(\theta) = \mathbb{E}_{(x,y)\sim D}[L(\theta, x, y)]$$

In this case, the difference between adversarial training and traditional training is simply whether the minimization objective finds the parameters for the loss of perturbed data points $\tilde{x} = x + \delta$ or of clean, in-distribution data points x. As such, the objective of adversarial training specifies a clear goal of obtaining the parameters such that the expected loss (population risk) is extremely low given an adversarial example. When a model reaches this goal, it is *adversarially robust* and is not expected to be fooled by adversarial examples.

### 2.2.2 Adversarial Training Implementation

To solve the inner maximization problem, we can directly employ Projected Gradient Descent, which consistently succeeds in finding a high loss value. Because PGD is an 'ultimate' and first-order optimization attack, as explained in section 2.1.3, Madry et. al. suggest that robustness against PGD attacks guarantees

universal robustness against first-order attacks. This guarantee would provide us with a strong basis because 1) attacks relying on first-order information are most common for the current practice of deep learning, 2) black-box attacks, which do not even use first-order information, will fall under the robustness guarantee, and 3) transfer attacks, which tend to perform worse than fully white-box first order attacks, also fall under the guarantee. As such, PGD is commonly used as the methodology to solve the inner maximization problem, so that the resulting model weights minimize the population risk against the 'ultimate' PGD attack and thus form a universal robustness guarantee against common types of attacks.

Now that we have solved the inner maximization problem, the next logical step would be to find a gradient for the model parameters that solves this saddle point optimization question. However, the true descent direction initially seems ambiguous, as the minimization and maximization optimization should happen simultaneously for data point $x$ with model parameters $\theta$. However, Madry et. al. suggest that we could solve the outer maximization problem sequentially after solving the inner maximization - Danskin's theorem states the gradients at inner maximizes correspond to the descent directions for a saddle point problem. Thus, if $\tilde{x}$ is the adversarial example obtained by PGD, the objective function for adversarial training is reduced to

$$\min_\theta \rho(\theta), \text{ where } \rho(\theta) = \mathbb{E}_{(x,y)\sim D}[L(\theta, \tilde{x}, y)]$$

Thus, PGD can be thought of as a data augmentation method, where we train the model on the augmented $x$. However, unlike normal data augmentation, a new adversarial example must be generated for each $x$ every epoch, as the adversarial example that produces the highest loss will change every time the model parameters are updated. Thus, adversarial training is simply a slightly modified implementation of traditional training methodologies, where each data is perturbed (augmented) through PGD before every epoch.

## 2.3 Standardization

Beyond Adversarial Training [8], there have been many other research efforts to improve the robustness of a model against adversarial examples. However, the evaluation metrics on these models are often non-uniformed, which makes it difficult to compare different defense strategies. This calls for a standardized metric that accurately and comprehensively measures the robustness of these models under evasion attacks.

### 2.3.1 AutoAttack

Introduced in the same paper as APGD, AutoAttack is an evaluation framework developed by Croce and Hein [4]. AutoAttack contains an ensemble of attacks, including APGD, a targetted variant of APGD (APGD-T), Fast Adaptive Boundary Attack (FAB)[5], and square attack[2]. The first three attacks are white-box - while APGD and APGD-T attempt to find the adversarial example with the highest loss, FAB attempts to find an adversarial example with the minimal norm of perturbation required for misclassification. Meanwhile, square attack is a score-based black-box attack that is query-efficient and is known to have a good success rate. This diversity of attacks within AutoAttack ensures that a classifier is tested comprehensively - even if some attacks fail, AutoAttack will attempt another attack that may succeed. Furthermore, this diversity allows us to understand a trend of vulnerabilities in some models - even if the average robustness against all four attacks is the same for two different models, we may gain insight into the model's tendency for vulnerability depending on which attacks a model fails against. Beyond the diversity of the attacks, AutoAttack's biggest trait is that it is parameter-free. When researchers evaluate a model's robustness against adversaries, there are often parameters that can be chosen for the attacks. This could cause an inconsistency in the adversarial robustness evaluations because the models would be tested against different attacks. Thus, AutoAttack provides a truly standardized method for evaluation.

From the experiments conducted by Croce and Hein, we observe that AutoAttack seems like a comprehensive evaluation technique. Using AutoAttack, they re-evaluated a variety of 'adversarially robust' models with a reported robustness accuracy from their respective papers. The final robustness accuracy reported by AutoAttack was almost always lower or around the same compared to the reported accuracy, which showed that AutoAttack was a stricter(and thus most extensive) evaluation metric than the various evaluation metrics initially used.

