Adversarial Vulnerability of Randomized Ensembles

Hassan Dbouk [1] **Naresh R. Shanbhag** [1]

Abstract

Despite the tremendous success of deep neural networks across various tasks, their vulnerability to imperceptible adversarial perturbations has hindered their deployment in the real world. Recently, works on randomized ensembles have empirically demonstrated significant improvements in adversarial robustness over standard adversarially trained (AT) models with minimal computational overhead, making them a promising solution for safety-critical resource-constrained applications. However, this impressive performance raises the question: *Are these robustness gains provided by randomized ensembles real?* In this work we address this question both theoretically and empirically. We first establish theoretically that commonly employed robustness evaluation methods such as adaptive PGD provide a false sense of security in this setting. Subsequently, we propose a theoretically-sound and efficient adversarial attack algorithm (ARC) capable of compromising random ensembles even in cases where adaptive PGD fails to do so. We conduct comprehensive experiments across a variety of network architectures, training schemes, datasets, and norms to support our claims, and empirically establish that randomized ensembles are in fact *more vulnerable* to ℓ_p-bounded adversarial perturbations than even standard AT models. Our code can be found at `https://github.com/hsndbk4/ARC`.

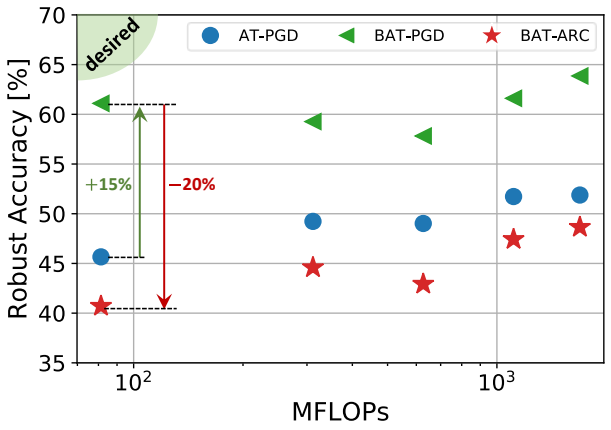


Figure 1. The adversarial *vulnerability* of randomized ensembles across various network architectures (from left to right: ResNet-20, MobileNetV1, VGG-16, ResNet-18, and WideResNet-28-4) against ℓ_∞ norm-bounded perturbations on CIFAR-10. Compared to a standard adversarially trained (AT) single model (●), a randomized ensemble of two models obtained via BAT (◀) (Pinot et al., 2020) offers significant improvement in robustness at iso-FLOPs, when evaluated using an adaptive PGD adversary. However, the robust accuracy of the BAT ensemble suffers a massive drop and becomes worse than even the single model AT baseline when evaluated using the proposed ARC (⋆) adversary, rendering the ensemble obsolete.

1. Introduction

Deep neural networks (DNNs) are ubiquitous today, cementing themselves as the *de facto* learning model for various machine learning tasks (Sarwar et al., 2001; He et al., 2016; Girshick, 2015; Farhadi & Redmon, 2018; Devlin et al., 2018; Zhang et al., 2017). Yet, their vulnerability to *imperceptible* adversarial perturbations (Szegedy et al., 2013; Goodfellow et al., 2014; Ilyas et al., 2019) has caused major concerns regarding their security, and hindered their deployment in safety-critical applications.

Various attack algorithms for crafting such perturbations have been proposed in the literature (Kurakin et al., 2016; Moosavi-Dezfooli et al., 2016; Carlini & Wagner, 2017); the majority of which rely on moving the input along the gradient to maximize a certain loss function. Powerful attacks, such as projected gradient descent (PGD) (Madry et al., 2018), perform this process iteratively until a perturbation is found, and can successfully fool undefended DNNs causing them to misclassify *every* test sample in the test set.

In order to robustify DNNs against adversarial perturbations, several defense strategies have been proposed (Cisse et al., 2017; Pang et al., 2019; Yang et al., 2019; Zhang

[1]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, USA. Correspondence to: Hassan Dbouk <hdbouk2@illinois.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

et al., 2019; Madry et al., 2018). While some of them were subsequently broken with more powerful attacks (adaptive attacks) (Tramèr et al., 2020; Athalye et al., 2018), adversarial training (AT) (Goodfellow et al., 2014; Zhang et al., 2019; Madry et al., 2018) remains the strongest empirical defense that thus far has stood the test of time.

Recently, the work of (Pinot et al., 2020) showed, from a game-theoretic point of view, that the robustness of any deterministic classifier can be outperformed by a randomized one. Specifically, (Pinot et al., 2020) proposed constructing random classifiers by randomly sampling from an ensemble of classifiers trained via boosted adversarial training (BAT), and empirically demonstrated *impressive* ($\sim$15%) improvements in robust accuracy over single model standard AT classifiers (see Fig. 1). Randomized ensembles also present a promising way for achieving high robustness with *limited* compute resources since the number of FLOPs per inference is fixed[1] – a problem that has garnered much interest recently (Dbouk & Shanbhag, 2021; Sehwag et al., 2020; Guo et al., 2020). However, this recent success of randomized ensembles raises the following question:

Are the robustness gains provided by randomized ensembles real?

In this work, we tackle this question both theoretically and empirically. Specifically, we make the following contributions:

- We prove that standard attack algorithms such as PGD suffer from a fundamental flaw when employed to attack randomized ensembles of linear classifiers – there are no guarantees that it will find an ℓ_p-bounded adversarial perturbation even when one exists (*inconsistency*).
- We propose a provably (for linear classifiers) *consistent* and efficient adversarial perturbation algorithm – the **A**ttacking **R**andomized ensembles of **C**lassifiers (**ARC**) algorithm – that is tailored to evaluate the robustness of randomized ensembles against ℓ_p-bounded perturbations.
- We employ the ARC algorithm to demonstrate empirically that randomized ensembles of DNNs are in fact *more* vulnerable to ℓ_p-bounded perturbations than standard AT DNNs (see Fig. 1).
- We conduct comprehensive experiments across a variety of network architectures, training schemes, datasets, and norms to support our observations.

[1]Assuming all the models in the ensemble share the same architecture.

Our work suggests the need for improved adversarial training algorithms for randomized ensembles.

2. Background and Related Work

Adaptive Attacks: Benchmarking new defenses against adaptive attacks, i.e., attacks carefully designed to target a given defense, has become standard practice today, thanks to the works of (Athalye et al., 2018; Tramèr et al., 2020). The authors in (Athalye et al., 2018) identify the phenomenon of obfuscated gradients, a type of gradient masking, that leads to a false sense of security in defenses against adversarial examples. By identifying and addressing different types of obfuscated gradients such as stochastic gradients due to randomization, they were able to circumvent and fool most defenses proposed at the time. More recently, (Tramèr et al., 2020) identified and circumvented *thirteen* defenses from top venues (NeurIPS, ICML, and ICLR), despite most employing adaptive attacks, by customizing these attacks to the specific defense. Therefore, it is imperative when evaluating a new defense to not only re-purpose existing adaptive attacks that circumvented a prior defense, but also "adapt" these attacks to target the new defense. Our work falls in the category of adaptive attacks for randomized ensembles. Specifically, we identify an unforeseen vulnerability of randomized ensembles by demonstrating the shortcomings of commonly used PGD-based attacks and then proposing an adaptive attack to successfully circumvent them.

Ensembles and Robustness: Traditionally, the use of ensembling techniques such as bagging (Breiman, 1996) and boosting (Dietterich, 2000; Freund & Schapire, 1997) has been a popular technique for improving the performance of machine learning models. Building on that success, and in order to address the apparent vulnerability of deep nets, a recent line of work (Kariyappa & Qureshi, 2019; Pang et al., 2019; Sen et al., 2019; Yang et al., 2020; 2021) has proposed the use of DNN ensembles. The intuition is that promoting some form of diversity between the different members of the ensemble would alleviate the adversarial transferability that DNNs exhibit, and as a byproduct improve overall robustness. Specifically, GAL (Kariyappa & Qureshi, 2019) proposes maximizing the cosine distance between the members' gradients, whereas EMPIR (Sen et al., 2019) leverages extreme model quantization to enforce ensemble diversity. ADP (Pang et al., 2019) proposes a training regularizer that promotes different members to have high diversity in the non-maximal predictions. These earlier works have been subsequently broken by adaptive and more powerful attacks (Tramèr et al., 2020). More recently, DVERGE (Yang et al., 2020) presents a robust ensemble training approach that diversifies the non-robust features of the different members via an adversarial training objective function. TRS (Yang et al., 2021), on the other hand, provides a theoretical

framework for understanding adversarial transferability and establishes bounds that show model smoothness and input gradient diversity are both required for low transferability. The authors also propose an ensemble training algorithm to empirically reduce transferability. While none of these works deal with randomized ensembles explicitly, their proposed techniques can be seamlessly adapted to that setting, where our work shows that existing powerful attacks such as PGD can provide a *false* sense of robustness.

Randomization and Robustness: Randomization has been shown, both theoretically and empirically, to have a very strong connection with adversarial robustness. Randomized smoothing (Lecuyer et al., 2019; Cohen et al., 2019) proves a tight robustness guarantee in ℓ_2 norm for smoothing classifiers with Gaussian noise. Prior to (Lecuyer et al., 2019), no certified defense has been shown feasible on ImageNet. In the same spirit, SNAP (Patil et al., 2021) enhances the robustness of single attack ℓ_∞ AT frameworks against the union of perturbation types via shaped noise augmentation. Recently, the work of (Pinot et al., 2020) showed, from a game-theoretic point of view, that the robustness of any deterministic classifier can be outperformed by a randomized one when evaluated against deterministic attack strategies. A follow-up work from (Meunier et al., 2021) removes any assumptions on the attack strategy and further motivates the use of randomization to improve robustness. Building on their framework, the authors of (Pinot et al., 2020) propose training random ensembles of DNNs using their BAT algorithm and *empirically* demonstrate impressive robustness gains over individual adversarially trained DNNs on CIFAR-10, when evaluated against a strong PGD adversary. However, our work demonstrates that these robustness gains give a *false* sense of security and propose a new attack algorithm better suited for randomized ensembles.

3. Preliminaries

3.1. Problem Setup

Let $\mathcal{F} = \{f_1, f_2, ..., f_M\}$ be a collection of $M \in \mathbb{N}$ arbitrary C-ary differentiable classifiers $f_i : \mathbb{R}^D \to \mathbb{R}^C$. A classifier f_i assigns the label $m \in [C]$ to data-point $\mathbf{x} \in \mathbb{R}^D$ if and only if $\big[f_i(\mathbf{x})\big]_m > \big[f_i(\mathbf{x})\big]_j \;\; \forall j \in [C] \setminus \{m\}$, where $[f_i(\mathbf{x})]_j$ is the j^{th} component of $f_i(\mathbf{x})$.

A randomized ensemble classifier (REC) g is defined over the tuple $(\mathcal{F}, \boldsymbol{\alpha})$ as: $\Pr\{g(\mathbf{x}) = f_d(\mathbf{x})\} = \alpha_d$ where $d \in [M]$ is sampled *independent* of $\mathbf{x}$. This independence assumption is crucial as the choice of the sampled classifier cannot be tampered by any adversary.

For any labeled data $(\mathbf{x}, y) \in \mathbb{R}^D \times [C]$, define L as the expected classification accuracy of g:

$$\begin{aligned} L(\mathbf{x}, y, \boldsymbol{\alpha}) &= \mathbb{E}\left[\mathbb{1}\left\{\underset{j\in[C]}{\arg\max}\,\big[g(\mathbf{x})\big]_j = y\right\}\right] \\ &= \sum_{i=1}^{M} \alpha_i \mathbb{1}\left\{\underset{j\in[C]}{\arg\max}\,\big[f_i(\mathbf{x})\big]_j = y\right\} \end{aligned} \tag{1}$$

where $\mathbb{1}\{.\}$ is the indicator function. Note that we have $L(\mathbf{x}, y, \boldsymbol{\alpha}) \leq 1$ with equality if and only if all the classifiers in $\mathcal{F}$ correctly classify $\mathbf{x}$. Similarly $L(\mathbf{x}, y, \boldsymbol{\alpha}) \geq 0$ with equality if and only if all all the classifiers misclassify $\mathbf{x}$. We always assume that $\alpha_i > 0$, otherwise we can remove f_i from the ensemble as it is not used.

Definition 3.1. A perturbation $\boldsymbol{\delta} \in \mathbb{R}^D$ is called adversarial to $(\mathcal{F}, \boldsymbol{\alpha})$ for the labeled data-point $(\mathbf{x}, y)$ if:

$$L(\mathbf{x} + \boldsymbol{\delta}, y, \boldsymbol{\alpha}) < L(\mathbf{x}, y, \boldsymbol{\alpha}) \tag{2}$$

That is, the mean classification accuracy of the randomized ensemble strictly decreases when $\mathbf{x}$ is perturbed by $\boldsymbol{\delta}$. To that end, an adversary's objective is to search for a norm-bounded adversarial perturbation $\boldsymbol{\delta}^*$ such that:

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_p \leq \epsilon}{\arg\min}\, L(\mathbf{x} + \boldsymbol{\delta}, y, \boldsymbol{\alpha}) \tag{3}$$

Let $\mathcal{A}$ be an adversarial perturbation generation algorithm that takes as input an REC $(\mathcal{F}, \boldsymbol{\alpha})$, a labeled data-point $(\mathbf{x}, y)$, and a norm bound ϵ and generates a perturbation $\boldsymbol{\delta}_{\mathcal{A}} : \|\boldsymbol{\delta}_{\mathcal{A}}\|_p \leq \epsilon$ while attempting to solve the optimization problem in (3).

Definition 3.2. Given REC $(\mathcal{F}, \boldsymbol{\alpha})$, a labeled data-point $(\mathbf{x}, y)$, and a norm bound ϵ: An adversarial perturbation generation algorithm $\mathcal{A}$ is said to be *consistent* if it finds a norm-bounded adversarial $\boldsymbol{\delta}_{\mathcal{A}}$ whenever $L(\mathbf{x}, y, \boldsymbol{\alpha}) = 1$ and a norm-bounded adversarial $\boldsymbol{\delta}$ exists.

Definition 3.2 implies that a consistent algorithm will always find an adversarial perturbation, if one exists, when all the classifiers in the ensemble correctly classify $\mathbf{x}$. It does not imply that it will find the optimal adversarial perturbation $\boldsymbol{\delta}^*$. Conversely, an *inconsistent* adversarial algorithm will fail to find a norm-bounded adversarial perturbation under these conditions and provide an overly optimistic estimate of adversarial robustness. Note that the condition $L(\mathbf{x}, y, \boldsymbol{\alpha}) = 1$ is not too restrictive, since DNNs typically exhibit very high accuracies on clean samples.

3.2. Projected Gradient Descent

Optimization-based attacks, such as PGD (Madry et al., 2018; Maini et al., 2020), search for adversarial perturbations around the data sample $(\mathbf{x}, y)$ for some classifier f and loss function l, by solving the following maximization:

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_p \leq \epsilon}{\arg\max}\, l\big(f(\mathbf{x} + \boldsymbol{\delta}), y\big) \tag{4}$$

in an iterative fashion $\forall k \in [K]$:

$$\boldsymbol{\delta}^{(k)} = \Pi_\epsilon^p \left(\boldsymbol{\delta}^{(k-1)} + \eta \mu_p \left(\nabla_{\mathbf{x}} l(\mathbf{x} + \boldsymbol{\delta}^{(k-1)}, y) \right) \right) \quad (5)$$

where $\boldsymbol{\delta}^{(0)}$ is typically a random initial guess inside the ℓ_p ball of radius ϵ, η is the step size, μ_p computes the unit-norm ℓ_p steepest direction, and Π_ϵ^p computes the projection onto the ℓ_p ball of radius ϵ. A typical choice of loss function is the cross-entropy loss (Madry et al., 2018). To simplify notation, we will use $l(\mathbf{x} + \boldsymbol{\delta}, y)$ instead of $l\big(f(\mathbf{x} + \boldsymbol{\delta}), y\big)$. For instance when $p = \infty$, the update equation in (5) reduces to:

$$\boldsymbol{\delta}^{(k)} = \text{clip} \left(\boldsymbol{\delta}^{(k-1)} + \eta \text{sgn}\left(\mathbf{g}\right), -\epsilon, \epsilon \right) \quad (6)$$

where $\mathbf{g}$ is the gradient of the loss as in (5).

