Lattice Climber Attack: Adversarial attacks for randomized mixtures of classifiers

Lucas Gnecco Heredia (🖂), Benjamin Negrevergne, and Yann Chevaleyre

LAMSADE, CNRS, Université Paris Dauphine - PSL
Correspondence to lucas.gnecco-heredia@dauphine.psl.eu

Abstract. Finite mixtures of classifiers (a.k.a. randomized ensembles) have been proposed as a way to improve robustness against adversarial attacks. However, existing attacks have been shown to not suit this kind of classifier. In this paper, we discuss the problem of attacking a mixture in a principled way and introduce two desirable properties of attacks based on a geometrical analysis of the problem (effectiveness and maximality). We then show that existing attacks do not meet both of these properties. Finally, we introduce a new attack called *lattice climber attack* with theoretical guarantees in the binary linear setting, and demonstrate its performance by conducting experiments on synthetic and real datasets.

Keywords: adversarial robustness · adversarial attacks · randomized classifiers · mixtures.

1 Introduction

Deep neural networks have been shown to be vulnerable to adversarial attacks [10], i.e. small perturbations that, although imperceptible to humans, manage to drastically change the predictions of the model. This observation has led to numerous efforts to understand this phenomenon [4] and started a series of publications introducing various techniques to train robust models [14] as well as new algorithms to attack them [22].

One research direction that has been explored is the use of *randomized classifiers*, which include a source of randomness in their predictions. Examples of such classifiers include stochastic pruning [9, 17], noise injection classifiers [12], classifiers with random input transformations [23, 20], finite mixtures [19, 15], among others. Unfortunately, the robustness of these randomized classifiers is not well understood, and most of them have been shown to be less robust than originally claimed under the white-box threat model. The classifier based on random input transformations by Xie et al. [23], which won the 2017 Neurips *Adversarial Attacks and Defences* Competition, was broken by Athalye et al. [2], together with stochastic pruning [9]. The defense by Panousis et al. [17] was debated on a Github issue[1] by the authors of Robustbench [6], who found that

[1] Available at https://github.com/fra31/auto-attack/issues/58

the classifier was not robust using a simple adaptation of AutoAttack [6]. The defense *Barrage of random transforms* by Raff et al. [20], which had impressive robustness results on Imagenet, was broken three years later by Sitawarin et al. [21], and Dbouk et al. [7] debate the robustness of finite mixtures.

In addition to showing that the real robustness of randomized classifiers is not yet fully understood, these results also highlight the lack of adaptive attacks for randomized models. The attacks used to evaluate the robustness of these models are often not suitable for the task, leading to an overestimation of their robustness, a phenomenon that is well known nowadays [2, 22]. The lack of strong adaptive attacks for randomized classifiers has undermined research on this family of classifiers and limited its practical applications. As an example, one of the criteria used by the Robustbench benchmark to filter out defenses is the use of randomness in the forward pass, because such defenses often "only make gradient-based attacks harder but do not substantially improve robustness" [6].

Arguably, finite mixtures of classifiers [19, 15] are one of the simplest kind of randomized classifiers, and yet they are not trivial to attack. Dbouk et al. [7] showed that the adaptations of projected gradient descent (PGD) [14] used by Pinot et al. [19] or Meunier et al. [15] were weak, thus overestimating the robustness of finite mixtures. They design ARC, the current state-of-the-art attack for finite mixtures of classifiers, and their evaluation shows a considerable drop in robustness with respect to the results reported by Pinot et al. [19].

In this work, we take a principled approach to understand adversarial attacks for finite mixtures of classifiers using a set-theoretic perspective and the concept of *vulnerability regions*. We show that the problem of attacking a finite mixture can be seen as the problem of climbing a lattice. Using this perspective, we identify a series of desirable properties and limitations of existing attacks. Afterward, we leverage the lattice reformulation to devise a new attack with better theoretical guarantees in binary classification with linear classifiers. More specifically, we make the following 3 contributions: First in Section 3 we model the problem of attacking a mixture using a semi-lattice, which allows us to better characterize the limitations of existing attacks like adaptations of PGD [14] and ARC [7]. Second, we propose in Section 4 a new attack algorithm called *lattice climber* that has strong guarantees in the binary linear setting compared to existing attacks. We then generalize to multiclass differentiable classifiers like neural networks. Finally, we provide extensive experimental results showing that our proposed attack is better at simultaneously attacking a finite mixture compared to existing attacks. Our code is available in `https://github.com/lucasgneccoh/lattice_climber_attack`, which also contains a reference to the extended version of this article that includes supplementary material.

2 Preliminaries

Notations. For a predicate C, we denote by $\mathbb{1}\{C\}$ the function that returns 1 if the predicate C is true and 0 otherwise. For an integer m, we use the notation

$[m] = \{1, \cdots, m\}$. For a vector $u \in \mathbb{R}^d$, we denote by $u^{(j)}$ the j-th component of u. We denote by Δ^n the probability simplex in $\mathbb{R}^n$. For a probability vector $p \in \Delta^n$, we denote $\mathrm{Cat}(p)$ the categorical distribution on n elements, where $p^{(i)}$ is the probability of sampling element i. To alleviate the notation, when z is a random variable following the distribution $\mathrm{Cat}(p)$, we write $z \sim p$.

Problem setting. Given a d-dimensional input space $\mathcal{X} \subset \mathbb{R}^d$ and a set $\mathcal{Y} = [K]$ of K class labels, a *deterministic* classifier $h : \mathcal{X} \to \mathcal{Y}$ is a function that maps each input point x to a predicted label $h(x)$. To measure the quality of the prediction of h at a point $(x, y) \in \mathcal{X} \times \mathcal{Y}$, we use the zero-one loss :

$$\mathcal{L}^{0\text{-}1}(h, x, y) = \mathbb{1}\{h(x) \neq y\} \tag{1}$$

In this paper, we consider finite mixtures of classifiers [19, 15], which are a type of randomized classifier inspired by mixed strategies in game theory. Given a base set of deterministic classifiers $\mathbf{h} = \{h_1, \ldots, h_m\}$ and a probability distribution over them $\mathbf{q} \in \Delta^m$, the mixture classifier $\mathbf{h_q}$ will map any input x by first sampling a classifier $i \sim \mathbf{q}$, and then returning $h_i(x)$. Note that, unlike classical deterministic classifiers, mixtures may predict different labels for the same input x over repeated calls, thus the prediction $\mathbf{h_q}(x)$ is a random variable $\hat{y}$ over $\mathcal{Y}$, and the zero-one loss needs to be adapted to measure the expected error:

$$\mathcal{L}^{0\text{-}1}(\mathbf{h_q}, x, y) = \mathop{\mathbb{E}}_{i \sim \mathbf{q}} [\mathcal{L}^{0\text{-}1}(h_i, x, y)] = \sum_{i=1}^{m} \mathbf{q}^{(i)} \cdot \mathcal{L}^{0\text{-}1}(h_i, x, y). \tag{2}$$

In other words, the zero-one loss of a mixture represents the *probability* of predicting an incorrect label for input x.

