Adversarial Learning

AmirHossein Bagheri
Radmehr Karimian
Zahra Sodagar

July 11, 2023

Abstract

Adversarial training has become an increasingly critical area of research in recent years as deep learning models have become more ubiquitous and have started to play a more significant role in our daily lives. One of the most significant challenges we face with these models is that they are often vulnerable to attacks from adversarial examples, which are input data that has been modified or distorted to deceive the model and produce incorrect results. This can have devastating consequences, particularly in critical applications, such as healthcare, finance, and cybersecurity.

Fortunately, there are several approaches that we can take to defend against these sorts of attacks, and adversarial training is perhaps one of the most effective. This technique involves training the model on adversarial examples to increase its robustness, meaning it becomes better at identifying these examples and producing the correct output.

Despite its effectiveness, there are still some areas where adversarial training could be improved further, and researchers have been working hard to explore these areas in more detail. For example, there have been recent studies exploring ways to make adversarial training more efficient, to reduce the time and computational resources required to train these models. Similarly, other researchers have been developing new methods of generating adversarial examples that are more challenging for models to identify, which would test the improved resilience of models trained using adversarial training techniques.

Although some of these improvements to adversarial training have been proposed and discussed in various forums, they have not yet been explored in-depth through existing reviews or surveys of the field. Therefore, it's essential to continue building our understanding in this area and identifying new ways to enhance the robustness of deep learning models against adversarial examples, ultimately making these models more reliable and trustworthy for real-world applications.

Adversarial Training-generalization error-radmacher complexity

1 Introduction

Adversarial robustness is a critical topic in the field of deep learning, as it addresses a significant issue that modern machine learning systems face. These systems are often vulnerable to adversarial attacks, which are malicious perturbations of the input data designed to deceive the model into making incorrect predictions. This issue is particularly concerning in real-world applications, such as self-driving cars or medical diagnosis systems, where the consequences of incorrect predictions can be dire.

To address this issue, researchers have been working on developing methods for training models that are robust to adversarial attacks. These methods typically involve adding regularization terms to the objective function during training, which encourage the model to be less sensitive to small perturbations in the input. The tutorial serves as a valuable resource for individuals seeking to gain a comprehensive understanding of the topic of adversarial robustness in deep learning. The tutorial combines mathematical presentations and illustrative code examples to provide a broad, hands-on introduction to the topic. The sections are also available as downloadable Jupyter Notebooks, which allows readers to try out and build upon the presented ideas.

While the tutorial attempts to cover most high-level ideas driving research in this area, it is possible that some highly relevant work may be omitted. However, the tutorial serves as a valuable starting point for individuals new to the area and a launching pad for those seeking to pursue the ideas more deeply. Additionally, the

tutorial emphasizes the importance of examining the robustness and reliability of machine learning systems as they are deployed in real systems.

Example Targeted Attack

The goal of the attacker is to force the model to predict a specific target label, rather than just causing the model to make an incorrect prediction.

To achieve this goal, the attacker modifies the input in a way that maximizes the difference between the loss with respect to the true label and the loss with respect to the target label. Mathematically, the targeted adversarial loss function can be written as:

$$\underset{\delta}{\text{maximize}} \left[\ell(h_\theta(x+\delta), y_{\text{target}}) - \ell(h_\theta(x+\delta), y)\right] \tag{1}$$

where ℓ is the loss function, h_θ is the machine learning model, x is the original input, δ is the adversarial perturbation, y is the true label, and y_{target} is the target label.

To solve this optimization problem, the attacker can use a variety of techniques such as gradient ascent or optimization algorithms like L-BFGS. The resulting adversarial perturbation can then be added to the original input to create an adversarial example that will force the model to predict the target label.

Targeted attacks can be more difficult to defend against than untargeted attacks, as they require the model to predict a specific target label rather than just making an incorrect prediction. Defenses against targeted attacks often involve adding additional layers of security such as input validation or detecting outlier inputs.

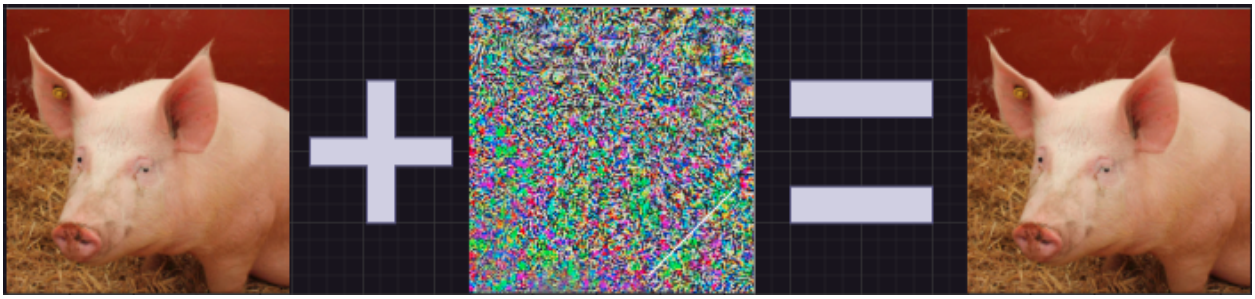

Figure 1: Targeted attack makes model predicts Airline

1.1 Notation

Equation 2 represents the loss function, which measures the difference between the predicted output $h_\theta(x)$ and the true output y:

$$\ell(h_\theta(x), y) \tag{2}$$

Equation 3 represents the empirical risk, which is the average loss over the training set D:

$$\underset{\theta}{\text{minimize}}, \frac{1}{m}\sum_{i=1}^{m} \ell\left(h_\theta\left(x_i\right), y_i\right) \tag{3}$$

Equation 4 represents the update rule for the parameters θ using stochastic gradient descent:

$$\theta := \theta - \frac{\alpha}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla_\theta \ell\left(h_\theta\left(x_i\right), y_i\right) \tag{4}$$

Equation 5 represents the adversarial loss function, which maximizes the loss with respect to a perturbation $\hat{x}$:

$$\underset{\hat{x}}{\text{maximize}}, \ell\left(h_\theta(\hat{x}), y\right) \tag{5}$$

Equation 6 represents the adversarial loss function, which maximizes the loss with respect to a perturbation δ within a bounded set Δ:

$$\underset{\delta\in\Delta}{\text{maximize}}, \ell\left(h_\theta(x+\delta), y\right) \tag{6}$$

Equation 7 defines the set of allowable perturbations for the adversarial loss:

$$\Delta = \delta : |\delta|_p \leq \epsilon \tag{7}$$

Equation 8 represents the adversarial loss function with respect to a bounded perturbation δ:

$$\underset{\delta \in \Delta}{\text{maximize}}, \ell\left(h_\theta(x+\delta), y\right) \tag{8}$$

1.1.1 Risk Minimization

Equation 9 represents the traditional notion of risk as it is used in machine learning, which is the expected loss over the distribution $\mathcal{D}$:

$$R\left(h_\theta\right) = \mathbf{E}(x, y) \sim \mathcal{D}\left[\ell\left(h\theta(x)\right), y)\right] \tag{9}$$

Equation 10 represents the training set D:

$$D = \left(x_i, y_i\right) \sim \mathcal{D}\ , i = 1, \ldots, m \tag{10}$$

Equation 11 represents the empirical risk, which is the average loss over the training set D:

$$\hat{R}(h_\theta) = \frac{1}{m} \sum_{i=1}^{m} \ell(h_\theta(x_i), y_i) \tag{11}$$

Equation 12 represents the regularized risk, which adds a penalty term to the empirical risk to prevent overfitting:

$$\underset{\theta}{\text{minimize}}, \left(\frac{1}{m} \sum_{i=1}^{m} \ell(h_\theta(x_i), y_i) + \lambda \Omega(\theta) \right) \tag{12}$$

where λ is the regularization hyperparameter and $\Omega(\theta)$ is the regularization term, such as L1 or L2 regularization.
Equation 13 represents the stochastic gradient descent update rule for the regularized risk:

$$\theta := \theta - \alpha \nabla_\theta \left(\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \ell(h_\theta(x_i), y_i) + \lambda \Omega(\theta) \right) \tag{13}$$

where α is the learning rate and $\mathcal{B}$ is the mini-batch of size $|\mathcal{B}|$ used in each iteration.
Equation 14 represents the generalization error, which is the expected difference between the true risk and the empirical risk:

$$\text{GenErr}(h_\theta) = |R(h_\theta) - \hat{R}(h_\theta)| \tag{14}$$

where $R(h_\theta)$ is the true risk, which is the expected loss over the distribution $\mathcal{D}$.

1.2 Linear Models

$$h_\theta(x) = Wx + b, \tag{15}$$

where $\theta = W \in \mathbb{R}^{k \times n}, b \in \mathbb{R}^k$.
The robust optimization framework for this hypothesis class, where the perturbation set is a norm ball:

$$\Delta = \delta : |\delta| \leq \epsilon, \tag{16}$$

where the specific norm (e.g. ℓ_∞, ℓ_2, ℓ_1) is not specified. The resulting mix-max problem is formulated as:

$$\underset{W,b}{\text{minimize}} \frac{1}{|D|} \sum_{x,y \in D} \max_{\delta \in \Delta} \ell(W(x+\delta) + b, y), \tag{17}$$

where D is the training set and ℓ is the loss function.

