Reliable Robustness Evaluation via Automatically Constructed Attack Ensembles

Shengcai Liu[1,2], **Fu Peng**[2], **Ke Tang**[1,2*]

[1]Research Institute of Trustworthy Autonomous Systems,
Southern University of Science and Technology, Shenzhen 518055, China
[2]Department of Computer Science and Engineering,
Southern University of Science and Technology, Shenzhen 518055, China
liusc3@sustech.edu.cn, pengf2022@mail.sustech.edu.cn, tangk3@sustech.edu.cn

Abstract

Attack Ensemble (AE), which combines multiple attacks together, provides a reliable way to evaluate adversarial robustness. In practice, AEs are often constructed and tuned by human experts, which however tends to be sub-optimal and time-consuming. In this work, we present AutoAE, a conceptually simple approach for automatically constructing AEs. In brief, AutoAE repeatedly adds the attack and its iteration steps to the ensemble that maximizes ensemble improvement per additional iteration consumed. We show theoretically that AutoAE yields AEs provably within a constant factor of the optimal for a given defense. We then use AutoAE to construct two AEs for l_∞ and l_2 attacks, and apply them without any tuning or adaptation to 45 top adversarial defenses on the RobustBench leaderboard. In all except one cases we achieve equal or better (often the latter) robustness evaluation than existing AEs, and notably, in 29 cases we achieve better robustness evaluation than the best known one. Such performance of AutoAE shows itself as a reliable evaluation protocol for adversarial robustness, which further indicates the huge potential of automatic AE construction. Code is available at https://github.com/LeegerPENG/AutoAE.

Introduction

Deep neural networks (DNNs) have exhibited vulnerability to adversarial examples, which are crafted by maliciously perturbing the original input (Szegedy et al. 2014). Such perturbations are nearly imperceptible to humans but can fool DNNs into producing unexpected behavior, thus raising major concerns about their utility in security-critical applications (Liu et al. 2022; Dai et al. 2022). Over the past few years, defense strategies against adversarial attacks have attracted rapidly increasing research interest (Machado, Silva, and Goldschmidt 2021). Among them the notable ones include adversarial training (Madry et al. 2018) that uses different losses (Carlini and Wagner 2017; Zhang and Wang 2019) and generates additional training data (Carmon et al. 2019; Alayrac et al. 2019), as well as provable robustness (Ruan et al. 2019; Croce, Andriushchenko, and Hein 2019; Gowal et al. 2018; Zhang et al. 2019).

On the other hand, there also exists much evidence showing that many proposed defenses seem to be effective initially but broken later, i.e., exhibiting significant drops of robust accuracy (often by more than 10%) when evaluated by more powerful or adapted attacks (Carlini and Wagner 2017; Athalye, Carlini, and Wagner 2018). With the fast development of this field, it is therefore of utmost importance to reliably evaluate new defenses, such that the really effective ideas can be identified. Recently, Croce and Hein (2020b) proposed to assess adversarial robustness using an attack ensemble (AE), dubbed AutoAttack, and has achieved better robustness evaluation for many defenses than the original evaluation. More specifically, when applied to a defense, an AE would run its component attacks individually, and choose the best of their outputs. It is thus conceivable that the performance of an AE heavily relies on the complementarity and diversity among its component attacks. In general, one would choose those attacks with different nature as the component attacks (Croce and Hein 2020b). However, given the rich literature of attacks, manual construction of high-quality AEs such as AutoAttack is often a challenging task, requiring domain experts (with deep understanding of both attacks and defenses) to explore the vast design space of AEs, which can be time-consuming and sub-optimal.

The main goal of this work is to develop an easy-to-use

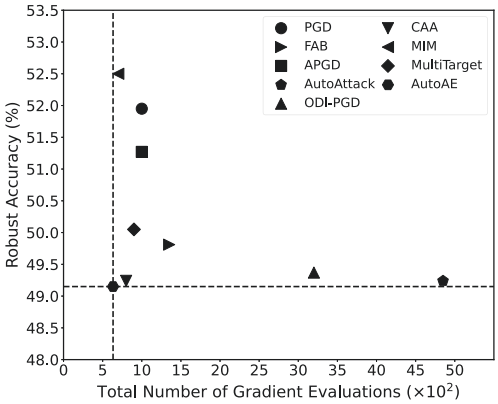

Figure 1: Comparison of AutoAE and the recently proposed attacks on the CIFAR-10 adversarial training model. AutoAE achieves the best robustness evaluation with only a small number of gradient evaluations.

*Corresponding Author.

approach that not only can reliably evaluate adversarial robustness, but also can significantly reduce the human efforts required in building AEs. Specifically, we present AutoAE, a simple yet efficient approach for the automatic construction of AEs. The approach is implemented with a design space containing several candidate attacks; for each attack, there is one hyper-parameter, i.e., iteration steps. To construct an AE, AutoAE starts from an empty ensemble and repeatedly adds the candidate attack associated with its iteration steps to the ensemble that maximizes ensemble improvement per additional iteration spent.

In addition to its conceptual simplicity, AutoAE is appealing also due to its theoretical performance guarantees. Given a defense model, the AE crafted by AutoAE simultaneously achieves $(1-1/e)$-approximation for the success rate achievable at any given time $t \leq T$ (where T is the total iteration steps consumed by the AE), as well as 4-approximation for the optimal iteration steps.