Adversarial attacks on classifiers and mixtures. Given an input point $x \in \mathcal{X}$ and its true label y, attacking a classifier h (deterministic or randomized) consists of crafting a norm bounded perturbation $\delta \in \mathbb{R}^d$ (with $\|\delta\| \leq \epsilon$) that increases the 0-1 loss at (x, y). Various norms can be used to measure the magnitude of the perturbation δ, the most common being ℓ_p norms with $p = 2$ or $p = \infty$. For a given p-norm, we denote $B_p(x, \epsilon)$ the ball centered at x with radius ϵ *i.e.* $B_p(x, \epsilon) = \{x + \delta \in \mathcal{X} \ \textit{s.t.} \ \|\delta\| \leq \epsilon\}$. The adversarial zero-one loss $\mathcal{L}^{0\text{-}1}_{\epsilon}$ is defined as the zero-one loss under attack by an *optimal adversary*:

$$\mathcal{L}^{0\text{-}1}_{\epsilon,p}(h, x, y) = \sup_{x' \in B_p(x, \epsilon)} \mathcal{L}^{0\text{-}1}(h, x', y) \tag{3}$$

3 Failures of existing attacks

We start by analyzing of the limitations of existing attacks such as Expectation Over Transformation (EOT) [3] and ARC [7].

3.1 Attacks based on Expectation Over Transformation (EOT)

Expectation Over Transformation (EOT) was initially developed by Athalye et al. [3], in order to craft attacks that are robust to real world perturbations. EOT introduces a set of transformations T that may be applied to the input, and optimizes the expected loss $\mathcal{L}$ over T:

$$\mathop{\mathbb{E}}_{t \sim T} \left[\mathcal{L}(h, t(x'), y)\right]. \tag{4}$$

The EOT principle was later adapted to attack finite mixture of classifiers by Pinot et al. [19]. Instead of considering random transformations, the idea is to account for the sampling of the classifier as the source of randomness. To attack the mixture $\mathbf{h}_{\mathbf{q}}$, our objective function becomes the *Expectation Of the Loss* (EOL), as follows:

$$\mathop{\mathbb{E}}_{i \sim \mathbf{q}} \left[\mathcal{L}(h_i, x, y)\right] = \sum_{i=1}^{m} \mathbf{q}^{(i)} \; \mathcal{L}(h_i, x, y), \tag{5}$$

Note that if we choose $\mathcal{L}$ to be the zero-one loss, then maximizing the objective in (5) directly corresponds to maximizing the classification error of the mixture in (2). Therefore, *maximizing the EOL with the zero-one loss is the correct objective for attacking a finite mixture.* However, as we will demonstrate, this stops being true if we replace the zero-one loss with a surrogate loss function such as the cross-entropy loss.

Practical adaptations for attacking finite mixtures. To generate an adversarial example in practice, one can directly maximize the EOL objective in (5) (as in (6)), or maximize the loss computed on the expected output of the mixture ((7)) [19, 22].

$$\sup_{x' \in B_p(x,\epsilon)} \quad \sum_{i=1}^{m} \mathbf{q}^{(i)} \; \mathcal{L}(h_i, x', y). \tag{6}$$

$$\sup_{x' \in B_p(x,\epsilon)} \quad \mathcal{L}\left(z \mapsto \sum_{i=1}^{m} \mathbf{q}^{(i)} \; h_i(z), x', y\right). \tag{7}$$

In practice, both of these problems are solved using first-order optimization methods like PGD. We refer to these attacks as EOL-PGD and LOE-PGD, respectively. The problem in doing so is that there is an underlying assumption that all classifiers can be attacked simultaneously and that their vulnerabilities are *aligned*, a concept we illustrate in Figure 1. This is because the gradient of either objective can be rewritten as a linear combination of the gradients of the loss of the individual classifiers with positive coefficients. Thus, using first-order methods to solve problems (6) or (7) is intuitively trying to attack all classifiers simultaneously, and the success of such attack relies on the assumption that this linear combination is a good attack direction. This can be effective in a scenario like the one depicted in Figure 1 (left), in which all classifiers are vulnerable and their vulnerabilities are aligned, but fail when the vulnerabilities lie outside the set of admissible perturbations, as demonstrated in Figure 1 (right).

This issue with EOL-PGD and LOE-PGD is the starting point for the development of ARC [7], a stronger attack against mixtures of classifiers, which we will discuss in the following section.

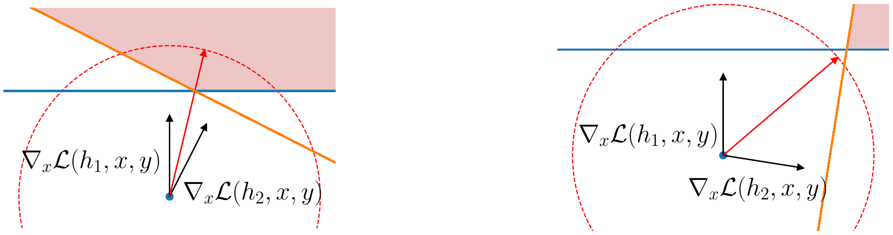
$\nabla_x\mathcal{L}(h_1,x,y)$
$\nabla_x\mathcal{L}(h_2,x,y)$
$\nabla_x\mathcal{L}(h_1,x,y)$
$\nabla_x\mathcal{L}(h_2,x,y)$

Fig. 1: Example of two linear classifiers with high (left) and low (right) alignment. The red arrow represents the gradient of the loss of the mixture, while the black arrows represent the gradient of the loss of individual classifiers. The red region in the top-right corner represents the points that are adversarial for both classifiers simultaneously. In the case of high alignment (left), the linear combination of the individual gradients is a successful attack direction, while in the case of low alignment (right), it is *not* an effective attack direction as it leads to a non-adversarial perturbation, even though adversarial perturbations do exist.