For binary classification, the inner maximization problem can be solved exactly, while for multi-class classification, a relatively tight upper bound can be provided. Furthermore, the resulting minimization problem is convex in θ, allowing for an optimal solution of the robust training procedure.
By noting the importance of understanding the linear case, as it provides insights into the theory and practice of adversarial robustness and its connections to other machine learning methods, such as support vector machines.

1.2.1 Binary classification

Let us first consider the case of binary classification, where k = 2 in the multiclass setting we described earlier. In this case, we use the binary cross entropy, or logistic loss. Our hypothesis function is given by:

$$h_\theta(x) = w^T x + b \tag{18}$$

for $\theta = w \in \mathbb{R}^n, b \in \mathbb{R}$, class label $y \in +1, -1$, and loss function:

$$\ell(h_\theta(x), y) = \log(1 + \exp(-y \cdot h_\theta(x))) \equiv L(y \cdot h_\theta(x)) \tag{19}$$

where we define the function $L(z) = \log(1 + \exp(-z))$ for convenience. The semantics of this setup are such that for a data point x, the classifier predicts class $+1$ with probability:

$$p(y = +1 \mid x) = \frac{1}{1 + \exp(-h_\theta(x))} \tag{20}$$

It is worth noting that if we use the traditional multiclass cross entropy loss with two classes, then the probabilities of predicting class 1 and class 2 are given by the above equation with $y = +1$ and $y = -1$, respectively. We can thus define a single scalar-valued hypothesis:

$$h'_\theta(x) \equiv h_\theta(x)1 - h\theta(x)_2 \tag{21}$$

with the associated probabilities:

$$p(y \mid x) = \frac{1}{1 + \exp(-y \cdot h'_\theta(x))} \tag{22}$$

where y is defined as $y \in +1, -1$ as written above. Taking the negative log of this quantity gives:

$$-\log \frac{1}{1 + \exp(-y \cdot h'_\theta(x))} = \log(1 + \exp(-y \cdot h'_\theta(x))) \tag{23}$$

which is exactly the logistic loss we defined earlier.

1.2.2 Solving the inner maximization problem

Let us now consider the inner maximization problem in the robust optimization problem, which takes the form:

$$\underset{|\delta| \leq \epsilon}{\text{maximize}} \ell(w^T(x + \delta), y) \equiv \underset{|\delta| \leq \epsilon}{\text{maximize}} L(y \cdot (w^T(x + \delta) + b)) \tag{24}$$

The key point to note here is that it is possible to solve this inner maximization problem exactly. Firstly, note that L is a scalar function that is monotonically decreasing and has the following form:

$$L(z) = np \cdot \log(1 + np \cdot \exp(-z)) \tag{25}$$

Since the function is monotonically decreasing, maximizing this function applied to a scalar is equivalent to minimizing the scalar quantity. Therefore, we have:

$$\max_{|\delta| \leq \epsilon} L(y \cdot (w^T(x + \delta) + b)) = L(\min_{|\delta| \leq \epsilon} y \cdot (w^T(x + \delta) + b)) \quad = L(y \cdot (w^T x + b) + \min_{|\delta| \leq \epsilon} y \cdot w^T \delta) \tag{26}$$

To solve the optimization problem $\min_{|\delta| \leq \epsilon} y \cdot w^T \delta$, consider the case where $y = +1$ and the $|\cdot|\infty$ norm constraint $|\delta|\infty \leq \epsilon$. In this case, we minimize the quantity by setting $\delta_i = -\epsilon$ for $w_i \geq 0$ and $\delta_i = \epsilon$ for $w_i < 0$. For $y = -1$, we flip these quantities. The optimal solution to the optimization problem for the $|\cdot|_\infty$ norm is therefore given by:

$$\delta^\star = -y\epsilon \cdot \text{sign}(w) \tag{27}$$

We can also determine the function value achieved by this solution:

$$y \cdot w^T \delta^\star = -y^2 \epsilon \sum_i |w_i| = -\epsilon |w|_1 \tag{28}$$

Thus, we can analytically compute the solution of the inner maximization problem:

$$\max_{|\delta|\infty \leq \epsilon} L(y \cdot (w^T(x + \delta) + b)) = L(y \cdot (w^T x + b) - \epsilon |w|1) \tag{29}$$

Therefore, we can convert the robust min-max problem to a pure minimization problem:

$$\underset{w,b}{\text{minimize}} \frac{1}{D} \sum_{(x,y) \in D} L(y \cdot (w^T x + b) - \epsilon |w|_1) \tag{30}$$

This problem is still convex in w, b and can be solved exactly or using SGD to approach the globally optimal solution. More generally, it turns out that the optimization problem in robust optimization can be solved using similar techniques.
To solve the optimization problem $\min_{|\delta| \leq \epsilon} y \cdot w^T \delta$, we can use the dual norm $|\cdot|*$ of our original norm bound on $\theta(|\cdot|p)$ and $|\cdot|_q$, where $1/p + 1/q = 1$. The solution is given by:

$$\min_{|\delta| \leq \epsilon} y \cdot w^T \delta = -\epsilon |w|_* \tag{31}$$

This means that regardless of our norm constraint, we can solve the robust optimization problem via a single minimization problem and find the analytical solution to the worst-case adversarial attack, without the need to explicitly solve a min-max problem. Note that the final robust optimization problem, adopting the general form, is as follows:

$$\underset{w,b}{\text{minimize}} \frac{1}{D} \sum_{(x,y) \in D} L(y \cdot (w^T x + b) - \epsilon |w|_*) \tag{32}$$

This looks very similar to the typical norm-regularized objective commonly considered in machine learning, with the exception that the regularization term is inside the loss function. This means that in the robust optimization case, if a point is far from the decision boundary, we don't penalize the norm of the parameters, but we do penalize the norm of the parameters (transformed by the loss function) for a point where we are close to the decision boundary. The connections between such formulations and support vector machines have been extensively studied.

1.3 Solving MiniMax

Neural Networks [Bai+21] are especially prone to adversarial examples due to the nature of their loss surfaces. To solve the inner maximization problem:

$$\underset{|\delta| \leq \epsilon}{\text{maximize}}, \ell\left(h_\theta(x), y\right) \tag{33}$$

where ℓ is the loss function, there are three main strategies:

1. Lower bounds (Only focus on this)
2. Exact solutions
3. Upper bounds

Each strategy has its own advantages and disadvantages, and the choice of strategy depends on the specific problem and resources available.

1.3.1 Lower Bound

By definition, any feasible δ will give us a lower bound for the inner maximization problem. This method is equivalent to "trying to empirically solve the optimization problem" or "find an adversarial example". It is the most common strategy for solving the inner maximization problem, motivated largely by the fact that for neural networks in general, problems of local optima don't seem as bad as initially thought [GSS15].
The idea is quite simple: using backpropagation, we can compute the gradient of the loss function with respect to the perturbation δ itself. Thus, we can perform gradient descent on δ to maximize our objective. However, we also need to ensure that δ stays within the norm bound ϵ. Thus, after each step, we can project δ back into this space [Mad+19].

1.3.2 Fast Gradient Sign Method

The Fast Gradient Sign Method (FGSM) [GSS15] suggests that for a given example x, we adjust δ in the direction of its gradient. We first compute the gradient:

$$g := \nabla_\delta \ell \left(h_\theta(x + \delta), y \right) \tag{34}$$

for some step size α, and then project it back into the norm ball defined by $|\delta| \leq \epsilon$. For the particular case of the ℓ_∞ norm $|\delta|_\infty \leq \epsilon$, projecting onto this norm ball simply involves clipping values of δ to lie within the range $[-\epsilon, \epsilon]$:

$$\delta^t := \delta^{t-1} + \text{clip}(\alpha g, [-\epsilon, \epsilon]) \tag{35}$$

However, doing it until convergence is slow, so we estimate the sign of the gradient with only one step:

$$\delta := \epsilon \cdot \text{sign}(g) \tag{36}$$

Catastrophic overfitting can occur with FGSM [KM21; KLL20].

1.3.3 Projected Gradient Descent (PGD)

The more powerful adversary inspired by PGD [Mad+17] is the multi-step variant, which is essentially projected gradient descent (PGD) on the negative loss function. The basic PGD algorithm simply iterates the updates:

$$\delta := \mathcal{P} \left(\delta + \alpha \nabla_\delta \ell \left(h_\theta(x + \delta), y \right) \right) \tag{37}$$

where $\mathcal{P}$ denotes the projection onto the ball of interest (for example, clipping in the case of the ℓ_∞ norm). At the initial zero point, we need to scale it by a relatively large α to make any progress at all. Once we "break out" of the initial region around $\delta = 0$, the gradients typically increase in magnitude substantially, and at this point our large α is too large, and the method takes too big a step toward the boundary.

1.3.4 Steepest Descent in Adversarial Attacks and Defenses

Steepest Descent is a commonly used optimization algorithm in the context of adversarial attacks and defenses in machine learning. Adversarial attacks aim to find input perturbations that can cause the model to misclassify the input, while adversarial defenses aim to improve the robustness of the model against such attacks [Mad+17; Pap+16; GSS15].
In the context of adversarial attacks, Steepest Descent can be used to compute the direction of the input perturbation that maximizes the model's loss function. This direction can then be used to generate adversarial examples by adding a small perturbation to the original input in the opposite direction of the gradient. The resulting adversarial example can then be used to test the robustness of the model or to perform targeted attacks [GSS15].

