Building Robust Ensembles via Margin Boosting

Dinghuai Zhang [1] **Hongyang Zhang** [2] **Aaron Courville** [1] **Yoshua Bengio** [1]
Pradeep Ravikumar [3] **Arun Sai Suggala** [4]

Abstract

In the context of adversarial robustness, a single model does not usually have enough power to defend against all possible adversarial attacks, and as a result, has sub-optimal robustness. Consequently, an emerging line of work has focused on learning an ensemble of neural networks to defend against adversarial attacks. In this work, we take a principled approach towards building robust ensembles. We view this problem from the perspective of margin-boosting and develop an algorithm for learning an ensemble with maximum margin. Through extensive empirical evaluation on benchmark datasets, we show that our algorithm not only outperforms existing ensembling techniques, but also large models trained in an end-to-end fashion. An important byproduct of our work is a margin-maximizing cross-entropy (MCE) loss, which is a better alternative to the standard cross-entropy (CE) loss. Empirically, we show that replacing the CE loss in state-of-the-art adversarial training techniques with our MCE loss leads to significant performance improvement.

1. Introduction

The output of deep neural networks can be vulnerable to even a small amount of perturbation to the input (Szegedy et al., 2013). These perturbations, usually referred to as adversarial perturbations, can be imperceptible to humans and yet deceive even state-of-the-art models into making incorrect predictions. Owing to this vulnerability, there has been a great interest in understanding the phenomenon of these adversarial perturbations (Fawzi et al., 2018; Bubeck et al., 2019; Ilyas et al., 2019), and in designing techniques to defend against adversarial attacks (Goodfellow et al., 2014; Madry et al., 2017; Raghunathan et al., 2018; Zhang et al., 2019b). However, coming up with effective defenses has turned out to be a hard problem as many of the proposed defenses were eventually circumvented by novel adversarial attacks (Athalye et al., 2018a; Tramer et al., 2020). Even the well performing techniques, such as adversarial training (AT) (Madry et al., 2017), are unsatisfactory as they do not yet achieve high enough adversarial robustness.

One of the reasons for the unsatisfactory performance of existing defenses is that they train a single neural network to defend against all possible adversarial attacks. Such single models are often not powerful enough to defend against all the moves of the adversary and as a result, have sub-optimal robustness. Consequently, an emerging line of work in adversarial robustness has focused on constructing ensembles of neural networks (Kariyappa & Qureshi, 2019; Verma & Swami, 2019; Pinot et al., 2020). These ensembling techniques work under the hypothesis that an ensemble with a diverse collection of classifiers can be more effective at defending against adversarial attacks, and can lead to better robustness guarantees. This hypothesis was in fact theoretically proven to be true by Pinot et al. (2020). However, many of the existing ensemble based defenses were not successful, and were defeated by stronger attacks (Tramer et al., 2020). In this paper, we show that the recently proposed ensembling technique of Pinot et al. (2020) can be defeated (Appendix F.6), thus adding the latter to the growing list of "circumvented ensemble defenses". This shows that the problem of designing ensembles that are robust to adversarial attacks is pretty much unsolved.

In this work, we take a principled approach towards constructing robust ensembles. We view this problem from the perspective of boosting and ask the following question:

How can we combine multiple base classifiers into a strong classifier that is robust to adversarial attacks?

Our answer to this question relies on the key machine learning notion of margin, which is known to govern the generalization performance of a model (Bartlett, 1998). In particular, we develop a margin-maximizing boosting framework that aims to find an ensemble with maximum margin via a two-player zero-sum game between the learner and the adversary, the solution to which is the desired max-margin ensemble. One of our key contributions is

[1]Mila and Université de Montréal [2]University of Waterloo [3] Carnegie Mellon University [4]Google Research. Correspondence to: Dinghuai Zhang <dinghuai.zhang@mila.quebec>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

to provide an efficient algorithm (MRBOOST) for solving this game. Through extensive empirical evaluation on benchmark datasets, we show that our algorithm not only outperforms existing ensembling techniques, but also large models trained in an end-to-end fashion. An important byproduct of our work is a margin-maximizing cross-entropy (MCE) loss, which is a better alternative to the standard cross-entropy (CE) loss. Empirically, we demonstrate that replacing the CE loss in state-of-the-art adversarial training techniques with our MCE loss leads to significant performance improvement. Our code is available at https://github.com/zdhNarsil/margin-boosting.

Contributions. Here are the key contributions of our work:

- We propose a margin-boosting framework for learning max-margin ensembles. Moreover, we prove the optimality of our framework by showing that it requires the weakest possible conditions on base classifiers to output an ensemble with strong adversarial performance;
- We derive a computationally efficient algorithm (MRBOOST.NN) from our margin-boosting framework. Through extensive empirical evaluation, we demonstrate the effectiveness of our algorithm;
- Drawing from our boosting framework, we propose the MCE loss, a new variant of the CE loss, which improves the adversarial robustness of state-of-the-art defenses.

2. Preliminaries

In this section, we set up the notation and review necessary background on adversarial robustness and ensembles. A consolidated list of notation can be found in Appendix A.

Notation. Let $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ denote a feature-label pair following an unknown probability distribution P. In this work, we consider the multi-class classification problem where $\mathcal{Y} = \{0, \dots K-1\}$, where K is the number of classes, and assume $\mathcal{X} \subseteq \mathbb{R}^d$. Let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be n i.i.d samples drawn from P, and P_n the empirical distribution of S. We let $h : \mathcal{X} \to \mathcal{Y}$ denote a generic classifier which predicts the label of $\mathbf{x}$ as $h(\mathbf{x})$. In practice, such a classifier is usually constructed by first constructing a *score-based* classifier $g : \mathcal{X} \to \mathbb{R}^K$ which assigns a confidence score to each class, and then mapping its output to an appropriate class using the argmax operation: $\operatorname{argmax}_{j \in \mathcal{Y}} [g(\mathbf{x})]_j$.[1] We denote the resulting classifier by g^{am}.

Standard Classification Risk. The expected classification risk of a score-based classifier g is defined as $\mathbb{E}_{(X,Y)\sim P}[\ell_{0-1}(g(X), Y)]$, where $\ell_{0-1}(g(X), Y) = 0$ if $g^{\mathrm{am}}(X) = Y$, and 1 otherwise. Since optimizing $0/1$ risk is computationally intractable, it is often replaced with convex surrogates, which we denote by $\ell(g(X), Y)$. A popular choice for ℓ is the cross-entropy loss: $\ell_{\mathrm{CE}}(g(\mathbf{x}), y) \coloneqq -[g(\mathbf{x})]_y + \log\left(\sum_{j\in\mathcal{Y}} \exp[g(\mathbf{x})]_j\right)$. The population and empirical risks of classifier g w.r.t loss ℓ are defined as $R(g;\ell) \coloneqq \mathbb{E}_{(X,Y)\sim P}[\ell(g(X), Y)]$, $\widehat{R}_n(g;\ell) \coloneqq \mathbb{E}_{(X,Y)\sim P_n}[\ell(g(X), Y)]$.

Adversarial Risk. We consider the following robustness setting in this work: given a classifier, there is an adversary which corrupts the inputs to the classifier with the intention of making the model misclassify the inputs. Our goal is to design models that are robust to such adversaries. Let $\mathcal{B}(\epsilon)$ be the set of perturbations that the adversary is allowed to add to the input. Popular choices for $\mathcal{B}(\epsilon)$ for instance include ℓ_p norm balls $\{\omega : \|\omega\|_p \le \epsilon\}$ for $p \in \{2, \infty\}$. In this work, we assume $\mathcal{B}(\epsilon)$ is a compact set (this is satisfied by ℓ_p norm balls). Given this setting, the population and empirical adversarial risks of a classifier g w.r.t loss ℓ are defined as

$$R^{\mathrm{adv}}(g;\ell) \coloneqq \mathbb{E}_{(X,Y)\sim P}\left[\max_{\delta\in\mathcal{B}(\epsilon)} \ell(g(X+\delta), Y)\right],$$

$$\widehat{R}_n^{\mathrm{adv}}(g;\ell) \coloneqq \mathbb{E}_{(X,Y)\sim P_n}\left[\max_{\delta\in\mathcal{B}(\epsilon)} \ell(g(X+\delta), Y)\right].$$

Ensembles. Ensembling is a popular technique in machine learning for constructing models with good generalization performance (*i.e.*, small population risk). Ensembling typically involves linearly combining the predictions of several base classifiers. Let $\mathcal{H}$ be the set of base classifiers, where each $h \in \mathcal{H}$ maps from $\mathcal{X}$ to $\mathcal{Y}$. In ensembling, we place a probability distribution Q over $\mathcal{H}$ which specifies the weight of each base classifier. This distribution defines a score-based classifier $h_Q : \mathcal{X} \to \mathbb{R}^K$ with $[h_Q(\mathbf{x})]_j = \mathbb{E}_{h\sim Q}[\mathbb{I}(h(\mathbf{x}) = j)]$. This can be converted into a standard classifier using the argmax operation: $h_Q^{\mathrm{am}}(\mathbf{x}) = \operatorname{argmax}_{j\in\mathcal{Y}}[h_Q(\mathbf{x})]_j$. If we have score-based base classifiers $\mathcal{G} = \{g_1, g_2, \dots\}$, where each $g \in \mathcal{G}$ maps $\mathcal{X}$ to $\mathbb{R}^K$, we can simply linearly combine them via a set of real-valued weights W over $\mathcal{G}$, and define a score-based classifier g_W as $[g_W(\mathbf{x})]_j = \sum_{g\in\mathcal{G}} W(g)[g(\mathbf{x})]_j$, where $W(g) \in \mathbb{R}$ is the weight of the base classifier g. g_W can be converted into a standard classifier using the argmax operation: $g_W^{\mathrm{am}}(\mathbf{x}) = \operatorname{argmax}_{j\in\mathcal{Y}}[g_W(\mathbf{x})]_j$.

Boosting. Boosting is perhaps the most popular technique for creating ensembles. Boosting aims to address the following question: *"Given a set of base classifiers, how can we combine them to produce an ensemble with the best predictive performance?"* Numerous techniques have been proposed to answer this question, with the most popular ones being *margin*-boosting (Freund et al., 1996; Breiman, 1999; Rätsch et al., 2005) and *greedy*-boosting (Mason et al., 2000b; Friedman, 2001). The technique developed in this work falls in the category of margin-boosting. Margin-boosting works under the hypothesis that a large-margin

[1] If there are multiple optimizers to this problem, we pick one of the optimizers uniformly at random.

classifier has good generalization performance (Bartlett et al., 1998; Bartlett, 1998). Consequently, it aims to learn an ensemble with large margin. Let $\mathcal{H}$ be the set of base classifiers, where each $h \in \mathcal{H}$ maps $\mathcal{X}$ to $\mathcal{Y}$. The margin of the ensemble h_Q, for some probability distribution Q over $\mathcal{H}$, at point $(\mathbf{x}, y)$, is defined as

$$\operatorname{mg}(Q, \mathbf{x}, y) := [h_Q(\mathbf{x})]_y - \max_{y' \neq y} [h_Q(\mathbf{x})]_{y'}. \quad (1)$$

Intuitively, margin captures the confidence with which h_Q assigns $\mathbf{x}$ to class y. We ideally want $\operatorname{mg}(Q, \mathbf{x}, y)$ to be large for all $(\mathbf{x}, y) \in S$. To capture this, we introduce the notion of *minimum margin* over the dataset S which is defined as $\operatorname{mg}(Q, S) := \min_{(\mathbf{x},y)\in S} \operatorname{mg}(Q, \mathbf{x}, y)$. In margin-boosting, we aim to find a Q with the largest possible minimum margin. This leads us to the following optimization problem: $\max_{Q\in\Delta(\mathcal{H})} \operatorname{mg}(Q, S)$, where $\Delta(\mathcal{H})$ is the set of all probability distributions over $\mathcal{H}$. AdaBoost.MR, a popular boosting algorithm for multi-class classification, can be viewed as solving this optimization problem (Schapire & Singer, 1999; Mukherjee & Schapire, 2013).

2.1. Related Work

Adversarial Robustness. Numerous techniques have been proposed to defend neural networks against adversarial attacks (Szegedy et al., 2014). These techniques broadly fall into two categories. The first category of techniques called *empirical defenses* rely on heuristics and do not provide any guarantee on the robustness of the learned models. Adversarial training (AT) (Madry et al., 2017) is by far the most popular defense in this category. AT has been successfully improved by many recent works such as Zhang et al. (2019b;a); Wang et al. (2019b); Carmon et al. (2019); Zhang et al. (2020b); Shi et al. (2020). The second category of techniques called *certified defenses* output models whose robustness can be certified in the following sense: at any given point, they can provide a certificate proving the robustness of the learned model to adversarial attacks at that point. Early computationally efficient approaches in this category were developed by Raghunathan et al. (2018); Wong & Kolter (2018). Several recent techniques such as randomized smoothing (Cohen et al., 2019; Salman et al., 2019; Zhang et al., 2020a; Blum et al., 2020; Yang et al., 2021) have improved upon these early works and even scale to ImageNet size datasets.