Adaptive PGD for REC: As pointed out by (Athalye et al., 2018; Tramèr et al., 2020), randomized defenses need to take caution when evaluating robustness. In particular, one should use the expectation over transformation (EOT) method to avoid gradient masking. The discrete nature of our setup allows for exact computation of expectations, and eliminates the need for Monte Carlo sampling (Pinot et al., 2020). Therefore, in order to solve the optimization in (3) and evaluate the robustness of randomized ensembles, (Pinot et al., 2020) adapted the PGD update rule (5) by replacing the loss function with its expected value $\mathbb{E}\left[l(\mathbf{x} + \boldsymbol{\delta}, y)\right]$, which can be computed efficiently. Interestingly, we find that it is in fact this adaptation of PGD that leads to an overly optimistic estimate of the robustness of randomized ensembles – a point we address in the next section.

4. Limitations of Adaptive PGD

We analyze the nature of adversarial perturbations obtained via adaptive PGD (APGD) for the special case of an REC – an ensemble of binary linear classifiers (BLCs). In doing so, we unveil the reason underlying APGD's weakness in attacking RECs.

Assume an ensemble $(\mathcal{F}, \boldsymbol{\alpha})$ of M BLCs:

$$f_i(\mathbf{x}) = \mathbf{w}_i^{\mathrm{T}} \mathbf{x} + b_i \quad \forall i \in [M] \quad (7)$$

where $\mathbf{w}_i \in \mathbb{R}^D$ and $b_i \in \mathbb{R}$ are the weight and bias, respectively, of the i^{th} BLC. The BLC f_i assigns label 1 to $\mathbf{x}$ if and only if $f_i(\mathbf{x}) > 0$, and -1 otherwise [2]. The associated REC is given by $\Pr\{g(\mathbf{x}) = f_d(\mathbf{x})\} = \alpha_d$ where $d \in [M]$. We direct the reader to Appendix A for detailed derivations and proofs of the results in this section.

4.1. Attacking the Auxiliary Classifier

The APGD update rule employs the expected loss $\mathbb{E}\left[l(\mathbf{x} + \boldsymbol{\delta}, y)\right]$ in (5), where l is the binary cross-entropy loss. Now consider the gradient $\mathbf{g}$ of the expected loss. By linearity of the expectation, we have:

$$\mathbf{g} = -y \sum_{i=1}^{M} \alpha_i \lambda_i \mathbf{w}_i \quad (8)$$

where λ_i's belong to the open interval $(0, 1)$. Examining the expression in (8), we observe that the direction of $\mathbf{g}$ is determined purely by a linear combination of the classifiers' weight vectors. With that mind, if we define an *auxiliary* classifier $\bar{f}$ such that:

$$\bar{f}(\mathbf{x}) = \left(\sum_{i=1}^{M} \alpha_i \lambda_i \mathbf{w}_i \right)^{\mathrm{T}} \mathbf{x} + \bar{b} \quad (9)$$

for some arbitrary $\bar{b} \in \mathbb{R}$, then it is easy to see that the APGD attack is actually equivalent to the standard PGD attack evaluated against the auxiliary classifier $\bar{f}$. More generally, we have the following result:

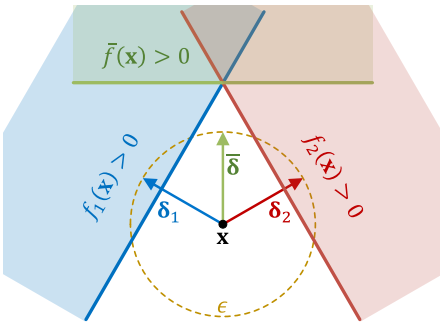


Figure 2. Illustration of an auxiliary classifier $\bar{f}$ for a randomized ensemble of two BLCs f_1 and f_2 with equiprobable sampling. The APGD output $\bar{\boldsymbol{\delta}}$ when attacking f_1 and f_2 is equivalent to the output of PGD when attacking $\bar{f}$.

Theorem 4.1 (Auxiliary Classifiers)**.** *For any REC $(\mathcal{F}, \boldsymbol{\alpha})$ consisting of BLCs and any data-point $(\mathbf{x}, y)$, there exists a sequence of auxiliary BLCs $\{\bar{f}^{(k)}\}_{k=1}^{K}$ such that in the k^{th} iteration, the output of APGD against the REC is equal to the output of PGD against $\bar{f}^{(k)}$.*

4.2. Inconsistency of Adaptive PGD

Theorem 4.1 highlights a fundamental flaw underlying APGD when applied to RECs: *APGD iterates target an auxiliary classifier that may or may not exist in the REC being attacked*. For instance, Fig. 2 illustrates the case of an $M = 2$ REC. Assuming equiprobable sampling, clearly choosing either of $\boldsymbol{\delta}_1$ or $\boldsymbol{\delta}_2$ will fool the ensemble half the time on average. However, due to the norm radius choice, the APGD generated perturbation cannot fool either, and thus will give an *incorrect* and *optimistic* estimate of the REC's robustness.

[2] For notional convenience, we assume $y \in \{-1, 1\}$ for binary classification.

Another implication of Theorem 4.1 is that APGD might prematurely converge to a fixed-point, that is $\boldsymbol{\delta}^{(k)} = \boldsymbol{\delta}^{(k-1)}$, which can happen when the auxiliary classifier becomes ill-defined and reduces to $\bar{f}^{(k)}(\mathbf{x}) = \bar{b}$. Furthermore, these fundamental flaws in APGD can be exploited by ensemble training techniques to artificially boost the robustness of the randomized ensemble. In fact, prime examples of this phenomena include adapting standard ensembling techniques (Kariyappa & Qureshi, 2019; Yang et al., 2021) that promote input gradient dissimilarity to the randomized setting, something we explore in Section 6.

One final implication of Theorem 4.1 is the following result regarding APGD:

Theorem 4.2 (Inconsistency)**.** *The APGD algorithm for randomized ensembles of classifiers is inconsistent.*

Theorem 4.2 provides further theoretical justification of the shortcomings of APGD when attacking randomized ensembles. The inconsistency of APGD leads to overly optimistic robustness evaluation in case of DNNs (see Fig. 1).

5. The ARC Algorithm

To properly assess the robustness of randomized ensembles, and avoid the pitfalls of APGD as seen in Section 4, we present a novel attack algorithm for solving (3), namely the Attacking Randomized ensembles of Classifiers (ARC) algorithm.

5.1. Binary Linear Classifiers

Analogous to the APGD analysis in Section 4, we first present a simpler version of ARC (Algorithm 1) tailored for randomized ensembles of BLCs.

In this setting, the ARC algorithm iterates over all members of the ensemble *once* in a greedy fashion. At iteration $i \in [M]$, a candidate perturbation $\hat{\boldsymbol{\delta}}$ is obtained by updating the perturbation $\boldsymbol{\delta}$ from the previous perturbation as follows:

$$\hat{\boldsymbol{\delta}} = \gamma \left(\boldsymbol{\delta} + \beta \mathbf{g} \right) \tag{10}$$

where $\gamma > 0$ such that $\|\hat{\boldsymbol{\delta}}\|_p = \epsilon$, $\mathbf{g}$ is the unit ℓ_p norm optimal perturbation (Melachrinoudis, 1997) for fooling classifier f_i, and β is adaptively chosen based on lines (10)-(13) to guarantee that $\hat{\boldsymbol{\delta}}$ can fool f_i, irrespective of $\boldsymbol{\delta}$, as long as the shortest ℓ_p distance between f_i and $\mathbf{x}$ is less than the norm radius ϵ. Subsequently, the ARC algorithm updates its iterate $\boldsymbol{\delta}$ with $\hat{\boldsymbol{\delta}}$ as long as the average classification accuracy satisfies:

$$L(\mathbf{x} + \hat{\boldsymbol{\delta}}, y, \boldsymbol{\alpha}) \leq L(\mathbf{x} + \boldsymbol{\delta}, y, \boldsymbol{\alpha}) \tag{11}$$

that is $\hat{\boldsymbol{\delta}}$ does not *increase* the robust accuracy. Note that $\rho > 0$ in line (13) is an arbitrarily small positive constant to avoid boundary conditions. In our implementation, we set $\rho = 0.05\epsilon$.

Algorithm 1 The ARC Algorithm for BLCs

1: **Input:** REC $(\mathcal{F}, \boldsymbol{\alpha})$, labeled data-point $(\mathbf{x}, y)$, norm p, and radius ϵ.
2: **Output:** Adversarial perturbation $\boldsymbol{\delta}$ such that $\|\boldsymbol{\delta}\|_p \leq \epsilon$.
3: Initialize $\boldsymbol{\delta} \leftarrow \mathbf{0}$, $v \leftarrow L(\mathbf{x}, y, \boldsymbol{\alpha})$, $q \leftarrow \frac{p}{p-1}$
4: Define $\mathcal{I}$ such that $\alpha_i \geq \alpha_j \; \forall i, j \in \mathcal{I}$ and $i \leq j$.
5: **for** $i \in \mathcal{I}$ **do**
6: /* optimal unit ℓ_p norm adversarial direction for f_i
7: $\mathbf{g} \leftarrow -y \frac{|\mathbf{w}_i|^{q-1} \odot \operatorname{sgn}(\mathbf{w}_i)}{\|\mathbf{w}_i\|_q^{q-1}}$
8: /* shortest ℓ_p distance between $\mathbf{x}$ and f_i
9: $\zeta \leftarrow \frac{|f_i(\mathbf{x})|}{\|\mathbf{w}_i\|_q}$
10: **if** $\zeta \geq \epsilon \vee i = 1$ **then**
11: $\beta \leftarrow \epsilon$
12: **else**
13: $\beta \leftarrow \frac{\epsilon}{\epsilon - \zeta} \left| \frac{y \mathbf{w}_i^{\mathrm{T}} \boldsymbol{\delta}}{\|\mathbf{w}_i\|_q} + \zeta \right| + \rho$
14: **end if**
15: $\hat{\boldsymbol{\delta}} \leftarrow \epsilon \frac{\boldsymbol{\delta} + \beta \mathbf{g}}{\|\boldsymbol{\delta} + \beta \mathbf{g}\|_p}$ ▷ candidate $\hat{\boldsymbol{\delta}}$ such that $\|\hat{\boldsymbol{\delta}}\|_p = \epsilon$
16: $\hat{v} \leftarrow L(\mathbf{x} + \hat{\boldsymbol{\delta}}, y, \boldsymbol{\alpha})$
17: /* if robustness does not increase, update $\boldsymbol{\delta}$
18: **if** $\hat{v} \leq v$ **then**
19: $\boldsymbol{\delta} \leftarrow \hat{\boldsymbol{\delta}}$, $v \leftarrow \hat{v}$
20: **end if**
21: **end for**

In contrast to APGD, the ARC algorithm is provably consistent as stated below:

Theorem 5.1 (Consistency)**.** *The ARC algorithm for randomized ensembles of binary linear classifiers is consistent.*

The implication of Theorem 5.1 (see Appendix A for proof) is that the perturbation generated via the ARC algorithm is *guaranteed* to be adversarial to $(\mathcal{F}, \boldsymbol{\alpha})$, given that $\mathbf{x}$ is correctly classified by all members provided such a perturbation exists. Thus, we have constructed a theoretically-sound algorithm for attacking randomized ensembles of binary linear classifiers.

5.2. Differentiable Multiclass Classifiers

We now extend the ARC algorithm presented in Algorithm 1 to the more general case of C-ary differentiable classifiers, such as DNNs. When dealing with a non-linear differentiable classifier f, we can "linearize" its behavior around an input $\mathbf{x}$ in order to provide *estimates* of both the shortest ℓ_p distance to the decision boundary $\tilde{\zeta}$, and the corresponding unit ℓ_p norm direction $\tilde{\mathbf{g}}$. Using a first-order Taylor series expansion at $\mathbf{x}$ we approximate f:

$$f(\mathbf{u}) \approx \tilde{f}(\mathbf{u}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^{\mathrm{T}} (\mathbf{u} - \mathbf{x}) \tag{12}$$

where $\nabla f(\mathbf{x}) = [\nabla [f(\mathbf{x})]_1 | ... | \nabla [f(\mathbf{x})]_C] \in \mathbb{R}^{D \times C}$ is the Jacobian matrix. Let $m \in [C]$ be the assigned label to $\mathbf{x}$ by f, we can construct the approximate decision region $\tilde{\mathcal{R}}_m(\mathbf{x})$ around $\mathbf{x}$ as follows:

$$\begin{aligned}\tilde{\mathcal{R}}_m(\mathbf{x}) &= \bigcap_{\substack{j=1 \\ j \neq m}}^{C} \left\{ \mathbf{u} \in \mathbb{R}^D : \left[\tilde{f}(\mathbf{u})\right]_m > \left[\tilde{f}(\mathbf{u})\right]_j \right\} \\ &= \bigcap_{\substack{j=1 \\ j \neq m}}^{C} \left\{ \mathbf{u} \in \mathbb{R}^D : \tilde{\mathbf{w}}_j^{\mathrm{T}} (\mathbf{u} - \mathbf{x}) + \tilde{h}_j > 0 \right\}\end{aligned} \tag{13}$$

where $\tilde{h}_j = [f(\mathbf{x})]_m - [f(\mathbf{x})]_j$ and $\tilde{\mathbf{w}}_j = \nabla [f(\mathbf{x})]_m - \nabla [f(\mathbf{x})]_j \ \forall j \in [C] \setminus \{m\}$. The decision boundary of $\tilde{\mathcal{R}}_m(\mathbf{x})$ is captured via the $C-1$ hyper-planes $\mathcal{H}_j$ defined by:

$$\mathcal{H}_j = \left\{ \mathbf{u} \in \mathbb{R}^D : \tilde{\mathbf{w}}_j^{\mathrm{T}} (\mathbf{u} - \mathbf{x}) + \tilde{h}_j = 0 \right\} \tag{14}$$

Therefore, in order to obtain $\tilde{\zeta}$ and $\tilde{\mathbf{g}}$, we find the closest hyper-plane $\mathcal{H}_n$:

$$n = \underset{j \in [C] \setminus \{m\}}{\arg\min} \frac{\left|\tilde{h}_j\right|}{\|\tilde{\mathbf{w}}_j\|_q} = \underset{j \in [C] \setminus \{m\}}{\arg\min} \ \tilde{\zeta}_j \tag{15}$$

and then compute:

$$\tilde{\zeta} = \tilde{\zeta}_n \quad \& \quad \tilde{\mathbf{g}} = -\frac{|\tilde{\mathbf{w}}_n|^{q-1} \odot \operatorname{sgn}(\tilde{\mathbf{w}}_n)}{\|\tilde{\mathbf{w}}_n\|_q^{q-1}} \tag{16}$$

where $q \geq 1$ is the dual norm of p, $\odot$ is the Hadamard product, and $\mathbf{m} = |\mathbf{w}|^r$ denotes the element-wise operation $m_i = |w_i|^r \ \forall i \in [D]$. Using these equations, we generalize Algorithm 1 for BLCs to obtain ARC in Algorithm 2. The ARC algorithm is iterative. At each iteration $k \in [K]$, ARC performs a local linearization approximation. Due to this approximation, we limit the local perturbation radius to η, a hyper-paramater often referred to as the step size akin to PGD.