A possible concern for standardized usage of AutoAttack is the overadaption of new defenses to AutoAttack. This would cause us to overestimate the true robustness of a model if the model performs well against AutoAttack but not against other attacks such as an adaptive attack, where there is a real-time adaption of the attack (and thus adapted attacks cannot be standardized). To prevent this overadaption, yet encourage the development of models with higher robustness against AutoAttack, Croce and Hein hosted an online leaderboard called the RobustBench [4]. In this leaderboard, models are first ranked in terms of their performance against AutoAttack. However, it also reports the 'best known robust accuracy' which reports the lowest known robustness accuracy against an attack outside of AutoAttack. With these two robustness metrics, Croce and Hein encourages external researchers to improve adversarial robustness techniques while creating better adaptive attacks such that the model will perform much worse than against AutoAttack. Currently, most models on the leaderboard have around AutoAttack accuracy and best-known robust accuracy, which shows how AutoAttack is mostly a comprehensive evaluation metric fit for standardized use. However, there is currently a single model on the leaderboard that has a significant difference in the two metrics (best known robust metric is 7.08% lower). This serves as either an encouragement for a search for an even better standardization metric or a reminder that a single standardization metric may never be fully trusted.

### 2.3.2 Evaluation metrics

In Table 2 below, I summarize common evaluation metrics. I encourage the use of these terminologies to ensure consistency when evaluating models. For explanations on the notations in table 2, refer to Table 3.

| Metric | Symbol | Definition/Equation | Relations | Objectives |
|---|---|---|---|---|
| Clean Accuracy | $a$ | The accuracy of a standard classifier on clean data<br>$a$ = eval$(f_{\theta^c}, D^c) = \frac{\sum_{j=1}^{N} 1_{c_j==y_j}}{N}$,<br>where $c_j = f_{\theta^c}(x_j)$, $x_j \in D^c$, and $N = \lvert D^c \rvert$ | | max(a) |
| Attacked Accuracy | $a^{c\alpha}$ | The accuracy of a standard classifier on attacks.<br>$a^{c\alpha}$ = eval$(f_{\theta^c}, D^a) = \frac{\sum_{j=1}^{N} 1_{c_j==y_j}}{N}$,<br>where $c_j = f_{\theta^c}(x_j)$, $x_j \in D^a$, and $N = \lvert D^a \rvert$ | $a^{c\alpha} < a$. Let<br>$\delta_1 = a - a^{c\alpha} > 0$. | Attacker:<br>max$(\delta_1)$ |
| Robust Clean Accuracy | $a^{\alpha c}$ | The accuracy of an adv. trained classifier on clean data.<br>$a^{\alpha c}$ = eval$(f_{\theta^a}, D^c) = \frac{\sum_{j=1}^{N} 1_{c_j==y_j}}{N}$,<br>where $c_j = f_{\theta^c}(x_j)$, $x_j \in D^c$, and and $N = \lvert D^c \rvert$ | $a^{\alpha c} \leq a$. Let<br>$\epsilon_1 = a - a^{\alpha c} > 0$.<br>Ideally, $a^{c\alpha} < a^{\alpha c}$<br>and $\delta_1 \gg \epsilon_1$ | max$(a^c)$ |
| Robust Accuracy | $a^r$ | The accuracy of adv. trained classifier on non-adaptive attacks.<br>$a^r$ = eval$(f_{\theta^a}, D^a) = \frac{\sum_{j=1}^{N} 1_{c_j==y_j}}{N}$,<br>where $c_j = f_{\theta^a}(x_j)$, $x_j \in D^a$, and $N = \lvert D^a \rvert$ | $a^r > a^{c\alpha}$. Let<br>$\delta_2 = a^r - a^{c\alpha} > 0$.<br>$\delta_1 > \delta_2$. Ideally,<br>$\delta_2 \gg \epsilon_1$. Let<br>$\epsilon_2 = \lvert a^r - a^{\alpha c} \rvert > 0$. | min$(\epsilon_1)$<br>min$(\epsilon_2)$<br>max$(\delta_2)$ |
| Best Known Robust Accuracy | $a^B$ | The accuracy of an adv. trained classifier on the worst possible attack.<br>$a^B$ = eval$(f_{\theta^c}, D^{aa}) = \frac{\sum_{j=1}^{N} 1_{c_j==y_j}}{N}$,<br>where $c_j = f_{\theta^c}(x_j)$, $x_j \in D^{aa}$, and $N = \lvert D^{aa} \rvert$ | $a^{c\alpha} < a^B \leq a^r$ | min$(a^r - a^B)$. |
| Clean Loss | $L^c$ | Loss value (measure of unhappiness in scores) in standard classfiers.<br>$L^c = L_{x \in D^c}(\theta, x, y)$ | | min$_\theta(L^c)$ |
| Adversarial Loss | $L^a$ | Loss value when input is adversarial.<br>Loss value of inner optimization for Eq 2.1<br>$L^a = L_{x^{adv} \in D^a}(\theta, x^{adv}, y)$, where $x^{adv} = x + \delta$<br>s.t. $x \in D^c$ and $\delta \in S$, where S is a threat model. | $L^a \gg L^c$ | max$_\delta(L^a)$. |
| Adversarial Training Loss | $L^{at}$ | Loss value of outer optimization for Eq 2.1<br>$L^{at} = \rho(\theta) = \mathbb{E}_{(x,y)\sim D}[\max_{\delta \in S} L(\theta, x + \delta, y)]$<br>$= \frac{1}{\lvert D \rvert} \sum_{j=1}^{\lvert D \rvert} \max_{\delta \in S}(L_j^a)$, where $\rho$ is from Eq. 2.1. | | min$_\theta(L^{at})$. |