Several recent works have attempted to use ensembles to defend against adversarial attacks. Sen et al. (2020) train the base classifiers of the ensemble independent of each other, which ignores the interaction between base classifiers (Pang et al., 2019). Other works (Verma & Swami, 2019; Kariyappa & Qureshi, 2019; Pang et al., 2019; Meng et al., 2020) simultaneously learn all the components of the ensemble, which requires huge memory and computational resources. These works add diversity promoting regularizers to their training objectives to learn good ensembles. A number of these defences are known to be rather weak (Tramer et al., 2020). There are also works which use boosting inspired approaches and construct ensembles in a sequential manner (Pinot et al., 2020; Abernethy et al., 2021). Many of these techniques rely on heuristics and are rather weak. In Appendix F.6, we empirically show that the defense of Pinot et al. (2020) can be circumvented with carefully designed adversarial attacks. Moreover, we show that our boosting algorithm has better performance than the algorithm of Abernethy et al. (2021).

Boosting. Boosting has a rich history in both computer science (CS) and statistics. The CS community takes a game-theoretic perspective of boosting and views boosting algorithms as playing a game against a base/weak learner (Freund & Schapire, 1995). The statistical community views it as greedy stagewise optimization (Friedman, 2001; Mason et al., 2000b). Both these views have contributed to the development of popular boosting techniques such as AdaBoost (Freund & Schapire, 1995), XGBoost (Chen & Guestrin, 2016). In this work, we take the game-theoretic perspective to build robust ensembles. Recently, boosting has seen a revival in the deep learning community. This is because boosting techniques consume less memory than end-to-end training of deep networks and can accommodate much larger models in limited memory (Huang et al., 2017; Nitanda & Suzuki, 2018; Suggala et al., 2020). Moreover, boosting techniques are easier to understand from a theoretical and optimization standpoint and can make neural networks easy to adopt in critical applications.

3. Margin-Boosting for Robustness

In this section, we present our margin-boosting framework for building robust ensembles. Let $\mathcal{H}$ be a compact set of base classifiers. Typical choices for $\mathcal{H}$ include the set of all decision trees of certain depth, and the set of all neural networks of bounded depth and width. Given $\mathcal{H}$, our goal is to design an ensemble (*i.e.,* identify a $Q \in \Delta(\mathcal{H})$) which has the best possible robustness towards adversarial attacks. The starting point for our boosting framework is the observation that large margin classifiers tend to generalize well to unseen data. In fact, large margin has been attributed to the success of popular ML techniques such as SVMs and boosting (Bartlett, 1998; Bartlett et al., 1998; Mason et al., 2000a). So, in this work, we learn ensembles with large margins to defend against adversarial attacks. However, unlike ordinary boosting, ensuring large margins for data points in S does not suffice for adversarial robustness. In robust boosting, we need *large margins even at the perturbed points* for the ensemble to have good adversarial generalization (Khim & Loh, 2018; Yin et al., 2019). Letting h_Q be our ensemble, we want $\operatorname{mg}(Q, \mathbf{x}+\delta, y)$ to be large for all $(\mathbf{x}, y) \in S$,

$\delta \in \mathcal{B}(\epsilon)$. To capture this, we again introduce the notion of *minimum robust margin* of h_Q which is defined as $\mathrm{mg}_{\mathrm{rob}}(Q, S) \coloneqq \min_{(\mathbf{x},y)\in S} \min_{\delta\in\mathcal{B}(\epsilon)} \mathrm{mg}(Q, \mathbf{x}+\delta, y)$. In our boosting framework, we aim to find a Q with the largest possible $\mathrm{mg}_{\mathrm{rob}}(Q, S)$, which leads us to the following optimization problem: $\max_{Q\in\Delta(\mathcal{H})} \mathrm{mg}_{\mathrm{rob}}(Q, S)$. This problem can be equivalently written as

$$\max_{Q\in\Delta(\mathcal{H})} \min_{\substack{(\mathbf{x},y)\in S,\\ y'\in\mathcal{Y}\setminus\{y\},\delta\in\mathcal{B}(\epsilon)}} [h_Q(\mathbf{x}+\delta)]_y - [h_Q(\mathbf{x}+\delta)]_{y'}. \tag{2}$$

In this work, we often refer to such max-min problems as two-player zero-sum games.

3.1. Margin-Boosting is Optimal

Before we present our algorithm for solving the max-min problem in Equation (2), we try to understand the margin-boosting framework from a theoretical perspective. In particular, we are interested in studying the following questions pertaining to the quality of our boosting framework:

1. Under what conditions on $\mathcal{H}$ does the boosting framework output an ensemble with 100% adversarial accuracy on the training set?
2. Are these conditions on $\mathcal{H}$ optimal? Can there be a different boosting framework that outputs an 100% accurate ensemble under milder conditions on $\mathcal{H}$?

Understanding these questions can aid us in designing appropriate base hypothesis classes $\mathcal{H}$ for our boosting framework. The choice of $\mathcal{H}$ is crucial as it can significantly impact learning and generalization. If $\mathcal{H}$ is too weak to satisfy the required conditions, then the boosting framework cannot learn a robust ensemble. On the other hand, using a more complex $\mathcal{H}$ than necessary can result in overfitting. Understanding the second question can thus help us compare various boosting frameworks. For instance, consider two boosting frameworks B_1 and B_2. If B_1 requires more powerful hypothesis class $\mathcal{H}$ than B_2 to output an 100% accurate ensemble, then the latter should be preferred over the former as using a less powerful $\mathcal{H}$ can prevent overfitting. The following theorems answer these questions.

Theorem 1. *The following is a necessary and sufficient condition on $\mathcal{H}$ that ensures that any maximizer of Equation (2) achieves 100% adversarial accuracy on S: for any probability distribution P' over points in the set $S_{aug} \coloneqq \{(\mathbf{x}, y, y', \delta) : (\mathbf{x}, y) \in S, y' \in \mathcal{Y}\setminus\{y\}, \delta \in \mathcal{B}(\epsilon)\}$, there exists a classifier $h \in \mathcal{H}$ which achieves slightly-better-than-random performance on P'*

$$\mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P'}[\mathbb{I}(h(\mathbf{x}+\delta) = y)] \geq \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P'}[\mathbb{I}(h(\mathbf{x}+\delta) = y')] + \tau.$$

Here $\tau > 0$ is some constant.

Theorem 2. *The margin-based boosting framework is* optimal *in the following sense: there does not exist any other boosting framework which can guarantee a solution with 100% adversarial accuracy with milder conditions on $\mathcal{H}$ than the above margin-based boosting framework.*

Discussion. The condition in Theorem 1 holds if for any weighting of points in S_{aug}, there exists a base classifier in $\mathcal{H}$ which performs slightly better than *random guessing*, where performance is measured with respect to an appropriate metric. In the case of binary classification, this metric boils down to ordinary classification accuracy, and the condition in Theorem 1 can be rewritten as

$$\mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P'}[\mathbb{I}(h(\mathbf{x}+\delta) = y)] \geq \frac{1+\tau}{2}.$$

Such conditions are referred to as *weak learning conditions* in the literature of boosting and have played a key role in the design and analysis of boosting algorithms (Freund & Schapire, 1995; Telgarsky, 2011; Mukherjee & Schapire, 2013).

Theorem 2 shows that the margin-based boosting framework is *optimal* in the sense that among all boosting frameworks, the margin-boosting framework requires the weakest possible weak learning condition. In a recent work, Mukherjee & Schapire (2013) obtained a similar result in the context of standard multi-class classification. In particular, they develop optimal boosting frameworks and identify minimal weak learning conditions for standard multi-class classification. Our work extends their results to the setting of adversarial robustness. Moreover, the results of Mukherjee & Schapire (2013) can be obtained as a special case of our results by setting $\mathcal{B}(\epsilon) = \{0\}$.

Comparison with Abernethy et al. (2021). Recently, Abernethy et al. (2021) developed a boosting framework for adversarial robustness. We now show that their framework is *strictly* sub-optimal to our framework. Abernethy et al. (2021) require $\mathcal{H}$ to satisfy the following weak learning condition to guarantee that their boosting framework outputs an ensemble with 100% adversarial accuracy: for any probability distribution P' over points in the set $\tilde{S}_{\text{aug}} \coloneqq \{(\mathbf{x}, y, y') : (\mathbf{x}, y) \in S, y' \in \mathcal{Y}\setminus\{y\}\}$, there exists a classifier $h \in \mathcal{H}$ which satisfies the following for some $\tau > 0$:

$$\mathbb{E}_{(\mathbf{x},y,y')\sim P'}[\mathbb{I}(\forall\delta \in \mathcal{B}(\epsilon) : h(\mathbf{x}+\delta) = y)] \geq \mathbb{E}_{(\mathbf{x},y,y')\sim P'}[\mathbb{I}(\exists\delta \in \mathcal{B}(\epsilon) : h(\mathbf{x}+\delta) = y')] + \tau.$$

Proposition 3. *Let $\mathrm{WL}_{\text{MRBoost}}$ denote the necessary and sufficient weak learning condition of our margin-boosting framework and $\mathrm{WL}_{\text{RobBoost}}$ denote the weak learning condition of the boosting framework of Abernethy et al. (2021). Then $\mathrm{WL}_{\text{RobBoost}} \implies \mathrm{WL}_{\text{MRBoost}}$. The implication does not hold the other way round; that is, $\mathrm{WL}_{\text{MRBoost}} \not\Longrightarrow \mathrm{WL}_{\text{RobBoost}}$.*

Algorithm 1 MRBOOST

Input: training data S, boosting iterations T, learning rate η.
Let P_1 be the uniform distribution over S_{aug}.
for $t = 1 \dots T$ **do**
  Compute $h_t \in \mathcal{H}$ as the minimizer of:

$$\min_{h \in \mathcal{H}} \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P_t}[\text{mg}_{\text{L}}\left(h(\mathbf{x}+\delta), y, y'\right)].$$

  Exponential Weights. Compute probability distribution P_{t+1}, supported on S_{aug}, as:

$$P_{t+1}(\mathbf{x}, y, y', \delta) \propto \exp\left(\eta \sum_{j=1}^{t} \text{mg}_{\text{L}}\left(h_j(\mathbf{x}+\delta), y, y'\right)\right),$$

end for
Output: return the classifier $h^{\text{am}}_{Q(T)}(\mathbf{x})$, where $Q(T)$ is the uniform distribution over $\{h_t\}_{t=1\dots T}$.

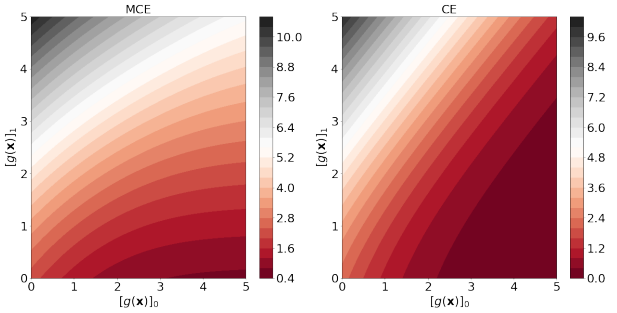

Figure 1. Contour plots of $\ell_{\text{MCE}}(g(\mathbf{x}), y, y')$ and $2\times\ell_{\text{CE}}(g(\mathbf{x}), y)$ for $K = 3, y = 0, y' = 1$. The plots show how the loss functions vary with $[g(\mathbf{x})]_0, [g(\mathbf{x})]_1$ when we set $[g(\mathbf{x})]_2$ to 0. It can be seen that the two losses differ significantly on the right side of their plots where ℓ_{MCE} aggressively penalizes points whose $[g(\mathbf{x})]_0, [g(\mathbf{x})]_1$ are close to each other.

3.2. Robust Boosting Algorithm

In this section, we present our algorithm MRBOOST for optimizing Equation (2). The pseudocode of this is shown in Algorithm 1. Define the set S_{aug} as

$$\{(\mathbf{x}, y, y', \delta) : (\mathbf{x}, y) \in S, y' \in \mathcal{Y} \setminus \{y\}, \delta \in \mathcal{B}(\epsilon)\}.$$

To simplify the notation, we define the following pairwise 0-1 margin loss

$$\text{mg}_{\text{L}}\left(h(\mathbf{x}), y, y'\right) := \mathbb{I}(h(\mathbf{x}) \neq y) - \mathbb{I}(h(\mathbf{x}) \neq y').$$

In the t-th round of our algorithm, the following distribution P_t is computed over S_{aug}:

$$P_t(\mathbf{x}, y, y', \delta) \propto \exp\left(\eta \sum_{j=1}^{t-1} \text{mg}_{\text{L}}\left(h_j(\mathbf{x}+\delta), y, y'\right)\right).$$

Note that P_t uses a distribution over adversarial perturbations rather than simply choose a single adversarial perturbation. Intuitively, for any given $(\mathbf{x}, y, y')$, this distribution places more weight on perturbations that are adversarial to the ensemble constructed till now, and less weight on perturbations that are non-adversarial to the ensemble. Once we have P_t, a new classifier h_t is computed to minimize the weighted error relative to P_t and added to the ensemble. Learning h_t in this way helps us fix the mistakes of the past classifiers, and eventually leads to a robust ensemble.

To get a better understanding of P_t, we consider the following optimization problem. It can be easily shown that P_t is an optimizer of this problem (Catoni, 2004; Audibert, 2009):

$$\max_{P' \in \Delta(S_{\text{aug}})} \mathbb{E}_{P'}\left[\sum_{j=1}^{t-1} \text{mg}_{\text{L}}\left(h_j(\mathbf{x}+\delta), y, y'\right)\right] - \frac{KL(P'||P_1)}{\eta}$$

where $\Delta(S_{\text{aug}})$ is the set of all probability distributions over S_{aug}, and $KL(P'||P_1)$ is the KL divergence between P', P_1. Here, P_1 is the uniform distribution over S_{aug}. Without the KL term, P_t would have placed all its weight on the worst possible perturbations (*i.e.,* perturbations which fool the ensemble the most). However, the presence of KL term makes P_t assign "soft weights" to points in S_{aug} based on how poorly they are classified by the existing ensemble. This regularization actually plays a key role in the convergence of our algorithm to an optimal ensemble.