It is worth mentioning that there also exists another attempt on automatically ensembling attacks. Mao et al. (2021) proposed an approach named Composite Adversarial Attack (CAA) that uses multi-objective NSGA-II algorithm to search for attack policies with high attack success rates and low iteration steps. Here an attack policy also consists of multiple attacks, but differs from AEs in that it is a serial attack connection, where the output of previous attack is used as the initialization input for the successor. Since the perturbations computed by attacks are accumulated sequentially, it is necessary for CAA to clip or re-scale the perturbations when they exceed the l_p-norm constraints. Note such repair procedure is unnecessary for AutoAE because the component attacks in an AE are run individually. The main limitation of CAA is that, due to its multi-objective optimization scheme, the algorithm will output a set of non-dominated attack policies; therefore one needs to select a final policy from these policies by weighting multiple objectives, while the best weighting parameter is difficult to determine. Besides, for CAA the policy length is also a parameter that needs to be set by users, while in AutoAE, the ensemble size is automatically determined.

We conduct a large-scale evaluation to assess whether AutoAE can reliably evaluate adversarial robustness. Specifically, we use AutoAE to construct two AEs with a CIFAR-10 l_∞- (ϵ=$8/255$) and a CIFAR-10 l_2- (ϵ=0.5) adversarial training model, and then apply them to 45 top defense models on the RobustBench leaderboard (Croce et al. 2021). Although using only one restart for each component attack, these two AEs consistently achieve high success rates across various defense models trained on different datasets (CIFAR-10, CIFAR-100 and ImageNet) with different constraints (e.g., ϵ=$8/255$ and $\epsilon = 4/255$ for l_∞ attack). In particular, in all except one cases our AEs achieve equal or better (often the latter) robustness evaluation than existing AEs (AutoAttack and CAA); notably, in 29 cases our AEs achieve *new* best known robustness evaluation. Such performance is achieved by these AEs without any tuning or adaptation to any particular defense at hand, suggesting that they are well suited as minimal tests for any new defense.

We do not argue that AutoAE is the ultimate adversarial attack and this work is not intended for boosting the state-of-the-art attack performance. As aforementioned, our main purpose is develop a easy-to-use technique for automatic construction of high-quality AEs that can reliably evaluate adversarial robustness. As a result, AutoAE can significantly reduce human efforts in AE construction and also enables researchers to easily obtain their own AEs to meet specific demands.

Preliminaries and Related Works

Adversarial Attack

Let $\mathcal{F} : [0,1]^D \rightarrow \mathbb{R}^K$ be a K-class image classifier taking decisions according to $\arg\max_{k=1,\ldots,K} \mathcal{F}_k(\cdot)$ and $x \in [0,1]^D$ be an input image which is correctly classified by $\mathcal{F}$ as y. Given a metric $d(\cdot,\cdot)$ and $\epsilon > 0$, the goal of adversarial attack is to find an adversarial example z such that the target model makes wrong predictions on z:

$$\arg\max_{k=1,\ldots,K} \mathcal{F}_k(z) \neq y \ \text{ s.t. } \ d(x,z) \leq \epsilon \wedge z \in [0,1]^D. \quad (1)$$

The problem in Eq. (1) can be rephrased as maximizing some loss function L to enforce z not to be assisigned to class y:

$$\max_{z \in [0,1]^D} L(\mathcal{F}_k(z), y) \ \text{ s.t. } \ d(x,z) \leq \epsilon \wedge z \in [0,1]^D. \quad (2)$$

In image classification, the commonly considered metrics are based on l_p-distances, i.e., $d(x,z) := ||x-z||_p$. Since the projection on the l_p-ball for $p \in \{2,\infty\}$ is available in close form, the Projected Gradient Descent (PGD) attack (Madry et al. 2018), currently the most popular white-box attack, iteratively performs updates on the adversarial example z along the direction of the gradient of loss function.

Hence, the computational complexity of an attack could be characterized by the number of times it computes the gradient of the target model, which typically equals to the number of iteration steps. In this work, we use the number of gradient evaluations (number of iteration steps) as the complexity metric for attacks and AEs.

Existing Attack Ensembles and RobustBench

This work is mostly inspired by AutoAttack (Croce and Hein 2020b), which improves the evaluation of adversarial defenses by using an ensemble of four fixed attacks, APGD_{CE}, APGD_{DLR}, FAB (Croce and Hein 2020a) and Square Attack (Andriushchenko et al. 2020). The key property of AutoAE lies in the diversity among its component attacks with different nature. In this work, we improve the manual construction of AutoAttack to an automatic construction scheme AutoAE, where the attacks in the former are used to implement the design space.

As aforementioned, CAA (Mao et al. 2021) is a multi-objective optimization based approach for automatically ensembling attacks in a serial connection, i.e., attack policy. In spirit, AutoAE and CAA are similar because they both seek to achieve a high success rate with low iteration steps. In particular, CAA optimizes these two values as separate objectives, while AutoAE optimizes them simultaneously

by maximizing the success rate gain per additional iteration spent. Compared to CAA, AutoAE is simpler and more straightforward, with theoretical performance guarantees on both success rate and iteration steps. In practice, AutoAE is easier to use than CAA because the latter requires users to set objective-weighting parameter and policy length. Besides, the experiments show that AutoAE is more efficient than CAA, i.e., it consumes fewer GPU hours to construct AEs with stronger performance (see Figure 2).

In this work, we use RobustBench (Croce et al. 2021) as the testbed, which is a standardized adversarial robustness benchmark that maintains a leaderboard based on more than 120 defense models. It provides a convenient way to track the progress in this field. For each defense, the leaderboard presents the robust test accuracy assessed by AutoAttack and its best known accuracy.

Finally, this work is also inspired by the recent advances in AutoML, such as in the domain of neural architecture search (Elsken, Metzen, and Hutter 2019). These automation techniques can effectively help people get rid of the tedious process of architecture design. This is also true in our work, where searching for high-quality AEs can help better evaluate adversarial robustness.

Methods

We first formulate the AE construction problem and then detail the AutoAE approach, as well as the theoretical analysis.