5.3. Practical Considerations

The ARC algorithm is extremely efficient to implement in practice. Automatic differentiation packages such as PyTorch (Paszke et al., 2017) can carry out the linearization procedures in a seamless fashion. However, datasets with large number of classes C, e.g., CIFAR-100 and ImageNet, can pose a practical challenge for Algorithm 2. This is due to the $\mathcal{O}(C)$ search performed in (15), which recurs for each iteration k and classifier f_i in the ensemble. However, we have observed that (15) can be efficiently and accurately approximated by limiting the search only to a fixed subset of hyper-planes of size $G \in [C-1]$, e.g., $G = 4$ reduces the evaluation time by more than $\sim 14\times$ (for CIFAR-100) without compromising on accuracy. We shall use this version of ARC for evaluating such datasets, and refer the reader to Appendix D for further details.

Algorithm 2 The ARC Algorithm

Input: REC $(\mathcal{F}, \boldsymbol{\alpha})$, labeled data-point $(\mathbf{x}, y)$, number of steps K, step size η, norm p, and radius ϵ.
Output: Adversarial perturbation $\boldsymbol{\delta}$ such that $\|\boldsymbol{\delta}\|_p \leq \epsilon$.
Initialize $\boldsymbol{\delta} \leftarrow \mathbf{0}$, $v \leftarrow L(\mathbf{x}, y, \boldsymbol{\alpha})$, $q \leftarrow \frac{p}{p-1}$
Define $\mathcal{I}$ such that $\alpha_i \geq \alpha_j \ \forall i, j \in \mathcal{I}$ and $i \leq j$.
for $k \in [K]$ **do**
  $\boldsymbol{\delta}_l \leftarrow \mathbf{0}$, $v_l \leftarrow v$ ▷ initialize for local search
  for $i \in \mathcal{I}$ **do**
    $\tilde{\mathbf{x}} \leftarrow \mathbf{x} + \boldsymbol{\delta} + \boldsymbol{\delta}_l$
    /* the label assigned to $\tilde{\mathbf{x}}$ by f_i
    $m \leftarrow \arg\max_{j \in [C]} [f_i(\tilde{\mathbf{x}})]_j$
    $\tilde{\mathbf{w}}_j \leftarrow \nabla [f_i(\tilde{\mathbf{x}})]_m - \nabla [f_i(\tilde{\mathbf{x}})]_j \quad \forall j \in [C] \setminus \{m\}$
    $\tilde{h}_j \leftarrow [f_i(\tilde{\mathbf{x}})]_m - [f_i(\tilde{\mathbf{x}})]_j \quad \forall j \in [C] \setminus \{m\}$
    /* get closest hyper-plane to $\tilde{\mathbf{x}}$ when f_i is linearized
    $n \leftarrow \arg\min_{j \in [C] \setminus \{m\}} \frac{|\tilde{h}_j|}{\|\tilde{\mathbf{w}}_j\|_q}$
    $\mathbf{g} \leftarrow -\frac{|\tilde{\mathbf{w}}_n|^{q-1} \odot \operatorname{sgn}(\tilde{\mathbf{w}}_n)}{\|\tilde{\mathbf{w}}_n\|_q^{q-1}}$
    $\zeta \leftarrow \frac{|\tilde{h}_n|}{\|\tilde{\mathbf{w}}_n\|_q}$
    if $\zeta \geq \eta \vee i = 1$ **then**
      $\beta \leftarrow \eta$
    else
      $\beta \leftarrow \frac{\eta}{\eta - \zeta} \left| \frac{\tilde{\mathbf{w}}_n^{\mathrm{T}} \boldsymbol{\delta}_l}{\|\tilde{\mathbf{w}}_n\|_q} + \zeta \right| + \rho$
    end if
    $\hat{\boldsymbol{\delta}}_l \leftarrow \eta \frac{\boldsymbol{\delta}_l + \beta \mathbf{g}}{\|\boldsymbol{\delta}_l + \beta \mathbf{g}\|_p}$
    /* project onto the ℓ_p ball of radius ϵ and center $\mathbf{x}$
    $\hat{\boldsymbol{\delta}} \leftarrow \Pi_\epsilon^p \left(\boldsymbol{\delta} + \hat{\boldsymbol{\delta}}_l \right)$, $\hat{v}_l \leftarrow L(\mathbf{x} + \hat{\boldsymbol{\delta}}, y, \boldsymbol{\alpha})$
    if $\hat{v}_l \leq v_l$ **then**
      $\boldsymbol{\delta}_l \leftarrow \hat{\boldsymbol{\delta}}_l$, $v_l \leftarrow \hat{v}_l$ ▷ update the local variables
    end if
  end for
  /* update the global variables after localized search
  $\hat{\boldsymbol{\delta}} \leftarrow \Pi_\epsilon^p (\boldsymbol{\delta} + \boldsymbol{\delta}_l)$, $\hat{v} \leftarrow L(\mathbf{x} + \hat{\boldsymbol{\delta}}, y, \boldsymbol{\alpha})$
  if $\hat{v} \leq v$ **then**
    $\boldsymbol{\delta} \leftarrow \hat{\boldsymbol{\delta}}$, $v \leftarrow \hat{v}$
  end if
end for

6. Experiments

6.1. Setup

We conduct comprehensive experiments to demonstrate the effectiveness of ARC in generating norm-bounded adversarial perturbations for RECs, as compared to APGD. Specif-

Table 1. Comparison between ARC and adaptive PGD when attacking randomized ensembles trained via BAT (Pinot et al., 2020) across various network architectures and norms on the CIFAR-10 dataset. We use the standard radii $\epsilon_2 = 128/255$ and $\epsilon_\infty = 8/255$ for ℓ_2 and ℓ_∞-bounded perturbations, respectively.

Network	Norm	Robust Accuracy [%] AT ($M=1$) PGD	REC ($M=2$) APGD	REC ($M=2$) ARC	Diff
ResNet-20	ℓ_2	62.43	69.21	55.44	**−13.77**
	ℓ_∞	45.66	61.10	40.71	**−20.39**
MobileNetV1	ℓ_2	66.39	67.92	59.43	**−8.49**
	ℓ_∞	49.23	59.27	44.59	**−14.68**
VGG-16	ℓ_2	66.08	66.96	59.20	**−7.76**
	ℓ_∞	49.02	57.82	42.93	**−14.89**
ResNet-18	ℓ_2	69.16	70.16	65.88	**−4.28**
	ℓ_∞	51.73	61.61	47.43	**−14.18**
WideResNet-28-4	ℓ_2	69.91	71.48	62.95	**−8.53**
	ℓ_∞	51.88	63.86	48.65	**−15.21**

ically, we experiment with a variety of network architectures (VGG-16 (Simonyan & Zisserman, 2014), ResNet-20, ResNet-18 (He et al., 2016), WideResNet-28-4 (Zagoruyko & Komodakis, 2016), and MobileNetV1 (Howard et al., 2017)), datasets (SVHN (Netzer et al., 2011), CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100, and ImageNet (Krizhevsky et al., 2012)), and norms (ℓ_2 and ℓ_∞). We use the tried and tested publicly available implementation of PGD from (Rice et al., 2020). For all our experiments, the same ℓ_p norm is used during both training and evaluation. Further details on the training/evaluation setup can be found in Appendix B.

For BAT (Pinot et al., 2020) comparisons, we first train a standard AT model using PGD (Rice et al., 2020), which serves both as an independent robust baseline as well as the first model (f_1) in the REC to be constructed. Following the BAT procedure, we then train a second model (f_2) using the adversarial samples of the *first* model only. The same network architecture will be used for both models. When evaluating the robust accuracy of an REC, we compute the true expectation via (1) in accordance with (Pinot et al., 2020).

6.2. Results

We first showcase the effectiveness of ARC on CIFAR-10 using five network architectures and both ℓ_2 and ℓ_∞ norms. Table 1 summarizes the robust accuracies for various models and adversaries. Following the approach of (Pinot et al., 2020), we find the optimal sampling probability $\boldsymbol{\alpha} = (\alpha, 1-\alpha)$ that maximizes the robust accuracy via a grid search using APGD. We report the corresponding robust accuracies using both APGD and ARC for the same optimal sampling probability, to ensure a fair comparison. We consistently find that $\alpha^* \approx 0.9$, that is the REC heavily favors the robust model (f_1).

Table 2. Comparison between ARC and adaptive PGD when attacking randomized ensembles trained via BAT (Pinot et al., 2020) across various datasets and norms. We use ResNet-18 for ImageNet and and ResNet-20 for SVHN, CIFAR-10, and CIFAR-100 datasets.

Dataset	Norm	Radius (ϵ)	Robust Accuracy [%] AT ($M=1$) PGD	REC ($M=2$) APGD	REC ($M=2$) ARC	Diff
SVHN	ℓ_2	128/255	68.35	74.66	60.15	**−14.51**
	ℓ_∞	8/255	53.55	65.99	52.01	**−13.98**
CIFAR-10	ℓ_2	128/255	62.43	69.21	55.44	**−13.77**
	ℓ_∞	8/255	45.66	61.10	40.71	**−20.39**
CIFAR-100	ℓ_2	128/255	34.60	41.91	28.92	**−12.99**
	ℓ_∞	8/255	22.29	33.37	17.45	**−15.92**
ImageNet	ℓ_2	128/255	47.61	49.62	42.09	**−7.53**
	ℓ_∞	4/255	24.33	35.92	19.54	**−16.38**

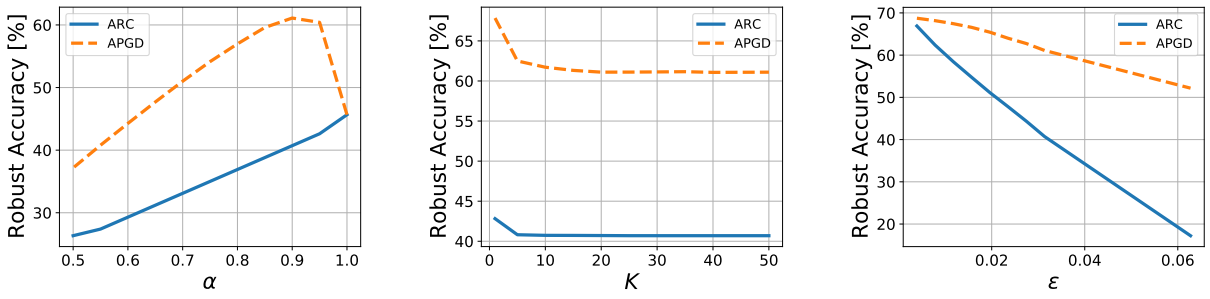
Table 1 provides *overwhelming* evidence that: 1) BAT trained RECs perform significantly better than their respective AT counterparts, when APGD is used for robustness evaluation, 2) employing ARC instead of APGD results in a massive drop in robust accuracy of the *same* REC, and 3) the *true robust accuracy of the REC* (obtained via ARC) *is worse than that of the AT baseline* $M = 1$. For example, a randomized ensemble of ResNet-20s achieves 61.1% robust accuracy when using APGD, a $\sim 15\%$ increase over the AT baseline. However, when using ARC, we find that the robustness of the ensemble is in fact 40.7% which is: 1) much lower ($\sim 20\%$) than what APGD claims and 2) also lower ($\sim 5\%$) than the AT baseline.

The conclusions of Table 1 are further corroborated by Table 2, where we now compare ARC and APGD across four datasets. These empirical results illustrate both the validity and usefulness of our theoretical results, as they convincingly show that APGD is indeed ill-suited for evaluating the robustness of RECs and can provide a *false* sense of robustness.

6.3. Robustness Stress Tests

In this section, we conduct robustness stress tests to further ensure that our claims are in fact well-founded and not due to a technicality.

Parameter Sweeps: We sweep the sampling probability (Fig. 3a), number of iterations (Fig. 3b), and norm radius (Fig. 3c) for an REC of two ResNet-20s obtained via ℓ_∞ BAT on the CIFAR-10 dataset. For similar experiments across more norms and datasets, please refer to Appendix C.2. In all cases, we find that the ARC adversary is consistently a much stronger attack for RECs than APGD. Moreover, we observe that $\alpha = 1$ always results in the high-



(a) (b) (c)

Figure 3. Comparison between ARC and APGD using an REC of ResNet-20s obtained via ℓ_∞ BAT ($M = 2$) on CIFAR-10. Robust accuracy vs.: (a) sampling probability α, (b) number of iterations K, and (c) norm radius ϵ.

Table 3. Robust accuracy of different versions of APGD compared to ARC for ResNet-20 RECs on CIFAR-10. We use the standard radii $\epsilon_2 = 128/255$ and $\epsilon_\infty = 8/255$ for ℓ_2 and ℓ_∞-bounded perturbations, respectively.

Attack Method	Robust Accuracy [%] ℓ_2	ℓ_∞
APGD	69.21	61.10
APGD + R	69.28	62.80
APGD + RR	70.14	62.95
ARC	55.44	40.71

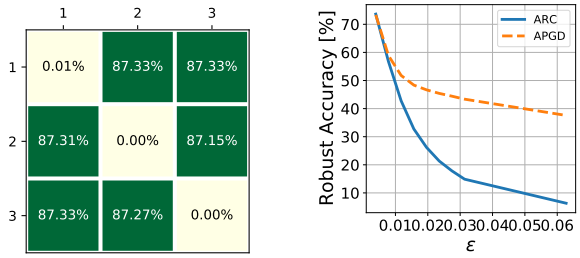
est ensemble robustness when properly evaluated with ARC. This renders BAT RECs obsolete since the best policy is to choose the deterministic AT trained model f_1.

Stronger PGD: We also experiment with stronger versions of PGD using random initialization (R) (Madry et al., 2018) and 10 random restarts (RR) (Maini et al., 2020). Table 3 reports the corresponding robust accuracies. Interestingly enough, we find that APGD+RR is actually slightly less effective than vanilla APGD and APGD+R. While this observation might seem counter-intuitive at first, it can be reconciled using our analysis from Section 4. By design, we have that the expected loss associated with APGD+RR perturbations is guaranteed to be larger than that of APGD+R[3]. However, we have established (see Fig. 2) that the perturbation that maximizes the expected loss can perform worse than other sub-optimal perturbations, which provides further evidence that APGD is indeed fundamentally flawed, and that ARC is better suited for evaluating the robustness of RECs.

6.4. Other Ensemble Training Methods

As alluded to in Section 2, one can construct randomized ensembles from any pre-trained set of classifiers, regardless

[3]Assuming they share the same random seed.



(a) (b)

Figure 4. Robustness performance of an REC constructed from ℓ_∞ DVERGE trained models ($M = 3$): (a) cross-robustness matrix, and (b) robust accuracy vs. norm radius ϵ using both ARC and APGD.

of the training procedure. To that end, we leverage the publicly available DVERGE (Yang et al., 2020) trained ResNet-20 models on CIFAR-10 to construct and evaluate such RECs.

Figure 4a plots the cross-robustness matrix of three DVERGE models. As expected, each model is quite robust ($\sim 87\%$) to other models' adversarial perturbations (obtained via PGD). However, the models are completely compromised when subjected to their own adversarial perturbations.

Due to the apparent symmetry between the models, we construct a randomized ensemble with equiprobable sampling $\alpha_i = 1/3$. We evaluate the robustness of the REC against both APGD and ARC adversaries, and sweep the norm radius ϵ in the process. As seen in Fig. 4b, the ARC adversary is much more effective at fooling the REC, compared to APGD. For instance, for the typical radius $\epsilon = 8/255$, the APGD robust accuracy is 43.36%, on-par with the single model AT baseline (see Table 1). In contrast, the ARC robust accuracy is 14.88%, yielding a *massive* difference in robustness ($\sim 28\%$). In the interest of space, we defer

the ADP (Pang et al., 2019) and TRS (Yang et al., 2021) experiments to Appendix C.4. We also explore constructing RECs using independent adversarially trained models in Appendix C.5.

7. Conclusion

We have demonstrated both theoretically and empirically that the proposed ARC algorithm is better suited for evaluating the robustness of randomized ensembles of classifiers. In contrast, current adaptive attacks provide an overly optimistic estimate of robustness. Specifically, our work successfully *compromises* existing empirical defenses such as BAT (Pinot et al., 2020). While we also experiment with other methods for constructing RECs, we point out the paucity of work on randomized ensemble defenses.