Our algorithm has its roots in the framework of online learning (Hazan, 2016). We relegate related details to Appendix C. One important thing to note here is that, when $\mathcal{B}(\epsilon) = \{0\}$, our algorithm boils down to AdaBoost in binary classification setting, and to AdaBoost.MR in multiclass setting (Schapire & Singer, 1999).[2] Despite this connection, the analysis and implementation of our algorithm is significantly harder than AdaBoost. This is because $\mathcal{B}(\epsilon)$ is an infinite set in the adversarial setting, which thus forms a much more challenging zero-sum game.

We now present the following Theorem which characterizes the rate of convergence of our Algorithm. In this Theorem, we assume the set of perturbations $\mathcal{B}(\epsilon)$ has finitely many elements. Later on, we discuss techniques to extend this result to continuous perturbation sets.

Theorem 4. *Let $|\mathcal{B}(\epsilon)|$ denote the number of elements in $\mathcal{B}(\epsilon)$. Suppose Algorithm 1 is run with $\eta = \sqrt{\frac{\log(nK|\mathcal{B}(\epsilon)|)}{T}}$. Then the ensemble $h_{Q(T)}$ output by the algorithm is an approximate optimizer of Equation 2. In particular, the* minimum robust margin *of $h_{Q(T)}$ is $O(T^{-1/2})$ close to the best possible robust margin:*

$$\max_{Q \in \Delta(\mathcal{H})} \text{mg}_{\text{rob}}\left(Q, S\right) \leq \text{mg}_{\text{rob}}\left(Q(T), S\right) + \xi(T),$$

where $\xi(T) = 2\sqrt{\frac{\log(nK|\mathcal{B}(\epsilon)|)}{T}}$. Moreover, the mixture distribution $P_{avg} = \sum_{t=1}^{T} \frac{1}{T} P_t$ is close to being adversarial;

[2] MRBOOST and AdaBoost only differ in the choice of η.

that is, the distribution places most of its weight on the worst possible adversarial perturbations of $h_{Q(T)}$

$$\max_{P\in\Delta(S_{aug})} \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P}\left[\sum_{t=1}^{T}\frac{1}{T}\mathrm{mg}_{\mathrm{L}}\left(h_t(\mathbf{x}+\delta),y,y'\right)\right]$$
$$\leq \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P_{avg}}\left[\sum_{t=1}^{T}\frac{1}{T}\mathrm{mg}_{\mathrm{L}}\left(h_t(\mathbf{x}+\delta),y,y'\right)\right]+\xi(T).$$

The above Theorem shows that the ensemble $h_{Q(T)}$ output by the algorithm is $O(T^{-1/2})$ close to the max-margin solution. The Theorem also shows that the mixture distribution P_{avg} concentrates around the worst adversarial examples of $h_{Q(T)}$.

Extension to continuous perturbation sets. The analysis technique used in Theorem 4 doesn't extend to the continuous case. This is mainly because our objective function $\mathrm{mg}_{\mathrm{L}}\left(h(\mathbf{x}),y,y'\right)$ is a discontinuous function of $\mathbf{x}$, and for exponential weights to obtain non-vacuous regret bound in continuous case, the loss functions need to be continuous. A work around for this is to consider a discretization of $\mathcal{B}(\epsilon)$ and run Algorithm 1 on the discretized set. To be precise, let $\mathcal{B}_\kappa(\epsilon)$ be the κ-cover of $\mathcal{B}(\epsilon)$[3]. For ℓ_p balls with $p\geq 1$, $|\mathcal{B}_\kappa(\epsilon)| = \Theta((c/\kappa)^d)$, for some positive constant c. Under mild assumptions on the hypothesis class $\mathcal{H}$, it can be shown that any Nash equilibrium (NE) of the discretized problem will be approximate NE of the original problem in Equation (2). Consequently, it suffices to solve the discretized problem using Algorithm 1. From Theorem 4, it can be seen that for appropriate choice of κ (= $O(T^{-1})$), Algorithm 1 converges to an optimum at $O(d\log(nKT)T^{-1/2})$ rate.

3.3. Practical MRBOOST for Neural Networks

Note that lines 4, 5 of Algorithm 1 are computationally expensive to implement, especially when $\mathcal{H}, \mathcal{B}(\epsilon)$ are not finite element sets. This intractability arises due to the presence of the discontinuous margin loss $\mathrm{mg}_{\mathrm{L}}\left(h(\mathbf{x}),y,y'\right)$ in the optimization and sampling objectives. We now attempt to make this algorithm computationally tractable by replacing $\mathrm{mg}_{\mathrm{L}}\left(h(\mathbf{x}),y,y'\right)$ with a smooth and differentiable surrogate loss.

We focus on neural network score-based base classifiers: $\mathcal{G} = \{g_\theta : \theta\in\Theta\subseteq\mathbb{R}^D\}$, where $g_\theta : \mathcal{X}\to\mathbb{R}^K$ is a neural network parameterized by θ.

To begin with, we introduce the following differentiable surrogate for pairwise margin loss $\mathrm{mg}_{\mathrm{L}}\left(h(\mathbf{x}),y,y'\right)$ (which serves a key role in our boosting framework), and which we refer to as margin cross entropy loss:

[3]Note that running the algorithm on the $\mathcal{B}_\kappa(\epsilon)$ is infeasible in practice. This discretization is done purely for the sake of analysis. In practice, one can run the algorithm on the original set $\mathcal{B}(\epsilon)$.

Algorithm 2 MRBOOST.NN

1: **Input:** training data S, boosting iterations T, learning rate η, SGD iterations E, SGD step size γ, sampling sub-routine: SAMPLER.
2: **for** $t = 1\dots T$ **do**
3: $\quad\theta_t \leftarrow \begin{cases}\text{random initialization} & \text{(RNDINIT)}\\ \theta_{t-1} & \text{(PERINIT)}\end{cases}$
4: $\quad$**for** $e = 1\dots E$ **do**
5: $\quad\quad$Generate mini-batch
$$\{(\mathbf{x}_b,y_b,y'_b,\delta_b)\}_{b=1}^B \leftarrow \text{SAMPLER}\left(S,\{\theta_j\}_{j=1}^{t-1},\eta\right)$$
6: $\quad\quad$Update g_{θ_t} using SGD:
$$\theta_t \leftarrow \theta_t - \frac{\gamma}{B}\sum_{b=1}^{B}\nabla_\theta \ell_{\mathrm{MCE}}(g_{\theta_t}(\mathbf{x}_b+\delta_b),y_b,y'_b).$$
7: $\quad$**end for**
8: **end for**
9: **Output:** Let $Q(T)$ be the uniform distribution over $\{g_{\theta_t}\}_{t=1\dots T}$. Output the classifier $g^{\text{am}}_{Q(T)}(\mathbf{x})$.

Algorithm 3 SAMPLER.EXP (Exponential)

Input: training data S, base models $\{\theta_j\}_{j=1}^{t-1}$, learning rate η.
Randomly sample a batch of points $\{(\mathbf{x}_b,y_b,y'_b,\delta_b)\}_{b=1}^B$ from the following distribution:
$$P_t(\mathbf{x},y,y',\delta)\propto\exp\left(\eta\ell_{\mathrm{MCE}}\left(\sum_{j=1}^{t-1}g_{\theta_j}(\mathbf{x}+\delta),y,y'\right)\right).$$
Output: $\{(\mathbf{x}_b,y_b,y'_b,\delta_b)\}_{b=1}^B$

$$\ell_{\mathrm{MCE}}(g_\theta(\mathbf{x}),y,y') := \ell_{\mathrm{CE}}(g_\theta(\mathbf{x}),y) + \ell_{\mathrm{CE}}(-g_\theta(\mathbf{x}),y').$$

For binary classification, ℓ_{MCE} is equivalent to ℓ_{CE} (to be precise, $\ell_{\mathrm{MCE}} = 2\times\ell_{\mathrm{CE}}$). However, both the losses differ when $K > 2$. Our margin cross entropy loss encourages $[g_\theta(\mathbf{x})]_y$ to be large while simultaneously forcing $[g_\theta(\mathbf{x})]_{y'}$ to be small, thus increasing the pairwise margin between the two logits. This is in contrast to standard cross entropy loss which doesn't have an explicit term for y' and doesn't necessarily increase the pairwise margin (see Figure 1). In our experiments, we show that ℓ_{MCE} loss is interesting even outside the context of margin boosting. In particular, we show that using ℓ_{MCE} in the objectives of existing defenses significantly improves their performance.

Remark 5. Several recent works have showed that performing logistic regression on linearly separable data leads to max-margin solutions (Soudry et al., 2018). At a first glance, these results seem to suggest that ℓ_{CE} loss suffices for max-margin solutions. However, it should be noted that these works study linear classifiers in the binary classification setting. It is not immediately clear if these results extend to non-linear classifiers in the multi-class setting.

Algorithm 4 SAMPLER.ALL

Input: training data S, base models $\{\theta_j\}_{j=1}^t$.
$\widehat{S}_B \leftarrow \{\}$
Uniformly sample a batch of points $\{(\mathbf{x}_b, y_b)\}_{b=1}^B$ from S.
for $b = 1 \dots B$ **do**
  compute δ_b as

$$\delta_b \in \underset{\delta \in \mathcal{B}(\epsilon)}{\operatorname{argmax}} \sum_{y' \in \mathcal{Y} \setminus \{y_b\}} \ell_{\text{MCE}} \left(\sum_{j=1}^{t} g_{\theta_j}(\mathbf{x}_b + \delta), y_b, y' \right)$$

  $\widehat{S}_B \leftarrow \widehat{S}_B \cup \{(\mathbf{x}_b, y_b, y', \delta_b)\}_{y' \in \mathcal{Y} \setminus \{y_b\}}$.
end for
Output: $\widehat{S}_B$

We now get back to Algorithm 1 and replace the margin loss $\text{mg}_\text{L}(h(\mathbf{x}), y, y')$ in the Algorithm with the surrogate loss ℓ_{MCE}. In particular, we replace line 4 in Algorithm 1 with the following

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P_t}[\ell_{\text{MCE}}(g_\theta(\mathbf{x}+\delta), y, y')].$$

We solve this optimization problem using SGD. In each iteration of SGD, we randomly sample a mini-batch according to the probability distribution P_t and descend along the gradient of the mini-batch. One caveat here is that sampling from P_t can be computationally expensive due to the presence of $\text{mg}_\text{L}(h(\mathbf{x}), y, y')$ loss. To make this tractable, we replace it with ℓ_{MCE} and sample from the following distribution

$$P_t(\mathbf{x}, y, y', \delta) \propto \exp\left(\eta \ell_{\text{MCE}} \left(\sum_{j=1}^{t-1} g_{\theta_j}(\mathbf{x}+\delta), y, y' \right) \right).$$

Algorithm 2, when invoked with Algorithm 3 as the sampling sub-routine, describes this procedure. Notice that in line 3 of Algorithm 2, we consider two different initializations for the t-th base classifier θ_t: RNDINIT, and PERINIT. In RNDINIT, we randomly initialize θ_t using techniques such as Xavier initialization. In PERINIT (Persistent Initialization), we initialize θ_t to θ_{t-1}. This makes θ_t stand on the shoulders of its precursors, and benefits its training.

While this algorithm is computationally tractable, it can be further improved by relying on the following heuristics:

- **Aggressive Defense.** Note that P_t in SAMPLER.EXP (Algorithm 3) relies only on $\{\theta_j\}_{j=1}^{t-1}$ and not on θ_t. That is, the t-th base classifier θ_t is trained to be robust only to the "soft" adversarial examples generated from the classifiers $\{\theta_j\}_{j=1}^{t-1}$. In our experiments, we noticed that making θ_t to be also robust to its own adversarial examples makes the optimization more stable and improves its convergence speed. So, we make our sampler rely on θ_t as well.
- **Efficient Samplers.** The key computational bottleneck in our algorithm is the sampler in Algorithm 3. In order to scale up our algorithm to large datasets, we replace the sampling sub-routine in Algorithm 3 with a more efficient sampler described in Algorithm 4. This sampler approximates the "soft weight" assignments via appropriate "hard weights". To be precise, we first uniformly sample $(\mathbf{x}, y)$ from S and find a perturbation whose pairwise margin loss w.r.t all classes $y' \in \mathcal{Y} \setminus \{y\}$ is the highest, and use it to train our base classifier. In our experiments, we tried two other samplers - SAMPLER.RND, SAMPLER.MAX - which differ in the way they perform the hard weight assignment (see Appendix E.1 for a more thorough discussion on these samplers). We noticed that SAMPLER.ALL is the best performing (in terms of accuracy and speed) variant among the three, and we use it in all our experiments.