In the context of adversarial defenses, Steepest Descent can be used to compute the direction of the input perturbation that minimizes the model's loss function, subject to a constraint on the size of the perturbation. This direction can then be used to generate a perturbation that can be added to the input to improve its robustness against adversarial attacks [Mad+17; Pap+16].
However, Steepest Descent can be computationally expensive, especially for high-dimensional input spaces, and may require careful tuning of the step size and other parameters to ensure convergence. Therefore, other optimization algorithms such as Projected Gradient Descent and Adam have also been used for adversarial attacks and defenses in machine learning [Mad+17; Pap+16].

1.4 A Taxonomy for Adversarial Training in Deep Learning

Adversarial training is one of the most effective approaches to defending deep learning models against adversarial examples. A variety of improvements and developments of adversarial training are proposed, in which each address the generalization problems in adversarial training from three perspectives:
Standard Generalization. Adversarial training, while improving the robustness of neural networks to adversarial attacks negatively affects their standard accuracy. Some researchers argue that there is a trade-off between adversarial robustness and standard accuracy, showing a negative correlation between model classification accuracy and model robustness. However, other studies suggest that adversarial robustness and standard accuracy are not opposing but can coexist. They highlight the existence of adversarial examples on the manifold of natural data and the potential for robust and accurate classifiers. Different training methods have been proposed to address this trade-off, such as adaptive ϵ, robust local features, and L1 penalty, which have shown better standard generalization in empirical evaluations.
Adversarially Robust Generalization. Adversarially trained models exhibit poor performance on adversarially perturbed test data, indicating a significant gap between training accuracy and test accuracy. This phenomenon is referred to as adversarially robust generalization. Researchers have observed this overfitting in various datasets and have made efforts to improve generalization empirically through techniques such as semi/unsupervised learning, AVmixup, and robust local features. Different tools, such as Rademacher complexity and VC dimension, have been used to analyze the generalization problem, but theoretical progress has been limited, and the problem remains unsolved. Early stopping is found to be the most effective technique for improving generalization.
Generalization on Unseen Attacks. Adversarial training methods often fail to generalize to unseen attacks beyond the specific type of attack they were trained on. It is proven that a specific type of attack is not sufficient to represent the full space of possible perturbations. Adversarially trained models, which are robust to a specific attack, can be easily circumvented by different types of attacks or variations in attack parameters. Combining perturbations with different norms in adversarial training has been found to be ineffective. This poor generalization to other attacks undermines the reliability of adversarial training. The limitation arises from how the inner maximization problem is handled in adversarial training. Various approaches have been proposed to approximate optimal solutions to the inner problem, such as increasing the diversity of targeted models, modeling the distribution of adversarial examples, and calibrating confidence scores during training. However, the generalization problem on unseen attacks is still not extensively studied, partly due to the limited understanding of adversarial examples and the need for further research efforts.
Now we review the recent advances of adversarial training in last few years, categorized by different understandings on adversarial training.

1.4.1 Adversarial Regularization

Adversarial regularization is a flexible approach that requires understanding of adversarial robustness. It offers the potential to enhance robustness using unlabeled data through the decomposition of robust error. The basis of these methods is the fact that more robust models usually have smaller values of local linearity, and the linearity of neural networks is attributed to the existence of adversarial examples. [GSS15] added a regularization term based on FGSM:

$$L(\theta, x + \varepsilon sign(\nabla_x L(\theta, x, y))) \tag{38}$$

[Zha+19] decomposed the robust error into natural and boundary errors. They introduced TRADES to minimize the boundary error (a local linearity regularization), outperforming PGD-AT:

$$\min_f \mathbb{E}\left\{\mathcal{L}(f(x), y) + \max_{x' \in \mathbb{B}(x,\epsilon)} \mathcal{L}\left(f(x), f\left(x'\right)\right)/\lambda\right\} \tag{39}$$

[Wan+19b] addressed the issue of misclassified examples with Misclassification Aware adveRsarial Training (MART). This method emphasizes on misclassified examples with weights of $1 - P_y(x, \theta)$
Deep models amplify imperceptible noises, affecting the feature space. Kannan et al. (2018) proposed Adversarial Logit Pairing (ALP), but its formulation was incorrect. [Mao+19] used triplet loss for aligning natural and adversarial representations. Both methods are based on encouraging logits for pairs of examples to be similar and the alignment of representations of natural data and their adversarial counterparts.

1.4.2 Curriculum-based Adversarial Training

Adversarial examples generated by strong attacks significantly cross over the decision boundary, this leads to overfitting of adversarial examples, so the worst-case samples are not always suitable for adversarial training. We can improve the generalization of clean data while preserving adversarial robustness by using weaker attacks in early training. This problem leads to overfitting of adversarial examples in PGD-AT (Projected Gradient Descent Adversarial Training).
Curriculum Adversarial Training (CAT) [Cai+18], gradually increasing the iteration steps of PGD to generate stronger adversarial examples. Friendly Adversarial Training (FAT) [Zha+20], employs early stopping in PGD attacks and uses adversarial data near the decision boundary for training. Both CAT and FAT adjust the strength of attacks in a practical manner, although a quantitative criterion is missing. Instead of ad-hoc iterative methods, First-Order Stationary Condition (FOSC) [Wan+21] was introduced to estimate the convergence quality of the inner maximization problem, with a lower FOSC indicating a stronger attack.

1.4.3 Ensemble Adversarial Training

Ensemble Adversarial Training (EAT) was introduced by [Tra+17] as a method that augments training data with adversarial examples generated from multiple target models instead of just one. EAT helps mitigate the sharp curvatures caused by single-step attacks like FGSM (Fast Gradient Sign Method). However, it neglects the interaction among different target models, which can be problematic since standardly trained models often have similar predictions or representations and share the adversarial subspace.
There are variations of this method since similar predictions or representations and share the adversarial subspace:

- Forcing different models to be diverse in non-maximal predictions.
- Maximizing the cosine distances among each target models' input gradients.
- Maximizing the vulnerability diversity (by measuring the sum of losses for two models using crafted images containing non-robust features).

Ensemble methods in adversarial training, are valuable for approximating the optimal solution of the inner maximization problem, but exhibit better generalization abilities regardless of the type of perturbations. In conclusion, increasing the number and diversity of target models in training is a practical and effective approach for approximating the space of adversarial examples.

1.4.4 Adversarial Training with Adaptive ϵ

In adversarial training, the parameters of attacks, such as ϵ (perturbation magnitude), are typically predefined and fixed for all data points during training. However, some studies argue that individual data points may have different intrinsic robustness, with varying distances to the classifier's decision boundary. Adversarial training with fixed ϵ treats all data equally, disregarding their individual characteristics of adversarial robustness.

To address the individual characteristic of adversarial robustness, researchers propose adversarial training at the instance level. Instance Adaptive Adversarial Training (IAAT) [BGH19] selects ϵ to be as large as possible, ensuring that images within the ϵ-ball of a data point belong to the same class.
Another approach called Margin Maximization Adversarial Training (MMA) directly maximizes the margin-distances between data points and the model's decision boundary, estimated using adversarial perturbations with the least magnitudes. MMA's choice of a small ϵ is more reasonable as small ϵ in the spatial domain minimally changes the classes of images, especially for high-resolution images. Customized Adversarial Training (CAT) further incorporates adaptive label uncertainty to prevent over-confident predictions based on adaptive ϵ.
Adversarial training with adaptive ϵ is an interesting exploration. However, empirical evidence indicates that many standard datasets exhibit distributional separation, meaning that the distances between classes are larger than the ϵ used for attacks. This highlights the limitations of current adversarial training methods in finding appropriate decision boundaries.

1.4.5 Adversarial Training with Semi/Unsupervised Learning

Supervised adversarial training methods exhibit a significant generalization gap between training and testing, limiting their effectiveness, however collecting labeled data can be expensive. As an alternative, recent works have investigated the use of additional unlabeled data to improve adversarial training.
Several studies have theoretically shown that incorporating unlabeled data reduces the sample complexity gap between standard and adversarial training. These approaches decompose adversarial robustness and utilize unlabeled data for stability while relying on labeled data for classification. Empirical investigations have examined factors such as label noise, distribution shift, and the amount of additional data to understand their impact on adversarial training and complexity measures for theoretical analysis, and self-supervised training has been observed to benefit adversarial robustness.
While the use of additional unlabeled data shows promise in improving adversarial robustness, precise theoretical or empirical guarantees regarding the required amount of additional data are still lacking. Leveraging unlabeled data has the potential to enhance adversarial training and bridge the gap between standard and adversarial performance.