Problem Formulation

Given a classifier $\mathcal{F}$ and an annotated training set D, where $\mathcal{F}$ can correctly classify the instances in D. Suppose we have a candidate attack pool $\mathbb{S} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$, which can be constructed by collecting existing attacks. The goal is to automatically select attacks from $\mathbb{S}$ to form an AE with strong performance. On the other hand, as a common hyperparameter in attacks, the iteration steps can have a significant impact on the attack performance (Madry et al. 2018). Hence, we set an upper bound $\mathcal{T}$ on the total iteration steps of all attacks in the AE and also integrate the attack's iteration steps into problem formulation. Concretely, the AE, denoted as $\mathbb{A}$, is a list of $\langle$attack, iteration steps$\rangle$ pairs, i.e., $\mathbb{A} = [\langle \mathcal{A}_1, t_1 \rangle, ..., \langle \mathcal{A}_m, t_m \rangle]$, where $\mathcal{A}_i \in \mathbb{S}$, $t_i \in \mathbb{Z}^+$ is the iteration steps of $\mathcal{A}_i$, m is the ensemble size and $\sum_{i=1}^{m} t_i \leq \mathcal{T}$.

We denote the set of all possible pairs by W, i.e., $W = \{\langle \mathcal{A}, t \rangle | \mathcal{A} \in \mathbb{S} \wedge t \in \mathbb{Z}^+ \wedge t \leq \mathcal{T}\}$, and denote the AE that results from appending (adding) a pair $\langle \mathcal{A}, t \rangle$ and another AE $\mathbb{A}'$ to $\mathbb{A}$ by $\mathbb{A} \oplus \langle \mathcal{A}, t \rangle$ and $\mathbb{A} \oplus \mathbb{A}'$, respectively. Generally, an attack $\mathcal{A}$ can be considered as an operation that transforms an input image x to an adversarial example z which is within the ϵ-radius l_p ball around x, i.e., $\mathcal{A} : [0,1]^D \rightarrow \{z | z \in [0,1]^D \wedge ||z - x||_p \leq \epsilon\}$. Supposing $\mathcal{A}$ runs with iteration steps t to attack $\mathcal{F}$ on the instance $\pi = (x, y) \in D$ and generates an adversarial example z, we define $Q(\langle \mathcal{A}, t \rangle, \pi)$ as an indicator function of whether z can successfully fool the target model:

$$Q(\langle \mathcal{A}, t \rangle, \pi) := \mathbb{I}_{\arg\max_{k=1,...,K} \mathcal{F}_k(z) \neq y}(z).$$

When using an ensemble $\mathbb{A} = [\langle \mathcal{A}_1, t_1 \rangle, ..., \langle \mathcal{A}_m, t_m \rangle]$ to attack $\mathcal{F}$ on π, all the attacks in $\mathbb{A}$ are run individually and the best of their outputs is returned. In other words, the performance of $\mathbb{A}$ on π, denoted as $Q(\mathbb{A}, \pi)$, is:

$$Q(\mathbb{A}, \pi) = \max_{i=1,...,m} Q(\langle \mathcal{A}_i, t_i \rangle, \pi).$$

Then the success rate of $\mathbb{A}$ for attacking $\mathcal{F}$ on dataset D, denoted as $Q(\mathbb{A}, D)$, is:

$$Q(\mathbb{A}, D) = \frac{1}{|D|} \sum_{\pi \in D} Q(\mathbb{A}, \pi). \tag{3}$$

Note $Q(\mathbb{A}, D)$ is always 0 when $\mathbb{A}$ is empty. To make $Q(\cdot, \cdot)$ deterministic, in this work we fix the random seeds of all attacks in $\mathbb{S}$ to keep their outputs stable.

Another common concern of an attack is the number of iteration steps, which indicates the computational complexity of it. Ideally, an AE should achieve the best possible success rate (i.e., 100%) using as few iteration steps as possible. Let $\mathbb{A}(t)$ denote the AE resulting from truncating $\mathbb{A}$ at iteration steps t, e.g., for $\mathbb{A} = [\langle \mathcal{A}_1, 4 \rangle, \langle \mathcal{A}_2, 4 \rangle]$, $\mathbb{A}(5) = [\langle \mathcal{A}_1, 4 \rangle, \langle \mathcal{A}_2, 1 \rangle]$. We use the following expected iteration steps (denoted as $C(\mathbb{A}, D)$) needed by $\mathbb{A}$ to achieve 100% success rate on dataset D:

$$C(\mathbb{A}, D) = \sum_{t=0} 1 - Q(\mathbb{A}(t), D). \tag{4}$$

Finally, the AE construction problem is presented in Definition 1. Note the ensemble size is not predefined and should be determined by the construction algorithm. Since D is fixed for $Q(\cdot, D)$ and $C(\cdot, D)$, for notational simplicity, henceforth we omit the D in them and directly use $Q(\cdot)$ and $C(\cdot)$. The problem in Definition 1 is NP-hard, because maximizing only Q is equivalent to solving the NP-hard subset selection problem with general cost constraints (Nemhauser and Wolsey 1978; Qian et al. 2017, 2019).

Definition 1. *Given $\mathcal{F}$, D, $\mathbb{S}$ and $\mathcal{T}$, the AE construction problem is to find $\mathbb{A}^* = (\langle \mathcal{A}_1^*, t_1^* \rangle, ..., \langle \mathcal{A}_{m^*}^*, t_{m^*}^* \rangle)$ that maximizes $Q(\mathbb{A}^*, D)$ and minimizes $C(\mathbb{A}^*, D)$, s.t. $\langle \mathcal{A}_i^*, t_i^* \rangle \in W$, $m^* \in \mathbb{Z}^+$, and $\sum_{i=1}^{m^*} t_i^* \leq \mathcal{T}$.*

AutoAE: Automatic AE Construction

We now introduce AutoAE, a simple yet efficient approach for automatically constructing AEs.