We provide a PyTorch-style pseudocode of our algorithm (SAMPLER.ALL variant) for training a single network:

```
# net: classification neural network
# attack: Sampler.ALL adversarial attack (uses PGD)
# num_classes: the number of classes for the task

import torch.nn.functional as F
for inputs, targets in trainloader:
    adv_inp = attack(net, inputs, targets, num_classes)

    # standard adversarial training
    adv_outputs = net(adv_inp)
    loss = F.cross_entropy(adv_outputs, targets)

    if use_MCE:  # our method
        loss2 = - F.log_softmax(-adv_outputs, dim=1)
        loss2 *= (1 - F.one_hot(targets, num_classes))
        loss2 = loss2.mean() / (num_classes - 1)
        loss = loss + loss2

    # optimization step
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

As can be seen from the pseudocode, our algorithm does not need extra forward passes. It only performs a few simple operations in the logit ("adv_outputs") space. The SAMPLER.ALL attack (1st line in `for` loop) has similar runtime as the standard PGD attack and can be implemented using similar logic as in the pseudocode above. Therefore, our algorithm has only a slight computational overhead over baselines (see a comparison in Section F.1).

4. Experiments

In this section, we present experimental results showing the effectiveness of the proposed boosting technique. In addition, we demonstrate the efficacy of the proposed loss (ℓ_{MCE}) for standard adversarial training. Our focus here is on the ℓ_∞ threat model (*i.e.,* $\mathcal{B}(\epsilon)$ is the ℓ_∞ norm ball).

4.1. Effectiveness of ℓ_{MCE}

We first demonstrate the effectiveness of ℓ_{MCE} for training a single robust model. Here, we compare AT (Madry

Table 1. Experiments with ResNet-18 on different datasets. † denotes that the results are from the last epoch checkpoint.

METHOD	SVHN				CIFAR-10				CIFAR-100			
	CLEAN	ADV	CLEAN†	ADV†	CLEAN	ADV	CLEAN†	ADV†	CLEAN	ADV	CLEAN†	ADV†
AT	89.07	52.09	90.60	42.76	**82.06**	50.95	84.38	46.39	54.49	27.09	57.22	22.96
AT + MCE	**89.69**	**53.25**	**90.89**	**46.78**	81.13	**51.86**	84.37	**48.47**	54.37	**28.25**	57.01	**24.93**

Table 2. Experiments with WideResNet-34-10 on CIFAR10.

METHOD	CLEAN	FGSM	CW	PGD-20	PGD-100	AUTOATTACK
AT	**86.31**	64.01	53.28	54.12	53.75	50.13
AT + MCE	85.56	64.20	53.46	**55.40**	**55.14**	**52.07**
TRADES	83.25	62.48	49.51	54.97	54.80	51.92
TRADES + MCE	84.76	**64.63**	49.49	**56.23**	**55.99**	52.40
MART	83.12	63.68	52.57	55.75	55.49	50.85
MART + MCE	**83.65**	**64.3**	**54.24**	**56.31**	**56.15**	**52.81**
GAIR	83.91	65.79	49.44	58.99	58.97	44.04
GAIR + MCE	**84.55**	**67.96**	**49.94**	**61.79**	**61.93**	44.22
AWP	**85.32**	65.89	55.40	57.37	57.08	53.67
AWP + MCE	84.97	**66.53**	**56.23**	**58.40**	**58.12**	**54.69**

et al., 2017) with AT + MCE, which is a defense technique obtained by replacing ℓ_{CE} in AT with ℓ_{MCE}, and relying on SAMPLER.ALL to generate mini-batches during training. We consider three datasets: SVHN, CIFAR-10 and CIFAR-100. We train ResNet-18 using SGD with 0.9 momentum for 100 epochs. The initial leaning rate is set to 0.1 and it is further decayed by the factor of 10 at the 50-th and 75-th epoch. The batch size is set to 128 in this work. We also use a weight decay of 5×10^{-4}. For the ℓ_∞ threat model, we use $\epsilon = 8/255$. The step size of attacks is $1/255$ for SVHN and $2/255$ for CIFAR-10 and CIFAR-100. PGD-10 (Madry et al., 2017) attack is used for adversarial training and PGD-20 is used during testing period. In Table 1, we report both the best checkpoint model as well as the last checkpoint model, the gap between which is known as the robust overfitting phenomenon (Rice et al., 2020). For each checkpoint, we report its clean accuracy and adversarial accuracy. We use bold numbers when the accuracy gap is $> 0.20\%$. We can see that AT + MCE consistently improves the robustness of AT across different datasets, with only a tiny drop in clean accuracy. This indicates that ℓ_{MCE} is broadly applicable even outside the context of boosting. Note that although the computation of adversarial perturbations in SAMPLER.ALL involves all the classes $y' \in \mathcal{Y} \setminus \{y\}$, it can be computed with only one forward pass, resulting in less than 5% increase in runtime.

We now train larger capacity models, as they lead to state-of-the-art results in the literature. Concretely, we use WideResNet-34-10 on CIFAR-10 and use the same setting as Madry et al. (2017). We consider a number of state-of-the-art baseline algorithms: 1) AT (Madry et al., 2017); 2) TRADES (Zhang et al., 2019b); 3) MART (Wang et al., 2019b); 4) AWP (Wu et al., 2020), and 5) GAIR (Zhang et al., 2020c). We combine these defenses with ℓ_{MCE} to improve their robustness (see Appendix F.3 for more details). We use the same hyperparameters as stated previously, to train these defenses. With regard to attacking the trained models, we choose FGSM, CW_∞ (Carlini & Wagner, 2017), PGD-20 and PGD-100 attacks with $\epsilon = 8/255$. Additionally, we test the model performance against the AutoAttack (Croce & Hein, 2020b), which is a strong and reliable evaluation suite; stable performances under such threat models often show that the robustness of our defenses is not caused by the "obfuscated gradient" phenomenon (Athalye et al., 2018b). The results from this experiment are reported in Table 2. It can be seen that the robustness of all the baseline algorithms can be improved using ℓ_{MCE}, demonstrating the effectiveness of the proposed loss. We also notice that for GAIR, there is a remarkable drop in performance on the AutoAttack. This suggests that some form of obfuscated gradients could potentially be the reason behind its robustness.

4.2. Effectiveness of Margin-Boosting

We now present experimental results showing the effectiveness of MRBOOST.NN on CIFAR-10 (experimental results on SVHN and CIFAR-100 can be found in Appendix F.4). In our experiments, we use SAMPLER.ALL to generate mini-batches (see Appendix F.4 for more details). We consider the following baselines: 1) larger models trained end-to-end using AT, and 2) ROBBOOST, which is the boosting algorithm of Abernethy et al. (2021) (see Appendix F.5 for the details). For the boosting techniques, we set the total number of boosting iterations to 5, and choose ResNet-18 as our base classifier. For end-to-end trained larger models, we consider: 1) an ensemble of five ResNet-18 models

Table 3. Boosting experiments on CIFAR-10 with ResNet-18 being the base classifier.

METHOD	ITERATION 1		ITERATION 2		ITERATION 3		ITERATION 4		ITERATION 5	
	CLEAN	ADV	CLEAN	ADV	CLEAN	ADV	CLEAN	ADV	CLEAN	ADV
WIDER MODEL	82.61	51.73	—	—	—	—	—	—	—	—
DEEPER MODEL	82.67	52.32	—	—	—	—	—	—	—	—
ROBBOOST + RNDINIT	82.00	51.05	84.58	49.95	83.87	51.66	82.56	52.72	81.44	52.92
ROBBOOST + PERINIT	82.18	50.97	85.60	50.13	84.59	51.77	84.21	52.79	82.78	53.28
MRBOOST.NN + RNDINIT	81.04	51.83	84.61	52.68	84.93	53.51	85.01	53.95	85.35	54.13
MRBOOST.NN + PERINIT	81.34	51.92	84.97	52.97	85.28	53.62	85.99	54.26	86.16	54.42

("wider model"), and 2) a ResNet-152 model which has slightly larger number of parameters than the wider model ("deeper model"). The hyperparameter settings, including the threat model setup and optimization schedule for each boosting iteration, are identical to those used in the earlier experiments.

Table 3 presents the results from this experiment. For all the techniques, we report the best robustness checkpoint results after each boosting iteration (for end-to-end trained models, number of boosting iterations is 1, and longer training could not bring further improvement (Pang et al., 2021)). Several conclusions can be drawn from Table 3. Firstly, MRBOOST.NN has significantly better performance than all the baselines, on both clean and adversarial accuracies. This shows the effectiveness of our margin-boosting framework over the boosting framework of Abernethy et al. (2021). It also shows that boosting techniques can outperform end-to-end trained larger networks. Next, persistent initialization (PERINIT) not only improves MRBOOST.NN, but also ROBBOOST (Abernethy et al. (2021) only study RNDINIT in their work). This makes it a helpful technique for boosting in the context of deep learning.

5. Conclusion and Future Work

We proposed a margin-boosting framework for building robust ensembles. Theoretically, we showed that our boosting framework is optimal and derived a computationally efficient boosting algorithm (MRBOOST.NN) from our framework. Through extensive empirical evaluation, we showed that our algorithm outperforms both existing boosting techniques and larger models trained end-to-end.

Future Work. We believe the performance of our algorithm can be further improved by designing better samplers, and in particular, faster techniques to implement Algorithm 3. In our experiments, we noticed that SAMPLER.MAX achieves good robustness, but is computationally expensive (its computational complexity scales linearly with the number of classes). One interesting future direction would be to speed up SAMPLER.MAX. Next, it'd be interesting to understand, both theoretically and empirically, why boosting techniques outperform end-to-end trained larger models.

Acknowledgement

We thank Chen Dan, Feng Zhu and Mounica for helpful discussions. Dinghuai Zhang thanks the never-ending snow storm in Montreal for preventing him from any form of outdoor activity. Hongyang Zhang is supported in part by an NSERC Discovery Grant. Aaron Courville thanks the support of Microsoft Research, Hitachi and CIFAR. Yoshua Bengio acknowledges the funding from CIFAR, Samsung, IBM and Microsoft. Pradeep Ravikumar acknowledges the support of ARL and NSF via IIS-1909816.

References

Abernethy, J., Awasthi, P., and Kale, S. A multiclass boosting framework for achieving fast and provable adversarial robustness. *arXiv preprint arXiv:2103.01276*, 2021.

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. Square attack: a query-efficient black-box adversarial attack via random search. *ArXiv*, abs/1912.00049, 2020.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018a.

Athalye, A., Carlini, N., and Wagner, D. A. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018b.

Audibert, J.-Y. Fast learning rates in statistical inference through aggregation. *The Annals of Statistics*, 37(4): 1591–1646, 2009.

Bartlett, P., Freund, Y., Lee, W. S., and Schapire, R. E. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5): 1651–1686, 1998.

Bartlett, P. L. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 44(2):525–536, 1998.

Blum, A., Dick, T., Manoj, N., and Zhang, H. Random smoothing might be unable to certify ℓ_∞ robustness for high-dimensional images. *Journal of Machine Learning Research*, 21:1–21, 2020.

Breiman, L. Prediction games and arcing algorithms. *Neural computation*, 11(7):1493–1517, 1999.

Bubeck, S., Lee, Y. T., and Eldan, R. Kernel-based methods for bandit convex optimization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 72–85, 2017.

Bubeck, S., Lee, Y. T., Price, E., and Razenshteyn, I. Adversarial examples from computational constraints. In *International Conference on Machine Learning*, pp. 831–840. PMLR, 2019.

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

Carmon, Y., Raghunathan, A., Schmidt, L., Liang, P., and Duchi, J. C. Unlabeled data improves adversarial robustness. *ArXiv*, abs/1905.13736, 2019.

Catoni, O. *Statistical learning theory and stochastic optimization: Ecole d'Eté de Probabilités de Saint-Flour, XXXI-2001*, volume 1851. Springer Science & Business Media, 2004.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, learning, and games*. Cambridge university press, 2006.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320. PMLR, 2019.