2 Generalization Properties

2.1 Rademacher Complexity

Theorem 1 [YRB20]: Suppose that the range of $\ell(f(\mathrm{x}), y)$ is $[0, B]$. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$, the following holds for all $f \in \mathcal{F}$:

$$[R(f) \leq R_n(f) + 2B\mathbb{E}\sigma \left[\sup f \in \mathcal{F} \frac{1}{n} \sum_{i=1}^{\mathcal{S}} \sigma_i \ell(f(\mathbf{x}_i), y_i) \right] + 3B\sqrt{\frac{\log(2/\delta)}{2n}}.]$$

Define the adversarial loss $\widetilde{\ell}(f(\mathbf{x}), y) := \max_{\mathbb{B}\mathbf{x}^{\infty}(\boldsymbol{\epsilon})} \ell\left(f\left(\mathbf{x}'\right), y\right)$ and the function class $\widetilde{\ell}\mathcal{F} \subseteq [0, B]^{\mathcal{X}\times\mathcal{Y}}$ as $\widetilde{\ell}_{\mathcal{F}} = \widetilde{\ell}(f(\mathbf{x}), y) : f \in \mathcal{F}$.
Corollary 1: Suppose that the range of $\ell(f(\mathrm{x}), y)$ is $[0, B]$. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$, the following holds for all $f \in \mathcal{F}$:

$$[\widetilde{R}(f) \leq \widetilde{R}n(f) + 2B\mathbb{E}\sigma \left[\sup_{f\in\mathcal{F}} \frac{1}{n} \sum_{i=1}^{\mathcal{S}} \sigma_i \widetilde{\ell}(f(\mathbf{x}_i), y_i) \right] + 3B\sqrt{\frac{\log(2/\delta)}{2n}}.]$$

Rademacher complexity of the adversarial loss function class is the Keyquantity for the generalization ability of the learning problem which highlights the importance of considering the worst-case perturbations in the training data when analyzing the robustness and generalization ability of machine learning models in adversarial settings.

2.1.1 Binary Classification

Rademacher complexity of Binary Linear classifier follows from new setting that you set labels -1 and 1 then problem convert to below:
theorem 2: Let $\mathcal{F} := f_{\mathbf{w}}(\mathbf{x}) : |\mathbf{w}|p \leq W$ and $\widetilde{\mathcal{F}} := \min \mathbf{x}' \in \mathbb{B}\mathbf{x}^{\infty}(\epsilon) y \langle \mathbf{w}, \mathbf{x}' \rangle : |\mathbf{w}|p \leq W$. Suppose that $\frac{1}{p} + \frac{1}{q} = 1$. Then, there exists a universal constant $c \in (0, 1)$ such that

$$\max \mathfrak{RS}(\mathcal{F}), c\epsilon W \frac{d^{\frac{1}{q}}}{\sqrt{n}} \leq \mathfrak{RS}(\widetilde{\mathcal{F}}) \leq \mathfrak{RS}(\mathcal{F}) + \epsilon W \frac{d^{\frac{1}{q}}}{\sqrt{n}}.$$

The adversarial Rademacher complexity, $\mathfrak{RS}(\widetilde{\mathcal{F}})$ is always at least as large as the Rademacher complexity in the natural setting. This implies that uniform convergence in the adversarial setting is at least as hard as that in the natural setting. In addition, since $\max a, b \geq \frac{1}{2}(a + b)$, we have

$$\frac{c}{2}\left(\mathfrak{RS}(\mathcal{F}) + \epsilon W \frac{d^{\frac{1}{q}}}{\sqrt{n}}\right) \leq \mathfrak{RS}(\widetilde{\mathcal{F}}) \leq \mathfrak{R}_{\mathcal{S}}(\mathcal{F}) + \epsilon W \frac{d^{\frac{1}{q}}}{\sqrt{n}}.$$

Theorem 2 provides a bound on the adversarial Rademacher complexity of a binary linear classifier. The adversarial Rademacher complexity measures the classifier's ability to make accurate predictions on adversarial examples, which are examples that have been intentionally modified to mislead the classifier. The bound says that the adversarial Rademacher complexity of a binary linear classifier is at least as large as its natural Rademacher complexity, which measures its ability to make accurate predictions on randomly sampled examples. This means that it is harder to achieve uniform convergence in the adversarial setting than in the natural setting, as the classifier needs to perform well on a wider range of examples.
On the other hand, the bound also shows that the generalization error of the binary linear classifier will asymptotically decrease as the number of training examples n increases, and the dimensionality d of the input space decreases. Specifically, the bound says that the adversarial Rademacher complexity of the binary linear classifier scales as $\frac{d^{\frac{1}{q}}}{\sqrt{n}}$, where q is the conjugate exponent of p in the assumption $\frac{1}{p} + \frac{1}{q} = 1$. As n increases or d decreases, the complexity of the classifier decreases, which means that it becomes easier to achieve uniform convergence and the generalization error decreases. The bound also shows that the generalization error can be further reduced by increasing the regularization parameter W or decreasing the maximum allowed perturbation ϵ in the adversarial examples.
In summary, Theorem 2 provides a theoretical guarantee that a binary linear classifier can achieve good generalization performance in the presence of adversarial examples, as long as the number of training examples is sufficiently large and the regularization parameter is properly chosen.

2.1.2 Multi-Class Classification

In the multi-class classification setting, a linear classifier maps an input vector $\mathbf{x}$ to a vector of scores $\mathbf{z} = (z_1, z_2, \ldots, z_K)$, where K is the number of classes. The predicted class label is the index of the highest score in the vector $\mathbf{z}$. That is,

$$\hat{y} = \arg \max_{y \in [K]} z_y.$$

The margin parameter γ measures the difference between the score of the true class label y and the maximum score over the other class labels. Specifically, for a given input vector $\mathbf{x}$ and a true class label y, the margin operator $M(\mathbf{z}, y)$ is defined as

$$M(\mathbf{z}, y) = z_y - \max_{y' \neq y} z_{y'}.$$

The margin parameter γ is the minimum margin over all training examples, that is,

$$\gamma = \min_{i=1}^{n} M(\mathbf{z}_i, y_i),$$

where $(\mathbf{x}_i, y_i)$ is the i-th training example.
The margin parameter plays a crucial role in the generalization error bound for multi-class linear classifiers in the presence of adversarial examples. Intuitively, a larger margin implies a greater degree of separation

between the scores of the true class label and the scores of the other labels. This makes it more difficult for an attacker to construct adversarial examples that fool the classifier. Therefore, the generalization error bound for multi-class linear classifiers in the presence of adversarial examples improves as the margin parameter γ increases.

Corollary 2 Consider the multi-class classification setting. For any fixed $\gamma > 0$, we have, with probability at least $1 - \delta$, for all $f \in \mathcal{F}$,

$$\mathbb{P}(\mathbf{x}, y) \sim \mathcal{D} y \neq \arg\max y' \in [K][f(\mathbf{x})]y' \leq$$

$$\frac{1}{n} \sum i = 1^{n} \mathbb{1} \left([f\left(\mathbf{x} i\right)] y_i \leq \gamma + \max y' \neq y \left[f\left(\mathbf{x} i\right)\right] y' \right) + 2\Re\mathcal{S}\left(\ell_{\mathcal{F}}\right) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2n}},$$

where γ is the least margin in the margin operator for true label $M(\mathbf{z}, y) : \mathbb{R}^K \times [K] \to \mathbb{R}$ as $M(\mathbf{z}, y) = z_y - \max_{y' \neq y} z_{y'}$.

This corollary provides a bound on the generalization error for multi-class linear classifiers in the presence of adversarial examples. The bound guarantees that the probability of misclassification on a test example drawn from the distribution $\mathcal{D}$ is small, with high probability, for any classifier f in the hypothesis class $\mathcal{F}$. The bound depends on the margin parameter γ and the Rademacher complexity of the hypothesis class. The margin parameter measures the distance between the score of the true label and the scores of the other labels. The bound shows that the probability of misclassification decreases as the margin parameter increases, and the Rademacher complexity of the hypothesis class decreases.

We can see the Emprical study also align with theoretical part

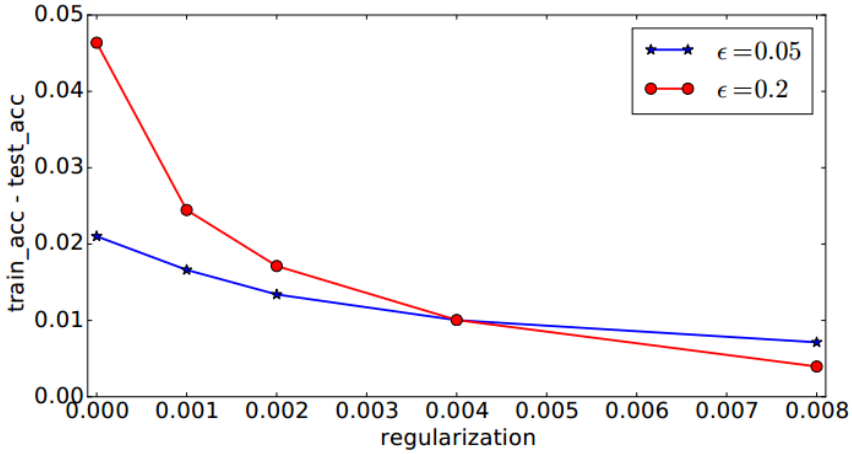

Figure 2: Adversarial generalization error vs regularization coefficient λ for a four-layer ReLU neural network with two convolutional layers and two fully connected layers. The results show that ℓ_1 regularization can effectively reduce the adversarial generalization error under the PGD attack.

$$\min_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^{n} \max_{\mathbf{x}_i' \in \mathbb{B}_{\mathbf{x}_i}^{\mathbf{x}}(\epsilon)} \ell\left(f_{\mathbf{W}}\left(\mathbf{x}_i'\right), y_i\right) + \lambda \|\mathbf{W}\|_1$$

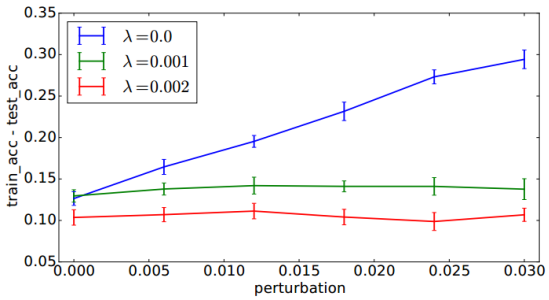

Figure 3: Adversarial generalization error vs ℓ_∞ perturbation ϵ and regularization coefficient λ.