Despite the vulnerability of existing methods, the hardware efficiency of randomized ensembles is an attractive feature. We believe that constructing robust randomized ensembles is a promising research direction, and our work advocates the need for improved defense methods including *certifiable* defenses, i.e., defenses that are *provably* robust against a specific adversary.

Acknowledgements

This work was supported by the Center for Brain-Inspired Computing (C-BRIC) and the Artificial Intelligence Hardware (AIHW) program funded by the Semiconductor Research Corporation (SRC) and the Defense Advanced Research Projects Agency (DARPA).

References

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, 2018.

Breiman, L. Bagging predictors. *Machine learning*, 24(2): 123–140, 1996.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pp. 854–863. PMLR, 2017.

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

Dbouk, H. and Shanbhag, N. Generalized depthwise-separable convolutions for adversarially robust and efficient neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dietterich, T. G. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer, 2000.

Farhadi, A. and Redmon, J. YOLOv3: An incremental improvement. In *Computer Vision and Pattern Recognition*, pp. 1804–02767, 2018.

Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Girshick, R. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

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

Guo, M., Yang, Y., Xu, R., Liu, Z., and Lin, D. When NAS meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 631–640, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Ilyas, A., Santurkar, S., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

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

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 656–672. IEEE, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Maini, P., Wong, E., and Kolter, Z. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*, pp. 6640–6650. PMLR, 2020.

Melachrinoudis, E. An analytical solution to the minimum lp-norm of a hyperplane. *Journal of Mathematical Analysis and Applications*, 211(1):172–189, 1997.

Meunier, L., Scetbon, M., Pinot, R. B., Atif, J., and Chevaleyre, Y. Mixed nash equilibria in the adversarial examples game. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7677–7687. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/meunier21a.html.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

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.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pp. 582–597. IEEE, 2016.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Patil, A. D., Tuttle, M., Schwing, A. G., and Shanbhag, N. R. Robustifying ℓ_∞ adversarial training to the union of perturbation models. *arXiv preprint arXiv:2105.14710*, 2021.

Pinot, R., Ettedgui, R., Rizk, G., Chevaleyre, Y., and Atif, J. Randomization matters how to defend against strong adversarial attacks. In *International Conference on Machine Learning*, pp. 7717–7727. PMLR, 2020.

Rauber, J., Brendel, W., and Bethge, M. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017. URL http://arxiv.org/abs/1707.04131.

Rice, L., Wong, E., and Kolter, Z. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pp. 8093–8104. PMLR, 2020.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, 2001.

Sehwag, V., Wang, S., Mittal, P., and Jana, S. HYDRA: Pruning adversarially robust neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 7, 2020.

Sen, S., Ravindran, B., and Raghunathan, A. EMPIR: Ensembles of mixed precision deep networks for increased robustness against adversarial attacks. In *International Conference on Learning Representations*, 2019.

Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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.

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

Yang, H., Zhang, J., Dong, H., Inkawhich, N., Gardner, A., Touchet, A., Wilkes, W., Berry, H., and Li, H. DVERGE: Diversifying vulnerabilities for enhanced robust generation of ensembles. *Advances in Neural Information Processing Systems*, 33, 2020.

Yang, Y., Zhang, G., Katabi, D., and Xu, Z. ME-net: Towards effective adversarial robustness with matrix estimation. *arXiv preprint arXiv:1905.11971*, 2019.

Yang, Z., Li, L., Xu, X., Zuo, S., Chen, Q., Zhou, P., Rubinstein, B., Zhang, C., and Li, B. TRS: Transferability reduced ensemble via promoting gradient diversity and model smoothness. *Advances in Neural Information Processing Systems*, 34, 2021.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

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, 2019.

Zhang, Y., Suda, N., Lai, L., and Chandra, V. Hello edge: Keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*, 2017.

A. Proofs

In this section, we provide proofs for Theorems 4.1, 4.2 & 5.1 as well as any omitted derivations. We first state the following result due to (Melachrinoudis, 1997) on the ℓ_p norm projection of any point onto a hyper-plane:

Lemma A.1 (Melachrinoudis)**.** *For any hyper-plane $\mathcal{H} = \{\mathbf{u} \in \mathbb{R}^D : \mathbf{w}^{\mathrm{T}}\mathbf{u} + b = 0\} \subseteq \mathbb{R}^D$, point $\mathbf{x} \in \mathbb{R}^D$, and $p \geq 1$, then the ℓ_p norm projection $\mathbf{z}^*$ of $\mathbf{x}$ onto $\mathcal{H}$ can be obtained analytically as follows:*

$$\mathbf{z}^* = \mathbf{x} + \zeta\mathbf{g} = \mathbf{x} - \zeta \mathrm{sgn}(f(\mathbf{x})) \frac{|\mathbf{w}|^{q-1} \odot \mathrm{sgn}(\mathbf{w})}{\|\mathbf{w}\|_q^{q-1}} \tag{17}$$

where

$$\zeta = \min_{\mathbf{z} \in \mathcal{H}} \|\mathbf{z} - \mathbf{x}\|_p = \|\mathbf{z}^* - \mathbf{x}\|_p = \frac{\left|\mathbf{w}^{\mathrm{T}}\mathbf{x} + b\right|}{\|\mathbf{w}\|_q} = \frac{|f(\mathbf{x})|}{\|\mathbf{w}\|_q} \tag{18}$$

is the shortest ℓ_p distance between $\mathbf{x}$ and $\mathcal{H}$, $q \geq 1$ satisfying $\frac{1}{q} + \frac{1}{p} = 1$ defines the dual norm of p, $\odot$ is the Hadamard product, $\mathbf{g}$ is the unit ℓ_p norm vector from $\mathbf{x}$ towards the decision boundary, and $\mathbf{m} = |\mathbf{w}|^r$ denotes the element-wise operation $m_i = |w_i|^r \; \forall i \in [D]$.

The only dependence $\mathbf{g}$ has on $\mathbf{x}$ is the sign of $f(\mathbf{x})$, which essentially determines which side of the hyper-plane $\mathbf{x}$ lies in. The nice thing about Lemma A.1 is that it provides us with the tools for computing the optimal adversarial perturbations for linear classifiers, for all valid norms $p \geq 1$.

A.1. Fooling a Binary Linear Classifier

We also state and prove a simple and useful result for fooling binary linear classifiers. Let $(\mathbf{x}, y) \in \mathbb{R}^D \times \{-1, 1\}$ be a labeled data point that is correctly classified by f. That is we have $y(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) > 0$. Let $\boldsymbol{\delta} \in \mathbb{R}^D$ such that $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ is misclassified by f. That is, $y(\mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}} + b) < 0$. What can we say about $\boldsymbol{\delta}$?

Lemma A.2. *For any $(\mathbf{x}, y) \in \mathbb{R}^D \times \{-1, 1\}$ correctly classified by f and valid norm $p \geq 1$, $\boldsymbol{\delta}$ is adversarial to f at $(\mathbf{x}, y)$ if and only if the following conditions hold*

$$-y\mathbf{w}^{\mathrm{T}}\boldsymbol{\delta} > 0 \quad \& \quad \frac{\left|\mathbf{w}^{\mathrm{T}}\boldsymbol{\delta}\right|}{\|\mathbf{w}\|_q} > \zeta \tag{19}$$

where

$$\zeta = \frac{|f(\mathbf{x})|}{\|\mathbf{w}\|_q} = y\frac{f(\mathbf{x})}{\|\mathbf{w}\|_q} \tag{20}$$

is the shortest ℓ_p distance between $\mathbf{x}$ and the decision boundary of f.

Proof. We can write:

$$y(\mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}} + b) = y(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) + y\mathbf{w}^{\mathrm{T}}\boldsymbol{\delta} \tag{21}$$

Since $y(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) > 0$ holds by assumption, $y(\mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}} + b)$ is negative if and only if $-y\mathbf{w}^{\mathrm{T}}\boldsymbol{\delta} > y(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) > 0$. This implies we need $\boldsymbol{\delta}$ to have a positive projection on $-y\mathbf{w}$. Dividing both sides by $\|\mathbf{w}\|_q$, establishes the second condition as well. □

A.2. Derivation of (8)

Let $(\mathcal{F}, \boldsymbol{\alpha})$ be an REC of M BLCs:

$$f_i(\mathbf{x}) = \mathbf{w}_i^{\mathrm{T}}\mathbf{x} + b_i \quad \forall i \in [M] \tag{22}$$

where $\mathbf{w}_i \in \mathbb{R}^D$ and $b_i \in \mathbb{R}$ are the weight and bias, respectively, of the i^{th} BLC. The BLC f_i assigns label 1 to $\mathbf{x}$ if and only if $f_i(\mathbf{x}) > 0$, and -1 otherwise.

Define the binary cross-entropy loss l for classifier f evaluated at $(\mathbf{x}, y)$:

$$\begin{aligned} l(\mathbf{x}, y) = l\left(f(\mathbf{x}), y\right) &= -\log\left(\frac{e^{f(\mathbf{x})}}{1 + e^{f(\mathbf{x})}}\right)\left(\frac{1+y}{2}\right) - \log\left(\frac{1}{1 + e^{f(\mathbf{x})}}\right)\left(\frac{1-y}{2}\right) \\ &= -\log\left(z\right)\tilde{y} - \log\left(1 - z\right)\left(1 - \tilde{y}\right) \end{aligned} \tag{23}$$

where $z = \frac{e^{f(\mathbf{x})}}{1+e^{f(\mathbf{x})}}$ is the empirical probability of assigning label 1 to $\mathbf{x}$ by f and $\tilde{y} = \frac{1+y}{2} \in \{0, 1\}$ is the binarized version of the signed label $y \in \{-1, 1\}$.

Computing the gradient of l:

Let us first compute:

$$\begin{aligned}\nabla_{\mathbf{x}} \log(z) &= \nabla_{\mathbf{x}} \log\left(e^{f(\mathbf{x})}\right) - \nabla_{\mathbf{x}} \log\left(1 + e^{f(\mathbf{x})}\right) \\ &= \nabla_{\mathbf{x}} f(\mathbf{x}) - \frac{1}{1+e^{f(\mathbf{x})}} \nabla_{\mathbf{x}} e^{f(\mathbf{x})} \\ &= \mathbf{w} - \frac{e^{f(\mathbf{x})}}{1+e^{f(\mathbf{x})}} \mathbf{w} = (1-z)\mathbf{w}\end{aligned} \tag{24}$$

Similarly, we compute:

$$\begin{aligned}\nabla_{\mathbf{x}} \log(1-z) &= \mathbf{0} - \nabla_{\mathbf{x}} \log\left(1 + e^{f(\mathbf{x})}\right) \\ &= -\frac{1}{1+e^{f(\mathbf{x})}} \nabla_{\mathbf{x}} e^{f(\mathbf{x})} \\ &= -\frac{e^{f(\mathbf{x})}}{1+e^{f(\mathbf{x})}} \mathbf{w} = -z\mathbf{w}\end{aligned} \tag{25}$$

The gradient of $l(\mathbf{x}, y)$ w.r.t. $\mathbf{x}$ can thus be computed:

$$\begin{aligned}\nabla_{\mathbf{x}} l(\mathbf{x}, y) &= -\nabla_{\mathbf{x}} \log(z)\, \tilde{y} - \nabla_{\mathbf{x}} \log(1-z)\,(1-\tilde{y}) \\ &= -(1-z)\tilde{y}\mathbf{w} + z(1-\tilde{y})\mathbf{w} \\ &= (-(1-z)\tilde{y} + z(1-\tilde{y}))\mathbf{w} \\ &= (-\tilde{y} + z\tilde{y} + z - z\tilde{y})\mathbf{w} \\ &= (-\tilde{y} + z)\mathbf{w}\end{aligned} \tag{26}$$

Note that $z \in (0, 1)$ and $\tilde{y} \in \{0, 1\}$, therefore $-\tilde{y} + z$ has the same sign as $-y$. If $y = 1$, then $\tilde{y} = 1$, and $-1 + z \in (-1, 0)$ is negative. If $y = -1$, then $\tilde{y} = 0$, and $z \in (0, 1)$ is positive. Therefore, we can write:

$$\nabla_{\mathbf{x}} l(\mathbf{x}, y) = -y\lambda\mathbf{w} \tag{27}$$

where

$$\begin{aligned}\lambda &= |-(1-z)\tilde{y} + z(1-\tilde{y})| = (1-z)\tilde{y} + z(1-\tilde{y}) \\ &= \frac{1}{2}\frac{1+y+(1-y)e^{f(\mathbf{x})}}{1+e^{f(\mathbf{x})}} \in (0, 1)\end{aligned} \tag{28}$$

Gradient of the expected loss:

Using (27), we can compute the gradient $\mathbf{g}$ of the expected loss $\mathbb{E}[l(\mathbf{x}+\boldsymbol{\delta}, y)]$ employed by APGD:

$$\begin{aligned}\mathbf{g} &= \nabla_{\mathbf{x}} \mathbb{E}[l(\mathbf{x}+\boldsymbol{\delta}, y)] = \nabla_{\mathbf{x}} \sum_{i=1}^{M} [\alpha_i l_i(\mathbf{x}+\boldsymbol{\delta}, y)] \\ &= \sum_{i=1}^{M} [\alpha_i \nabla_{\mathbf{x}} l_i(\mathbf{x}+\boldsymbol{\delta}, y)] = \sum_{i=1}^{M} [\alpha_i(-y\lambda_i \mathbf{w}_i)] \\ &= -y \sum_{i=1}^{M} \alpha_i \lambda_i \mathbf{w}_i\end{aligned} \tag{29}$$

where $\forall i \in [M]$:

$$\lambda_i = \frac{1}{2}\frac{1+y+(1-y)e^{f_i(\mathbf{x}+\boldsymbol{\delta})}}{1+e^{f_i(\mathbf{x}+\boldsymbol{\delta})}} \in (0, 1) \tag{30}$$

A.3. Proof of Theorem 4.1

We provide a more detailed proof for Theorem 4.1, restated below:

Theorem (Restated). For any REC $(\mathcal{F}, \boldsymbol{\alpha})$ consisting of BLCs and any data-point $(\mathbf{x}, y)$, there exists a sequence of auxiliary BLCs $\{\bar{f}^{(k)}\}_{k=1}^{K}$ such that in the k^{th} iteration, the output of APGD against the REC is equal to the output of PGD against $\bar{f}^{(k)}$.