Croce, F. and Hein, M. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, 2020a.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *ArXiv*, abs/2003.01690, 2020b.

Fan, K. Minimax theorems. *Proceedings of the National Academy of Sciences of the United States of America*, 39 (1):42, 1953.

Fawzi, A., Fawzi, O., and Frossard, P. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508, 2018.

Freund, Y. and Schapire, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pp. 23–37. Springer, 1995.

Freund, Y., Schapire, R. E., et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pp. 148–156. Citeseer, 1996.

Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Hazan, E. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016.

Huang, F., Ash, J., Langford, J., and Schapire, R. Learning deep resnet blocks sequentially using boosting theory. *arXiv preprint arXiv:1706.04964*, 2017.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019.

Kalai, A. and Vempala, S. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

Kariyappa, S. and Qureshi, M. K. Improving adversarial robustness of ensembles with diversity training. *arXiv preprint arXiv:1901.09981*, 2019.

Khim, J. and Loh, P.-L. Adversarial risk bounds via function transformation. *arXiv preprint arXiv:1810.09519*, 2018.

Krichene, W., Balandat, M., Tomlin, C., and Bayen, A. The hedge algorithm on a continuum. In *International Conference on Machine Learning*, pp. 824–832, 2015.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Mason, L., Bartlett, P. L., and Baxter, J. Improved generalization through explicit optimization of margins. *Machine Learning*, 38(3):243–255, 2000a.

Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pp. 512–518, 2000b.

McMahan, H. B. A survey of algorithms and analysis for adaptive online learning. *The Journal of Machine Learning Research*, 18(1):3117–3166, 2017.

Meng, Y., Su, J., O'Kane, J., and Jamshidi, P. Athena: A framework based on diverse weak defenses for building adversarial defense. *arXiv preprint arXiv:2001.00308*, 2020.

Mukherjee, I. and Schapire, R. E. A theory of multiclass boosting. *Journal of Machine Learning Research*, 14 (Feb):437–497, 2013.

Nitanda, A. and Suzuki, T. Functional gradient boosting based on residual network perception. *arXiv preprint arXiv:1802.09031*, 2018.

Pang, T., Xu, K., Du, C., Chen, N., and Zhu, J. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pp. 4970–4979. PMLR, 2019.

Pang, T., Yang, X., Dong, Y., Su, H., and Zhu, J. Bag of tricks for adversarial training. *ArXiv*, abs/2010.00467, 2021.

Pinot, R., Ettedgui, R., Rizk, G., Chevaleyre, Y., and Atif, J. Randomization matters. how to defend against strong adversarial attacks. In *ICML*, 2020.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Rätsch, G., Warmuth, M. K., and Shawe-Taylor, J. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6(12), 2005.

Rice, L., Wong, E., and Kolter, J. Z. Overfitting in adversarially robust deep learning. In *ICML*, 2020.

Salman, H., Yang, G., Li, J., Zhang, P., Zhang, H., Razenshteyn, I., and Bubeck, S. Provably robust deep learning via adversarially trained smoothed classifiers. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 11292–11303, 2019.

Schapire, R. E. and Singer, Y. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37 (3):297–336, 1999.

Sen, S., Ravindran, B., and Raghunathan, A. EMPIR: ensembles of mixed precision deep networks for increased robustness against adversarial attacks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJem3yHKwH.

Shi, B., Zhang, D., Dai, Q., Zhu, Z., Mu, Y., and Wang, J. Informative dropout for robust representation learning: A shape-bias perspective. *ArXiv*, abs/2008.04254, 2020.

Sion, M. On general minimax theorems. *Pacific Journal of mathematics*, 8(1):171–176, 1958.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Suggala, A., Liu, B., and Ravikumar, P. Generalized boosting. *Advances in neural information processing systems*, 2020.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

Telgarsky, M. The fast convergence of boosting. In *NIPS*, pp. 1593–1601, 2011.

Tramer, F., Carlini, N., Brendel, W., and Madry, A. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33, 2020.

Verma, G. and Swami, A. Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. *Advances in Neural Information Processing Systems*, 32:8646–8656, 2019.

Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., and Gu, Q. On the convergence and robustness of adversarial training. *ArXiv*, abs/2112.08304, 2019a.

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2019b.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5286–5295. PMLR, 2018.

Wu, D., Xia, S., and Wang, Y. Adversarial weight perturbation helps robust generalization. *arXiv: Learning*, 2020.

Yang, Z., Li, L., Xu, X., Kailkhura, B., Xie, T., and Li, B. On the certified robustness for ensemble models and beyond. *arXiv preprint arXiv:2107.10873*, 2021.

Yin, D., Kannan, R., and Bartlett, P. Rademacher complexity for adversarially robust generalization. In *International conference on machine learning*, pp. 7085–7094. PMLR, 2019.

Zhang, D., Zhang, T., Lu, Y., Zhu, Z., and Dong, B. You only propagate once: Accelerating adversarial training via maximal principle. *ArXiv*, abs/1905.00877, 2019a.

Zhang, D., Ye, M., Gong, C., Zhu, Z., and Liu, Q. Black-box certification with randomized smoothing: A functional optimization based framework. *ArXiv*, abs/2002.09169, 2020a.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pp. 7472–7482. PMLR, 2019b.

Zhang, J., Xu, X., Han, B., Niu, G., zhen Cui, L., Sugiyama, M., and Kankanhalli, M. S. Attacks which do not kill training make adversarial learning stronger. In *ICML*, 2020b.

Zhang, J., Zhu, J., Niu, G., Han, B., Sugiyama, M., and Kankanhalli, M. S. Geometry-aware instance-reweighted adver-. 2020c.

A. Notation

Symbol	Description
$\mathbf{x}$	feature vector
y	label
$\mathcal{X}$	domain of feature vector
$\mathcal{Y}$	domain of the label
K	number of classes in multi-class classification problem
S	training data set
P	true data distribution
P_n	empirical distribution
$h : \mathcal{X} \to \mathcal{Y}$	standard classifier
$g : \mathcal{X} \to \mathbb{R}^K$	score based classifier
ℓ_{0-1}	$0/1$ classification loss
ℓ	convex surrogate of ℓ_{0-1}
ℓ_{CE}	cross-entropy loss
ℓ_{MCE}	margin cross-entropy loss
$R(g;\ell)$	population risk of classifier g, measured w.r.t ℓ
$\widehat{R}_n(g;\ell)$	empirical risk of classifier g, measured w.r.t ℓ
$\mathcal{B}(\epsilon)$	Set of valid perturbations of an adversary
S_{aug}	$\{(\mathbf{x}, y, y', \delta) : (\mathbf{x}, y) \in S, y' \in \mathcal{Y} \setminus \{y\}, \delta \in \mathcal{B}(\epsilon)\}$
$\tilde{S}_{\text{aug}}$	$\{(\mathbf{x}, y, y') : (\mathbf{x}, y) \in S, y' \in \mathcal{Y} \setminus \{y\}\}$
$R^{\text{adv}}(g;\ell)$	population adversarial risk of classifier g, measured w.r.t ℓ
$\widehat{R}^{\text{adv}}_n(g;\ell)$	empirical adversarial risk of classifier g, measured w.r.t ℓ
$\mathcal{H}$	compact set of base classifiers
$\mathcal{G}$	compact set of score-based base classifiers
$\Delta(\mathcal{H})$	set of all probability distributions over $\mathcal{H}$
Q	probability distribution over base classifiers $\mathcal{H}$
h_Q	ensemble constructed by placing probability distribution Q over elements in $\mathcal{H}$

Table 4. Margin related terminology

Symbol	Description
$\mathrm{mg}_{\mathrm{L}}(h(\mathbf{x}), y, y')$	$\mathbb{I}(h(\mathbf{x}) \neq y) - \mathbb{I}(h(\mathbf{x}) \neq y')$
$\mathrm{mg}(Q, \mathbf{x}, y)$	$[h_Q(\mathbf{x})]_y - \max_{y' \neq y}[h_Q(\mathbf{x})]_{y'}$
$\mathrm{mg}(Q, S)$	$\min_{(\mathbf{x},y)\in S} \mathrm{mg}(Q, \mathbf{x}, y)$
$\mathrm{mg}_{\mathrm{rob}}(Q, S)$	$\min_{(\mathbf{x},y)\in S} \min_{\delta \in \mathcal{B}(\epsilon)} \mathrm{mg}(Q, \mathbf{x}+\delta, y)$

B. Proofs of Section 3.1

B.1. Intermediate Results

We first present the following intermediate result that helps us prove Theorems 1, 2 and Proposition 3.

Lemma 6. *The following three statements are equivalent:*

1. *There exists a $Q \in \Delta(\mathcal{H})$ such that the ensemble h^{am}_Q achieves 100% adversarial accuracy on the training set S.*
2. *For any maximizer Q^* of Equation (2), the ensemble $h^{\mathrm{am}}_{Q^*}$ achieves 100% adversarial accuracy on the training set S.*
3. *The hypothesis class $\mathcal{H}$ satisfies the following condition: for any probability distribution P' over points in the set*

$S_{aug} \coloneqq \{(\mathbf{x}, y, y', \delta) : (\mathbf{x}, y) \in S, y' \in \mathcal{Y} \setminus \{y\}, \delta \in \mathcal{B}(\epsilon)\}$, there exists a classifier $h \in \mathcal{H}$ which achieves slightly-better-than-random performance on P'

$$\mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P'}[\mathbb{I}(h(\mathbf{x}+\delta) = y)] \geq \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P'}[\mathbb{I}(h(\mathbf{x}+\delta) = y')] + \tau.$$

Here $\tau > 0$ is some constant.

Proof. To prove the Lemma, it suffices to show that $(1) \iff (2), (2) \iff (3)$.

- **Proof of** $(1) \implies (2)$**.** Let Q be the ensemble which achieves 100% adversarial accuracy on S. We first show that this is equivalent to: $\mathrm{mg}_{\mathrm{rob}}(Q, S) > 0$. To see this, first note that the following should hold for any $(\mathbf{x}, y) \in S$

$$h_Q^{\mathrm{am}}(\mathbf{x}+\delta) = y, \quad \text{for all } \delta \in \mathcal{B}(\epsilon).$$

Now consider the following, for any $(\mathbf{x}, y) \in S$

$$\begin{aligned}
&\forall \delta \in \mathcal{B}(\epsilon),\ h_Q^{\mathrm{am}}(\mathbf{x}+\delta) = y \\
&\overset{(a)}{\iff} \forall \delta \in \mathcal{B}(\epsilon),\ [h_Q(\mathbf{x}+\delta)]_y > \max_{y' \neq y} [h_Q(\mathbf{x}+\delta)]_{y'} \\
&\iff \min_{\delta \in \mathcal{B}(\epsilon)} [h_Q(\mathbf{x}+\delta)]_y - \max_{y' \neq y} [h_Q(\mathbf{x}+\delta)]_{y'} > 0 \\
&\iff \min_{\delta \in \mathcal{B}(\epsilon)} \mathrm{mg}(Q, \mathbf{x}+\delta, y) > 0,
\end{aligned}$$

where (a) follows from the definition of $h_Q^{\mathrm{am}}(\mathbf{x})$. Since the last statement in the above display holds for any $(\mathbf{x}, y) \in S$, we have

$$\begin{aligned}
&\min_{(\mathbf{x},y)\in S} \min_{\delta \in \mathcal{B}(\epsilon)} \mathrm{mg}(Q, \mathbf{x}+\delta, y) > 0 \\
&\iff \mathrm{mg}_{\mathrm{rob}}(Q, S) > 0.
\end{aligned}$$

This shows that statement (1) is equivalent to: $\mathrm{mg}_{\mathrm{rob}}(Q, S) > 0$. Now, let Q^* be any maximizer of Equation (2). Since $\mathrm{mg}_{\mathrm{rob}}(Q^*, S) \geq \mathrm{mg}_{\mathrm{rob}}(Q, S)$ and $\mathrm{mg}_{\mathrm{rob}}(Q, S) > 0$, we have

$$\mathrm{mg}_{\mathrm{rob}}(Q^*, S) > 0.$$

From our above argument, we know that this is equivalent to saying that the ensemble $h_{Q^*}^{\mathrm{am}}$ achieves 100% adversarial accuracy on training set S. This shows that $(1) \implies (2)$.

- **Proof of** $(2) \implies (1)$**.** This statement trivially holds.

- **Proof of** $(2) \iff (3)$**.** Suppose $\mathcal{H}$ satisfies the weak learning condition stated in statement (3). Then we have

$$\begin{aligned}
&\forall P' \in \Delta(S_{\text{aug}}), \exists h \in \mathcal{H}, \text{ such that } \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P'}[\mathbb{I}(h(\mathbf{x}+\delta) = y) - \mathbb{I}(h(\mathbf{x}+\delta) = y')] \geq \tau \\
&\iff \forall P' \in \Delta(S_{\text{aug}}),\ \max_{h \in \mathcal{H}} \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P'}[\mathbb{I}(h(\mathbf{x}+\delta) = y) - \mathbb{I}(h(\mathbf{x}+\delta) = y')] \geq \tau \\
&\iff \min_{P' \in \Delta(S_{\text{aug}})} \max_{h \in \mathcal{H}} \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P'}[\mathbb{I}(h(\mathbf{x}+\delta) = y) - \mathbb{I}(h(\mathbf{x}+\delta) = y')] \geq \tau \\
&\overset{(a)}{\iff} \min_{P' \in \Delta(S_{\text{aug}})} \max_{Q \in \Delta(\mathcal{H})} \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P', h\sim Q} [\mathbb{I}(h(\mathbf{x}+\delta) = y) - \mathbb{I}(h(\mathbf{x}+\delta) = y')] \geq \tau \\