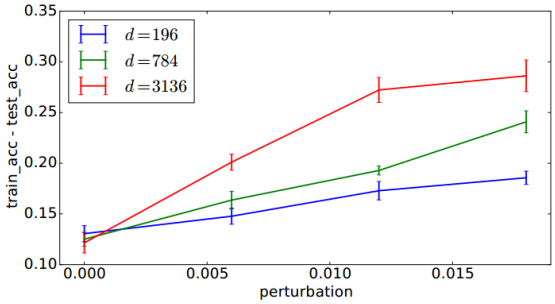

Figure 4: Adversarial generalization error vs ℓ_∞ perturbation ϵ and dimension of feature space d.

Figure 5: $\min_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^{n} \max_{\mathbf{x}_i' \in \mathbb{B}_{\mathbf{x}_i}^{\mathbf{x}}(\epsilon)} \ell\left(f_{\mathbf{W}}\left(\mathbf{x}_i'\right), y_i\right) + \lambda \|\mathbf{W}\|_1$

2.2 Trade Off of Adversarial Risk & Standard Risk

Let's take a new approach [JSH20] instead of only optimizing the adversarial risk, we can combine the clean and noisy risks to find a better trade-off between accuracy and robustness. This new approach to adversarial training involves minimizing a combined risk function that takes into account both the standard and adversarial risks. By finding the optimal trade-off between these two risks, we can improve the overall performance of the linear regression model. The authors of "Precise Trade-offs in Adversarial Training for Linear Regression" propose a method for achieving this by minimizing an objective function that includes both the combined risk function and a regularization term. This approach has been shown to achieve better trade-offs between accuracy and robustness on several benchmark datasets compared to existing methods for adversarial training for linear regression.

$$y_i = \langle \boldsymbol{x}_i, \boldsymbol{\theta}_0 \rangle + w_i, \quad w_i \sim \mathrm{N}(0, \sigma_0^2), \tag{40}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{\theta}_0$ are the input vector and the true parameter vector, respectively.
The adversarial training problem is formulated as:

$$\widehat{\boldsymbol{\theta}}^{\varepsilon} \in \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^p} \max_{|\boldsymbol{\delta} i|_{\ell_2} \leq \varepsilon} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \langle \boldsymbol{x}_i + \boldsymbol{\delta}_i, \boldsymbol{\theta} \rangle\right)^2, \tag{41}$$

where ε is the maximum allowed perturbation size.
The standard risk (SR) and the adversarial risk (AR) are defined as:

$$\mathrm{SR}(\widehat{\boldsymbol{\theta}}) = \mathbb{E}(y - \langle \boldsymbol{x}, \widehat{\boldsymbol{\theta}} \rangle)^2, \tag{42}$$

$$\mathrm{AR}(\widehat{\boldsymbol{\theta}}) = \mathbb{E} \max_{|\boldsymbol{\delta}|_{\ell_2} \leq \varepsilon \mathrm{test}} (y - \langle \boldsymbol{x} + \boldsymbol{\delta}, \widehat{\boldsymbol{\theta}} \rangle)^2, \tag{43}$$

where $\varepsilon_{\mathrm{test}}$ is the maximum perturbation size during testing.
The Pareto optimal curve is obtained by minimizing a weighted combination of these two accuracies for different values of the weight λ:
Solving it will reaches to The solution $\boldsymbol{\theta}^\lambda$ of the optimization problem is given by $\boldsymbol{\theta}^\lambda = \left(1 + \gamma_0^\lambda\right)^{-1} \boldsymbol{\theta}_0$, with γ_0^λ the fixed point of the following two equations:

$$\gamma_0^\lambda = \frac{\varepsilon_{\mathrm{test}}^2 + \sqrt{\frac{2}{\pi}} \varepsilon_{\mathrm{test}} A^\lambda}{1 + \lambda + \sqrt{\frac{2}{\pi}} \frac{\varepsilon_{\mathrm{test}}}{A^\lambda}} \qquad A^\lambda = \frac{1}{|\boldsymbol{\theta} 0|_{\ell_2}} \left(\left(1 + \gamma_0^\lambda\right)^2 \sigma_0^2 + \left(\gamma_0^\lambda\right)^2 |\boldsymbol{\theta} 0|_{\ell_2}{}^2\right)^{1/2}.$$

The optimal trade-off between the standard risk and the adversarial risk can be interpreted as the **margin of the linear regression model**. This is similar to the margin in support vector machines (SVM), where the

margin is defined as $\frac{1}{|\boldsymbol{\theta}|}$, with $\boldsymbol{\theta}$ being the weight vector of the SVM. In the case of our linear regression model, paying more attention to the adversarial risk will lead to a larger margin, while paying more attention to the standard risk will lead to a smaller margin. This trade-off between margin and accuracy is a fundamental concept in machine learning and is crucial for achieving good performance on real-world datasets.

2.3 A Minimax Approach With Wasserstein Distance

Instead of dealing with adversarial risk we define new distribution of data and deal with that [TZT19]. The method begins by pushing forward the original distribution P into a new distribution P' which is dependent on h using a transport map $T_h : \mathcal{Z} \to \mathcal{Z}$ satisfying

$$R_P(h, \mathcal{B}) = R_{P'}(h),$$

where $R_P(h, \mathcal{B})$ is the risk of hypothesis h under the adversarial radius $\mathcal{B}$, and $R_{P'}(h)$ is the risk of hypothesis h under the distribution P'.
Next, the method defines T_h as

$$z = (x, y) \to (x^*, y),$$

where $x^* = \arg\max_{x' \in N(x)} l(h(x'), y)$, and $N(x)$ is a neighborhood of x.
The method then defines the radius of the adversary $\mathcal{B}$ as $\epsilon_{\mathcal{B}} := \sup_{x \in \mathcal{B}} d_{\mathcal{X}}(x, 0)$, where $d_{\mathcal{X}}(x, 0)$ is the distance between x and the origin in the input space $\mathcal{X}$. For any hypothesis h and the corresponding $P' = T_h \# P$, the method shows that

$$W_p(P, P') \leq \epsilon_{\mathcal{B}},$$

where W_p is the Wasserstein distance between distributions.
Finally, the method derives the bound

$$R_P(h, \mathcal{B}) \leq R_{\epsilon_{\mathcal{B}},1}(P, h), \quad \forall h \in \mathcal{H},$$

where $R_{\epsilon_{\mathcal{B}},1}(P, h)$ is a modified risk that takes into account the adversarial radius $\mathcal{B}$.
Now with three assumptions we can reach new bound for risk which includes Duldly integration we will explain how in the two example this will be helpful and its effect. For any upper semicontinuous function $f : \mathcal{Z} \to \mathbb{R}$ and for any $P \in \mathcal{P}_p(\mathcal{Z})$
$R_{\epsilon_{\mathcal{B}},1}(P, f) = \min_{\lambda \geq 0} \{\lambda \epsilon_{\mathcal{B}} + \mathbb{E}_P[\varphi_{\lambda,f}(z)]\}$, where $\varphi_{\lambda,f}(z) := \sup_{z' \in \mathcal{Z}} \{f(z') - \lambda \cdot d_{\mathcal{Z}}(z, z')\}$
The assumptions made in this analysis are:

1. The instance space $\mathcal{Z}$ is bounded: $\operatorname{diam}(\mathcal{Z}) := \sup_{z, z' \in \mathcal{Z}} d\mathcal{Z}(z, z') < \infty$
2. The functions in $\mathcal{F}$ are upper semicontinuous and uniformly bounded: $0 \leq f(z) \leq M < \infty$ for all $f \in \mathcal{F}$ and $z \in \mathcal{Z}$
3. For any function $f \in \mathcal{F}$ and any $z \in \mathcal{Z}$, there exists a constant λ such that $f(z') - f(z) \leq \lambda d_{\mathcal{Z}}(z, z')$ for any $z' \in \mathcal{Z}$

Theorem Define the function class $\Phi := \{\varphi_{\lambda,f} : \lambda \in [a, b], f \in \mathcal{F}\}$ where $b \geq a \geq 0$. Then, the expected Rademacher complexity of the function class Φ satisfies

$$\Re_n(\Phi) \leq \frac{12\mathfrak{C}(\mathcal{F})}{\sqrt{n}} + \frac{6\sqrt{\pi}}{\sqrt{n}}(b - a) \cdot \operatorname{diam}(Z)$$

where $\mathfrak{C}(\mathcal{F}) := \int_0^\infty \sqrt{\log \mathcal{N}(\mathcal{F}, \|\cdot\|_\infty, u/2)} du$ and $\mathcal{N}(\mathcal{F}, \|\cdot\|_\infty, u/2)$ denotes the covering number of $\mathcal{F}$. If the assumptions $[1 - 3]$ hold, then for any $f \in \mathcal{F}$, we have below with probability at least $1 - \delta$.

$$\begin{aligned} R_{e_B,1}(P, f) - R_{e_B,1}(P_n, f) \leq & \frac{24\mathfrak{C}(\mathcal{F})}{\sqrt{n}} + M\sqrt{\frac{\log\left(\frac{1}{6}\right)}{2n}} + \\ & \frac{12\sqrt{\pi}}{\sqrt{n}} \Lambda_{e_B} \cdot \operatorname{diam}(Z) \end{aligned}$$

So the key factor here is $\mathfrak{C}(\mathcal{F}) := \int_0^\infty \sqrt{\log \mathcal{N}(\mathcal{F}, \|\cdot\|_\infty, u/2)}du$ for example for these two cases this is computable but for Deep model is it not.