Table 2: Evaluation Metrics

| Notation | Definition |
|---|---|
| $f_{\theta^c}$ | A classifier trained normally |
| $f_{\theta^a}$ | An adversarially-trained classifier |
| $D^c$ | Clean, in-distribution data |
| $D^a$ | Adversarially perturbed data (non-adaptive attack) |
| $D^{aa}$ | Adversarially perturbed data using adaptive attacks |
| eval$(f, D)$ | Function to evaluate classifier $f$ on given data $D$. (Mathematically defined in row 1 of table 1) |
| $L_{x \in D}(\theta, x, y)$ | Standard Loss function given a single input, its correct label, and network parameters. |
| Eq. 2.1 | min$_\theta \rho(\theta)$, where $\rho(\theta) = \mathbb{E}_{(x,y)\sim D}[\max_{\delta \in S} L(\theta, x + \delta, y)]$ |

Table 3: Notations

# 3 Personal Impacts

Studying counter AI over this quarter had a major impact on my academics and my social views. Furthermore, counter-AI added an extra layer of complexity to my knowledge of AI, which allows me to think critically when considering real-world applications. In this section, I would like to cover in detail how this research experience impacted me over this quarter.

## 3.1 Academics

This research experience most noticeably enriched my academic experience even inside the classroom. Particularly, it impacted my understanding of the materials in CSE 493G1: Deep Learning. As this was my first

quarter taking an artificial intelligence-related class, I learned 'vanilla' machine learning algorithms concurrent with counter-AI topics from guided research. This was truly a mind-expanding experience since I was constantly able to connect between the new ideas I learned in class and the Counter-AI research. Comparing and studying the differences between the objective functions of adversarial training and vanilla training, as discussed in section 2.2.1, is one such example. Another example was making connections between the gradient of the Loss with respect to the weights/parameters to update the model, in contrast to the gradient of the Loss with respect to the data point $x$ to obtain an adversarial example. Because these comparisons required attention to minor details, they allowed me to learn the material at a deeper level than if I were to take the class by itself. This alone demonstrates how the study on counter-AI enriched my classroom experience.

Beyond classroom material, Counter-AI also influenced my final project for CSE 493G1. In my project, I investigated the relationship between corruption and adversarial robustness of point cloud classifiers of various models and further attempted to train a model that improved on both robustness metrics. While the corruption robustness of point cloud classifiers was measured using a pre-corrupted dataset of point clouds, I measured the adversarial robustness using AutoAttack, which I described in Section 2.3.1. Since AutoAttack was made specifically for image classification tasks, my tasks involved modifying the AutoAttack source code to create adversarial examples for point clouds. After measuring both of these robustness metrics and analyzing their relationship, my next goal was to train a model to improve both robustness by employing adversarial training(Section 2.2) using APGD(Section 2.1.4 to solve the inner maximization problem. Adversarial training using APGD diverged from traditional methods of using PGD, but I hypothesized that it would perform better in terms of adversarial robustness because APGD succeeds in obtaining a higher lass value, meaning it solves the objective function more optimally. Furthermore, since a high adversarial robustness seemed to imply a high corruption robustness based on our previous analysis, we hoped our new adversarially trained model using APGD would achieve a higher dual robustness. Though our methodology showed the potential of adversarial training using APGD as a method for improving corruption and adversarial robustness, we identified several open discussions that we would want further insight into. A more detailed explanation of our project can be viewed in this paper in the following URL, which I have also uploaded onto the Counter-AI Dropbox, https://www.overleaf.com/read/dmbpqnxgmxpq#5365fd.

## 3.2 Social Views

Two of the biggest concerns as more AI-powered services and products become more accessible to society are the safety/integrity of AI and its privacy of personal information. As a part of my research in Counter-AI, I investigated the recent executive order by Joe Biden on "Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence" [1], which demonstrated the urgency and importance of safety for real deployment of AI. Through this investigation, I gained a new perspective that is completely different from what we learn in a deep-learning classroom environment, where we typically only care about the performance of a model in a controlled setting. I learned that what is equally important to performance is the trust of the consumers, so that everyone can comfortably coexist with artificial intelligence. This realization further motivates me to continue research in counter-AI and support regulations that ensure AI safety such as through standardized methods discussed in Section 2.3.