Proof. Let $(\mathcal{F}, \boldsymbol{\alpha})$ be an arbitrary REC of BLCs: $f_i(\mathbf{x}) = \mathbf{w}_i^{\mathsf{T}}\mathbf{x} + b_i$ and $(\mathbf{x}, y) \in \mathbb{R}^D \times \{-1, 1\}$ be an arbitrary labeled data point. Recall the APGD update rule is:

$$\begin{aligned}
\boldsymbol{\delta}^{(k)} &= \Pi_\epsilon^p\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\nabla_{\mathbf{x}}\mathbb{E}\left[l(\mathbf{x}+\boldsymbol{\delta}^{(k-1)}, y)\right]\right)\right) \\
&= \Pi_\epsilon^p\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\nabla_{\mathbf{x}}\sum_{i=1}^{M}\left[\alpha_i l_i(\mathbf{x}+\boldsymbol{\delta}^{(k-1)}, y)\right]\right)\right) \\
&= \Pi_\epsilon^p\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\sum_{i=1}^{M}\left[\alpha_i \nabla_{\mathbf{x}} l_i(\mathbf{x}+\boldsymbol{\delta}^{(k-1)}, y)\right]\right)\right) \\
&= \Pi_\epsilon^p\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\sum_{i=1}^{M}\left[\alpha_i \mathbf{g}_i^{(k-1)}\right]\right)\right)
\end{aligned} \tag{31}$$

where $l_i(\mathbf{x}, y) = l(f_i(\mathbf{x}), y)$ is the binary cross-entropy loss evaluated at $f_i(\mathbf{x})$. We can compute the gradients $\mathbf{g}_i^{(k-1)}$ analytically $\forall i \in [M], \forall k \in [K]$:

$$\mathbf{g}_i^{(k-1)} = -y\lambda_i^{(k-1)}\mathbf{w}_i \tag{32}$$

where

$$\lambda_i^{(k-1)} = \frac{1}{2}\frac{1 + y + (1-y)e^{f_i(\mathbf{x}+\boldsymbol{\delta}^{(k-1)})}}{1 + e^{f_i(\mathbf{x}+\boldsymbol{\delta}^{(k-1)})}} \in (0, 1) \tag{33}$$

We construct the k^{th} auxiliary classifier as follows:

$$\bar{f}^{(k)} = \bar{\mathbf{w}}_k^{\mathsf{T}}\mathbf{x} + \bar{b}_k = \left(\sum_{i=1}^{M}\alpha_i\lambda_i^{(k-1)}\mathbf{w}_i\right)^{\mathsf{T}}\mathbf{x} + \bar{b}_k \tag{34}$$

for some arbitrary $\bar{b}_k \in \mathbb{R}$. We now proceed to prove the result via induction on k, assuming both the APGD and PGD algorithms share the same random initialization $\boldsymbol{\delta}^{(0)}$. Let $\{\bar{\boldsymbol{\delta}}^{(k)}\}$ be the sequence of PGD outputs when attacking the auxiliary classifier $\bar{f}^{(k)}$. Assume that $\bar{\boldsymbol{\delta}}^{(k-1)} = \boldsymbol{\delta}^{(k-1)}$ for some k, i.e., the perturbation generated by APGD applied to the REC is equal to the perturbation generated by PGD applied to the auxiliary classifier. Thus, the k^{th} PGD output will be:

$$\begin{aligned}
\bar{\boldsymbol{\delta}}^{(k)} &= \Pi_\epsilon^p\left(\bar{\boldsymbol{\delta}}^{(k-1)} + \eta\mu_p\left(\nabla_{\mathbf{x}} l\left(\bar{f}^{(k)}(\mathbf{x}+\bar{\boldsymbol{\delta}}^{(k-1)}), y\right)\right)\right) \\
&= \Pi_\epsilon^p\left(\bar{\boldsymbol{\delta}}^{(k-1)} + \eta\mu_p\left(-y\bar{\lambda}^{(k)}\bar{\mathbf{w}}_k\right)\right) \\
&= \Pi_\epsilon^p\left(\bar{\boldsymbol{\delta}}^{(k-1)} + \eta\mu_p\left(-y\bar{\mathbf{w}}_k\right)\right) \\
&= \Pi_\epsilon^p\left(\bar{\boldsymbol{\delta}}^{(k-1)} + \eta\mu_p\left(-y\left[\sum_{i=1}^{M}\alpha_i\lambda_i^{(k-1)}\mathbf{w}_i\right]\right)\right) \\
&= \Pi_\epsilon^p\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\sum_{i=1}^{M}\left[\alpha_i\mathbf{g}_i^{(k-1)}\right]\right)\right) \\
&= \boldsymbol{\delta}^{(k)}
\end{aligned} \tag{35}$$

which is equal to the APGD output in (31). Note that in (35) we use the following property:

$$\mu_p(\gamma\mathbf{u}) = \text{sgn}(\gamma)\mu_p(\mathbf{u}) \quad \forall\mathbf{u} \in \mathbb{R}^D, \gamma \in \mathbb{R} \tag{36}$$

coupled with the fact that $\bar{\lambda}^{(k)} \in (0, 1)$.

Therefore, we have established the inductive step $\forall k$. Since both algorithms start from the same random initialization, then we also establish the base case: $\bar{\boldsymbol{\delta}}^{(0)} = \boldsymbol{\delta}^{(0)}$.

□

A.4. Proof of Theorem 4.2

We provide a more detailed proof for Theorem 4.2, restated below:

Theorem (Restated). The APGD algorithm for randomized ensembles of classifiers is inconsistent.

Proof. To prove inconsistency, we construct an example where the algorithm fails to generate an adversarial perturbation given one exists. Consider the following setup:

1. two binary linear classifiers with $\mathbf{w}_1 = -\mathbf{w}_2 = \mathbf{w} \neq \mathbf{0}$, and $b_1 = b_2 = b > 0$
2. equiprobable sampling $\boldsymbol{\alpha} = (0.5, 0.5)$
3. $(\mathbf{x}, y) = (\mathbf{0}, 1)$ with norm-bound $\epsilon = 2b\frac{\|\mathbf{w}\|_p}{\|\mathbf{w}\|_2^2}$

It is easy to show that both BLCs correctly classify $\mathbf{x}$, since $f_1(\mathbf{x}) = f_2(\mathbf{x}) = b > 0$, thus we have the condition $L(\mathbf{x}, y, \boldsymbol{\alpha}) = 1$. We also have a trivial choice of $\boldsymbol{\delta} = \epsilon\mathbf{w}/\|\mathbf{w}\|_p$, which satisfies both necessary and sufficient conditions for fooling f_2 shown below:

$$-\mathbf{w}_2^{\mathrm{T}}\boldsymbol{\delta} = \mathbf{w}^{\mathrm{T}}\boldsymbol{\delta} = \epsilon\frac{\mathbf{w}^{\mathrm{T}}\mathbf{w}}{\|\mathbf{w}\|_p} = \epsilon\frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_p} > 0 \quad \& \quad \left|\mathbf{w}_2^{\mathrm{T}}\boldsymbol{\delta}\right| / \|\mathbf{w}_2\|_q = \frac{\mathbf{w}^{\mathrm{T}}\boldsymbol{\delta}}{\|\mathbf{w}\|_q} = \epsilon\frac{\|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_q\|\mathbf{w}\|_p} = \frac{2b}{\|\mathbf{w}\|_q} > \zeta_2 \tag{37}$$

However, we observe that $\boldsymbol{\delta}$ does not fool f_1, since $-y(\mathbf{w}_1{}^{\mathrm{T}}\boldsymbol{\delta}) = -y(2b) < 0$. Therefore, we have constructed a norm-bounded perturbation $\boldsymbol{\delta}$ such that $L(\mathbf{x} + \boldsymbol{\delta}, y, \boldsymbol{\alpha}) = 0.5 < 1$.

Next, we show that the APGD algorithm evaluated against this REC cannot generate a norm-bounded perturbation that is adversarial. Recall that, starting from some random $\boldsymbol{\delta}^{(0)}$ such that $\|\boldsymbol{\delta}^{(0)}\|_p \leq \epsilon$, the APGD algorithm performs the following update rule:

$$\begin{aligned}
\boldsymbol{\delta}^{(k)} &= \Pi_\epsilon^p\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\nabla_{\mathbf{x}}\mathbb{E}\left[l(\mathbf{x} + \boldsymbol{\delta}^{(k-1)}, y)\right]\right)\right) \\
&= \Pi_\epsilon^p\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\nabla_{\mathbf{x}}\sum_{i=1}^{2}\left[\alpha_i l_i(\mathbf{x} + \boldsymbol{\delta}^{(k-1)}, y)\right]\right)\right) \\
&= \Pi_\epsilon^p\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\sum_{i=1}^{2}\left[\alpha_i\nabla_{\mathbf{x}} l_i(\mathbf{x} + \boldsymbol{\delta}^{(k-1)}, y)\right]\right)\right) \\
&= \Pi_\epsilon^p\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\sum_{i=1}^{2}\left[\alpha_i\mathbf{g}_i^{(k-1)}\right]\right)\right)
\end{aligned} \tag{38}$$

where $l_i(\mathbf{x}, y) = l(f_i(\mathbf{x}), y)$ is the binary cross-entropy loss evaluated for $f_i(\mathbf{x})$. We can compute the gradients $\mathbf{g}_i^{(k-1)}$ analytically $\forall i \in \{1, 2\}, \forall k \in [K]$:

$$\mathbf{g}_i^{(k-1)} = -y\lambda_i^{(k-1)}\mathbf{w}_i \tag{39}$$

where

$$\lambda_i^{(k-1)} = \frac{1}{2}\frac{1 + y + (1 - y)e^{f_i(\mathbf{x}+\boldsymbol{\delta}^{(k-1)})}}{1 + e^{f_i(\mathbf{x}+\boldsymbol{\delta}^{(k-1)})}} \in (0, 1) \tag{40}$$

Assume that $\boldsymbol{\delta}^{(0)}$ satisfies $\mathbf{w}^{\mathrm{T}}\boldsymbol{\delta}^{(0)} = 0$, therefore we have $f_1(\mathbf{x}+\boldsymbol{\delta}^{(0)}) = f_2(\mathbf{x}+\boldsymbol{\delta}^{(0)}) = b$. The expected gradient below is employed by APGD to obtain the next perturbation $\boldsymbol{\delta}^{(1)}$:

$$\begin{aligned}
\sum_{i=1}^{2}\left[\alpha_i \mathbf{g}_i^{(0)}\right] &= -y\sum_{i=1}^{2}\left[\alpha_i \lambda_i^{(0)}\mathbf{w}_i\right] = -\frac{y}{2}\left(\lambda_1^{(0)}\mathbf{w} - \lambda_2^{(0)}\mathbf{w}\right) = -\frac{y}{2}\mathbf{w}\left(\lambda_1^{(0)} - \lambda_2^{(0)}\right) \\
&= -\frac{y}{2}\mathbf{w}\left(\frac{1}{2}\frac{1+y+(1-y)e^{f_1(\mathbf{x}+\boldsymbol{\delta}^{(0)})}}{1+e^{f_1(\mathbf{x}+\boldsymbol{\delta}^{(0)})}} - \frac{1}{2}\frac{1+y+(1-y)e^{f_2(\mathbf{x}+\boldsymbol{\delta}^{(0)})}}{1+e^{f_2(\mathbf{x}+\boldsymbol{\delta}^{(0)})}}\right) \\
&= -\frac{y}{2}\mathbf{w}\left(\frac{1}{2}\frac{1+y+(1-y)e^{b}}{1+e^{b}} - \frac{1}{2}\frac{1+y+(1-y)e^{b}}{1+e^{b}}\right) \\
&= \mathbf{0}
\end{aligned} \tag{41}$$

This implies that the APGD algorithm is stuck in a fixed-point $\forall k$: $\boldsymbol{\delta}^{(k)} = \boldsymbol{\delta}^{(0)}$, which means the final perturbation obtained via PGD $\boldsymbol{\delta}_{\text{APGD}} = \boldsymbol{\delta}^{(0)}$ is a non-adversarial perturbation since $\mathbf{w}^{\mathrm{T}}\boldsymbol{\delta}^{(0)} = 0$. Hence, we have $L(\mathbf{x}+\boldsymbol{\delta}_{\text{APGD}}, y, \boldsymbol{\alpha}) = L(\mathbf{x}, y, \boldsymbol{\alpha}) = 1$. □

A.5. Proof of Theorem 5.1

We now can prove the main result, restated below:

Theorem (Restated). The ARC algorithm for randomized ensembles of binary linear classifiers is consistent.

Proof. To prove consistency, we need to show that for all random BLCs $(\mathcal{F}, \boldsymbol{\alpha})$, valid norms $p \geq 1$, norm-bounds ϵ, and labeled data-points $(\mathbf{x}, y)$ such that $L(\mathbf{x}, y, \boldsymbol{\alpha}) = 1$, the ARC generated perturbation $\boldsymbol{\delta}_{\text{ARC}}$ will result in $L(\mathbf{x}+\boldsymbol{\delta}_{\text{ARC}}, y, \boldsymbol{\alpha}) < 1$, given that $\exists\boldsymbol{\delta}$ that satisfies $\|\boldsymbol{\delta}\|_p \leq \epsilon$ and $L(\mathbf{x}+\boldsymbol{\delta}, y, \boldsymbol{\alpha}) < 1$.

First, let us establish some properties of the ARC algorithm. Define $\boldsymbol{\delta}^{(i)}$ to be the perturbation vector $\boldsymbol{\delta}$ at the end of the i^{th} iteration of the ARC algorithm. Similarly, define $v^{(i)}$ to be the cost function $L(\mathbf{x}+\boldsymbol{\delta}^{(i)}, y, \boldsymbol{\alpha})$. The algorithm is initialized with $\boldsymbol{\delta}^{(0)} = \mathbf{0}$ and $v^{(0)} = L(\mathbf{x}, y, \boldsymbol{\alpha}) = 1$. Subsequently, the algorithm terminates with $\boldsymbol{\delta}_{\text{ARC}} = \boldsymbol{\delta}^{(M)}$ with a final average classification $v^{(M)} = L(\mathbf{x}+\boldsymbol{\delta}_{\text{ARC}}, y, \boldsymbol{\alpha})$. We note that the sequence $\{v^{(i)}\}$ is non-increasing by design, with $v^{(i)} \leq L(\mathbf{x}, y, \boldsymbol{\alpha})$. This implies that the algorithm will never *increase* the mean classification, and if $\exists m \in [M]$ such that $v^{(m)} < v^{(m-1)}$, then the algorithm terminates with $v^{(M)} \leq v^{(m)} < 1$. Therefore, to show consistency, it is enough to find one such $m \in [M]$.

Assume that $L(\mathbf{x}, y, \boldsymbol{\alpha}) = 1$ and $\exists\boldsymbol{\delta}$ such that $\|\boldsymbol{\delta}\|_p \leq \epsilon$ and $L(\mathbf{x}+\boldsymbol{\delta}, y, \boldsymbol{\alpha}) < 1$, then from Lemma A.2 we know that $\exists m \in [M]$ such that:

$$-y\mathbf{w}_m^{\mathrm{T}}\boldsymbol{\delta} > 0 \quad \& \quad \frac{\left|\mathbf{w}_m^{\mathrm{T}}\boldsymbol{\delta}\right|}{\|\mathbf{w}_m\|_q} > \zeta_m \tag{42}$$

where

$$\zeta_m = \frac{|f_m(\mathbf{x})|}{\|\mathbf{w}_m\|_q} = y\frac{f_m(\mathbf{x})}{\|\mathbf{w}_m\|_q} \tag{43}$$

Combining the second inequality from (42) with Hölder's inequality, we get:

$$\|\mathbf{w}_m\|_q\|\boldsymbol{\delta}\|_p \geq \left|\mathbf{w}_m^{\mathrm{T}}\boldsymbol{\delta}\right| > \zeta_m\|\mathbf{w}_m\|_q \tag{44}$$

which implies that $\zeta_m < \epsilon$ since $\|\boldsymbol{\delta}\|_p \leq \epsilon$. We now have two cases to consider:

Case 1: If $m = 1$, the candidate perturbation will be $\hat{\boldsymbol{\delta}} = \epsilon\mathbf{g}$, where

$$\mathbf{g} = -y\frac{|\mathbf{w}_m|^{q-1} \odot \operatorname{sgn}(\mathbf{w}_m)}{\|\mathbf{w}_m\|_q^{q-1}} \tag{45}$$

We will now show that $\hat{\boldsymbol{\delta}}$ satisfies both necessary and sufficient conditions to fool f_m. First, we compute the dot-product:

$$-y{\mathbf{w}_m}^{\mathrm{T}}\hat{\boldsymbol{\delta}} = -y{\mathbf{w}_m}^{\mathrm{T}}(\epsilon\mathbf{g}) = -y\epsilon{\mathbf{w}_m}^{\mathrm{T}}\mathbf{g} = \epsilon\|\mathbf{w}_m\|_q > 0 \tag{46}$$

which is strictly positive, and the last equality uses the following:

$$\begin{aligned} -y\mathbf{w}_m{}^{\mathrm{T}}\mathbf{g} &= \frac{(-y)^2}{\|\mathbf{w}_m\|_q^{q-1}} \sum_{i=1}^{D} |w_{m,i}|^{q-1} \mathrm{sgn}(w_{m,i}) w_{m,i} \\ &= \frac{1}{\|\mathbf{w}_m\|_q^{q-1}} \sum_{i=1}^{D} |w_{m,i}|^{q-1} |w_{m,i}| \\ &= \frac{1}{\|\mathbf{w}_m\|_q^{q-1}} \sum_{i=1}^{D} |w_{m,i}|^{q} \\ &= \frac{1}{\|\mathbf{w}_m\|_q^{q-1}} \|\mathbf{w}_m\|_q^q = \|\mathbf{w}_m\|_q \end{aligned} \tag{47}$$

Second, we compute the following quantity:

$$\frac{\left|\mathbf{w}_m{}^{\mathrm{T}}\hat{\boldsymbol{\delta}}\right|}{\|\mathbf{w}_m\|_q} = \frac{-y\mathbf{w}_m{}^{\mathrm{T}}\hat{\boldsymbol{\delta}}}{\|\mathbf{w}_m\|_q} = \frac{\epsilon\|\mathbf{w}_m\|_q}{\|\mathbf{w}_m\|_q} = \epsilon > \zeta_m \tag{48}$$

which is strictly greater than ζ_m. Therefore, the candidate $\hat{\boldsymbol{\delta}}$ satisfies both necessary and sufficient conditions for fooling f_m.