3.2 ARC attack and its limitations

Dbouk et al. [7] criticize EOL-PGD and LOE-PGD because of their lack of *consistency*[2] [7, Theorem 5.1], meaning that even if there exist adversarial attacks that increase the error of the mixture, EOL-PGD or LOE-PGD can miss them (*i.e.* Figure 1 (right)). This motivates the authors to create *Attacking Randomized ensembles of Classifiers* (ARC)[7], an attack against finite mixtures of classifiers that is guaranteed to be consistent in binary classification against linear classifiers. In the rest of this section, we will compare ARC to EOL-PGD and discuss the limitations of ARC. We omit the discussion of LOE-PGD, but our analysis applies to it as well.

The key difference between ARC and EOL-PGD is that ARC attacks the classifiers one by one, instead of trying to attack all of them simultaneously. At each iteration, ARC follows the direction that attacks the one classifier at hand, and at the end of the iteration, the new perturbation will be kept only if the error of the whole mixture was strictly increased (see [7, Algorithm 1] for full details). The greedy approach of ARC makes it provably consistent when attacking binary

[2] This is the term used by authors in the original work. We rather use the term *effectiveness*. See Section 3.3

linear classifiers (see [7, Theorem 5.1]). In Figure 2 (left) we revisit a situation akin to Figure 1 (right), and we can see that ARC is able to attack one of the classifiers successfully, while EOL-PGD fails to find an adversarial perturbation.

Dbouk et al. [7] train mixtures of two classifiers using the method proposed in [19] and find that when attacked with ARC[3], the robustness of these mixtures drops significantly compared to when they are attacked with EOL-PGD or LOE-PGD, which were the attacks used by Pinot et al. [19] for their evaluation. This behavior remains consistent across neural network architectures, datasets, and norms considered. Thus, ARC has proven to be a much stronger attack than EOL-PGD and LOE-PGD and remains, to this day, the state-of-the-art attack against finite mixtures of classifiers.

Although ARC resolves the main issue with EOL-PGD, it does so at the cost of losing the ability to attack multiple classifiers simultaneously in situations of medium alignment. Experimentally, one can verify that even for two linear classifiers in $\mathbb{R}^2$, ARC may fail to find a perturbation that is misclassified by both classifiers simultaneously when the region of common vulnerabilities within the ϵ-ball is very small. An example of such scenario is shown in Figure 2 (right).

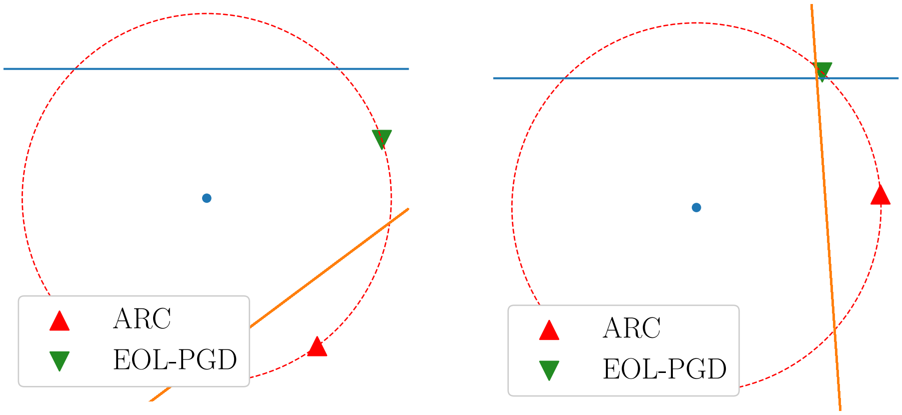

Fig. 2: Comparison between the perturbation proposed by EOL-PGD and ARC on a mixture of two linear classifiers with low to medium alignment.

To further demonstrate this issue, we test both EOL-PGD and ARC in situations that range from high to low alignment by changing the angle between the normal vectors of the linear classifiers from 0 degrees (perfect alignment) to 180 degrees (opposite normal vectors, low alignment). In Figure 3, we plot the error induced by the perturbations found by both EOL-PGD and ARC and compare it to the optimal error, which is determined by checking if the intersection of the

[3] ARC is born from an analysis in the binary linear case, but it is generalized to the case of multiclass differentiable classifiers like neural networks. In their experiments comparing to the results in Pinot et al. [19], they used the latter.

two linear classifiers lies within the ϵ-ball or not. It can be seen that no attack dominates the other and that each outperforms the other in some regime that depends on the alignment of the decision boundaries.

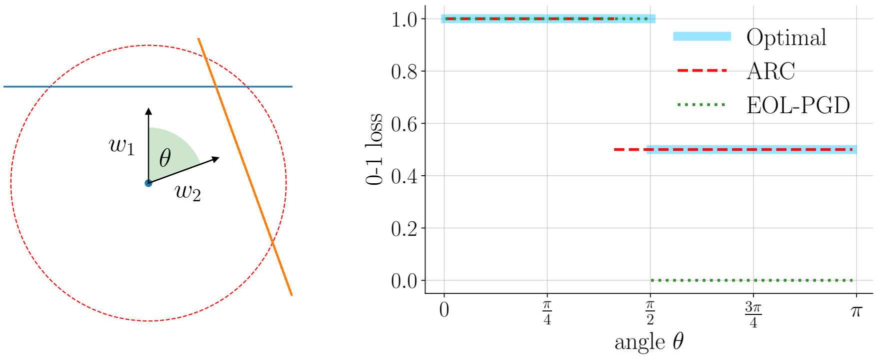

Fig. 3: Comparison between EOL-PGD and ARC w.r.t the level of alignment of the linear classifiers. On the left figure, a schema of the experiment that we run for a fixed angle θ. On the right figure, the error of the perturbations found by EOL-PGD and ARC compared to the optimal error w.r.t. θ.

We end this section with a comparison of ARC, EOL-PGD and LOE-PGD through the lens of the directions used to craft the perturbation at each iteration of the attack. Recall that the gradient directions used by EOL-PGD and LOE-PGD were linear combinations of the gradients of the loss of individual classifiers with non-zero coefficients for *all* of them. In the case of ARC, these coefficients are all zero except for one, which corresponds to the classifier being considered at the current step (See [7, Algorithm 2] for full details).