As shown in Algorithm 1, it starts with an empty ensemble $\mathbb{A}$ (line 1) and repeatedly finds the attack $\mathcal{A}^*$ and its iteration steps t^* that, if included in $\mathbb{A}$, maximizes ensemble improvement per additional iteration spent (line 3). Here, ties are broken by choosing the attack with the fewest iteration steps. After this, $\langle \mathcal{A}^*, t^* \rangle$ is subject to the following procedure: if adding it to $\mathbb{A}$ does not improve the success rate of the latter (which means the algorithm has converged), or results in a larger number of used iterations steps than $\mathcal{T}$, AutoAE will terminate and return $\mathbb{A}$ (line 4); otherwise $\langle \mathcal{A}^*, t^* \rangle$ is appended to $\mathbb{A}$ (line 5). For the AE returned by AutoAE, we can further shrink (simplify) it without impairing its performance. Specifically, we remove all pairs $\langle \mathcal{A}, t \rangle \in \mathbb{A}$ satisfying $\exists \langle \mathcal{A}', t' \rangle \in \mathbb{A}$ such that $\mathcal{A} = \mathcal{A}' \wedge t \leq t'$. This procedure can effectively reduce the computational complexity of $\mathbb{A}$, especially when $|\mathbb{S}|$ is small (see the experiments).

Algorithm 1: AutoAE

Input: classifier $\mathcal{F}$, annotated training set D, candidate attack pool $\mathbb{S}$, maximum total iteration steps $\mathcal{T}$

Output: $\mathbb{A}$

1 $\mathbb{A} \leftarrow []$, $t_{used} \leftarrow 0$;
2 **while** ***true*** **do**
3 Let $\langle \mathcal{A}^*, t^* \rangle \leftarrow \arg\max_{\langle \mathcal{A}, t \rangle \in W} \frac{Q(\mathbb{A} \oplus \langle \mathcal{A}, t \rangle) - Q(\mathbb{A})}{t}$;
4 **if** $Q(\mathbb{A} \oplus \langle \mathcal{A}^*, t^* \rangle) = Q(\mathbb{A})$ ***or*** $t_{used} + t^* > \mathcal{T}$ **then** return $\mathbb{A}$;
5 $\mathbb{A} \leftarrow \mathbb{A} \oplus \langle \mathcal{A}^*, t^* \rangle$;
6 $t_{used} \leftarrow t_{used} + t^*$;
7 **end**
8 **return** $\mathbb{A}$

The main computational costs of AutoAE are composed of the function queries to $Q(\cdot)$ (line 3). Specifically, each iteration of the algorithm will query $Q(\cdot)$ for $O(n\mathcal{T})$ times. In the worst case, AutoAE will run for $\mathcal{T}$ iterations where in each iteration t_{used} increases only by 1. Hence, the total number of function queries consumed by AutoAE is $O(n\mathcal{T}^2)$. In practice, we can discretize the range of iteration steps into a few uniform-spacing values to largely reduce the number of function queries; furthermore, we can run the attacks in $\mathbb{S}$ on D in advance to collect their performance data, which can be used to directly compute the value of $Q(\cdot)$ when it is being queried, instead of actually running the ensembles.

Theoretical Analysis

We now theoretically analyze the performance of AutoAE. Let V be the set of all possible AEs that can be constructed based on W. Define $\ell(\mathbb{A})$ the total iteration steps consumed by $\mathbb{A}$, i.e., $\ell(\mathbb{A}) = \sum_{\langle \mathcal{A}, t \rangle \in \mathbb{A}} t$. Our analysis is based on the following fact.

Fact 1. *$Q(\cdot)$ is a monotone submodular function, i.e., for any $\mathbb{A}, \mathbb{A}' \in V$ and any $\langle \mathcal{A}, t \rangle \in W$, it holds that $Q(\mathbb{A}) \leq Q(\mathbb{A} \oplus \mathbb{A}')$, and $Q(\mathbb{A} \oplus \mathbb{A}' \oplus \langle \mathcal{A}, t \rangle) - Q(\mathbb{A} \oplus \mathbb{A}') \leq Q(\mathbb{A} \oplus \langle \mathcal{A}, t \rangle) - Q(\mathbb{A})$.*

Proof. By definition of Q, for any given instance π, we have $Q(\mathbb{A} \oplus \mathbb{A}', \pi) = \max_{\langle \mathcal{A}, t \rangle \in \mathbb{A} \oplus \mathbb{A}'} Q(\langle \mathcal{A}, t \rangle, \pi) \geq \max_{\langle \mathcal{A}, t \rangle \in \mathbb{A}} Q(\langle \mathcal{A}, t \rangle, \pi) = Q(\mathbb{A}, \pi)$. Thus the monotonicity holds.

To prove submodulariy, we first notice that for any given instance π, if $Q(\mathbb{A} \oplus \langle \mathcal{A}, t \rangle, \pi) - Q(\mathbb{A}, \pi) = 1$, then $Q(\mathbb{A} \oplus \mathbb{A}' \oplus \langle \mathcal{A}, t \rangle) - Q(\mathbb{A} \oplus \mathbb{A}') \leq 1$; on the other hand, if $Q(\mathbb{A} \oplus \langle \mathcal{A}, t \rangle, \pi) - Q(\mathbb{A}, \pi) = 0$, then it must hold that $Q(\mathbb{A} \oplus \mathbb{A}' \oplus \langle \mathcal{A}, t \rangle) - Q(\mathbb{A} \oplus \mathbb{A}') = 0$. In either case, $Q(\mathbb{A} \oplus \langle \mathcal{A}, t \rangle, \pi) - Q(\mathbb{A}, \pi) \geq Q(\mathbb{A} \oplus \mathbb{A}' \oplus \langle \mathcal{A}, t \rangle) - Q(\mathbb{A} \oplus \mathbb{A}')$. Thus the submodularity holds. □

Intuitively, Q exhibits a so-called diminishing returns property (Qian, Yu, and Zhou 2015) that the marginal gain of adding $\langle \mathcal{A}, t \rangle$ diminishes as the ensemble size increases. Based on Fact 1, we can establish the following useful lemma that states in terms of the increase in Q per additional iteration step spent, the results from appending an AE to the current ensemble are always upper bounded by the results from appending the best $\langle \mathcal{A}, t \rangle$ pair to it.