Example

PCA:

$$R_P(f, \mathcal{B}) \leq \frac{1}{n}\sum_{i=1}^{n} f(z_i) + \lambda^+_{f,P_n}\epsilon_{\mathcal{B}} + \frac{576B^2k\sqrt{m}}{\sqrt{n}} + \frac{24B\sqrt{\pi}}{\sqrt{n}}\Lambda_{\epsilon_{\mathcal{B}}} + B^2\sqrt{\frac{\log(1/\delta)}{2n}}$$

SVM:

$$R_P(f, \mathcal{B}) \leq$$

$$\frac{1}{n}\sum_{i=1}^{n} f(z_i) + \lambda^+_{f,P_n}\epsilon_{\mathcal{B}} + \frac{144}{\sqrt{n}}\Lambda r\sqrt{d} + \frac{12\sqrt{\pi}}{\sqrt{n}}\Lambda_{\epsilon_{\mathcal{B}}}\cdot(2r+1) + (1+\Lambda r)\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2n}}$$

3 Stability of Adversarial training

3.1 Algorithmic Stability of Adversarial Training

Adversarial training To introduce adversarial training, let l denote the loss function and $f_\theta(x)$ be the model with parameter θ. The (population) adversarial loss is defined as

$$R(\theta, \epsilon) := \mathbb{E}\left[l\left(f_\theta\left[x + A_\epsilon\left(f_\theta, x, y\right)\right], y\right)\right],$$

where A_ϵ is an attack of strength $\epsilon > 0$ and intends to deteriorate the loss in the following way

$$A_\epsilon\left(f_\theta, x, y\right) := \underset{z\in B_p(0,\epsilon)}{\operatorname{argmax}}\left\{l\left(f_\theta(x+z), y\right)\right\},$$

where $B_p(x, r)$ is a $\mathcal{L}_p$ ball centering at x with radius r.

Given n i.i.d. samples $S = \{(x_i, y_i)\}_{i=1}^n$, the adversarial training minimizes the sample version of $R(\theta, \epsilon)$ w.r.t. θ :

$$R_S(\theta, \epsilon) = \frac{1}{n}\sum_{i=1}^{n} l\left(f_\theta\left[x_i + A_\epsilon\left(f_\theta, x_i, y_i\right)\right], y_i\right)$$

and the estimator $\widehat{\theta}$ aims to minimize $R_S(\theta, \epsilon)$. We rewrite $R_S(\theta, \epsilon)$ as $R_S(\theta)$ for simplicity when there is no confusion.

The minimization in upper problem is often implemented through an iterative two-step (min-max) update. In the t-th iteration, we calculate the adversarial sample $\widetilde{x}_i^{(t)} = x_i + A_\epsilon\left(f_{\theta^{(t)}}, x_i, y_i\right)$ based on the current $\theta^{(t)}$, and then update $\theta^{(t+1)}$ based on the gradient of the adversarial training loss while fixing $\widetilde{x}_i^{(t)}$'s with learning rate η_t. The algorithm runs for T iterations. A more detailed pseudocode is postponed to Algorithm 1 when introducing our adaptations. Note that for some loss function l or model f_θ, there may not be an analytic form for A_ϵ (e.g. deep neural networks), and numerical methods, e.g. FGM and PGD, are utilized to approximate A_ϵ.

Risk decomposition Define θ_0 and $\bar{\theta}$ as the minimizer of R and R_S respectively. Then for the algorithm output $\widehat{\theta}$, the excess risk can be decomposed into four parts as below:

$$R(\widehat{\theta}) - R\left(\theta_0\right) = \underbrace{R(\widehat{\theta}) - R_S(\widehat{\theta})}_{\mathcal{E}_{gen}} + \underbrace{R_S(\widehat{\theta}) - R_S(\bar{\theta})}_{\mathcal{E}_{opt}} + \underbrace{R_S(\bar{\theta}) - R_S\left(\theta_0\right)}_{\leq 0} + \underbrace{R_S\left(\theta_0\right) - R\left(\theta_0\right)}_{\mathbb{E}=0},$$

Since the last two parts are either negative or with zero expectation, we mainly focus on the first two parts, namely, generalization error $R(\widehat{\theta}) - R_S(\widehat{\theta})$ and optimization error $R_S(\widehat{\theta}) - R_S(\bar{\theta})$. Based on [HRS15], $\mathcal{E}_{\text{gen}}$ is upper bounded by algorithmic stability.

Uniform argument stability (UAS) UAS aims to quantify the output sensitivity in $\mathcal{L}_2$ norm w.r.t an arbitrary change in a single data point. An algorithm is λ-UAS if for neighboring datasets $S_1 \sim S_2$ (i.e., S_1 and S_2 differ only in a single data point), it satisfies that

$$\sup_{S_1 \sim S_2} \left\| \widehat{\theta}(S_1) - \widehat{\theta}(S_2) \right\| := \sup_{S_1 \sim S_2} \lambda(S_1, S_2) \leq \lambda.$$

where $\|\cdot\|$ represents the $\mathcal{L}_2$ norm. Under proper conditions, the UAS bound implies a generalization error bound (Bassily et al. 2020): if $P(\|\lambda(S_1, S_2)\| \geq \gamma) \leq \kappa_0$ for any neighboring (S_1, S_2), then for any κ,

$$P\left[|\mathcal{E}_{\text{gen}}| \geq c\left(\gamma(\log n)(\log(n/\kappa)) + \sqrt{\frac{\log(1/\kappa)}{n}} \right) \right] \leq \kappa + \kappa_0.$$

We presents the upper bound and lower bound of UAS of adversarial training when its natural counterpart is convex and smooth.

The upper bound of UAS of adversarial training can be directly extended from [Bas+20]as follows:

Proposition 1. Assume $l(f_\theta(x), y)$ is L-Lipschitz and convex w.r.t. θ, and $\theta \in B_2(0, r)$. The two models $\theta_1^{(t)}$ and $\theta_2^{(t)}$ are adversarial training estimators obtained using datasets S_1, S_2 respectively. For SGD,

$$\sup_{S_1 \sim S_2} \mathbb{E}\left[\left\| \theta_1^{(T)} - \theta_2^{(T)} \right\| \right] = O\left(\min\left\{ r, L\sqrt{\sum_{t=1}^{T} \eta_t^2} + L\frac{\sum_{t=1}^{T} \eta_t}{n} \right\} \right).$$

The upper bound of GD is the same.

The following theorem presents the lower bound of UAS. For simplicity, we consider the case of constant learning rate, i.e., $\eta_t = \eta$ for $t = 1, \ldots, T$.

Theorem 1. Assume $\theta \in B_2(0, r)$. There exist some $\epsilon > 0$ and some loss function $l(f_\theta(x), y)$ which is differentiable and convex w.r.t. θ, such that $\theta_1^{(t)}$ and $\theta_2^{(t)}$, which are SGD-based adversarial training estimators obtained using S_1, S_2 respectively under attack strength ϵ, satisfies that

$$\sup_{S_1 \sim S_2} \mathbb{E}\left\| \theta_1^{(T)} - \theta_2^{(T)} \right\| = \Omega\left(\min\left\{ 1, \frac{T}{n} \right\} \eta\sqrt{T} + \frac{\eta T}{n} \right).$$

For GD, the lower bound is

$$\sup_{S_1 \sim S_2} \left\| \theta_1^{(T)} - \theta_2^{(T)} \right\| = \Omega\left(\min\left\{ 1, \eta\sqrt{T} + \frac{\eta T}{n} \right\} \right)$$

To prove Theorem 1. similar to [Bas+20], we design a smooth clean loss function with two datasets $S_1 \sim S_2$ and study the exact change of the model parameters. The detailed proof is postponed to the Appendix D

As discussed by[Bas+20], the non-smoothness of the loss is the main reason for poor stability. The presented low bounds match the result of [Bas+20], but it is worth mentioning that the two results are not directly comparable since [Bas+20] studied the UAS of clean training when the loss function $l(f_\theta(x), y)$ is non-smooth, while our work studies the UAS of adversarial training when the loss function is smooth. On the other hand, the UAS of cleaning training under smooth loss, implied by Theorem 3.8 of [HRS16], is of order $O\left(\min\left\{ r, L\sum_{t=t_0}^{T} \eta_t/n \right\} \right)$. Therefore, we conclude that adversarial training has a worse stability than its natural counterpart.