In the same executive order, President Biden also addressed the issue of artificial intelligence potentially displacing workers. A couple of weeks ago, near the CSE Gates Building, I saw a poster that called for the stop of the development of artificial intelligence in fear of job displacement. Though the poster's call to stop AI development is extreme, this experience allowed me to realize the importance of social policies that mitigate harm to workers while maximizing the benefits of AI for workers. After all, as mentioned in the previous paragraph, the trust of the people in artificial intelligence is what will invigorate the further usage of AI among consumers, and thus encourage further developments.

## 3.3 Real World Applications

After learning about counter-AI, a natural thought process that comes to mind for me when dealing with AI models is to identify their potential vulnerabilities. For example, can we fool vision language models like LLaVas by generating adversarial images? View private training data from ChatGPT - large language models? Steal private model information such as model architecture and weights from common AI APIs, and apply white-box attacks based on the stolen information? These thoughts are not only entertaining thought experiments but can also be critical when identifying vulnerabilities of new AI systems that are planned to be deployed in the real world.

One example of a recent innovation that I went through the thought experiment is a smart speaker robot swarm that can mute different spatial areasm developed by a group of researchers at the University of Washington [7]. This swarm microphone employs a neural network that intakes multiple time-shifted audio signals aligned for a given location of interest, where the channel size of the data is the number of robots (number of audio signals). The neural network outputs a speech signal of a target speaker in the input location, essentially muting every other captured audio. Given this neural network, I particularly focused on the possibility of generating adversarial examples if the adversary has white-box access to the model and can edit the input audio signals. Since the neural network depends on the time interval differences of audio from the specified location to arrive at different robot microphones, adding a perturbation at time intervals where the delay occurs would cause the neural network to be unable to locate the targetted speaker. Using adversarial example generation techniques discussed by Carlini & Wagner for the audio domain [3], we could potentially add a small perturbation on the audio such that we can pretend a person is closer to a different robot in the room.

This perturbation would be inaudible to a human ear but could fool the neural network. If this attack is successful, it would be a severe privacy and integrity concern for this robot swarm because it could remove a person from the conversation or listen to another person who would otherwise be muted. This demonstrates that the ability to think critically about real-world AI applications in terms of their vulnerabilities can be crucial, as these risks could be mitigated before deployment. Hence, this ability is one of the most important skills I gained after conducting guided research in counter-AI.

# 4 Conclusion

Through my research on Counter-AI, I explored the two motivation questions from the introduction by specifically studying PGD and APGD evasion attacks and also by studying adversarial training, which defends models against these attacks. I also studied AutoAttack, which provided a standardized evaluation framework for image classification models by providing a comprehensive ensemble of parameter-free attacks(and thus free of human biases). In light of the numerous research efforts in creating adversarially robust models, I encourage the usage of standardized evaluation frameworks such as AutoAttack and also the usage of evaluation metrics described in my table in Section 2.3 for consistency purposes. Beyond gaining insight into Counter-AI that answered my motivation questions, I gained new perspectives that further enriched my academically and socially. Research in Counter-AI not only allowed me to gain a deeper understanding of the material in Deep Learning but also provided me with creative ideas for my final Project. Furthermore, investigating counter-AI and its impact on the world such as through President Joe Biden's executive order allowed me to develop a more mature social view of AI safety regulations and policies to help possibly displaced workers. Finally, by studying counter-AI, I developed critical thinking concerning possible AI vulnerabilities. This is something I would not have even considered if I had just taken Machine Learning/Deep Learning classes, but is crucial for mitigating risks for AI deployment in the real world. All in all, this research experience enriched my education and expanded my perspectives, which further motivates me to continue pursuing research in counter-AI.

# References


[1] Exec. order no. 14110, 88 fed. reg. 75191, 2023. 8

[2] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. Square attack: a query-efficient black-box adversarial attack via random search, 2020. 6

[3] N. Carlini and D. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, 2018. 8

[4] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein. RobustBench: a standardized adversarial robustness benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 6

[5] F. Croce and M. Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack, 2020. 6

[6] F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2206–2216. PMLR, 13–18 Jul 2020. 3, 4

[7] M. Itani, T. Chen, T. Yoshioka, and S. Gollakota. Creating speech zones with self-distributing acoustic swarms. *Nature Communications 14*, 14(5684), 2023. 8

[8] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. 3, 4, 5, 6

[9] A. Oprea and A. Vassilev. Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations. Technical Report NIST Artificial Intelligence (AI) 100-2 E2023 (Draft), National Institute of Standards and Technology, Mar. 2023. 1, 2

[10] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan. Theoretically principled trade-off between robustness and accuracy, 2019. 4