Lemma 1. *For any $\mathbb{A}, \mathbb{A}' \in V$, it holds that*

$$\frac{Q(\mathbb{A} \oplus \mathbb{A}') - Q(\mathbb{A})}{\ell(\mathbb{A}')} \leq \max_{\langle \mathcal{A}, t \rangle} \left\{ \frac{Q(\mathbb{A} \oplus \langle \mathcal{A}, t \rangle) - Q(\mathbb{A})}{t} \right\}.$$

Proof. Let r denote the right hand side of the inequality. Let $\mathbb{A} = [a_1, a_2, \ldots, a_L]$, where $a_l = \langle \mathcal{A}_l, t_l \rangle$. Let $\Delta_l = Q(\mathbb{A} \oplus [a_1, a_2, \ldots, a_l]) - Q(\mathbb{A} \oplus [a_1, a_2, \ldots, a_{l-1}])$. We have

$$\begin{aligned}
Q(\mathbb{A} \oplus \mathbb{A}') &= Q(\mathbb{A}) + \sum_{l=1}^{L} \Delta_l \qquad \text{(telescoping series)} \\
&\leq Q(\mathbb{A}) + \sum_{l=1}^{L} (Q(\mathbb{A} \oplus a_l) - Q(\mathbb{A})) \qquad \text{(submodularity)} \\
&\leq Q(\mathbb{A}) + \sum_{l=1}^{L} r \cdot \tau_l \qquad \text{(definition of } r \text{)} \\
&= Q(\mathbb{A}) + r \cdot \ell(\mathbb{A}').
\end{aligned}$$

Rearranging this inequality gives $\frac{Q(\mathbb{A} \oplus \mathbb{A}') - Q(\mathbb{A})}{\ell(\mathbb{A}')} \leq r$, as claimed. □

We now present the approximation bounds of AutoAE on maximizing $Q(\cdot)$ in Eq. (3) and minimizing $C(\cdot)$ in Eq. (4), in Theorem 1 and Theorem 2, respectively.

Theorem 1. *Let $\mathbb{A} = [\langle \mathcal{A}_1, t_1 \rangle, \langle \mathcal{A}_2, t_2 \rangle, ..., \langle \mathcal{A}_m, t_m \rangle]$ be the AE returned by AutoAE and let $T = \ell(\mathbb{A})$. It holds that $Q(\mathbb{A}) \geq (1 - \frac{1}{e}) \max_{\mathbb{A}' \in V} Q(\mathbb{A}'(T))$, where $\mathbb{A}'(T)$ is the AE resulting from truncating $\mathbb{A}'$ at iteration steps T. In other words, $\mathbb{A}$ achieves $(1 - 1/e)$-approximation for the optimal success rate achievable at any given iteration steps $t \leq T$.*

Proof. Let $Q^* = \max_{\mathbb{A}' \in V} Q(\mathbb{A}'(T))$. For integer $j \in [1, m]$, define $\mathbb{A}_j = [\langle \mathcal{A}_1, t_1 \rangle, ..., \langle \mathcal{A}_{j-1}, t_{j-1} \rangle]$, i.e., the AE at the beginning of the j-iteration of AutoAE. Define $Q_j = Q(\mathbb{A}_j)$ (note $Q_1 = 0$) and $\Delta_j = Q^* - Q_j$. Then let $s_j = \max_{\langle \mathcal{A}, t \rangle \in W} \frac{Q(\mathbb{A}_j \oplus \langle \mathcal{A}, t \rangle) - Q_j}{t}$, i.e., the maximum value obtained at the j-iteration in line 3 of Algorithm 1. It is straightforward to verify that $s_j = (\Delta_j - \Delta_{j+1})/t_j$.

We first prove for any $\mathbb{A}' \in V$ and $t \leq \mathcal{T}$, it holds that

$$Q(\mathbb{A}'(t)) \leq Q_j + t \cdot s_j. \tag{5}$$

By monotonicity in Fact 1, we have $Q(\mathbb{A}'(t)) \leq Q(\mathbb{A}_j \oplus \mathbb{A}'(t))$, while by Lemma 1, it further holds that $[Q(\mathbb{A}_j \oplus \mathbb{A}'(t)) - Q_j]/t \leq s_i$. The proof is complete.

Since $T \leq \mathcal{T}$, by Eq. (5), we have $Q^* \leq Q_j + T \cdot s_j$, which implies $\Delta_j \leq T \cdot s_j = T \cdot [(\Delta_j - \Delta_{j+1})/t_j]$. Rearranging this inequality gives $\Delta_{j+1} \leq \Delta_j(1 - t_j/T)$. Unrolling it, we have $\Delta_{m+1} \leq \Delta_1(\prod_{j=1}^{m} 1 - t_j/T)$. The product series is maximized when $t_j = T/m$ for all j, since $\sum_{j=1}^{m} t_j = T$. Thus it holds that

$$Q^* - G_{m+1} = \Delta_{m+1} \leq \Delta_1(1 - \frac{1}{m})^m < \Delta_1 \frac{1}{e} \leq Q^* \frac{1}{e}.$$

Thus $G_{m+1} = Q(\mathbb{A}) \geq (1 - \frac{1}{e}) Q^*$, as claimed. □

Theorem 2. *When $\mathcal{T} \to \infty$, $C(\mathbb{A}) \leq 4 \min_{\mathbb{A}' \in V} C(\mathbb{A}')$, i.e., $\mathbb{A}$ achieves 4-approximation for the optimal expected iteration steps to achieve the optimal success rate.*

The proof is similar to Streeter and Golovin (2007, Theorem 7) and is omitted here. The above Theorem 1 and Theorem 2 indicate that the AEs constructed by AutoAE have performance guarantees in both quality (success rate) and complexity (iteration steps).