Case 2: If $m > 1$, then assume that up until the $(m-1)^{\text{th}}$ iteration the ARC algorithm has not found any adversarial perturbation, that is $v^{(m-1)} = 1$, otherwise the algorithm will terminate with an adversarial perturbation. Let $\mathbf{u} = \boldsymbol{\delta}^{(m-1)}$, and by assumption we have $L(\mathbf{x}+\mathbf{u}, y, \boldsymbol{\alpha}) = 1$. During the m^{th} iteration, we have the candidate perturbation $\hat{\boldsymbol{\delta}} = \gamma(\mathbf{u}+\beta\mathbf{g})$ where

$$\mathbf{g} = -y\frac{|\mathbf{w}_m|^{q-1} \odot \mathrm{sgn}(\mathbf{w}_m)}{\|\mathbf{w}_m\|_q^{q-1}} \tag{49}$$

and

$$\gamma = \frac{\epsilon}{\|\mathbf{u}+\beta\mathbf{g}\|_p} > 0 \tag{50}$$

to ensure $\|\hat{\boldsymbol{\delta}}\|_p = \epsilon$.

Since $\zeta_m < \epsilon$, the choice of β for the m^{th} iteration will be:

$$\beta = \frac{\epsilon}{\epsilon - \zeta_m}\left|\frac{y\mathbf{w}_m{}^{\mathrm{T}}\mathbf{u}}{\|\mathbf{w}_m\|_q} + \zeta_m\right| + \rho \tag{51}$$

where $\rho > 0$ is a fixed arbitrarily small positive number. Note that $\|\mathbf{u}\|_p$ can either be 0 or ϵ by design. If $\|\mathbf{u}\|_p = 0$, then the same proof from case 1 follows, otherwise assume $\|\mathbf{u}\|_p = \epsilon$.

We will now show that $\hat{\boldsymbol{\delta}}$ satisfies both necessary and sufficient conditions to fool f_m. First, we compute the dot-product:

$$-y\mathbf{w}_m{}^{\mathrm{T}}\hat{\boldsymbol{\delta}} = -y\gamma\mathbf{w}_m{}^{\mathrm{T}}(\mathbf{u}+\beta\mathbf{g}) = -y\gamma\mathbf{w}_m{}^{\mathrm{T}}\mathbf{u} - y\beta\gamma\mathbf{w}_m{}^{\mathrm{T}}\mathbf{g} = -y\gamma\mathbf{w}_m{}^{\mathrm{T}}\mathbf{u} + \beta\gamma\|\mathbf{w}_m\|_q \tag{52}$$

Therefore, in order to satisfy the first condition in (42), we need $-y\mathbf{w}_m{}^{\mathrm{T}}\hat{\boldsymbol{\delta}}$ to be strictly positive, which is satisfied by any β satisfying:

$$\beta > \frac{y\mathbf{w}_m{}^{\mathrm{T}}\mathbf{u}}{\|\mathbf{w}_m\|_q} \tag{53}$$

However, using the fact that $\epsilon > \zeta_m > 0$, our choice of β satisfies this constraint since:

$$\beta = \frac{\epsilon}{\epsilon - \zeta_m}\left|\frac{y\mathbf{w}_m{}^{\mathrm{T}}\mathbf{u}}{\|\mathbf{w}_m\|_q} + \zeta_m\right| + \rho > \frac{\epsilon}{\epsilon - \zeta_m}\left(\frac{y\mathbf{w}_m{}^{\mathrm{T}}\mathbf{u}}{\|\mathbf{w}_m\|_q} + \zeta_m\right) > \frac{y\mathbf{w}_m{}^{\mathrm{T}}\mathbf{u}}{\|\mathbf{w}_m\|_q} \tag{54}$$

Second, we compute the following quantity required for the second condition:

$$\frac{\left|\mathbf{w}_m{}^{\mathrm{T}}\hat{\boldsymbol{\delta}}\right|}{\|\mathbf{w}_m\|_q} = \frac{-y\mathbf{w}_m{}^{\mathrm{T}}\hat{\boldsymbol{\delta}}}{\|\mathbf{w}_m\|_q} = \frac{-y\gamma\mathbf{w}_m{}^{\mathrm{T}}\mathbf{u} + \beta\gamma\|\mathbf{w}_m\|_q}{\|\mathbf{w}_m\|_q} \tag{55}$$

In order to satisfy the second condition, we need:

$$\frac{-y\gamma {\mathbf{w}_m}^{\mathrm{T}}\mathbf{u} + \beta\gamma\|\mathbf{w}_m\|_q}{\|\mathbf{w}_m\|_q} > \zeta_m \tag{56}$$

which is equivalent to:

$$\beta > \frac{y{\mathbf{w}_m}^{\mathrm{T}}\mathbf{u}}{\|\mathbf{w}_m\|_q} + \frac{\zeta_m}{\gamma} \tag{57}$$

Recall that:

$$\gamma = \frac{\epsilon}{\|\mathbf{u} + \beta\mathbf{g}\|_p} \geq \frac{\epsilon}{\|\mathbf{u}\|_p + |\beta|\|\mathbf{g}\|_p} = \frac{\epsilon}{\epsilon + \beta} \tag{58}$$

where the first inequality stems from the triangle inequality of the ℓ_p norm, and the second equality uses the fact that $\|\mathbf{u}\|_p = \epsilon$, $\|\mathbf{g}\|_p = 1$, and $\beta > 0$ by design. Using the inequality in (58), we can modify the inequality condition in (57) to:

$$\beta > \frac{y{\mathbf{w}_m}^{\mathrm{T}}\mathbf{u}}{\|\mathbf{w}_m\|_q} + \frac{\zeta_m(\epsilon + \beta)}{\epsilon} \geq \frac{y{\mathbf{w}_m}^{\mathrm{T}}\mathbf{u}}{\|\mathbf{w}_m\|_q} + \frac{\zeta_m}{\gamma} \tag{59}$$

which means it is enough to require that:

$$\beta\left(\frac{\epsilon - \zeta_m}{\epsilon}\right) > \frac{y{\mathbf{w}_m}^{\mathrm{T}}\mathbf{u}}{\|\mathbf{w}_m\|_q} + \zeta_m \tag{60}$$

for the second condition to hold. It is easy to see our choice of β satisfies this strict inequality (due to $\rho > 0$). In fact, we can easily show that $\frac{y{\mathbf{w}_m}^{\mathrm{T}}\mathbf{u}}{\|\mathbf{w}_m\|_q} + \zeta_m$ is guaranteed to be positive in this scenario due to our assumption that $\mathbf{x} + \mathbf{u}$ is correctly classified by f_m, which removes the looseness of the absolute value in our choice of β. Thus we have shown that, given the assumptions, we can find a $m \in [M]$ such that $\mathbf{x} + \boldsymbol{\delta}^{(m)}$ is misclassified by f_m, which implies $v^{(m)} < v^{(m-1)} = 1$, which concludes the proof that the ARC algorithm is consistent. □

B. Experimental Setup Details

B.1. Training Hyper-parameters

In this section, we provide the training details alongside the choice of hyper-parameters for all our experiments. For SVHN, CIFAR-10, and CIFAR-100 datasets, we establish strong adversarially trained baselines (via PGD) following the approach of (Rice et al., 2020) which utilizes early stopping to avoid robust over-fitting. As for ImageNet, we use the popular free training (FT) (Shafahi et al., 2019) method for faster and more efficient adversarial training. We employ standard data augmentation techniques (random cropping and random horizontal flips) for all models. Following standard practice, we also apply input normalization as part of the model, so that the adversary operates on physical images $\mathbf{x} \in [0,1]^D$. For all our experiments, the same ℓ_p norm is used during both training and evaluation.

For BAT training, as mentioned in Section 6, we re-use the AT models as the first model (f_1) in the REC, and then train a second model (f_2) using the adversarial samples of the *first* model only. The same network architecture, training algorithm, and hyper-parameters will be used for both models. A single workstation with two NVIDIA Tesla P100 GPUs is used for running all the training experiments. Below are the hyper-parameter details for each dataset:

SVHN: Models are trained for 200 epochs, using a PGD adversary with $K = 7$ iterations with: $\epsilon = 8/255$ and $\eta = 2/255$ for ℓ_∞ AT, and $\epsilon = 128/255$ and $\eta = 32/255$ for ℓ_2 AT. We use stochastic gradient descent (SGD) with momentum (0.9), 128 mini-batch size, and a step-wise learning rate decay set initially at 0.1 and divided by 10 at epochs 100 and 150. We employ weight decay of 2×10^{-4}.

CIFAR-10: Models are trained for 200 epochs, using a PGD adversary with $K = 7$ iterations with: $\epsilon = 8/255$ and $\eta = 2/255$ for ℓ_∞ AT, and $\epsilon = 128/255$ and $\eta = 32/255$ for ℓ_2 AT. We use SGD with momentum (0.9), 128 mini-batch size, and a step-wise learning rate decay set initially at 0.1 and divided by 10 at epochs 100 and 150. We employ weight decay of: 2×10^{-4} for ResNet-20 and MobileNetV1, and 5×10^{-4} for VGG-16, ResNet-18, and WideResNet-28-4.

CIFAR-100: Models are trained for 200 epochs, using a PGD adversary with $K = 7$ iterations with: $\epsilon = 8/255$ and $\eta = 2/255$ for ℓ_∞ AT, and $\epsilon = 128/255$ and $\eta = 32/255$ for ℓ_2 AT. We use SGD with momentum (0.9), 128 mini-batch size, and a step-wise learning rate decay set initially at 0.1 and divided by 10 at epochs 100 and 150. We employ weight decay of 2×10^{-4}.

ImageNet: Models are trained for a total of 90 epochs with mini-batch replay $m = 4$, using: $\epsilon = 4/255$ for ℓ_∞ FT, and $\epsilon = 76/255$ for ℓ_2 FT. We use stochastic gradient descent (SGD) with momentum (0.9), 256 mini-batch size and a step-wise learning rate decay set initially at 0.1 and decayed by 10 every 30 epochs. We employ weight decay of 1×10^{-4}.

B.2. Evaluation Details

We evaluate the robustness of standard AT models using PGD (Madry et al., 2018). As for randomized ensembles, we establish baselines for comparison using the adaptive version of PGD (APGD) (Pinot et al., 2020) which uses the expected loss function (see Section 3). We use our proposed ARC algorithm as well, to demonstrate the vulnerability of RECs. In Appendix C.3, we investigate different methods for constructing adaptive and non-adaptive attacks using both PGD and C&W (Carlini & Wagner, 2017). The PGD/APGD attack details for each dataset are listed below. The same configurations are used for ARC as well, except for ℓ_∞ evaluations, where we find that setting $\eta = \epsilon$ yields best results.

SVHN, CIFAR-10, CIFAR-100: We use $K = 20$ iterations, with: $\epsilon = 8/255$ with $\eta = 2/255$ for ℓ_∞ PGD and $\epsilon = 128/255$ with $\eta = 32/255$ ℓ_2 PGD.

ImageNet: We use $K = 50$ iterations, with: $\epsilon = 4/255$ with $\eta = 1/255$ for ℓ_∞ PGD and $\epsilon = 128/255$ with $\eta = 32/255$ for ℓ_2 PGD.

C. Additional Experiments and Comparisons

C.1. Individual Model Performance

Table 4. Clean and robust accuracies of the individual members of BAT trained RECs from Table 2. Model robustness is measured via both PGD and ARC.

DATASET	NORM	MODEL	CLEAN ACCURACY [%]	ROBUST ACCURACY [%] PGD	ARC
SVHN	ℓ_2	f_1	93.07	68.35	65.50
		f_2	91.23	0.00	0.00
	ℓ_∞	f_1	90.53	53.55	49.01
		f_2	88.31	0.00	0.00
CIFAR-10	ℓ_2	f_1	83.36	62.43	61.13
		f_2	91.26	0.13	0.00
	ℓ_∞	f_1	75.96	45.66	42.36
		f_2	77.64	0.00	0.00
CIFAR-100	ℓ_2	f_1	53.43	34.60	31.88
		f_2	63.74	0.00	0.00
	ℓ_∞	f_1	44.37	22.29	18.36
		f_2	34.1	0.00	0.00
IMAGENET	ℓ_2	f_1	62.91	47.60	45.96
		f_2	64.60	0.00	0.00
	ℓ_∞	f_1	52.01	24.33	21.03
		f_2	55.70	0.00	0.00

In this section, we provide more details on the performance of the individual models from the BAT RECs constructed in Table 2 across different datasets. Table 4 reports the clean and robust accuracies of all the individual models. The robust accuracy is measured via both standard PGD and our ARC adversary. We consistently find that the performance of ARC and PGD in the standard single model setting to be similar, with ARC being slightly better. One thing to note is that, due to the nature of BAT, the second model f_2 is always *not* robust to its own adversarial perturbation. However, as shown in Fig. 5, the second model is very successful at defending against the adversarial samples of the first model, which is expected.

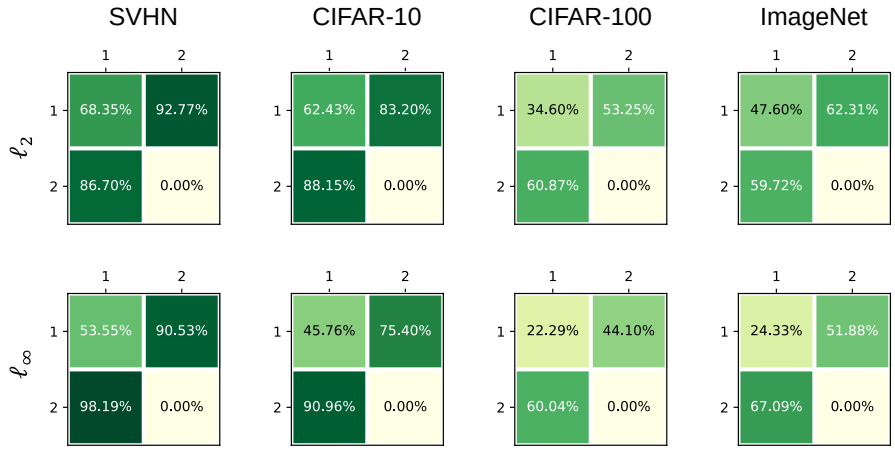


Figure 5. The cross-robustness matrices of all BAT REC from Table 2. As expected, BAT training induces an asymmetric ensemble were the first model is robust and the second model is completely compromised. The *false* sense of robustness from BAT RECs stems from the high robustness of the second model to the adversarial samples of the first model.

C.2. More Parameter Sweeps

In this section, we expand on the results in Section 6.3. Specifically, Fig. 6 provides a more complete version of Fig. 3 with added CIFAR-100 results, across two norms ℓ_∞ and ℓ_2. Across all datasets, norms, and parameter values, Fig. 6 consistently shows that the randomized ensembles obtained via BAT are more vulnerable than originally claimed. The ARC algorithm is much more effective at generating adversarial examples for RECs, in contrast to APGD.