&\overset{(b)}{\iff} \max_{Q \in \Delta(\mathcal{H})} \min_{P' \in \Delta(S_{\text{aug}})} \mathbb{E}_{h\sim Q, (\mathbf{x},y,y',\delta)\sim P'} [\mathbb{I}(h(\mathbf{x}+\delta) = y) - \mathbb{I}(h(\mathbf{x}+\delta) = y')] \geq \tau \\
&\iff \max_{Q \in \Delta(\mathcal{H})} \min_{(\mathbf{x},y,y',\delta) \in S_{\text{aug}}} \mathbb{E}_{h\sim Q} [\mathbb{I}(h(\mathbf{x}+\delta) = y) - \mathbb{I}(h(\mathbf{x}+\delta) = y')] \geq \tau,
\end{aligned}$$

where (a) follows since the optimum of a linear program over a convex hull can always be attained at an extreme point, and (b) follows from our compactness assumptions on $\mathcal{H}, \mathcal{B}(\epsilon)$ and from Sion's minimax theorem which says that the

max-min and min-max values of convex-concave games over compact domains are equal to each other (Sion, 1958). (b) can also be obtained using Ky Fan's minimax theorem (Fan, 1953). Continuing, we get

$$\begin{aligned}
&\forall P' \in \Delta(S_{\text{aug}}), \exists h \in \mathcal{H}, \text{ such that } \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P'}[\mathbb{I}(h(\mathbf{x}+\delta)=y) - \mathbb{I}(h(\mathbf{x}+\delta)=y')] \geq \tau \\
&\iff \max_{Q\in\Delta(\mathcal{H})} \min_{(\mathbf{x},y,y',\delta)\in S_{\text{aug}}} \mathbb{E}_{h\sim Q}\left[\mathbb{I}(h(\mathbf{x}+\delta)=y) - \mathbb{I}(h(\mathbf{x}+\delta)=y')\right] \geq \tau \\
&\iff \max_{Q\in\Delta(\mathcal{H})} \text{mg}_{\text{rob}}\left(Q, S\right) \geq \tau \\
&\overset{(c)}{\iff} \max_{Q\in\Delta(\mathcal{H})} \text{mg}_{\text{rob}}\left(Q, S\right) > 0,
\end{aligned}$$

where the reverse implication in (c) follows from our assumption that $\mathcal{H}, \mathcal{B}(\epsilon)$ are compact sets (Sion, 1958). In our proof for $(1) \implies (2)$ we showed that the last statement in the above display is equivalent to saying h_Q^{am} achieves 100% adversarial accuracy on S. This shows that $(2) \iff (3)$.

□

B.2. Proof of Theorem 1

The proof of this Theorem follows directly from Lemma 6. In particular, the equivalence between statements (2) and (3) in the Lemma proves the Theorem.

B.3. Proof of Theorem 2

The proof of this Theorem again follows from Lemma 6. Suppose there exists a boosting framework which can guarantee 100% accurate solution with a milder condition on $\mathcal{H}$ than the condition in statement (3) of Lemma 6. Referring to this milder condition as statement 4, we have: statement $4 \implies \{$statement 1 in Lemma 6$\}$. Given the equivalence between statements 1 and 3 in Lemma 6, we can infer the following: statement $4 \implies$ statement 3. This shows that any hypothesis class satisfying statement 4 should also satisfy statement 3. Thus statement 4 can't be milder than the condition in statement 3. This shows that margin-boosting is optimal.

B.4. Proof of Proposition 3

The first part of the proposition on proving $\text{WL}_{\text{RobBoost}} \implies \text{WL}_{\text{MRBoost}}$ follows from Theorem 2. So, here we focus on proving $\text{WL}_{\text{MRBoost}} \not\Longrightarrow \text{WL}_{\text{RobBoost}}$. To prove this statement, it suffices to construct a base hypothesis class $\mathcal{H}$ which satisfies $\text{WL}_{\text{MRBoost}}$ for some $\tau > 0$, but doesn't satisfy $\text{WL}_{\text{RobBoost}}$ for any $\tau > 0$. Here is how we construct such a $\mathcal{H}$. We consider the following simple setting:

$$\mathcal{X} = \mathbb{R}, \mathcal{Y} = \{0, 1\}, \mathcal{B}(\epsilon) = \{\delta : |\delta| \leq \epsilon\}.$$

We let $\epsilon = 1$, and $S = \{(0, 1)\}$; that is, we only have 1 sample in our training data with feature $x = 0$ and label $y = 1$. Our base hypothesis class is the set $\mathcal{H} = \{h_\theta\}_{\theta\in[-1,0.9]}$, where $h_\theta : \mathcal{X} \to \mathcal{Y}$ is defined as

$$h_\theta(x) = \begin{cases} 0, & \text{if } x \in [\theta, \theta + 0.1] \\ 1, & \text{otherwise} \end{cases}.$$

In this setting, the weak learning condition $\text{WL}_{\text{RobBoost}}$ can be rewritten as follows: there exists a classifier $h_\theta \in \mathcal{H}$ which satisfies the following for some $\tau > 0$

$$\mathbb{I}(\forall \delta \in \mathcal{B}(\epsilon) : h_\theta(\delta) = 1) \geq \frac{1+\tau}{2}.$$

Based on our construction of $\mathcal{H}$, it is easy to see that there is no $h \in \mathcal{H}$ which can satisfy the above condition for any $\tau > 0$. So, $\text{WL}_{\text{RobBoost}}$ is not satisfied in this setting. Now consider the weak learning condition $\text{WL}_{\text{MRBoost}}$. This can be rewritten as follows: for any probability distribution P' over points in the set $\mathcal{B}(1)$, there exists a classifier $h_\theta \in \mathcal{H}$ which satisfies the following for some $\tau > 0$:

$$\mathbb{E}_{\delta\sim P'}[\mathbb{I}(h_\theta(\delta) = 1)] \geq \frac{1+\tau}{2}.$$

Algorithm 5 Exponential Weights Algorithm (EXP)

1: **Input:** learning rate η
2: **for** $t = 1 \dots T$ **do**
3: Sample $\mathbf{z}_t$ from the following distribution

$$P_t(\mathbf{z}) \propto \exp\left(-\eta \sum_{i=1}^{t-1} f_i(\mathbf{z})\right).$$

4: Play $\mathbf{z}_t$ and observe loss function f_t. Suffer loss $f_t(\mathbf{z}_t)$.
5: **end for**

We now show that for our choice of $\mathcal{H}$, the above weak learning condition holds for $\tau = 0.2$. To see this, divide $\mathcal{B}(1)$ into the following (overlapping) intervals of width 0.1: $[-1, -0.9], [-0.9, -0.8], \dots [0.8, 0.9], [0.9, 1]$. There are 20 such intervals. It is easy to see that for any probability distribution P' over $\mathcal{B}(1)$, there exists atleast one interval in which the the probability distribution P' has mass $\leq \frac{1}{10}$. By choosing a $h_\theta \in \mathcal{H}$ which assigns 0 to points in that particular interval (and 1 to all other points), we can show that the $\mathbb{E}_{\delta \sim P'}[\mathbb{I}(h_\theta(\delta) = 1)] \geq \frac{1.2}{2}$. This shows that $\text{WL}_{\text{MRBoost}}$ holds in this setting with $\tau = 0.2$. This shows that $\text{WL}_{\text{MRBoost}} \implies \text{WL}_{\text{RobBoost}}$.

C. Design of Algorithm 1

As previously mentioned, Algorithm 2 has its roots in the framework of online learning (Hazan, 2016). In this section, we present necessary background on online learning, game theory, and derive our algorithm for solving the max-min game in Equation (2).

C.1. Online Learning

The online learning framework can be seen as a repeated game between a learner/decision-maker and an adversary. In this framework, in each round t, the learner makes a prediction $\mathbf{z}_t \in \mathcal{Z}$, where $\mathcal{Z} \subseteq \mathbb{R}^d$, and the adversary chooses a loss function $f_t : \mathcal{Z} \to \mathbb{R}$ and observe each others actions. The goal of the learner is to choose a sequence of actions $\{\mathbf{z}_t\}_{t=1}^T$ so that the cumulative loss $\sum_{t=1}^T f_t(\mathbf{z}_t)$ is minimized. The benchmark with which the cumulative loss will be compared is called the best fixed policy in hindsight, which is given by $\inf_{\mathbf{z} \in \mathcal{Z}} \sum_{t=1}^T f_t(\mathbf{z})$. This results in the following notion of *regret*, which the learner aims to minimize

$$\sum_{t=1}^T f_t(\mathbf{z}_t) - \inf_{\mathbf{z} \in \mathcal{Z}} \sum_{t=1}^T f_t(\mathbf{z}).$$

Observe that there is a very simple strategy for online learning that guarantees 0 regret, but under the provision that f_t *was known* to the learner ahead of round t. Then, an optimal strategy for the learner is to predict $\mathbf{z}_t$ as simply a minimizer of $f_t(\mathbf{z})$. It is easy to see that this algorithm, known as Best Response (BR), has 0 regret. While this is an impractical algorithm in the framework of online learning, it can be used to solve min-max games, as we will see in Section C.3.

A number of practical and efficient algorithms for regret minimization have been developed in the literature of online learning (Hazan, 2016; McMahan, 2017; Krichene et al., 2015; Kalai & Vempala, 2005). In this work, we are primarily interested in the exponential weights update algorithm (Algorithm 5). In each round of this algorithm, the learner chooses its action $\mathbf{z}_t$ by uniformly sampling a point from the following distribution

$$P_t(\mathbf{z}) \propto \exp\left(-\eta \sum_{i=1}^{t-1} f_i(\mathbf{z})\right).$$

We now present the following theorem which bounds the regret of this algorithm when $\mathcal{Z}$ is finite. This is a classic result and its proof can be found in several prior works (see Bubeck et al., 2017, for example).

Theorem 7 (Regret Bound)**.** *Suppose the action space $\mathcal{Z}$ of the learner is a finite set. Moreover suppose the sequence of loss functions chosen by the adversary are uniformly bounded:* $\sup_{t \in T, \mathbf{z} \in \mathcal{Z}} |f_t(\mathbf{z})| \leq B$. *Suppose Algorithm 5 is run with* $\eta \leq \frac{1}{B}\sqrt{\frac{\log(|\mathcal{Z}|)}{T}}$. *Then its expected regret can be bounded as follows*

$$\sup_{\mathbf{z} \in \mathcal{Z}} \mathbb{E}\left[\sum_{t=1}^T f_t(\mathbf{z}_t) - \sum_{t=1}^T f_t(\mathbf{z})\right] \leq 2B\sqrt{\log(|\mathcal{Z}|)T}.$$

C.2. Game Theory

Consider the following two-player zero-sum game

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x},\mathbf{y})$$

A pair $(\mathbf{x}^*,\mathbf{y}^*)\in\mathcal{X}\times\mathcal{Y}$ is called a pure strategy Nash Equilibrium (NE) of the game, if the following holds

$$\max_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x}^*,\mathbf{y}) \le f(\mathbf{x}^*,\mathbf{y}^*) \le \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x},\mathbf{y}^*).$$

Intuitively, this says that there is no incentive for any player to change their strategy while the other player keeps hers unchanged. A pure strategy NE need not always exist though. What exists often is a mixed strategy NE (Sion, 1958), which is defined as follows. Let P^*,Q^* be probability distributions over $\mathcal{X},\mathcal{Y}$. The pair (P^*,Q^*) is called a mixed strategy NE of the above game if

$$\sup_{\mathbf{y}\in\mathcal{Y}} \mathbb{E}_{\mathbf{x}\sim P^*}\left[f(\mathbf{x},\mathbf{y})\right] \le \mathbb{E}_{\mathbf{x}\sim P^*}\left[\mathbb{E}_{\mathbf{y}\sim Q^*}\left[f(\mathbf{x},\mathbf{y})\right]\right] \le \inf_{\mathbf{x}\in\mathcal{X}} \mathbb{E}_{\mathbf{y}\sim Q^*}\left[f(\mathbf{x},\mathbf{y})\right].$$

Note that (P^*,Q^*) can also be viewed as a pure strategy NE of the following linearized game[4]

$$\inf_{P\in\Delta(\mathcal{X})} \sup_{Q\in\Delta(\mathcal{Y})} \mathbb{E}_{\mathbf{x}\sim P}\left[\mathbb{E}_{\mathbf{y}\sim Q}\left[f(\mathbf{x},\mathbf{y})\right]\right].$$

Finally, (P^*,Q^*) is called an ϵ-approximate mixed NE of the game if

$$\sup_{\mathbf{y}\in\mathcal{Y}} \mathbb{E}_{\mathbf{x}\sim P^*}\left[f(\mathbf{x},\mathbf{y})\right] - \epsilon \le \mathbb{E}_{\mathbf{x}\sim P^*}\left[\mathbb{E}_{\mathbf{y}\sim Q^*}\left[f(\mathbf{x},\mathbf{y})\right]\right] \le \inf_{\mathbf{x}\in\mathcal{X}} \mathbb{E}_{\mathbf{y}\sim Q^*}\left[f(\mathbf{x},\mathbf{y})\right] + \epsilon.$$

C.3. Deriving MRBOOST

In this work, we are interested in computing mixed strategy NE of the game in Equation (2), which we present here for convenience

$$\max_{Q\in\Delta(\mathcal{H})} \min_{(\mathbf{x},y,y',\delta)\in S_{\text{aug}}} [h_Q(\mathbf{x})]_y - [h_Q(\mathbf{x})]_{y'}.$$

This game can equivalently be written as follows

$$\max_{Q\in\Delta(\mathcal{H})} \min_{P\in\Delta(S_{\text{aug}})} [h_Q(\mathbf{x})]_y - [h_Q(\mathbf{x})]_{y'}.$$

This follows since the optimum of a linear program over a convex hull can always be attained at an extreme point. Using our definition of $\text{mg}_{\text{L}}(\cdot)$, we can rewrite the above game as

$$\min_{Q\in\Delta(\mathcal{H})} \max_{P\in\Delta(S_{\text{aug}})} \mathbb{E}_{h\sim Q}\left[\mathbb{E}_{(\mathbf{x},y,y',\delta)\in P}\left[\text{mg}_{\text{L}}\left(h(\mathbf{x}+\delta),y,y'\right)\right]\right]. \tag{3}$$