Experiments

The main goal of the experiments is to evaluate the potential of AutoAE as a standard robustness evaluation approach. In particular, when using AutoAE to construct AEs, *we are not providing it with the currently strongest defense models, but some seemingly outdated ones*. The purpose of such a setting is to assess whether AutoAE can transfer well from weaker defenses to previously unseen and stronger ones. This is an important characteristic for reliable robustness evaluation.

We evaluate the adversarial robustness with the l_∞ and l_2 distance constraints of 45 top defenses on the leaderboard of RobustBench (Croce et al. 2021) (up to Jan 2022). Note all these defenses have a fully deterministic forward pass; thus the results of running our AEs on them are stable (i.e., no randomness).

Constructing AEs with AutoAE

The candidate attack pool consists of 11 recently proposed individual attacks, including the l_∞ and the l_2 versions of APGD_{CE} (CE), APGD_{DLR} (DLR), FAB, Carlini & Wagner Attack (CW) (Carlini and Wagner 2017) and MultiTargeted Attack (MT) (Gowal et al. 2019), as well as the l_2 attack Decoupled Direction and Norm (DDN) (Rony et al. 2019). We use the implementations of APGD attack and FAB attack from the code repository of AutoAttack. For MT attack, CW attack, and DDN attack, we use the implementations from the repository of CAA. Following AutoAttack (Croce and Hein 2020b), for attacks that require step size parameter, we use the adaptive versions of them that are step size free. We allow each attack with only one restart and set the AE's maximum total iteration steps to 1000, forcing AutoAE to construct AEs with low computational complexity (see Figure 1). Besides, we discretize the range of iteration steps of these attacks into 8 uniform-spacing values to reduce the computational costs of AutoAE. Finally, we randomly select 5,000 instances from the training set of CIFAR-10 as the annotated training set and use two CIFAR-10 adversarial training models from Robustness library (Engstrom et al. 2019) to construct the AE for l_∞ and l_2 attacks, respectively. Note these two defense models only ranked 49 and 14 on their corresponding leaderbords.

Baselines and Defense Models

We compare AutoAE with the other AEs, i.e., AutoAttack and CAA. The former is constructed by human experts while the latter is constructed automatically (like ours). Both of them can achieve generally better robustness evaluation than individual attacks. With Robustbench as the testbed, we use the top 15 defenses on the CIFAR-10 ($l_\infty, \epsilon = 8/255$) and the CIFAR-10 ($l_2, \epsilon = 0.5$) leaderboards, and the top 5 defenses on the CIFAR-100 ($l_\infty, \epsilon = 8/255$) and the ImageNet ($l_\infty, \epsilon = 4/255$) learderboards. Following the guidelines of RobustBench, we use its integrated interface to test our AEs, where for CIFAR-10 and CIRAR-100 all the 10,000 test instances are used, while for ImageNet a fixed set of 5,000 test instances are used. Note the parameter ϵ of the attacks in our AEs are always set in line with the defense being attacked. Since the leaderboards also list the robust accuracy assessed by AutoAttack and the best known accuracy, we directly compare them with the results achieved by our AEs. Besides, CAA (Mao et al. 2021) has been shown to achieve its best performance when building a separate attack policy for each defense model, we collect such results for comparison, leading to five more defenses used in our experiments. Note for some defenses, CAA has not been applied to them, in which case its robust accuracy is unavailable.

AEs constructed by AutoAE	
l_∞	$\langle \text{CE}, t = 32 \rangle, \langle \text{FAB}, t = 63 \rangle, \langle \text{DLR}, t = 160 \rangle, \langle \text{MT}, t = 378 \rangle$
l_2	$\langle \text{CE}, t = 32 \rangle, \langle \text{DDN}, t = 625 \rangle, \langle \text{DLR}, t = 32 \rangle, \langle \text{MT}, t = 63 \rangle$

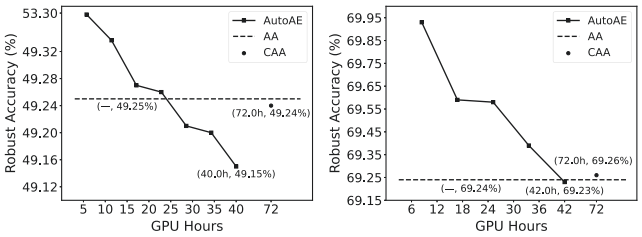

Figure 2: Visualization of the two constructed AEs (upper) and their *test* robust accuracy progress after each iteration of AutoAE (lower), on attacking l_∞ (left) and l_2 (right) CIFAR-10 adversarial training models. The results of AutoAttack (a horizontal line indicating its test robust accuracy) and CAA (a point indicating its test robust accuracy and GPU hours used to search for the attack policy) are also illustrated.

Analysis of the Constructed AEs

Figure 2 presents the two constructed AEs and the progress of their *test* robust accuracy along the iterations of AutoAE. The results of AutoAttack and CAA are also illustrated. For visualization, the GPU hours consumed by AutoAE to collect the candidate attacks' performance data are distributed evenly over all iterations. The first observation from Figure 2 is that both our AEs outperform the two baselines and their test performance improves monotonically from one iteration to the next, which accords to the monotonicity in Fact 1. Actually, AutoAE iterates for 7 times and 5 times in l_∞ and l_2 scenarios, respectively, and then shrink the sizes of the AEs to 4 and 3, respectively.