3.3 Desirable properties of an attack for mixtures

In this section we will formalize the problem of attacking a mixture of classifiers as a combinatorial optimization problem, specifically as the problem of enumerating maximal elements of a lattice. Using this formalism, we will discuss the desirable properties for attack algorithm. This will allow us to identify a new property unsatisfied by existing algorithms: *maximality*.

Formalizing the problem of attacking a finite mixture. Note that a deterministic classifier h partitions the input space into at most K non-overlapping regions $h^{-1}(j) \subseteq \mathcal{X}$, $j \in [K]$, where each region corresponds to a class. Moreover, an optimal untargeted attacker only cares about forcing a prediction *different* from the true label y inside the set of admissible perturbations $B_p(x, \epsilon)$. In that sense, each classifier h induces a partition of the set of admissible perturbation

$B_p(x,\epsilon)$ into two disjoint regions of interest:

$$B_p(x,\epsilon) = \underbrace{B_p(x,\epsilon)\cap h^{-1}(y)}_{\text{non-adversarial}} \quad \sqcup \quad \underbrace{B_p(x,\epsilon)\cap \bigcup_{j\neq y} h^{-1}(j)}_{\text{Vulnerability region } \mathrm{V}(h)}. \tag{8}$$

We call *vulnerability region* of h the set $B_p(x,\epsilon)\cap\bigcup_{j\neq y}h^{-1}(j)$, and denote it as $\mathrm{V}(h)$. With the definition of vulnerability region, attacking h is equivalent to finding a point $x'\in\mathrm{V}(h)$, so the error under attack of $\mathbf{h_q}$ at (x,y) becomes:

$$\mathcal{L}^{0\text{-}1}_{\epsilon,p}(\mathbf{h_q},x,y) = \sup_{x'\in B_p(x,\epsilon)} \sum_{i=1}^{m} \mathbf{q}^{(i)}\ \mathbb{1}\{x'\in\mathrm{V}(h_i)\}. \tag{9}$$

With Equation (9) it is clear that the optimal attack belongs to the intersection of the vulnerability regions of the classifiers that maximizes the total mass according to $\mathbf{q}$. Let us define the *common vulnerability region* of a subset of classifiers $\mathbf{h}'\subseteq\mathbf{h}$ as $\mathrm{CV}(\mathbf{h}')=\bigcap_{h\in\mathbf{h}'}\mathrm{V}(h)\setminus\bigcup_{h\notin\mathbf{h}'}\mathrm{V}(h)$. In simple terms, $\mathrm{CV}(\mathbf{h}')$ is the set of points that are adversarial for exactly the classifiers in $\mathbf{h}'$, and thus, if $x'\in\mathrm{CV}(\mathbf{h}')$, then $\mathcal{L}^{0\text{-}1}(\mathbf{h_q},x',y)=\sum_{i \text{ s.t. } h_i\in\mathbf{h}'}\mathbf{q}^{(i)}$, and if $\mathbf{h}_1\subseteq\mathbf{h}_2$ and $\mathrm{CV}(\mathbf{h}_2)\neq\emptyset$, then the following holds:

$$\forall x_1\in\mathrm{CV}(\mathbf{h}_2),\ \forall x_2\in\mathrm{CV}(\mathbf{h}_1),\ \mathcal{L}^{0\text{-}1}(\mathbf{h_q},x_2,y)\leq\mathcal{L}^{0\text{-}1}(\mathbf{h_q},x_1,y), \tag{10}$$

which means that an attacker prefers to attack in the common vulnerability region of a larger set of classifiers whenever it is not empty.

Equation (10) suggests that the subsets of classifiers can be ordered by their preference for the attacker. Formally speaking, we can define the partial order

$$\mathbf{h}_1\preceq\mathbf{h}_2 \iff \mathbf{h}_1\subseteq\mathbf{h}_2 \text{ and } \mathrm{CV}(\mathbf{h}_2)\neq\emptyset. \tag{11}$$

The order relation in Equation (11) induces a *lower semilattice structure* in the family of subsets $\mathcal{S}=\{\mathbf{h}'\subseteq\mathbf{h}\mid\mathrm{CV}(\mathbf{h}')\neq\emptyset\}\bigcup\{\emptyset\}$. For simplicity, we refer to it as *adversarial lattice* of $\mathbf{h_q}$ at (x,y).

The lattice object allows us to discuss desirable properties of an attack algorithm that faces mixtures of classifiers. The first has already been discussed under the name of consistency [7], but we redefine it as effectiveness:

Definition 1 (Effectiveness property). *An attack algorithm is effective if for any finite mixture $\mathbf{h_q}$ and point (x,y), it can generate an adversarial example increasing the error of the mixture whenever such a point exists.*

Definition 2 (Maximality property). *An attack algorithm is maximal if for any finite mixture $\mathbf{h_q}$ and any point (x,y), it can generate an adversarial example for a maximal subset of classifiers of the adversarial lattice.*

Neither EOL-PGD nor ARC is maximal for binary linear classifiers, as counterexamples can be constructed where they fail to find a maximal subset of

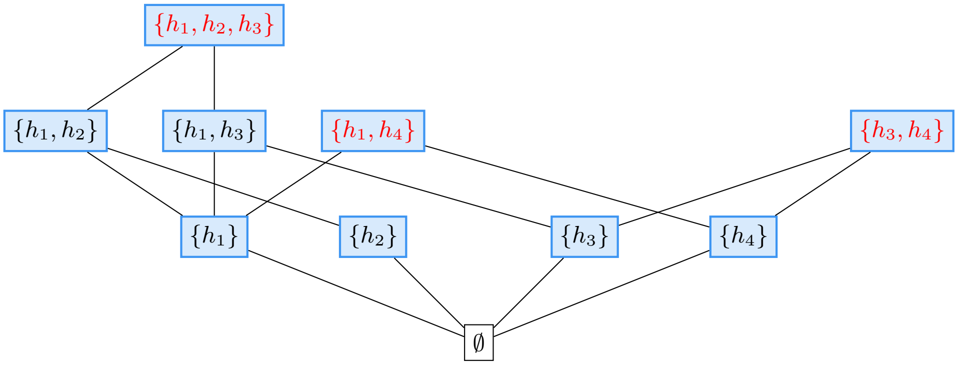

Fig. 4: Example of an adversarial lattice for a mixture of four classifiers.

classifiers to attack (Figure 2). Figure 4 illustrates an example of an adversarial lattice for a mixture of four classifiers. An *effective* attack is ensured to produce an adversarial attack x' that corresponds to *some* of the nodes highlighted with a blue rectangle. This property is a defining characteristic of ARC for binary linear classifiers —one that EOL-PGD lacks. However, there is no guarantee regarding the level of the lattice that is reached, and the effectiveness guarantee becomes less meaningful as the size of the lattice increases. The nodes with red font represent the maximal nodes, illustrating that the maximality guarantee is a stronger property than effectiveness.