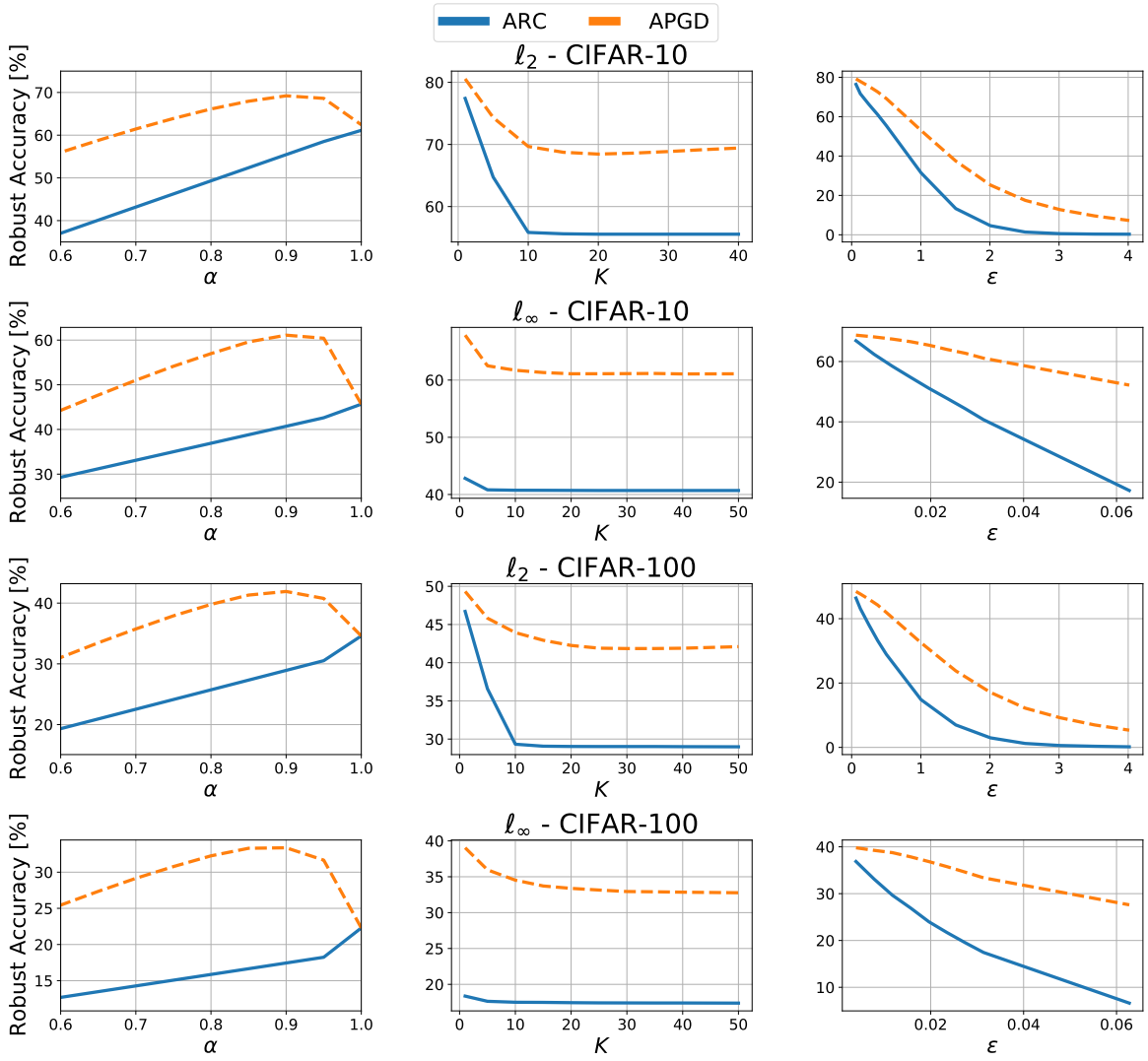


Figure 6. More comparisons between ARC and APGD using RECs of ResNet-20s obtained via BAT ($M = 2$).

C.3. More on Adaptive Attacks

In this section, we explore different ways of deploying PGD-based adversaries for attacking RECs and compare them with ARC. We also employ the C&W (Carlini & Wagner, 2017) attack for ℓ_2 norm-bounded perturbations, and adapt it to our setting.

C.3.1. Adaptive PGD

In Section 3, we introduced the standard PGD attack, which performs the following update rule $\forall k \in [K]$:

$$\boldsymbol{\delta}^{(k)} = \Pi_{\epsilon}^{p}\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\nabla_{\mathbf{x}} l(f(\mathbf{x}+\boldsymbol{\delta}^{(k-1)}), y)\right)\right) \tag{61}$$

where l is the loss function, f is the differentiable deterministic classifier, and $(\mathbf{x}, y)$ is the labeled data-point. It is clear that some changes have to made in order to use PGD for a randomized ensemble $(\mathcal{F}, \boldsymbol{\alpha})$. We propose some intuitive changes below.

PGD: Perhaps the simplest modification one can make is to randomize the attack as well! If the defender picks a model f_i at random from the ensemble with probability α_i, for some $i \in [M]$, then the attacker can also *independently* sample a model f_j at random with probability α_j and use it to construct the adversarial perturbation for $(\mathcal{F}, \boldsymbol{\alpha})$ via PGD. Again, the discrete nature of our setup allows us to compute exactly the expected robust accuracy, when both the attacker and defender are randomizing their strategies:

$$\bar{L} = \sum_{j=1}^{M} \alpha_j L(\mathbf{x}+\boldsymbol{\delta}_j, \boldsymbol{\alpha}) \tag{62}$$

where $\boldsymbol{\delta}_j$ is the adversarial perturbation obtained via PGD when attacking model f_j, and L is the conditional expected robust accuracy from (1).

PGD-1: Since the second model in BAT RECs is always not robust, one logical attack to consider is attacking the robust model only f_1, and ignore the second non-robust model. That is, we use standard PGD to attack f_1, and use the generated $\boldsymbol{\delta}$ to attack the REC.

APGD: The attacker can also adapt PGD by using the expected value of the loss function, and modifying (61) as follows:

$$\begin{aligned}\boldsymbol{\delta}^{(k)} &= \Pi_{\epsilon}^{p}\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\nabla_{\mathbf{x}}\mathbb{E}\left[l(f(\mathbf{x}+\boldsymbol{\delta}^{(k-1)}), y)\right]\right)\right) \\ &= \Pi_{\epsilon}^{p}\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\sum_{i=1}^{M}\left[\alpha_i \nabla_{\mathbf{x}} l(f_i(\mathbf{x}+\boldsymbol{\delta}^{(k-1)}), y)\right]\right)\right)\end{aligned} \tag{63}$$

Recall this is the adaptive PGD proposed in (Pinot et al., 2020) and what we adopted in this paper (see Sections 3 and 4) for robustness evaluation of randomized ensembles.

APGD-L: Another method for computed the EOT for randomized ensembles is to compute the expected classifier outputs (logits), instead of the expected loss, which will yield the following change to (61):

$$\begin{aligned}\boldsymbol{\delta}^{(k)} &= \Pi_{\epsilon}^{p}\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\nabla_{\mathbf{x}} l(\mathbb{E}\left[f(\mathbf{x}+\boldsymbol{\delta}^{(k-1)})\right], y)\right)\right) \\ &= \Pi_{\epsilon}^{p}\left(\boldsymbol{\delta}^{(k-1)} + \eta\mu_p\left(\nabla_{\mathbf{x}} l\left(\sum_{i=1}^{M}\alpha_i f_i(\mathbf{x}+\boldsymbol{\delta}^{(k-1)}), y\right)\right)\right)\end{aligned} \tag{64}$$

The authors of (Pinot et al., 2020) were also aware of this modification. However, they argued that averaging the logits may cause discrepancies, since logits from two different models can have completely different ranges for values. The compelling argument for using the loss function, is that it provides an elegant and natural way of normalizing the outputs. Note that, our analysis in Section 4 is also applicable for this version of adaptive PGD, and therefore we expect it to inherit the shortcomings of APGD and provide a *false* sense of robustness for RECs.

C.3.2. C&W Attack

The C&W attack (Carlini & Wagner, 2017) is another popular and powerful attack originally designed for compromising defensive distillation (Papernot et al., 2016) in neural networks. For a given classifier f and data-point $(\mathbf{x}, y)$, the C&W attack finds an adversarial perturbation by solving the following constraint optimization problem:

$$\boldsymbol{\delta}^{*} = \underset{\boldsymbol{\delta}+\mathbf{x}\in[0,1]^D}{\arg\min} \|\boldsymbol{\delta}\|_2 + ch(\mathbf{x}+\boldsymbol{\delta}) \tag{65}$$

where $h(\mathbf{x}+\boldsymbol{\delta})$ is some cost function such that $h(\mathbf{x}+\boldsymbol{\delta}) < 0$ if and only if $\arg\max[f(\mathbf{x}+\boldsymbol{\delta})]_i \neq y$, and $c > 0$ is a constant that is often optimized via binary search. Note that the C&W attack (Carlini & Wagner, 2017) tries to find the minimum

ℓ_2 norm perturbation that fools f, such that the perturbed image satisfies the box constraints. The resulting perturbation therefore is not guaranteed to be norm-bounded by some ϵ. However, a common workaround is to simply project the final perturbation onto the ℓ_2 ball of radius ϵ.

Solving (65) is very challenging from an optimization point of view. To facilitate the optimization, the authors propose a change of variable:

$$\boldsymbol{\delta} = \frac{1}{2}(\tanh(\boldsymbol{\theta}) + 1) - \mathbf{x} \tag{66}$$

where tanh is an element-wise hyperbolic tangent, which guarantees that $\boldsymbol{\delta} + \mathbf{x}$ satisfies the box constraints $\forall \boldsymbol{\theta} \in \mathbb{R}^D$. Therefore, the C&W attack solves for $\boldsymbol{\theta}^*$ via:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \|\frac{1}{2}(\tanh(\boldsymbol{\theta}) + 1) - \mathbf{x}\|_2^2 + ch(\frac{1}{2}(\tanh(\boldsymbol{\theta}) + 1)) \tag{67}$$

where h is:

$$h(\mathbf{u}) = \max\left\{[f(\mathbf{u})]_y - \max_{i \in [C] \setminus \{y\}} [f(\mathbf{u})]_i, -\kappa\right\} \tag{68}$$

and κ controls the confidence (usually set to 0). Finally, the optimization in (67) can be solved via gradient based algorithms, such as SGD or ADAM (Kingma & Ba, 2014), while $c > 0$ is chosen using binary search.

Analogous to adapting PGD, we will experiment with three version of the C&W attack for RECs:

C&W: The attacker randomly samples a classifier and attacks it via the classical C&W adversary.

C&W-1: The attacker only considers the first robust model (for BAT RECs) via the classical C&W adversary, akin to PGD-1.

AC&W: The attacker uses the expected value of h when solving (67), akin to APGD.

AC&W-L: The attacker uses the expected output of the REC (exepected logits) when solving (67), akin to APGD-L.

To ensure proper implementation, we adopt the open-source Foolbox (Rauber et al., 2017) implementation of the C&W attack. We adopt the common attack hyper-parameters: 9 steps for binary search with an initial constant $c = 0.001$, optimizer learning rate 0.01, confidence $\kappa = 0$, and run the attack for a total of $K = 50$ iterations.

C.3.3. ARC VS. ADAPTIVE ATTACKS

We now apply all proposed variants of PGD and C&W for attacking the RECs of ResNet-20s on CIFAR-10 (from Section 6.3). Note that the C&W variants will only be evaluated against the ℓ_2 trained REC. Figure. 7 plots the robust accuracy of all methods while sweeping the defender's sampling probability $\boldsymbol{\alpha} = (\alpha, 1 - \alpha)$. We make the following observations:

- The ARC algorithm outperforms all variants of PGD and C&W and across all values of α, and by massive margins ($\sim 20\%$) for $\alpha \in [0.5, 0.9]$.
- ARC demonstrates that there is no benefit in using RECs, since the highest robustness achieved corresponds to $\alpha = 1$. Note that $\alpha = 1$ reduces the REC to the deterministic adversarially trained classifier f_1.
- Adapting PGD via the expected loss, instead of computing the expected logits yields little to no difference, and the same can be said for C&W as well.
- Interestingly, attacking the first model only is a much more effective attack than any exsitng adaptive attack in the high sampling probabilty regime ($\alpha \in [0.8, 1]$). Despite this fact, ARC remains a much stronger attack for all values of α.
- The robustness evaluation of APGD and PGD follow completely different trends, with both providing *false* sense of robustness. The same can be said for AC&W and CW as well.

C.4. More on Ensemble Training Methods

In this section, we expand on the results from Section 6.4, where we constructed RECs from pre-trained DVERGE models (Yang et al., 2020). In addition to DVERGE, we experiment with ADP (Pang et al., 2019) and TRS (Yang et al., 2021) using

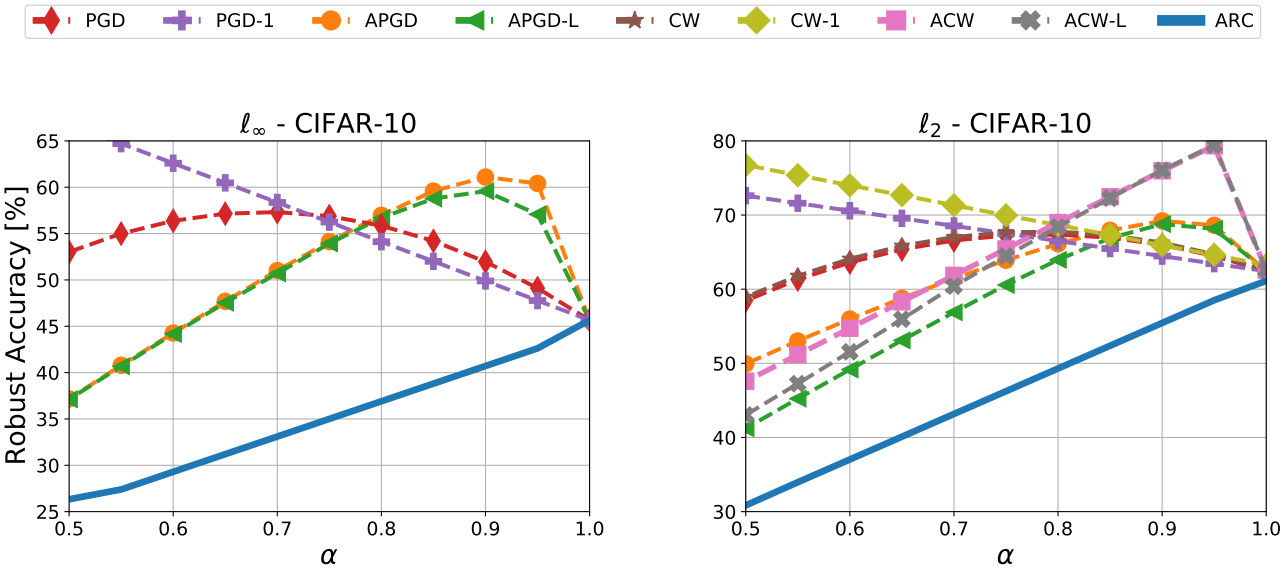
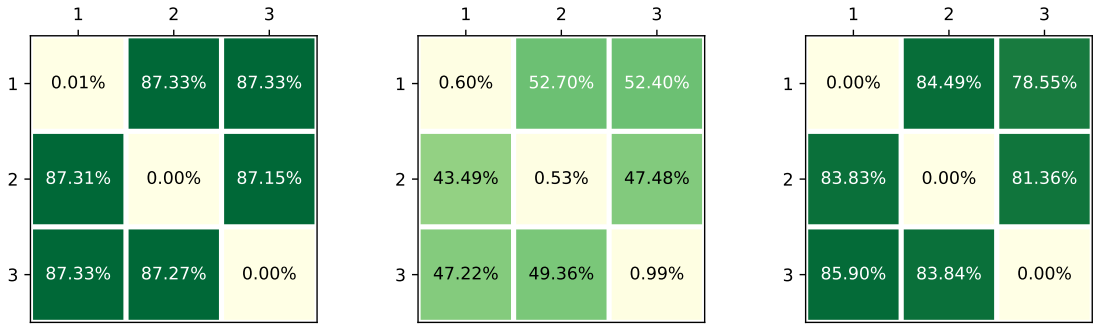


Figure 7. Robust accuracy vs. sampling probability of: ℓ_∞ (left) and ℓ_2 (right) BAT REC of ResNet-20s on CIFAR-10. Robust accuracy is evaluated using various adaptive attacks detailed in Appendix C.3 and the proposed ARC attack algorithm.

an ensemble of three ResNet-20s on CIFAR-10. We used the pre-trained ADP models that were publicly available thanks to (Yang et al., 2020), and trained TRS models using their publicly released code on GitHub (Yang et al., 2021).