In terms of efficiency, AutoAE generally only needs to consume around half GPU hours of CAA, making the former the more efficient approach for AE construction. One may observe that for both AEs, AutoAE chooses the three strong attacks (CE, DLR, and MT) that use different losses,

#	Paper	Clean	AutoAttack	CAA	Best Known	AutoAE
CIFAR-10 - l_∞ - $\epsilon = 8/255$						
1	Rebuffi et al. (2021)	92.23	66.58	—	66.56	**66.53***
2	Gowal et al. (2021)	91.10	65.88	65.87	65.87	**65.83***
3	Rebuffi et al. (2021)	88.50	64.64	—	64.58	**64.58**
4	Rebuffi et al. (2021)	88.54	64.25	—	64.20	**64.22**
5	Rade and Moosavi-Dezfooli (2021)	91.47	62.83	—	62.83	**62.82***
6	Gowal et al. (2021)	89.48	62.80	**62.77**	62.76	62.79
7	Rade and Moosavi-Dezfooli (2021)	88.16	60.97	—	60.97	**60.88***
8	Rebuffi et al. (2021)	87.33	60.75	—	60.73	**60.71***
9	Sridhar et al. (2022)	86.53	60.41	—	60.41	**60.32***
10	Wu, tao Xia, and Wang (2020)	88.25	60.04	60.04	60.04	**60.00***
11	Sridhar et al. (2022)	89.46	59.66	—	59.66	**59.58***
12	Zhang et al. (2021)	89.36	59.64	—	59.64	**59.18***
13	Carmon et al. (2019)	89.69	59.53	**59.38**	59.38	**59.38**
14	Sehwag et al. (2022)	85.85	59.09	—	59.09	**59.05***
15	Rade and Moosavi-Dezfooli (2021)	89.02	57.67	—	57.67	**57.63***
16	Gowal et al. (2021)	85.64	56.86	56.83	56.82	**56.82**
17	Wang et al. (2020)	87.50	56.29	56.29	56.29	**56.20***
18	Wu, tao Xia, and Wang (2020)	85.36	56.17	56.15	56.15	**56.11***
19	Pang et al. (2021)	86.43	54.39	54.26	54.26	**54.22***
20	Pang et al. (2020)	85.14	53.74	53.75	53.74	**53.68***
CIFAR-10 - l_2 - $\epsilon = 0.5$						
1	Rebuffi et al. (2021)	95.74	82.32	—	82.32	**82.31***
2	Gowal et al. (2021)	94.74	80.53	80.47	80.47	**80.43***
3	Rebuffi et al. (2021)	92.41	**80.42**	—	80.42	**80.42**
4	Rebuffi et al. (2021)	91.79	**78.80**	—	78.80	**78.80**
5	Augustin, Meinke, and Hein (2020)	93.96	78.79	—	78.79	**78.79**
6	Augustin, Meinke, and Hein (2020)	92.23	**76.25**	—	76.25	**76.25**
7	Rade and Moosavi-Dezfooli (2021)	90.57	**76.15**	—	76.15	**76.15**
8	Sehwag et al. (2022)	90.31	**76.12**	—	76.12	**76.12**
9	Rebuffi et al. (2021)	90.33	**75.86**	—	75.86	**75.86**
10	Gowal et al. (2021)	90.90	**74.50**	**74.50**	74.50	**74.50**
11	Wu, tao Xia, and Wang (2020)	88.51	**73.66**	**73.66**	73.66	**73.66**
12	Sehwag et al. (2022)	89.52	73.39	—	73.39	**73.38***
13	Augustin, Meinke, and Hein (2020)	91.08	**72.91**	72.93	72.91	**72.91**
14	Engstrom et al. (2019)	90.83	**69.24**	—	69.24	**69.24**
15	Rice, Wong, and Kolter (2020)	88.67	67.68	—	67.68	**64.40***
CIFAR-100 - l_∞ - $\epsilon = 8/255$						
1	Gowal et al. (2021)	69.15	36.88	—	36.88	**36.86***
2	Rebuffi et al. (2021)	63.56	34.64	—	34.64	**34.62***
3	Rebuffi et al. (2021)	62.41	32.06	—	32.06	**32.02***
4	Cui et al. (2021)	62.55	30.20	—	30.20	**30.13***
5	Gowal et al. (2021)	60.86	30.03	—	30.03	**30.00***
ImageNet - l_∞ - $\epsilon = 4/255$						
1	Salman et al. (2020)	68.46	38.14	—	38.14	**37.96***
2	Salman et al. (2020)	64.02	34.96	—	34.96	**34.36***
3	Engstrom et al. (2019)	62.56	29.22	—	29.22	**28.86***
4	Wong, Rice, and Kolter (2020)	55.62	26.24	—	26.24	**24.80***
5	Salman et al. (2020)	52.92	25.32	—	25.32	**25.00***

Table 1: Robustness evaluation of the top defenses listed on RobustBench Leaderboard, by AutoAttack, CAA (if available), and AutoAE, in terms of test robust accuracy (%). The clean test accuracy and the best known robust accuracy are also listed. For each defense model, the strongest AE is indicated in bold. A best known robust accuracy is underlined ("_") if it is not obtained by either AutoAttack or CAA, but some other attacks. The robust accuracy of AutoAE is marked with "*" if it provides lower (better) robust accuracy than the best known one. Note that CAA builds an attack policy for each defense separately, while for AutoAttack and AutoAE the AE remains the same under each attack setting.

Transfer	clean	AutoAttack	CAA	AutoAE
Semi-Adv → LBGAT	88.22	71.95	74.72	**70.78**
Semi-Adv → MMA	84.36	71.79	72.96	**70.61**
LBGAT → Semi-Adv	89.69	76.13	80.23	**74.93**
LBGAT → MMA	84.36	70.11	71.78	**68.53**
MMA → Semi-Adv	89.69	83.58	84.99	**82.52**
MMA → LBGAT	88.22	80.00	82.28	**78.78**

Table 2: Evaluation results on attack transferability, in terms of test accuracy (%). "A → B" means transferring adversarial examples generated based on model A to model B. The clean accuracy of target model is also listed. For each transfer scenario, the best value is indicated in bold.

which implies a combination of diverse attacks is preferred by it. Finally, we compare the AE for l_∞ attack with the recently proposed l_∞ attacks and illustrate the results in Figure 1, where among all the attacks, our AE achieves the best robustness evaluation with low computational complexity.