The most desirable property of an attack for mixtures is be *optimality*:

Definition 3 (Optimality property). *An attack algorithm is optimal if for any finite mixture over $\mathbf{h_q}$ and any point (x, y), it can generate an adversarial example that achieves the highest possible 0-1 loss.*

The optimal attack corresponds to a *subset* of the maximal elements of the adversarial lattice, depending on the weights of the mixture. Unfortunately, no polynomial-time algorithm is guaranteed to achieve optimality, even when using linear classifiers in binary classification, due to the following result:

Theorem 1 (Hardness of attacking linear classifiers). *Consider a binary classification setting, ℓ_p norm with $p > 1$ and ϵ to be defined. Given a labeled point (x, y), a set of m linear classifiers $x \mapsto \mathbb{1}\left\{\theta_i^\top x + b_i \geq 0\right\}$ where $(\theta_i, b_i) \in \mathbb{R}^{d+1}$, a uniform mixture $\mathbf{h_q}$ composed of these linear classifiers and a value $\beta > 0$, there exists a noise budget $\epsilon > 0$ such that the problem of checking if there exists $x' \in B_p(x, \epsilon)$ such that $\mathcal{L}^{0\text{-}1}(\mathbf{h_q}, x', y) \geq \beta$ is NP-hard.*

As a consequence of Theorem 1, the attack algorithm that we will propose will prioritize *maximality* over optimality, which can be guaranteed in the case of binary linear classifiers and provides effectiveness for free.

4 Lattice Climber attack

In this section, we introduce a new attack called Lattice Climber Attack (LCA). We first present a version of our attack that has maximality guarantees against mixtures of binary linear classifiers under mild assumptions, and then extend it to multiclass differentiable classifiers. Afterwards, we will present experimental results that compare LCA with EOL-PGD and ARC on synthetic and real-world datasets.

The idea behind LCA is to arrive at a maximal element of the lattice by climbing one level at a time. Similar problems have been studied under the name of *most specific sentences* or *maximal frequent itemsets* in the domain of data mining and knowledge discovery, where algorithms like `AllMSS` [11] have been proposed. Our approach is similar to `A_Random_MSS` in [11].

4.1 Binary linear classifiers

In this section and for simplicity, we consider class labels $y \in \{-1, 1\}$ and classifiers $h : \mathbb{R}^d \to \{-1, 1\}$ of the form $h(x) = \operatorname{sign}(f(x))$ for some linear function $f : \mathbb{R}^d \to \mathbb{R}$. In this setting, h correctly classifies the data point (x, y) if $yf(x) > 0$. Therefore, attacking h translates to minimizing $yf(x)$. The optimal attack direction and margin to the decision boundary of a single linear classifier are known for all ℓ_p norm with $p \geq 1$ [7, Appendix A].

The first component of our attack algorithm is a procedure to attack a subset of classifiers $\mathbf{h}'$ and find $x + \delta \in \mathrm{CV}(\mathbf{h}')$ whenever $\mathrm{CV}(\mathbf{h}') \neq \emptyset$. Similar to [18], we consider the *reverse hinge loss* $\mathcal{L}^{rev}(yf(x)) = \max(yf(x), 0)$ as an objective to *minimize* in order to attack *one* binary linear classifier. Attacking a binary linear classifier with the traditional hinge loss would imply maximizing a convex function, for which there is no guarantee of convergence to a global optimum using algorithms like PGD. On the other hand, using the reverse hinge loss is equivalent to *minimizing* a bounded convex function, for which PGD is guaranteed to converge to a global optimum.

Another useful property of the reverse hinge loss is that the expected reverse hinge loss over a finite mixture of classifiers is convex and equal to zero if and only if all the classifiers in the mixture misclassify x. Then, in order to attack a set of classifiers $\mathbf{h}'$, we can minimize the *sum of reverse hinge losses* [18]

$$\mathrm{SRH}(\mathbf{h}', x, y) = \frac{1}{|\mathbf{h}'|} \sum_{h \in \mathbf{h}'} \mathcal{L}^{rev}(yf(x)). \tag{12}$$

If there exists x' such that $\mathrm{SRH}(\mathbf{h}', x', y) = 0$, then PGD with an appropriately chosen step size and sufficient number of iterations will converge to some x'' that is misclassified by all classifiers in $\mathbf{h}'$ (see [18, Theorem 3] for details).

The second component of our attack algorithm is a way to navigate the adversarial lattice. We propose a bottom-up navigation mechanism to climb a branch of the adversarial lattice that consists of keeping a pool of fooled classifiers, attempting to add each classifier in a fixed order. A classifier joins the pool

Algorithm 1: LCA for binary linear classifiers

```
Require: Set $\mathbf{h}$ of $m$ binary linear classifiers in some order $(h_1, \dots, h_m)$, starting
    point $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$. $T$ number of iterations and $\eta$ step size for PGD.
1: Initialize pool $\mathbf{h}_{\text{pool}} = \emptyset$, $\delta = 0_d$
2: for $i = 1, 2 \cdots, m$ do
3:     $\mathbf{h}_{\text{pool}} = \mathbf{h}_{\text{pool}} \cup \{h_i\}$ {Add $h_i$ to the pool}
4:     Attack SRH($\mathbf{h}_{\text{pool}}, \cdot, y$) starting at $x + \delta$ with PGD $(T, \eta)$ to find new
    perturbation $\hat{\delta}$
5:     if SRH($\mathbf{h}_{\text{pool}}, x + \hat{\delta}, y$) = 0 i.e. succeeded then
6:         $\delta = \hat{\delta}$ {Update current attack, keep $h_i$ in pool}
7:     else
8:         $\mathbf{h}_{\text{pool}} = \mathbf{h}_{\text{pool}} \setminus \{h_i\}$ {Reset pool to last state}
9: return Adversarial example $x + \delta$
```

only if it can be fooled alongside all current members; otherwise, it is discarded. The algorithm terminates after evaluating all classifiers.