A popular and widely used approach for finding mixed NE of this game is to rely on online learning algorithms (Hazan, 2016; Cesa-Bianchi & Lugosi, 2006). In this approach, the minimization player and the maximization player play a repeated game against each other. Both the players rely on online learning algorithms to choose their actions in each round of the game, with the objective of minimizing their respective regret. In our work, we let the min player rely on Best Response (BR) and the max player rely on exponential weight update algorithm.

We are now ready to describe our algorithm for computing a mixed strategy NE of Equation (2) (equivalently the pure strategy NE of the linearized game in Equation (3)). Let (h_t,P_t) be the iterates generated by the algorithm in t-th iteration. The maximization player chooses distribution P_t over S_{aug} using exponential weights update

$$P_t(\mathbf{x},y,y',\delta) \propto \exp\left(\eta \sum_{j=1}^{t-1} \text{mg}_{\text{L}}\left(h_j(\mathbf{x}+\delta).y,y'\right)\right).$$

The minimization player chooses h_t using BR, which involves computing a minimizer of the following objective

$$h_t = \operatorname*{argmin}_{h\in\mathcal{H}} \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P_t}[\text{mg}_{\text{L}}\left(h(\mathbf{x}+\delta),y,y'\right)].$$

In Section D, we show that this algorithm converges to a mixed strategy NE of Equation (2).

[4] A linearized game is nothing but a game in the space of probability measures.

D. Proofs of Section 3.2

D.1. Proof of Theorem 4

The proof of this Theorem relies on the observation that the maximization player is using exponential weights update algorithm to generate its actions and the minimization player is using Best Response (BR) to generate its actions. To simplify the notation, we let $L(Q, P)$ denote

$$L(Q,P) = \mathbb{E}_{h\sim Q}\left[\mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P}[\mathrm{mg}_{\mathrm{L}}\left(h(\mathbf{x}+\delta),y,y'\right)]\right].$$

Note that $L(Q, P)$ is linear in both its arguments. With a slight overload of notation, we let $L(h, P), L(h, (\mathbf{x}, y, y', \delta))$ denote

$$L(h,P) = \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P}[\mathrm{mg}_{\mathrm{L}}\left(h(\mathbf{x}+\delta),y,y'\right)].$$

$$L(h,(\mathbf{x},y,y',\delta)) = \mathrm{mg}_{\mathrm{L}}\left(h(\mathbf{x}+\delta),y,y'\right).$$

Regret of min player. In the t-th iteration, the min player observes the loss function $L(\cdot, P_t)$ and chooses its action using BR

$$h_t \in \operatorname*{argmin}_{h\in\mathcal{H}} L(h,P_t).$$

Since $L(h_t, P_t) \leq \min_{h\in\mathcal{H}} L(h, P_t)$, we have

$$\sum_{t=1}^{T} L(h_t,P_t) - \min_{h\in\mathcal{H}} \sum_{t=1}^{T} L(h,P_t) \leq 0. \tag{4}$$

Regret of max player. In the t-th iteration, the min player chooses its action using exponential weights update algorithm and observes the loss function $L(h_t, \cdot)$. In particular, it plays action P_t, which is defined as follows

$$P_t(\mathbf{x},y,y',\delta) \propto \exp\left(\eta \sum_{j=1}^{t-1} L(h_t,(\mathbf{x},y,y',\delta))\right).$$

Relying on the regret bound for exponential weights update algorithm derived in Theorem 7, and using the fact that $|L(h, (\mathbf{x}, y, y', \delta))|$ is bounded by 1, we get

$$\max_{P\in\Delta(S_{\text{aug}})} \sum_{t=1}^{T} L(h_t,P) - \sum_{t=1}^{T} L(h_t,P_t) \leq 2\sqrt{\log(nK|\mathcal{B}(\epsilon)|)T}. \tag{5}$$

Combining the regret bounds of the min and max players in Equations (4), (5), we get

$$\max_{P\in\Delta(S_{\text{aug}})} \frac{1}{T}\sum_{t=1}^{T} L(h_t,P) - \min_{h\in\mathcal{H}} \frac{1}{T}\sum_{t=1}^{T} L(h,P_t) \leq 2\sqrt{\log(nK|\mathcal{B}(\epsilon)|)T}. \tag{6}$$

Let $\xi(T) = 2\sqrt{\log(nK|\mathcal{B}(\epsilon)|)T}$. We have the following from the above equation

$$\max_{P\in\Delta(S_{\text{aug}})} L(Q(T),P) \leq \min_{Q\in\Delta(H)} \max_{P\in\Delta(S_{\text{aug}})} L(Q,P) + \xi(T).$$

Rearranging the terms in the above equation and relying our definitions of $L(Q, P), \mathrm{mg}_{\mathrm{rob}}(\cdot)$, we get the following

$$\max_{Q\in\Delta(\mathcal{H})} \mathrm{mg}_{\mathrm{rob}}\left(Q,S\right) \leq \mathrm{mg}_{\mathrm{rob}}\left(Q(T),S\right) + \xi(T).$$

This proves the first part of the Theorem. The second part of the Theorem follows directly from the regret bound of the max player in Equation (5).

Algorithm 6 SAMPLER.RND

Input: training data S, base models $\{\theta_j\}_{j=1}^t$.
$\widehat{S}_B \leftarrow \{\}$
Uniformly sample a batch of points $\{(\mathbf{x}_b, y_b)\}_{b=1}^B$ from S.
for $b = 1 \dots B$ **do**
  uniformly sample y_b' from $\mathcal{Y} \setminus \{y_b\}$, and compute δ_b as

$$\delta_b \in \underset{\delta \in \mathcal{B}(\epsilon)}{\operatorname{argmax}} \ell_{\text{MCE}} \left(\sum_{j=1}^t g_{\theta_j}(\mathbf{x}_b + \delta), y_b, y_b' \right)$$

  $\widehat{S}_B \leftarrow \widehat{S}_B \cup \{(\mathbf{x}_b, y_b, y_b', \delta_b)\}$.
end for
Output: $\widehat{S}_B$

Algorithm 7 SAMPLER.MAX

Input: training data S, base models $\{\theta_j\}_{j=1}^t$.
$\widehat{S}_B \leftarrow \{\}$
Uniformly sample a batch of points $\{(\mathbf{x}_b, y_b)\}_{b=1}^B$ from S.
for $b = 1 \dots B$ **do**
  compute (y_b', δ_b) as

$$(y_b', \delta_b) \in \underset{\delta \in \mathcal{B}(\epsilon), y' \in \mathcal{Y} \setminus \{y_b\}}{\operatorname{argmax}} \ell_{\text{MCE}} \left(\sum_{j=1}^t g_{\theta_j}(\mathbf{x}_b + \delta), y_b, y' \right)$$

  $\widehat{S}_B \leftarrow \widehat{S}_B \cup \{(\mathbf{x}_b, y_b, y_b', \delta_b)\}$.
end for
Output: $\widehat{S}_B$

E. MRBOOST.NN

E.1. Computationally Efficient Samplers

As previously mentioned, the sampling sub-routine in Algorithm 3 is the key bottleneck in our Algorithm 2. So we design computationally efficient samplers which replace "soft weight" assignments in Algorithm 3 with "hard weight" assignments. Roughly speaking, these samplers first randomly sample a point $(\mathbf{x}, y, y') \in \tilde{S}_{\text{aug}}$, and find the worst perturbations for it and use it to train the base classifier.[5] In Algorithm 4 we described one of these samplers. Algorithms 6, 7 below present two other samplers. All these samplers differ in the way they perform hard weight assignment.

SAMPLER.RND. This is the most intuitive sampler. Here, we first uniformly sample $(\mathbf{x}, y)$ from S, and then randomly pick a false label $y' \in \mathcal{Y} \setminus \{y\}$. Next, we generate a perturbation with the highest pairwise margin loss for the ensemble $\{\theta_j\}_{j=1}^t$. This involves solving the following optimization problem

$$\delta^* \in \underset{\delta \in \mathcal{B}(\epsilon)}{\operatorname{argmax}} \ell_{\text{MCE}} \left(\sum_{j=1}^t g_{\theta_j}(\mathbf{x} + \delta), y, y' \right)$$

SAMPLER.MAX. This sampler differs from SAMPLER.RND in the way it chooses the label y'. Instead of randomly choosing y' from $\mathcal{Y} \setminus \{y\}$, it picks the worst possible y'. That is, it picks a y' which leads to the highest pairwise margin loss. This involves solving the following optimization problem

$$(y'^*, \delta^*) \in \underset{\delta \in \mathcal{B}(\epsilon), y' \in \mathcal{Y} \setminus \{y_b\}}{\operatorname{argmax}} \ell_{\text{MCE}} \left(\sum_{j=1}^t g_{\theta_j}(\mathbf{x} + \delta), y, y' \right)$$

[5] recall $\tilde{S}_{\text{aug}} = \{(\mathbf{x}, y, y') : (\mathbf{x}, y) \in S, y' \in \mathcal{Y} \setminus \{y\}\}$

Note that the computational complexity of this sampler scales linearly with the number of classes. So, this is not very practical for large-scale problems.

SAMPLER.ALL. This sampler smooths the "max" operator in SAMPLER.MAX by replacing it with the average of pairwise margin losses of all possible false labels. This involves solving the following optimization problem

$$\delta^* \in \operatorname*{argmax}_{\delta \in \mathcal{B}(\epsilon)} \sum_{y' \in \mathcal{Y} \setminus \{y\}} \ell_{\text{MCE}} \left(\sum_{j=1}^{t} g_{\theta_j}(\mathbf{x}+\delta), y, y' \right)$$

We found that the last sampler (SAMPLER.ALL) is both computationally efficient and achieves good robust accuracy (see related content in Section F.4.). So we use this it by default, unless mentioned otherwise.

F. More Details about Experiments

F.1. Efficient Implementation of MCE loss

As mentioned in the main text, for training a single network, our technique has as little as 5% additional computational overhead over baselines. For training an ensemble of T networks, the runtime scales linearly with T. The following table shows the running time (in seconds) of one epoch of adversarial training under different settings. These experiments were run on an NVIDIA A100 GPU.

SETTING	AT+CE (BASELINE)	AT+MCE (OURS)
SVHN+RES18	99S	101S
CIFAR10+RES18	71S	72S
CIFAR100+RES18	70S	72S

F.2. Adversarial Attacks

Given an input image $\mathbf{x}$, the target of an adversarial attack is to find an adversarial example $\mathbf{x}'$ within a local region of $\mathbf{x}$, such that it can mislead the classifier $g(\cdot)$ to make wrong decision making results. The local region is usually defined to be an $\epsilon-$norm ball centred at $\mathbf{x}$. In this work, we focus on ℓ_∞ cases, which means $\|\mathbf{x}' - \mathbf{x}\|_\infty \leq \epsilon$.

One of the earliest attacks is Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014), which is generated in the following way:

$$\mathbf{x}' = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} \ell(g(\mathbf{x}), y)),$$

which is essentially one-step gradient ascent in the input space. Another commonly used attack is Projected Gradient Descent (PGD) (Madry et al., 2017), which perturbs the data for K steps:

$$\mathbf{x}'^{k+1} = \Pi_\epsilon \left(\mathbf{x}^k + \alpha \cdot \text{sign}(\nabla_{\mathbf{x}^k} \ell(g(\mathbf{x}^k), y)) \right),$$

where α is the attack step size, and Π_ϵ denotes a projection to the ℓ_∞ norm ball. The Auto Attack (Croce & Hein, 2020b) is a strong and reliable evaluation suite, including a collection of three white-box attacks (APGD-CE (Croce & Hein, 2020b), APGD-DLR (Croce & Hein, 2020b) and FAB (Croce & Hein, 2020a)) and one black-box Square Attack (Andriushchenko et al., 2020).

F.3. Missing Details

In this section, we present more details about our experiments that are missing in the main paper. We first explain how we combine the proposed MCE loss into existing defense algorithms. To simplify the notation, we define $\ell_{\text{MCE-A}}$ as follows

$$\ell_{\text{MCE-A}}(g(\mathbf{x}), y) \coloneqq \left(\frac{1}{K-1} \sum_{y' \in \mathcal{Y} \setminus \{y\}} \ell_{\text{MCE}}(g(\mathbf{x}), y, y') \right).$$

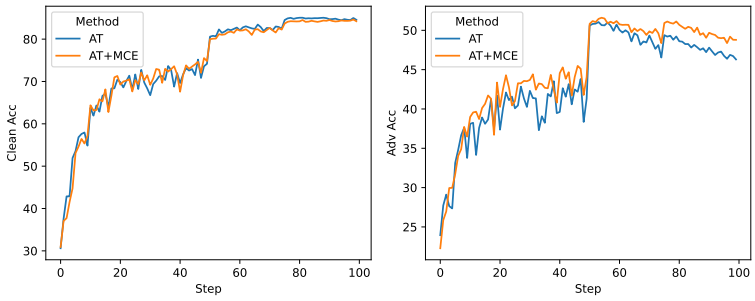

Figure 2. ResNet-18 robustness of AT and AT+MCE.

METHOD	CLEAN	ADV
SAMPLER.ALL	82.06	50.95
SAMPLER.RND	82.24	50.77
AT(2)	81.62	50.87

Table 5. Ablation of the proposed methods.

TRADES. TRADES (Zhang et al., 2019b) proposes to use the following training objective: $\ell_{\text{CE}}(g(\mathbf{x}), y) + \lambda \cdot \max_{\|\mathbf{x}'-\mathbf{x}\|\leq\epsilon} (D_{\text{KL}}(\mathbf{p}(\mathbf{x})\|\mathbf{p}(\mathbf{x}')))$, where $\mathbf{p} = \texttt{softmax}(g)$. We propose to incorporate MCE into this training framework as follows:

$$\ell_{\text{MCE-A}}(g(\mathbf{x}), y) + \lambda \cdot \max_{\|\mathbf{x}'-\mathbf{x}\|\leq\epsilon} (D_{\text{KL}}(\mathbf{p}(\mathbf{x})\|\mathbf{p}(\mathbf{x}')) + D_{\text{KL}}(\tilde{\mathbf{p}}(\mathbf{x})\|\tilde{\mathbf{p}}(\mathbf{x}'))),$$

where $\tilde{\mathbf{p}} = \texttt{softmax}(-g)$. In our experiments, we set $\lambda = 6$.

MART. MART (Wang et al., 2019b) uses the following objective: $\ell_{\text{BCE}}(g(\mathbf{x}'), y) + \lambda \cdot D_{\text{KL}}(\mathbf{p}(\mathbf{x})\|\mathbf{p}(\mathbf{x}')) \cdot (1 - \mathbf{p}(\mathbf{x})_y)$ where $\mathbf{x}'$ is an adversarial example from attack standard cross entropy loss. We refer readers to the original paper (Wang et al., 2019b) for more details on the BCE loss. We propose to incorporate MCE into this training framework as follows:

$$\ell_{\text{MCE-A}}(g(\mathbf{x}), y) + \lambda \cdot (D_{\text{KL}}(\mathbf{p}(\mathbf{x})\|\mathbf{p}(\mathbf{x}')) \cdot (1 - \mathbf{p}(\mathbf{x})_y) + D_{\text{KL}}(\tilde{\mathbf{p}}(\mathbf{x})\|\tilde{\mathbf{p}}(\mathbf{x}')) \cdot (1 - \tilde{\mathbf{p}}(\mathbf{x})_y)).$$

The notation of $\tilde{\mathbf{p}}$ is defined above. In our experiments, we set $\lambda = 1$.

AWP. AWP (Wu et al., 2020) proposes to optimize $\min_{\boldsymbol{\theta}} \max_{\|\tilde{\boldsymbol{\theta}}-\boldsymbol{\theta}\|\leq\gamma} \mathbb{E}_{\mathbf{x},y} \left[\max_{\|\mathbf{x}'-\mathbf{x}\|\leq\epsilon} \ell_{\text{CE}}(g_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}), y)\right]$, where γ is an additional hyperparameter to constrain the perturbation on model weights $\tilde{\boldsymbol{\theta}}$. As before, we simply replace cross entropy loss in this objective with $\ell_{\text{MCE-A}}$.

GAIR. GAIR (Zhang et al., 2020c) develops a reweighting method with cross entropy loss to improve adversarial training, which is orthogonal to our contribution. We simply replace the cross entropy loss in both training and attacks with $\ell_{\text{MCE-A}}$ to come up with GAIR+MCE method. For the hyperparameters, we use the default settings as the original paper suggests. We use the sigmoid-type decreasing function, set $\lambda = 0$, set the λ schedule to be the "fixed" schedule, and set the GAIR beginning epoch to be the time of first learning rate decay.

Training and Attack Details. In boosting experiments, the number of parameters of the five ResNet-18 ensemble is 55869810, while the deeper ResNet-158 model has 58156618 learnable parameters. We use 100 epochs for each boosting iteration (same as we do in Section 4.1) and run for five iterations. The output logit of the ensemble model is defined as the average of logits from different base classifiers. We test the robustness of all boosting methods with PGD-20 attack.

F.4. More Results

In Table 5, we perform an ablation study between SAMPLER.ALL and SAMPLER.RND. For this experiment, we use CIFAR-10 dataset. We report the results of running Algorithm 2 with different samplers on ResNet-18 for 1 boosting iteration (where the base classifier is trained for 100 epochs). From the table we could see that SAMPLER.RND is close to SAMPLER.ALL but with slightly lower robust accuracy and higher clean accuracy. We also compare with "AT(2)", which means we double the training loss of AT, to indicate that the improvement of SAMPLER.ALL is not a result of a different loss scale. Given these results, we use SAMPLER.ALL by default in our experiments presented in the main paper[6].

To show the difference between AT and AT+MCE, we also plot the performance curve of ResNet-18 for 100 epochs training in Figure F.3. As an evidence that our method is not fake defense, we try to attack the ResNet-18 model from AT+MCE with an adaptively designed attack (Tramer et al., 2020). To be concrete, we replace the cross entropy loss computation in PGD-20 attack with MCE loss. The adversarial accuracy from this modified attack almost remains the same ($50.95 \rightarrow 51.19$).

[6] we do not use SAMPLER.MAX as it is computationally expensive.

Table 6. Boosting experiments with ResNet-18 being the base classifier.

METHOD	ITERATION 1		ITERATION 2		ITERATION 3		ITERATION 4		ITERATION 5	
	CLEAN	ADV	CLEAN	ADV	CLEAN	ADV	CLEAN	ADV	CLEAN	ADV
WIDER MODEL	82.61	51.73	—	—	—	—	—	—	—	—
DEEPER MODEL	82.67	52.32	—	—	—	—	—	—	—	—
ROBBOOST + WHOLE + RNDINIT	82.00	51.05	84.58	49.95	83.87	51.66	82.56	52.72	81.44	52.92
MRBOOST.NN + WHOLE + RNDINIT	81.25	51.80	85.02	52.29	84.50	53.11	84.31	53.58	83.61	54.01
ROBBOOST + WHOLE + PERINIT	82.18	50.97	85.60	50.13	84.59	51.77	84.21	52.79	82.78	53.28
MRBOOST.NN + WHOLE + PERINIT	81.15	51.68	85.73	52.59	85.49	53.12	85.10	53.72	84.62	54.00
ROBBOOST + IND + RNDINIT	82.16	50.95	85.13	50.57	85.55	51.58	85.75	51.88	85.92	51.99
MRBOOST.NN + IND + RNDINIT	81.04	51.83	84.61	52.68	84.93	53.51	85.01	53.95	85.35	54.13
ROBBOOST + IND + PERINIT	82.12	51.04	85.73	50.91	85.98	51.89	85.97	52.31	85.81	52.52
MRBOOST.NN + IND + PERINIT	81.34	51.92	84.97	52.97	85.28	53.62	85.99	54.26	86.16	54.42

Table 7. Boosting experiments on SVHN

SVHN	ROBBOOST		MRBOOST.NN	
	CLEAN	ADV	CLEAN	ADV
ITER1	90.63	42.87	90.83	46.80
ITER2	90.91	44.56	90.71	48.87
ITER3	90.92	46.01	90.99	49.79
ITER4	91.20	46.75	91.12	50.59
ITER5	91.25	47.17	91.33	50.85

Considering our method has better adversarial accuracy even under AutoAttack, we believe our defense is not a result of gradient masking / obfuscated gradient.

We also present a more detailed version of boosting results in Table 6. Because the ROBBOOST algorithm of Abernethy et al. (2021) uses the whole ensemble $g_{1:t} = \sum_{j=1}^{t} g_j/t$ rather than only the new model g_t to calculate the loss for new model optimization (see Appendix F.5), we also try this version of MRBOOST.NN and name it MRBOOST.NN + WHOLE in this section. We then use ROBBOOST + IND to denote the version of ROBBOOST which only relies on g_t to calculate the loss of the new model optimization, where "ind" stands for individual update. To summarize, the improvement of our methodology is consistent for different settings.

MRBOOST.NN vs ROBBOOST. We now present more experimental results comparing our algorithm (MRBOOST.NN) with ROBBOOST. Tables 7, 8 present the results for SVHN, CIFAR-100 respectively. TODO: add details about the experiment (what base classifiers are used etc..)

Table 8. Boosting experiments on CIFAR-100

CIFAR100	ROBBOOST		MRBOOST.NN	
	CLEAN	ADV	CLEAN	ADV
ITER1	57.20	22.95	57.46	24.38
ITER2	60.94	27.89	60.07	27.73
ITER3	59.06	29.80	60.74	29.35
ITER4	60.19	30.72	61.20	31.13
ITER5	59.36	30.69	61.60	31.71

F.5. Boosting algorithm of Abernethy et al. (2021)

Abernethy et al. (2021) proposed a greedy stagewise boosting algorithm for building robust ensembles. Consider the following set of score-based base classifiers: $\mathcal{G} = \{g_\theta : \theta \in \mathbb{R}^D\}$, where $g_\theta : \mathcal{X} \to \mathbb{R}^K$ is a neural network parameterized by θ. Abernethy et al. (2021) construct ensembles of the form $g_{1:T}(\mathbf{x}) = \sum_{t=1}^{T} g_{\theta_t}(\mathbf{x})/T$.[7] The authors rely on a greedy stagewise algorithm to build this ensemble. In the t-th stage of this algorithm, the method picks a base classifier g_{θ_t} via the following greedy procedure

$$\theta_t = \operatorname*{argmin}_{\theta \in \mathbb{R}^D} \frac{1}{n} \sum_{i=1}^{n} \max_{\delta \in \mathcal{B}(\epsilon)} \ell_{\text{CE}} \left(g_{1:t-1}(\mathbf{x}_i + \delta) + \frac{1}{t} g_\theta(\mathbf{x}_i + \delta), y_i \right).$$

The authors solve this objective using AT (Madry et al., 2017). That is, for each step of gradient descent on w, θ, we solve the inner optimization problem via gradient ascent on the δ's, with projections on to $\mathcal{B}(\epsilon)$ at each step. Note that, at the beginning of t-th stage of the algorithm, θ_t is initialized randomly by Abernethy et al. (2021). In our experiments, we noticed that using persistent initialization (*i.e.,* initializing θ_t to θ_{t-1}) leads to more robust ensembles.

F.6. Circumventing the Defense of Pinot et al. (2020)

Table 9. Demonstration of the failure of Pinot et al. (2020) on CIFAR-10

Testing scenario	Accuracy (%)
Clean data	79.08
Adversarial examples of h_1	54.60
Pinot et al. (2020)'s ensemble attack	60.72
Our adaptive ensemble attack	37.37

Pinot et al. (2020) is one of the first works to develop boosting inspired algorithms for building robust ensembles. In this work, the authors build an ensemble by combining two base classifiers. The first one of these base classifiers is trained using PGD adversarial training (*i.e.,* AT (Madry et al., 2017)). The second base classifier is trained using standard empirical risk minimization on the adversarial dataset of the first base classifier. That is, the authors train the second base classifier to be robust *only* to the adversarial perturbations of the first base classifier. To be precise, after adversarially training the first neural network g_1, Pinot et al. (2020) propose to additionally train a second model g_2 using standard training with the adversarial dataset of the first model g_1. These two models are then randomly combined during evaluation, *i.e.*, for each test input, the algorithm selects one classifier at random and then outputs the corresponding predicted class. The authors claim that the proposed algorithm significantly boosts the performance of a single adversarially trained classifier.

In this work, we disprove this claim and show that the proposed defense can be circumvented using better adversarial attacks. Our key insight here is that there is a mismatch between the attack techniques used during training and inference phases of Pinot et al. (2020) (such phenomenon is referred to as "Obfuscated Gradient" phenomenon in Athalye et al. (2018b)[8]). This mismatch often leads to a false sense of robustness. Therefore, to properly evaluate the robustness of any defense technique, one should try to design adaptive attacks (Tramer et al., 2020) which take the algorithmic details of the defense into consideration.

Let g_1, g_2 be the base neural networks learned by the technique of Pinot et al. (2020). Let $w, (1-w)$ be the weights of these classifiers, where $w \in [0, 1]$. Given a test input, the authors first sample a base classifier according to these weights, and output the predicted class of the chosen model. So the loss of this ensemble at a point $(\mathbf{x}, y)$ is given by

$$w\ell_{0-1}(g_1(\mathbf{x}), y) + (1-w)\ell_{0-1}(g_2(\mathbf{x}), y). \tag{7}$$

To generate adversarial perturbation at a point $(\mathbf{x}, y)$ for this ensemble, the authors solve the following optimization problem using PGD

$$\max_{\delta \in \mathcal{B}(\epsilon)} \ell_{\text{CE}}(wg_1(\mathbf{x} + \delta) + (1-w)g_2(\mathbf{x} + \delta), y).$$

[7] Abernethy et al. (2021) also learn the weights of each component in the ensemble. In our experiments, we give equal weights to all the base classifiers in the ensemble.

[8] We refer the reader to Section 5.3 ("Stochastic Gradients") of Athalye et al. (2018b) for more discussion on the failure of defenses that rely on randomization.

Note that during the attack, the aggregation of the base classifiers is performed at the logit level. Whereas, during training the aggregation is performed at the level of predictions (Equation (7)). So, we design a slightly different attack which resolves this mismatch. In this attack the aggregation is performed at the probability level. This involves solving the following problem

$$\max_{\delta \in \mathcal{B}(\epsilon)} -\log \left(\frac{[w\mathbf{p}_1(\mathbf{x}+\delta) + (1-w)\mathbf{p}_2(\mathbf{x}+\delta)]_y}{\sum_{y'} [w\mathbf{p}_1(\mathbf{x}+\delta) + (1-w)\mathbf{p}_2(\mathbf{x}+\delta)]_{y'}} \right),$$

where $\mathbf{p}_1(\mathbf{x}+\delta) = \texttt{softmax}(g_1(\mathbf{x}+\delta)), \mathbf{p}_2(\mathbf{x}+\delta) = \texttt{softmax}(g_2(\mathbf{x}+\delta))$.

Table 9 presents the performance of the ensembling technique of Pinot et al. (2020) on various attacks. For this experiment, we use CIFAR-10 dataset and use a nine-layer convolutional neural network (as in Wang et al. (2019a)) as the base classifier. It can be seen that the performance of the ensemble takes a hit when we use our adaptive attack to generate perturbations. In particular, there is a 23% drop in robust accuracy when we switch from the attack of Pinot et al. (2020) to our attack. For comparison, performing standard AT on this model architecture results in a model with 80.29% clean accuracy and 45.08% PGD20 accuracy.