To ensure the convergence of optimization (i.e., ηT is not so small) and the generalization performance (Proposition 1 and Theorem 11, one may take $T = n^2$ and $\eta = 1/n^{3/2}$. The resulting optimization error and stability then become $O(1/\sqrt{n})$, which matches the minimax lower bound of excess risk (Chen et al. 2018). However, such a choice of (η, T) is impractical and needs to improve.

Corollary 1. Under the same conditions of Proposition [1 assume the algorithm uses an approximation A'_ϵ instead of the exact attack A_ϵ with attack error $\min \|A_\epsilon(x, y, w) - A'(x, y, w)\| \leq \Delta\varepsilon$ for any (x, y, w), where the minimum is taken when the exact attack (i.e., (1) is not unique. Assume $\nabla_\theta l(f_\theta(x), y)$ is κ-Lipschitz w.r.t. x. Then, for SGD

$$\sup_{S_1 \sim S_2} \mathbb{E}\left\| \theta_1^{(T)} - \theta_2^{(T)} \right\| = O\left(\min\left\{ r, L\sqrt{\sum_{t=1}^{T} \eta_t^2} + L\frac{\sum_{t=1}^{T} \eta_t}{n} + \kappa\Delta\varepsilon\sum_{t=1}^{T} \eta_t \right\} \right).$$

The upper bound of GD is the same.
Summarizing from the related works, we identify two important sources of non-smoothness in adversarial training even when the standard loss is smooth: (1) when the data are overfitted, i.e., the training loss is almost 0 and $\nabla_{x_i} l\left(f_\theta\left(x_i\right), y_i\right) \approx 0$, the adversary has no preference on the attack direction at x_i, and the numerical estimation of A_ϵ is not stable, which possibly leads to an unstable update iteration of adversarial training; (2) there exists a certain set of θ, such that the adversarial training loss is always non-differentiable regardless of the training data, even when its natural counterpart is smooth. For example, in linear regression, when θ_t is closed to the null model, the non-smoothness issue occurs ([XSC21]).
Algorithm 1 below illustrates the details of the noise injection method. The basic idea behind this is that: the non-smoothness of adversarial loss occurs only when θ and x_i ' belong to a certain special region (e.g., in linear regression, when θ is closed to either zero or when $\theta^\top x_i \equiv y_i$), thus injecting some small noise to both θ and x helps them to escape from such region where non-smoothness occurs, which further ensures the Lipschitz continuity property.
In the literature, there have been some applications of noise injection. For example, [RHF18]considered injecting noise to the weights as a regularization method to improve the adversarial robustness. Besides literature in supervised learning ([Wen+18]; [Wan+19a]; [Soh+20], injecting noise in data was also considered to stabilize the training process of GAN ([AB17];[JF19]).

Algorithm 1 Add noise to weight and data

Input: data $\{(x_i, y_i)\}_{i=1}^n$, number of iterations T, learning rate $\{\eta_t\}_{t=1}^T$, attack strength ϵ, noise size (ξ_θ, ξ_x), scale parameter r , initialization $\theta^{(0)}$.
for $t = 1$ **to** T **do**
Add Gaussian noise with variance ξ_θ^2 to each dimension of θ_t to form $\widetilde{\theta}^{(t)}$, and add Gaussian noise with variance ξ_x^2 to each dimension of x_t to obtain $\widetilde{x}_{i_t}$.
Calculate the attack (based on $\widetilde{x}_{i_t}$ and $\widetilde{\theta}^{(t)}$) as $\widehat{z}_{i_t}$.
Take gradient w.r.t $\widetilde{\theta}^{(t)}$ on $l(f_{\widetilde{\theta}^{(t)}}(\widehat{z}_{i_t}), y_{i_t})$.
Update $\theta^{(t)}$ to $\theta^{(t+1)}$ with rate η_t.
Project $\theta^{(t+1)}$ onto $B_2(0, r)$.
end for
Output: $\theta^{(T)}$.

Theorem 2. Assume $\theta \in B_2(0, r)$. Denote $\widehat{\theta}(S)$ as the model obtained based on dataset S using some algorithm. Assume $l\left(f_\theta(x), y\right) \in [0, M]$ when $\|x\| \leq \sqrt{d \log n}$, we have for any $i = 1, \ldots, n$,

$$\mathbb{E}\left[\left(R\left(\widehat{\theta}\left(S_1\right)\right) - R_{S_1}\left(\widehat{\theta}\left(S_1\right)\right)\right)^2\right] \leq \frac{M^2}{2n} + 4\mathbb{E}\left[\sup_{\theta \in B_2(0,r)} l^2\left(f_\theta\left(x + A_\epsilon\right), y\right) 1\{\|x\| \geq \sqrt{d \log n}\}\right] + 3M\mathbb{E}\left[\left|l\left(f_{\widehat{\theta}(S_1)}\left(x_i + A_\epsilon\right), y_i\right) - l\left(f_{\widehat{\theta}\left(S_2^i\right)}\left(x_i + A_\epsilon\right), y_i\right)\right|\right],$$

where S_2^i represents the neighboring dataset of S_1 whose ith sample is replaced. The notion A_ϵ is an abbreviation of the attack $A_\epsilon(f, x, y)$ or $A_\epsilon\left(f, x_i, y_i\right)$.

3.2 Wasserstein Distance in Adversarial Training

Let $\boldsymbol{x} \in [0, 1]^{n \times m}$ denote a two dimensional image, of height n and width m. We will normalize the image such that $\sum_i \sum_j x_{i,j} = 1$, so that $\boldsymbol{x}$ can be interpreted as a probability distribution on the discrete support of pixel coordinates of the two-dimensional image,Following the notation of [WSK20], we define the p-Wasserstein distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$ as:
Definition 1.1. Given two distributions $\boldsymbol{x}, \boldsymbol{x}' \in [0, 1]^{n \times m}$, and a distance metric $d \in ([n] \times [m]) \times ([n] \times [m]) \to \mathbb{R}$, the p-Wasserstein distance as:

$$W_p(\boldsymbol{x}, \boldsymbol{x}') = \min_{\Pi \in \mathbb{R}_+^{(n \cdot m) \times (n \cdot m)}} < \Pi, C >,$$
$$\Pi \mathbf{1} = \boldsymbol{x}, \Pi^T \mathbf{1} = \boldsymbol{x}',$$
$$C_{(i,j),(i',j')} := [d((i,j),(i',j'))]^p.$$

Intuitively, $\Pi_{(i,j),(i',j')}$ represents the amount of probability mass to be transported from pixel (i,j) to (i',j'), while $C_{(i,j),(i',j')}$ represents the cost per unit probability mass to transport this probability. We can choose $d(.,.)$to be any measure of distance between pixel positions in an image. For example, in order to represent the L_1 distance metric between pixel positions, we can choose:

$$d((i,j),(i',j')) = |i - i'| + |j - j'|.$$

Moreover, to represent the L_2 distance metric between pixel positions, we can choose:

$$d((i,j),(i',j')) = \sqrt{(i-i')^2 + (j-j')^2}.$$

To develop our certificate, we rely an alternative linear program formulation for the 1-Wasserstein distance on a two-dimensional image with the L_1 distance metric:

$$W_1(\boldsymbol{x}, \boldsymbol{x}') = \min_{\mathbf{g}} \sum_{(i,j)} \sum_{(i',j') \in \mathcal{N}(i,j)} \mathbf{g}_{(i,j),(i',j')}$$

where $\mathbf{g} \geq 0$ and $\forall (i,j)$,

$$\sum_{(i',j') \in \mathcal{N}(i,j)} \mathbf{g}_{(i,j),(i',j')} - \mathbf{g}_{(i',j'),(i,j)} = \mathbf{x}'_{i,j} - \mathbf{x}_{i,j}$$

Here, $\mathcal{N}(i,j)$ denotes the (up to) four immediate (nondiagonal) neighbors of the position (i,j); in other words, $\mathcal{N}(i,j) = \{(i',j') \,||i - i'| + |j - j' \,| = 1\}$. For the L_1 distance in two dimensions, Ling and Okada (2007) prove that this formulation is in fact equivalent to the linear program given in Equation. Note that only elements of $\mathbf{g}$ with $|i - i'| + |j - j'| = 1$ need to be defined: this means that the number of variables in the linear program is approximately $4nm$, compared to the n^2m^2 elements of Π in Equation. While this was originally used to make the linear program more tractable to be solved directly, we exploit the form of this linear program to devise a randomized smoothing scheme in the next section.