The order in which classifiers are considered is a parameter of the algorithm, and similarly to [7], we find that considering them in decreasing order of their associated weight yields good performance in general. For example, in the case $m = 2$, and with suitable parameters for the internal PGD, it ensures that LCA is optimal. Another reasonable option is to use a random permutation of the classifiers, which Dbouk et al. [8] found to be useful for ARC. The pseudocode of LCA in the binary linear setting is shown in Algorithm 1.

Recall that the step used by EOL-PGD can be expressed as linear combinations of the gradients of the loss of individual classifiers. For LCA, similar to ARC, the coefficients of this linear combination will be sparse: only the models within the pool $\mathbf{h}_{\text{pool}}$ will have non-zero coefficients. This behavior positions LCA as an intermediate approach between EOL-PGD and ARC: in LCA, the coefficients initially resemble those in ARC, but as the pool grows, they transition to resemble the denser pattern of EOL-PGD. This behavior makes it possible to adapt to different types of adversarial lattice: when classifiers are not simultaneously vulnerable, LCA will behave like ARC, which was created to ensure *effectiveness* for binary linear classifiers, and when they are simultaneously vulnerable, it will behave like EOL-PGD which attacks all classifiers simultaneously.

Maximality of LCA in the binary linear setting. In the binary linear classifier setting, PGD(T, η) is guaranteed to minimize the function SRH$(\mathbf{h}_{\text{pool}}, \cdot, y)$ because it is a convex optimization problem [18]. Thus, line 4 of Algorithm 1 ensures that we will find an adversarial attack $x + \hat{\delta} \in \text{CV}(\mathbf{h}_{\text{pool}})$ if $\text{CV}(\mathbf{h}_{\text{pool}}) \neq \emptyset$. This enables climbing to the top of the adversarial lattice along a branch determined by the classifier order. This is formalized in the following lemma:

Lemma 1. *Consider a set of m binary linear classifiers $\mathbf{h}$. Fix $\epsilon > 0$ the attack budget and a point (x, y). If there exists a point $x' \in B_p(x, \epsilon)$ such that*

$\mathrm{SRH}(\mathbf{h}, x', y) = 0$*, then there exist parameters* T *and* η *such that minimizing* $\mathrm{SRH}(\mathbf{h}, x, y)$ *w.r.t.* x *with* $\mathrm{PGD}(T, \eta)$ *will return* x'' *such that* $\mathrm{SRH}(\mathbf{h}, x'', y) = 0$.

Lemma 1 allows us to prove that LCA is maximal for the set of binary linear classifiers:

Theorem 2 (LCA is maximal in the binary linear setting). *Let* $\mathbf{h}$ *be a finite set of binary linear classifiers and* $\mathbf{h_q}$ *a mixture over* $\mathbf{h}$*. Fix* $\epsilon > 0$ *the attack budget and a p-norm with* $p > 1$*. For any* $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$*, there exist parameters* T *and* η *for the inner* PGD *such that Algorithm 1 returns an adversarial example* $x + \delta \in \mathrm{CV}(\mathbf{h}')$*, where* $\mathbf{h}'$ *is a maximal element of the adversarial lattice of* $\mathbf{h_q}$ *at* (x, y)*.*

4.2 Multiclass differentiable classifiers

The ideas developed in Section 4.1 for binary linear classifiers need to be adapted to the multiclass case with general differentiable classifiers, like neural networks, because we cannot have the guarantees provided by Lemma 1 and Theorem 2. Moreover, the reverse hinge loss was defined for binary classifiers and not for multiclass classifiers.

In order to attack a classifier h that predicts the class with the highest score according to the score function $f : \mathcal{X} \to \mathbb{R}^K$ at an arbitrary point (x, y), we choose the target label $y_{\mathrm{adv}} \in [K] \setminus \{y\}$ with the largest score according to $f(x)$ and minimize the reverse hinge loss of the margin between $f(x)^{(y)}$ and $f(x)^{(y_{\mathrm{adv}})}$, *i.e.* $\mathcal{L}^{rev}(f(x)^{(y)} - f(x)^{(y_{\mathrm{adv}})})$. This is a common choice that has been used by Perdomo et al. [18] and also by Carlini et al. [5], who found it to perform well as an objective function for crafting adversarial attacks. As there is no guarantee on the convergence of the attack to $\mathrm{SRH}(\mathbf{h}_{\mathrm{pool}}, \cdot, y)$, we change the criteria to update the pool of classifiers: each time we find a perturbation δ with a higher error for the mixture, we keep it and update the pool to be the classifiers that misclassify the current adversarial example $x + \delta$.

5 Experiments

In this section, we show the experimental results that support the theoretical guarantees of LCA in the binary linear setting. We also compare the performance of LCA with ARC and EOL-PGD in the multiclass differentiable setting, in which guarantees are not provided, and show that LCA generally performs better than existing state-of-the-art attacks.

5.1 Synthetic data: Linear classifiers

Binary linear classifiers in high dimension. In this experiment, we assess how the performance of EOL-PGD, ARC and LCA scale with the number of classifiers m in a higher dimension d. For each value m and a fixed value of $\epsilon = 1$, we repeat the following experiment 1000 times: we fix our point $(x, y) = (0_d, -1)$ and sample

m i.i.d linear classifiers (w_i, b_i) where w_i is uniformly sampled from the unit sphere in $\mathbb{R}^d$ and $b_i \sim -|\mathcal{N}(\alpha, \beta)|$ follows a folded Gaussian distribution, and we test the three attacks against the sampled mixture. The hyperparameter α controls the expected distance from x to the decision boundary of the classifiers, and β the variance of such distance.

Figure 5 shows that the average performance of all attacks deteriorates as the number of models increases, but LCA remains superior for all m. Note that depending on the difficulty of the configurations (α and β), the performance of the attacks can change drastically. Nevertheless, LCA remained superior in all configurations, regardless of the dimension.