We first plot the cross-robustness matrices for each ensemble in Fig. 8. None of the classifiers in the ensembles are *actually* robust, which is expected. These training methods are designed to improve the robustness to transfer attacks, which is why we notice high cross-robustness in all three matrices.

We construct RECs for each method via equiprobable sampling, and evaluate the robustness against both APGD and ARC in Fig. 9 for different norm radius ϵ values. A common observation to all three methods is that ARC consistently outperforms APGD, and with large margins for TRS and DVERGE. These experiments demonstrate that RECs are *vulnerable*, regardless of the training procedure thus motivating the need for improved adversarial training algorithms for randomized ensembles.



(a) (b) (c)

Figure 8. Cross-robustness matrices for: (a) DVERGE, (b) ADP, and (c) TRS trained ensembles of three ResNet-20s on CIFAR-10.

C.5. Randomized Ensembles from Independent Adversarially Trained Classifiers

In this section, we investigate the performance of RECs constructed from independent adversarially trained (IAT) deep nets. Specifically, we train M ResNet-20s using ℓ_2 AT on CIFAR-10. We use different random initializations for each network by

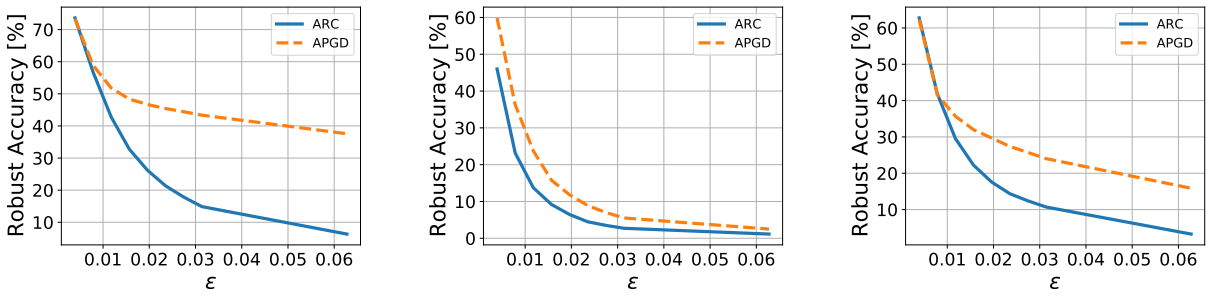


Figure 9. Robust accuracy vs. norm radius ϵ using both ARC and APGD for: (a) DVERGE, (b) ADP, and (c) TRS trained ensembles of three ResNet-20s on CIFAR-10.

changing the random seed. Doing so will result in a symmetric ensemble as seen via the cross-robustness matrix in Fig. 10a. In this setting, all the models are robust with $\sim 62\%$ robust accuracy, as opposed to the asymmetric BAT setting where the first model is robust and the second model is completely compromised. Using equiprobable sampling, we construct RECs from the M models and compare the performance of both ARC and APGD with varying ensemble size M in Fig. 10b, e.g., $M = 2$ means we only use the first 2 models and ignore the rest. Both ARC and APGD follow the same trend since the robustness of the ensemble increases with M. The ARC algorithm is consistently stronger than APGD, albeit the gap is relatively small compared to BAT. This is expected since all models in the ensemble are robust. We do note however that the improvement in robustness is limited and requires large ensemble size for it to be significant, e.g., $< 3\%$ improvement with $M = 8$. Finally, we strongly believe that future ensemble training methods need to employ IAT RECs as a baseline.

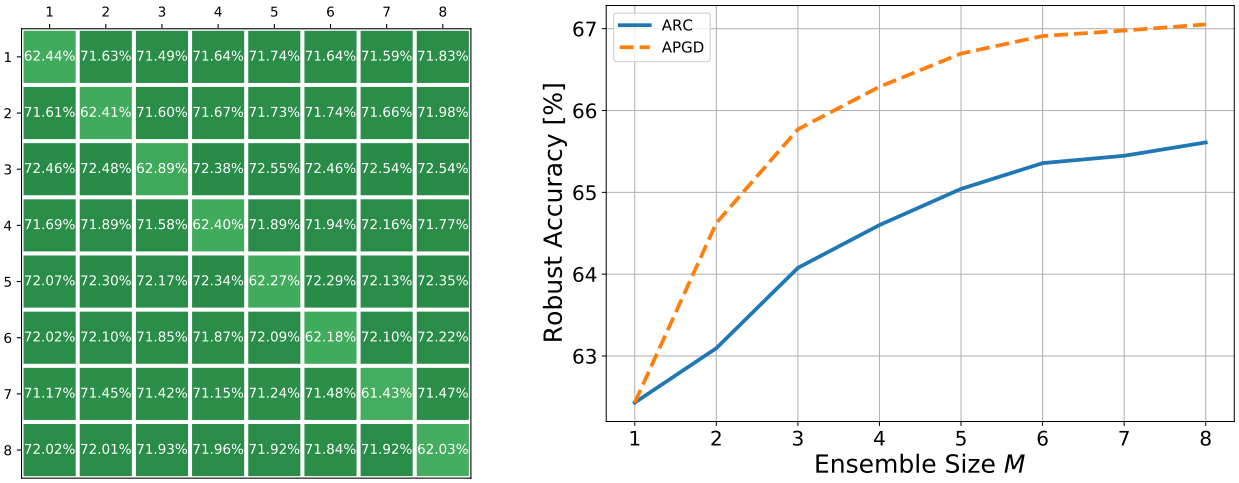


Figure 10. Robustness performance of an REC constructed from ℓ_2 independent adversarially trained models ($M = 8$): (a) cross-robustness matrix, and (b) robust accuracy vs. size of the ensemble M using both ARC and APGD.

C.6. Visualization of Adversarial Samples

In this section, we provide some visual examples of ℓ_∞ norm-bounded adversarial perturbations obtained via APGD and ARC. Specifically, Fig. 11 shows three examples from our ImageNet experiments, where each clean image (unperturbed) is correctly classified by a randomized ensemble of ResNet-18s (from Table 2). The APGD adversarial samples are also correctly classified by both networks in the ensemble, and thus are unable to fool the ensemble. In contrast, the ARC adversarial samples completely compromise the ensemble, i.e., they fool both the networks. All adversarial perturbations are ℓ_∞ norm-bounded with $\epsilon = 4/255$.

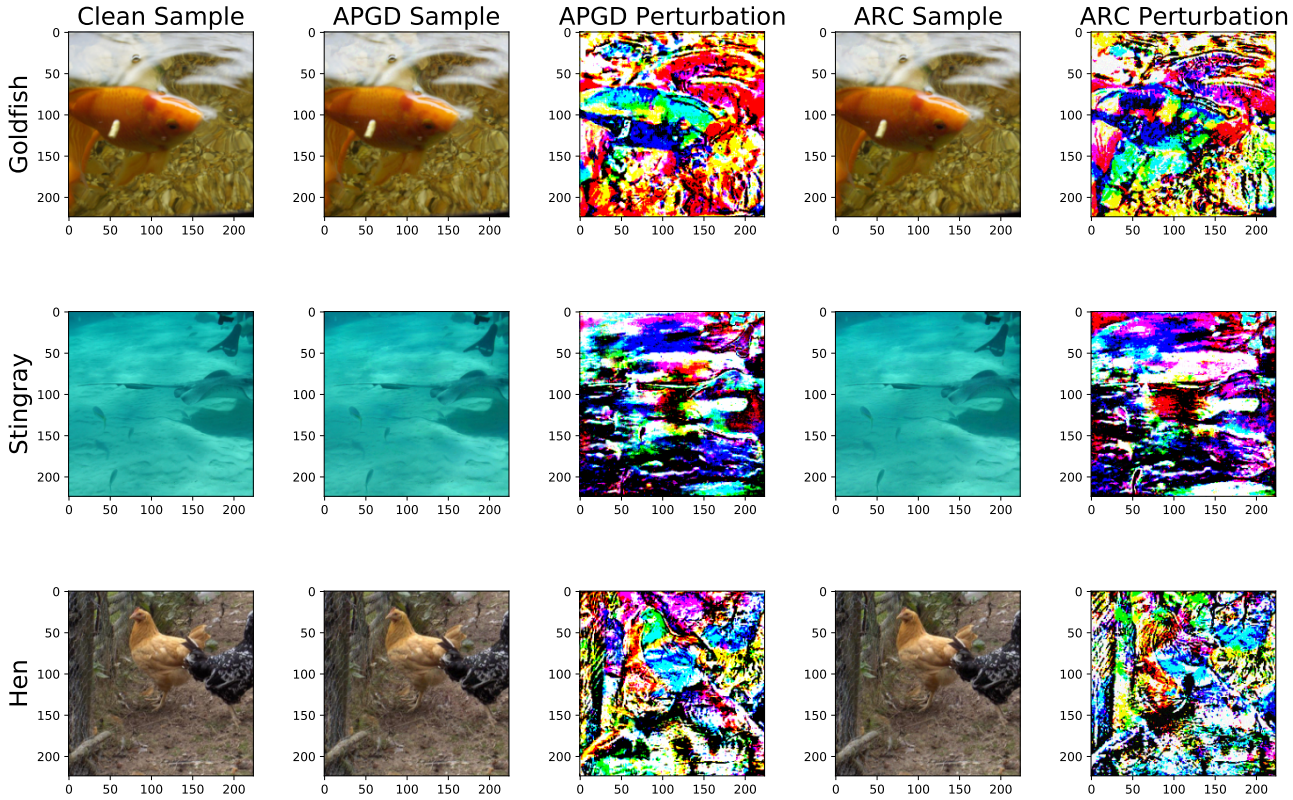


Figure 11. Visual examples of ℓ_∞ norm-bounded adversarial perturbations obtained via APGD and ARC evaluated against a randomized ensemble of ResNet-18s on ImageNet. All perturbations are ℓ_∞ norm-bounded with $\epsilon = 4/255$. The APGD and clean samples are correctly classified by all members of the ensemble, whereas the ARC samples are misclassified by all members of the same ensemble.

D. Scalability of the ARC Algorithm

In this section, we expand on Section 5.3 and provide more details on approximating the search procedure in (15) in the ARC Algorithm. Recall that the goal is to find the ℓ_p shortest distance $\tilde{\zeta}$ between $\mathbf{x}$ and the hyper-planes that capture the decision boundary of $\tilde{\mathcal{R}}_m(\mathbf{x})$, as well as the corresponding unit ℓ_p norm direction $\tilde{\mathbf{g}}$. The $C-1$ hyper-planes are defined by:

$$\mathcal{H}_j = \left\{ \mathbf{u} \in \mathbb{R}^D : \tilde{\mathbf{w}}_j^{\mathrm{T}} (\mathbf{u} - \mathbf{x}) + \tilde{h}_j = 0 \right\} \quad \forall j \in [C] \setminus \{m\} \tag{69}$$

where $m \in [C]$ is the label assigned to $\mathbf{x}$ by f, $\tilde{h}_j = [f(\mathbf{x})]_m - [f(\mathbf{x})]_j$ and $\tilde{\mathbf{w}}_j = \nabla [f(\mathbf{x})]_m - \nabla [f(\mathbf{x})]_j \; \forall j \in [C] \setminus \{m\}$. Therefore, in order to obtain $\tilde{\zeta}$ and $\tilde{\mathbf{g}}$, we find the closest hyper-plane:

$$n = \underset{j \in [C] \setminus \{m\}}{\arg\min} \frac{\left| \tilde{h}_j \right|}{\|\tilde{\mathbf{w}}_j\|_q} = \underset{j \in [C] \setminus \{m\}}{\arg\min} \; \tilde{\zeta}_j \tag{70}$$

and then compute:

$$\tilde{\zeta} = \tilde{\zeta}_n \quad \& \quad \tilde{\mathbf{g}} = -\frac{|\tilde{\mathbf{w}}_n|^{q-1} \odot \operatorname{sgn}(\tilde{\mathbf{w}}_n)}{\|\tilde{\mathbf{w}}_n\|_q^{q-1}} \tag{71}$$

The search in (70) can be computationally demanding for datasets with large number of classes, e.g., $C = 100$ for CIFAR-100 or $C = 1000$ for ImageNet, as the gradient computation routine required for finding $\tilde{\mathbf{w}}_j$ is very compute intensive, and has to be executed $C-1$ times, during every iteration k and classifier f_i. To alleviate this issue, we perform an approximate search procedure. Intuitively, we expect the closest hyper-plane n would correspond to the output logit $[f(\mathbf{x})]_n$ that is closest to $[f(\mathbf{x})]_m$. Therefore, instead of searching over all $C-1$ hyper-planes, we only need to search over $1 \leq G \leq C-1$ hyper-planes that correspond to the G largest logits $[f(\mathbf{x})]_j \; j \neq m$. Specifically, we propose approximating the search as follows:

1. compute the quantities $\tilde{h}_j \; \forall j \in [C] \setminus \{m\}$ as before
2. construct the sorted index set $\mathcal{J} = \{j_1, ..., j_G\} \subseteq [C] \setminus \{m\}$ such that:
$$0 < \tilde{h}_{j_1} \leq \tilde{h}_{j_2} \leq ... \leq \tilde{h}_{j_G} \leq \tilde{h}_t \quad \forall t \in [C] \setminus (\{m\} \cup \mathcal{J}) \tag{72}$$
3. search for the closest hyper-plane over the restricted set $\mathcal{J}$:
$$n = \underset{j \in \mathcal{J}}{\arg\min} \frac{\left| \tilde{h}_j \right|}{\|\tilde{\mathbf{w}}_j\|_q} = \underset{j \in \mathcal{J}}{\arg\min} \, \tilde{\zeta}_j \tag{73}$$

Table 5. The robust accuracy and run time of the proposed approximate version of ARC evaluated against two RECs on CIFAR-100. The evaluation was run on a single NVIDIA 1080 Ti GPU.

SEARCH SIZE (G)	RUN TIME [MIN]	ROBUST ACCURACY [%]	
		ℓ_2	ℓ_∞
1	2.06	29.79	18.59
2	2.56	29.08	17.71
3	3.05	28.97	17.55
4	3.56	28.92	17.45
10	6.44	28.88	17.32
50	26.46	28.88	17.32
99	49.91	28.88	17.32

The parameter G controls the accuracy-efficiency trade-off of the approximation, and setting $G = C - 1$ yields the exact search in (70).

To demonstrate the efficacy of this approximation, we use this version of ARC for evaluating the robustness of two RECs of ResNet-20's obtained via ℓ_2 BAT and ℓ_∞ BAT on CIFAR-100. We also measure the total evaluation time, i.e., the total time our script requires to evaluate the robustness of the REC using the entire 10000 samples of the CIFAR-100 testset. We use a workstation with a single NVIDIA 1080 Ti GPU and iterate over the testset with a mini-batch size of 256 for all evaluations. Table 5 reports the corresponding robustness and required evaluation time for different values of G. Table 5 demonstrates that using $G = 4$ provides the same robustness evaluation as the full ARC algorithm ($G = 99$), while requiring less than 4 minutes to run, as opposed to 50 minutes required by $G = 99$. This massive reduction in evaluation time, while maintaining the fidelity of robustness evaluation, allows us to scale the ARC algorithm to more complex datasets. Therefore, in all of our CIFAR-100 and ImageNet experiments, we will use this version of ARC with $G = 4$.