Definition 1.2. The local flow plan application function $\Delta \in \mathbb{R}^{n \times m} \times \mathbb{R}^r \to \mathbb{R}^{n \times m}$ is defined as:

$$\Delta(\boldsymbol{x}, \boldsymbol{\delta})_{i,j} = \boldsymbol{x}_{i,j} + \boldsymbol{\delta}^{\text{vert.}}_{i-1,j} - \boldsymbol{\delta}^{\text{vert.}}_{i,j} + \boldsymbol{\delta}^{\text{horiz.}}_{i,j-1} - \boldsymbol{\delta}^{\text{horiz.}}_{i,j}$$

where we let $\boldsymbol{\delta}^{\text{vert.}}_{0,j} = \boldsymbol{\delta}^{\text{vert.}}_{n,j} = \boldsymbol{\delta}^{\text{horiz.}}_{i,0} = \boldsymbol{\delta}^{\text{horiz.}}_{i,m} = 0$

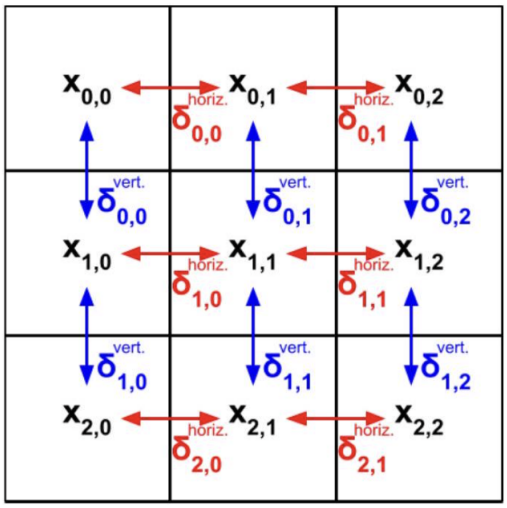

Figure 6: Indexing of the elements of the local flow map $\boldsymbol{\delta}$, in relation to the pixels of the image $\boldsymbol{x}$, with n = m = 3.

Given a classification score function $\boldsymbol{f} : \mathbb{R}^{n\times m} \to [0,1]^k$, we define $\bar{f}$ as the Wasserstein-smoothed classification function as follows:

$$\overline{\boldsymbol{f}} = \mathop{\mathbb{E}}_{\boldsymbol{\delta}\sim\mathcal{L}(\sigma)}[\boldsymbol{f}(\Delta(\boldsymbol{x},\boldsymbol{\delta}))]$$

Let i be the class assignment of $\boldsymbol{x}$ using the Wasserstein-smoothed classifier $\bar{f}$ (i.e. $i = \arg\max_{i'} \overline{\boldsymbol{f}}_{i'}(\boldsymbol{x})$).
Theorem 1. For any normalized probability distribution $\boldsymbol{x} \in [0,1]^{n\times m}$, if

$$\overline{\boldsymbol{f}}_i(\boldsymbol{x}) \geq e^{2\sqrt{2}\rho/\sigma} \max_{i'\neq i} \overline{\boldsymbol{f}}_{i'}(\boldsymbol{x})$$

then for any perturbed probability distribution $\tilde{\boldsymbol{x}}$ such that $W_1(\boldsymbol{x},\tilde{\boldsymbol{x}}) \leq \rho$, we have:

$$\overline{\boldsymbol{f}}_i(\tilde{\boldsymbol{x}}) \geq \max_{i'\neq i} \overline{\boldsymbol{f}}_{i'}(\tilde{\boldsymbol{x}}).$$

4 Discussion

In this Project, we have explored the topic of adversarial learning, which has become increasingly important in the field of machine learning due to the potential vulnerability of machine learning models to adversarial attacks. Our analysis of the current literature, including three influential works, has revealed important insights into the theoretical foundations and practical applications of adversarial learning. We have learned that adversarial learning algorithms can improve the robustness and generalization performance of machine learning models by incorporating adversarial examples into the training process. Additionally, the use of Rademacher complexity as a measure of generalization performance has been shown to provide a tight upper bound on the generalization error of adversarially robust classifiers. However, there are still many open questions and challenges in the field of adversarial learning, including the development of more efficient and scalable algorithms, the extension of adversarial learning to other machine learning models and applications, and the integration of adversarial learning into real-world systems. Overall, our analysis highlights the importance of continued research in this area to improve the security and reliability of machine learning systems in the face of adversarial attacks.
For future work, one suggestion is to consider the integration of duality for deep learning models, which could be a challenging but important topic. Additionally, we could explore the smoothing of Wasserstein distance into the minimax approach to see the effect of this new distance and how it affects the final results. These directions offer promising opportunities for further research in adversarial learning and have the potential to lead to significant advancements in the field.

References

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[HRS15] Moritz Hardt, Benjamin Recht, and Yoram Singer. "Train faster, generalize better: Stability of stochastic gradient descent". In: *CoRR* abs/1509.01240 (2015). arXiv: 1509.01240. URL: http://arxiv.org/abs/1509.01240.

[HRS16] Moritz Hardt, Benjamin Recht, and Yoram Singer. *Train faster, generalize better: Stability of stochastic gradient descent*. 2016. arXiv: 1509.01240 [cs.LG].

[Pap+16] Nicolas Papernot et al. *Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks*. 2016. arXiv: 1511.04508 [cs.CR].

[AB17] Martin Arjovsky and Léon Bottou. *Towards Principled Methods for Training Generative Adversarial Networks*. 2017. arXiv: 1701.04862 [stat.ML].

[Mad+17] Aleksander Madry et al. "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083* (2017).

[Tra+17] Florian Tramèr et al. "Ensemble adversarial training: Attacks and defenses". In: *arXiv preprint arXiv:1705.07204* (2017).

[Cai+18] Qi-Zhi Cai et al. "Curriculum adversarial training". In: *arXiv preprint arXiv:1805.04807* (2018).

[RHF18] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. *Parametric Noise Injection: Trainable Randomness to Improve Deep Neural Network Robustness against Adversarial Attack*. 2018. arXiv: 1811.09310 [cs.LG].

[Wen+18] Tsui-Wei Weng et al. *Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach*. 2018. arXiv: 1801.10578 [stat.ML].

[BGH19] Yogesh Balaji, Tom Goldstein, and Judy Hoffman. "Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets". In: *arXiv preprint arXiv:1910.08051* (2019).

[JF19] Simon Jenni and Paolo Favaro. *On Stabilizing Generative Adversarial Training with Noise*. 2019. arXiv: 1906.04612 [cs.CV].

[Mad+19] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019. arXiv: 1706.06083 [stat.ML].

[Mao+19] Chengzhi Mao et al. "Metric learning for adversarial robustness". In: *Advances in neural information processing systems* 32 (2019).

[TZT19] Zhuozhuo Tu, Jingwei Zhang, and Dacheng Tao. *Theoretical Analysis of Adversarial Learning: A Minimax Approach*. 2019. arXiv: 1811.05232 [stat.ML].

[Wan+19a] Bao Wang et al. *ResNets Ensemble via the Feynman-Kac Formalism to Improve Natural and Robust Accuracies*. 2019. arXiv: 1811.10745 [cs.LG].

[Wan+19b] Yisen Wang et al. "Improving adversarial robustness requires revisiting misclassified examples". In: *International conference on learning representations*. 2019.

[Zha+19] Hongyang Zhang et al. "Theoretically principled trade-off between robustness and accuracy". In: *International conference on machine learning*. PMLR. 2019, pp. 7472–7482.

[Bas+20] Raef Bassily et al. "Stability of Stochastic Gradient Descent on Nonsmooth Convex Losses". In: *CoRR* abs/2006.06914 (2020). arXiv: 2006.06914. URL: https://arxiv.org/abs/2006.06914.

[JSH20] Adel Javanmard, Mahdi Soltanolkotabi, and Hamed Hassani. *Precise Tradeoffs in Adversarial Training for Linear Regression*. 2020. arXiv: 2002.10477 [cs.LG].

[KLL20] Hoki Kim, Woojin Lee, and Jaewook Lee. *Understanding Catastrophic Overfitting in Single-step Adversarial Training*. 2020. arXiv: 2010.01799 [cs.LG].

[Soh+20] Kihyuk Sohn et al. *FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence*. 2020. arXiv: 2001.07685 [cs.LG].

[WSK20] Eric Wong, Frank R. Schmidt, and J. Zico Kolter. *Wasserstein Adversarial Examples via Projected Sinkhorn Iterations*. 2020. arXiv: 1902.07906 [cs.LG].

[YRB20] Dong Yin, Kannan Ramchandran, and Peter Bartlett. *Rademacher Complexity for Adversarially Robust Generalization*. 2020. arXiv: 1810.11914 [cs.LG].

[Zha+20] Jingfeng Zhang et al. "Attacks which do not kill training make adversarial learning stronger". In: *International conference on machine learning*. PMLR. 2020, pp. 11278–11287.

[Bai+21] Tao Bai et al. "Recent Advances in Adversarial Training for Adversarial Robustness". In: *CoRR* abs/2102.01356 (2021). arXiv: 2102.01356.

[KM21] Peilin Kang and Seyed-Mohsen Moosavi-Dezfooli. *Understanding Catastrophic Overfitting in Adversarial Training*. 2021. arXiv: 2105.02942 [cs.LG].

[Wan+21] Yisen Wang et al. "On the convergence and robustness of adversarial training". In: *arXiv preprint arXiv:2112.08304* (2021).

[XSC21] Yue Xing, Qifan Song, and Guang Cheng. *On the Generalization Properties of Adversarial Training*. 2021. arXiv: 2008.06631 [stat.ML].