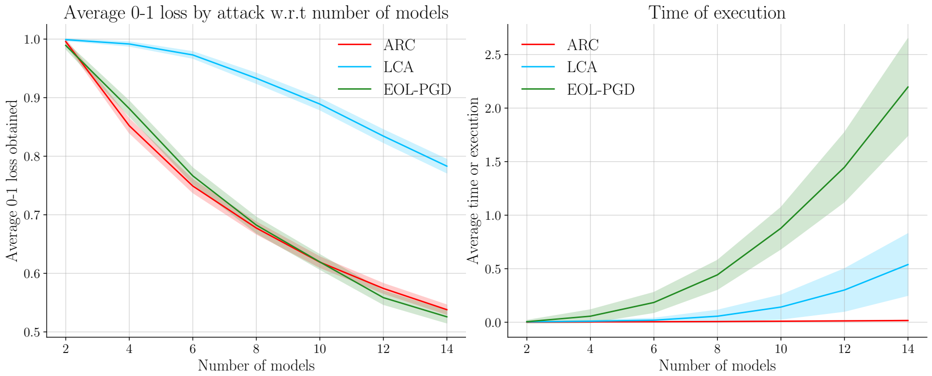

Fig. 5: On the left figure, the average error obtained by the attacks in $\mathbb{R}^{128}$ as a function of the number of classifiers in the mixture with 99% confidence intervals. The right figure shows the average time in seconds that each attack took to generate an adversarial example as a function of the number of classifiers with one standard deviation intervals. For each number of models, the experiment was repeated 1000 times. Note that the y-axis limits have been adapted for visualization purposes. In this case, $\alpha = 0.25$, $\beta = 0.2$.

5.2 CIFAR-10

Experimental setup. To test our attack against differentiable multiclass classifiers, we measure the accuracy of mixtures built using models trained with different ensemble diversity techniques. To have a wide variety of models and training methods, we take the pre-trained models from the DVERGE [24] repository[4], which compares baseline models trained without any defense (BASE) against ensemble diversity defenses such as ADP [16], GAL [13], plain adversarial training (AT) [14] and DVERGE (DV) [24]. The authors made available 3 independent runs of each method (except for BASE), and we report the average

[4] `https://github.com/zjysteven/DVERGE/tree/main`

Table 1: Expected accuracy of mixtures against EOL-PGD, ARC and LCA under the ℓ_∞ threat model. Attacker wants to minimize accuracy, so lower is better. All models use ResNet20 as the base architecture, except † which uses ResNet18.

		$\epsilon = 0.01$			$\epsilon = 0.03$		
Model	Nat	EOL	ARC	LCA	EOL	ARC	LCA
BASE/3	91.8%	15.0%	4.3%	**1.7%**	0.1%	0.2%	**0.0%**
BASE/5	91.9%	12.3%	6.1%	**2.7%**	**0.0%**	0.2%	0.1%
BASE/8	91.8%	10.6%	7.9%	**4.6%**	**0.0%**	0.2%	0.1%
ADP/3	89.4%	15.7%	**14.6%**	15.1%	**0.4%**	5.2%	1.8%
ADP/5 [16]	88.8%	15.4%	16.1%	**13.9%**	**0.2%**	2.5%	1.5%
ADP/8	88.5%	**15.2%**	18.4%	15.4%	**0.3%**	4.2%	3.3%
GAL/3	87.1%	43.0%	18.0%	**14.3%**	14.5%	5.8%	**0.9%**
GAL/5 [13]	88.7%	46.2%	45.1%	**36.0%**	11.5%	9.4%	**7.1%**
GAL/8	89.7%	**44.5%**	52.4%	50.2%	**3.9%**	11.7%	15.9%
AT/3	77.3%	68.6%	68.3%	**65.3%**	47.2%	47.7%	**42.4%**
AT/5 [14]	77.9%	69.0%	69.1%	**65.2%**	47.2%	49.2%	**42.7%**
AT/8	77.8%	69.0%	69.0%	**64.9%**	47.5%	50.3%	**44.3%**
DV/3	89.8%	52.6%	44.4%	**31.2%**	39.2%	5.8%	**1.9%**
DV/5 [24]	89.9%	59.0%	55.1%	**42.1%**	35.3%	10.1%	**6.0%**
DV/8	88.6%	62.1%	63.2%	**52.8%**	31.7%	18.4%	**15.0%**
DV+AT/3	81.7%	71.4%	71.3%	**69.0%**	44.4%	45.2%	**41.0%**
DV+AT/5 [24]	84.0%	72.8%	73.1%	**70.2%**	42.2%	44.6%	**39.9%**
DV+AT/8	84.0%	73.2%	73.4%	**70.9%**	44.3%	46.5%	**42.5%**
BARRE/5 [8]	76.3%	68.9%	69.4%	**66.2%**	50.6%	49.9%	**46.1%**
MR/5 [25]	73.7%	65.7%	66.4%	**62.3%**	46.3%	46.7%	**41.5%**
MR/5†	81.3%	72.7%	74.1%	**70.2%**	47.1%	51.4%	**46.5%**

accuracy over these 3 independent runs. We also trained a mixture of 5 models using the MRBoost [25] framework, using the ResNet20 and the ResNet18 architectures. All these methods are designed to promote ensemble diversity and reduce the joint adversarial vulnerability of the sub-models, so we consider them appropriate to evaluate the performance of attacks against mixtures. We also take pre-trained models from the BARRE [8] repository[5], in which authors propose a boosting algorithm to specifically train robust mixtures. All these models use the ResNet20 architecture as the base classifiers, except for MRBoost with ResNet18. We use the ℓ_∞ threat model with $\epsilon = 0.03$ as in standard practice, and also the $\epsilon = 0.01$ setting to compare with [24].

To provide a fair comparison, we adjusted the number of iterations T for each attack to ensure that LCA was not given any advantage. Note that the number of gradient computations is $T \cdot m$ for EOL-PGD, at most $4\, T \cdot m$ for ARC [7], and at most $T \cdot \frac{m(m+1)}{2}$ for LCA. In our case $m \in \{3, 5, 8\}$. Taking this into account, we set the number of iterations to 50 for LCA, and gave ARC and EOL-PGD up

[5] `https://github.com/hsndbk4/BARRE/tree/main`

to 200 and 500 iterations respectively. We also tested ARC and EOL-PGD with fewer iterations, and report only the best result. We further give more advantage to EOL-PGD by allowing it to perform random initialization and 5 restarts. In contrast, ARC and LCA do not use random initialization or restarts.

Results. Table 1 shows the results of our evaluation. First, we can confirm the observations made in [7]: EOL-PGD, even with the advantage given, tends to overestimate the robustness of mixtures, and it is more notorious for BASE models with $\epsilon = 0.01$ and DV models. In these cases, ARC dramatically outperforms EOL-PGD (as was seen in [7]), and *LCA outperforms both ARC and EOL-PGD*. Note however that EOL-PGD is better than both ARC and LCA when models are simultaneously vulnerable and the individual gradients used for the attack are highly aligned [7, 13], which seems to be the case for ADP models. This could be explained by the fact that ADP, contrary to methods like GAL, does not explicitly reduce gradient alignment during training.

GAL models show a peculiar behavior: robustness against EOL-PGD tends to *decrease* as the number of models increases, suggesting that the models become more aligned. A similar behavior was also reported in [24, Table 4], suggesting that this method does not scale well with the number of models. However, robustness against ARC or LCA increases with the number of models. This contrast might suggest that GAL models are indeed more diverse locally, but still simultaneously vulnerable inside the ϵ-ball so that EOL-PGD is able to exploit this vulnerability by naively attacking all classifiers enough times.

For the more robust models (AT, DV+AT, BARRE and MR), LCA shows a clear improvement in performance when compared to both ARC and EOL-PGD. The average gap between LCA and ARC in the $\epsilon = 0.03$ setting is 4.9%, with a minimum gap of 3.8%.

6 Conclusions and future work

In this paper, we explore the properties of attacks for randomized mixtures and introduce the LCA algorithm that is *maximal* in the binary linear setting and demonstrates good empirical performances against multiclass differentiable classifiers. Despite employing state-of-the-art diversity inducing techniques, our robustness evaluation confirms that the robustness of mixtures largely depends on the robustness of individual models, often matching that of a single model (the robustness of a 5-ResNet20 mixture in [8] is similar to the robustness of a single ResNet20 reported in [7] by the same authors), limiting the practical impact of mixtures. This discrepancy between the theoretical advantages of mixtures and their practical robustness underscores the persistent challenge of training robust mixtures. Moreover, adversarial training has limited effectiveness for mixtures, as training with stronger adaptive attacks does not produce more robust models. New theoretical and practical approaches are needed to improve the training of robust mixtures and fully leverage their potential, a direction that has been started in [8].

Acknowledgments. This work was carried out using HPC resources from GENCI–IDRIS (AD011014207 and AD011014207R1), and funded by the French National Research Agency (DELCO ANR-19-CE23-0016). Thanks to the reviewers for the valuable feedback.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Amaldi, E., Kann, V.: The complexity and approximability of finding maximum feasible subsystems of linear relations. Theoretical computer science **147**(1-2), 181–210 (1995)
2. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: ICML. pp. 274–283. PMLR (2018)
3. Athalye, A., Engstrom, L., Ilyas, A., Kwok, K.: Synthesizing robust adversarial examples. In: ICML. pp. 284–293. PMLR (2018)
4. Bubeck, S., Lee, Y.T., Price, E., Razenshteyn, I.: Adversarial examples from computational constraints. In: ICML. pp. 831–840. PMLR (2019)
5. Carlini, N., Wagner, D.A.: Towards evaluating the robustness of neural networks. In: Symposium on Security and Privacy. pp. 39–57. IEEE Computer Society (2017)
6. Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., Hein, M.: Robustbench: a standardized adversarial robustness benchmark. arXiv:2010.09670 (2020)
7. Dbouk, H., Shanbhag, N.: Adversarial vulnerability of randomized ensembles. In: ICML. pp. 4890–4917. PMLR (2022)
8. Dbouk, H., Shanbhag, N.R.: On the robustness of randomized ensembles to adversarial perturbations. arXiv:2302.01375 (2023)
9. Dhillon, G.S., Azizzadenesheli, K., Lipton, Z.C., Bernstein, J., Kossaifi, J., Khanna, A., Anandkumar, A.: Stochastic activation pruning for robust adversarial defense. arXiv:1803.01442 (2018)
10. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv:1412.6572 (2014)
11. Gunopulos, D., Mannila, H., Saluja, S.: Discovering all most specific sentences by randomized algorithms extended abstract. In: Int. Conf. on Database Theory. pp. 215–229. Springer (1997)
12. He, Z., Rakin, A.S., Fan, D.: Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In: CCVPR. pp. 588–597 (2019)
13. Kariyappa, S., Qureshi, M.K.: Improving adversarial robustness of ensembles with diversity training. CoRR **abs/1901.09981** (2019)
14. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv:1706.06083 (2017)
15. Meunier, L., Scetbon, M., Pinot, R.B., Atif, J., Chevaleyre, Y.: Mixed nash equilibria in the adversarial examples game. In: ICML. pp. 7677–7687. PMLR (2021)
16. Pang, T., Xu, K., Du, C., Chen, N., Zhu, J.: Improving adversarial robustness via promoting ensemble diversity. In: ICML. pp. 4970–4979. PMLR (2019)

17. Panousis, K.P., Chatzis, S., Theodoridis, S.: Stochastic local winner-takes-all networks enable profound adversarial robustness. arXiv:2112.02671 (2021)
18. Perdomo, J.C., Singer, Y.: Robust attacks against multiple classifiers. arXiv:1906.02816 (2019)
19. Pinot, R., Ettedgui, R., Rizk, G., Chevaleyre, Y., Atif, J.: Randomization matters how to defend against strong adversarial attacks. In: ICML. pp. 7717–7727. PMLR (2020)
20. Raff, E., Sylvester, J., Forsyth, S., McLean, M.: Barrage of random transforms for adversarially robust defense. In: CCVPR. pp. 6528–6537 (2019)
21. Sitawarin, C., Golan-Strieb, Z.J., Wagner, D.: Demystifying the adversarial robustness of random transformation defenses. In: ICML. pp. 20232–20252. PMLR (2022)
22. Tramer, F., Carlini, N., Brendel, W., Madry, A.: On adaptive attacks to adversarial example defenses. Neurips **33**, 1633–1645 (2020)
23. Xie, C., Wang, J., Zhang, Z., Ren, Z., Yuille, A.L.: Mitigating adversarial effects through randomization. In: ICLR (2018)
24. Yang, H., Zhang, J., Dong, H., Inkawhich, N., Gardner, A., Touchet, A., Wilkes, W., Berry, H., Li, H.: Dverge: diversifying vulnerabilities for enhanced robust generation of ensembles. Neurips **33**, 5505–5515 (2020)
25. Zhang, D., Zhang, H., Courville, A., Bengio, Y., Ravikumar, P., Suggala, A.S.: Building robust ensembles via margin boosting. In: ICML. vol. 162, pp. 26669–26692. PMLR (06 2022)