Results and Analysis

Table 1 presents the main results. Overall, AutoAE can provide consistently better robustness evaluation than the two baselines. In all except one cases AutoAE achieves the best robustness evaluation among the AEs, which is far more than AutoAttack with 10 best ones. Compared to CAA, in 9 out of 13 cases where it has been applied, AutoAE achieves better robustness evaluation, and in three out of the left four cases, AutoAE is as good as CAA. Recalling that CAA builds an attack policy for each defense separately while AutoAE builds a unified AE for different defenses, it is conceivable that the performance advantage of AutoAE would be greater if using it to build a separate AE for each defense.

Notably, in 29 cases AutoAE achieves better robustness evaluation than the best known one. Among them many defenses are ranked very highly on the leaderboard, demonstrating that our AEs can transfer well from the weaker defenses (i.e., the defenses used in the construction process) to newer and stronger ones. In summary, the strong performance of AutoAE proves itself a reliable evaluation protocol for adversarial robustness and also indicates the huge potential of automatic construction of AEs.

Attack Transferability

An evaluation approach for adversarial robustness is more desirable if it exhibits good attack transferability. We choose three recent CIFAR-10 defenses ($l_\infty, \epsilon = 8/255$) Semi-Adv (Carmon et al. 2019), LBGAT (Cui et al. 2021), and MMA (Ding et al. 2020), to which both AutoAttack and CAA have been applied, and transfer the adversarial examples (generated based on the 10,000 test instances) between them. As presented in Table 2, AutoAE consistently demonstrate the best transferability in all the transfer scenarios.

Constructing AEs for Specific Defenses

Although AutoAE is proposed for general robustness evaluation, it also enables researchers to easily build their own AEs to meet specific demands. For example, one can use AutoAE to build powerful AEs targeted at some specific defense model. Here, we provide AutoAE with the top-1 CIFAR-10 ($l_\infty, \epsilon = 8/255$) defense model in Table 1 (Rebuffi et al. 2021) (the candidate attack pool and the annotated dataset are the same as before), and the constructed AE obtains a test robust accuracy of 46.3% on this defense. Recalling that in Table 1 the corresponding robust accuracy is 66.53%, such improvement implies that AutoAE can achieve even better robustness evaluation when given newer and stronger defenses.

Discussion and Conclusion

This work focuses on developing an easy-to-use approach for AE construction that can provide reliable robustness evaluation and also can significantly reduce the efforts of human experts. To achieve this goal, we have proposed AutoAE, a conceptually simple approach with theoretical performance guarantees in both AE quality and complexity.

Compared with existing AE construction approach, AutoAE mainly has three advantages. First, it has no hard-to-set parameters. Second, it consumes fewer computation resources (measured by GPU hours) to construct high-quality AEs. Third, it could construct AEs that transfer well from weaker defenses to previously unseen and stronger ones. Also, compared to the fixed AE built by human experts, AutoAE enables researchers to conveniently build their own AEs to meet specific demands, e.g., achieving strong attack performance on specific defenses or incorporating the newly-proposed attacks into AEs to keep up with the field.

To conclude, let us discuss some of the limitations of our approach and possible directions for future work. First, since AutoAE builds an AE based on candidate attacks and a defense model, it may perform not well when the candidate attacks are homogeneous such that the complementarity among them is limited, or the defense is too weak to discriminate between good and bad AEs. Thus, it is possible to include more diverse attacks in the candidate pool to build even better AEs. Second, currently AutoAE only involves appending a $\langle \mathcal{A}, t \rangle$ pair to the ensemble, which may result in stagnation into the local optimum, due to the greedy nature of the approach. It is natural to integrate more pair operations such as deletion of a pair and exchange of two pairs into AutoAE to help it escape from the local optimum. Third, as a generic framework, AutoAE can be used to construct AEs for the black-box and even decision-based settings. Also, it can be used to construct AEs in other domains such as natural language processing and speech. Lastly, from the general perspective of problem solving, AEs fall into the umbrella of algorithm portfolio (AP) (Liu, Tang, and Yao 2019; Tang et al. 2021; Liu, Tang, and Yao 2022), which seeks to combine the advantages of different algorithms. Hence, it is valuable to study how to apply the approach proposed in this work to the automatic construction of APs, and vice versa.

Acknowledgments

This work was supported in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams un-

der Grant 2017ZT07X386, in part by the Shenzhen Peacock Plan under Grant KQTD2016112514355531, and in part by the Guangdong Provincial Key Laboratory under Grant 2020B121201001.

References

Alayrac, J.; Uesato, J.; Huang, P.; Fawzi, A.; Stanforth, R.; and Kohli, P. 2019. Are Labels Required for Improving Adversarial Robustness? In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems, NeurIPS'2019*, 12192–12202. Vancouver, Canada.

Andriushchenko, M.; Croce, F.; Flammarion, N.; and Hein, M. 2020. Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search. In *Proceedings of the 16th European Conference on Computer Vision, ECCV'2016*, 484–501. Glasgow, UK.

Athalye, A.; Carlini, N.; and Wagner, D. A. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML'2018*, 274–283. Stockholm, Sweden.

Augustin, M.; Meinke, A.; and Hein, M. 2020. Adversarial Robustness on In- and Out-Distribution Improves Explainability. In *Proceedings of the 16th European Conference on Computer Vision, ECCV'2016*, 228–245. Glasgow, UK.

Carlini, N.; and Wagner, D. A. 2017. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec'2017*, 3–14. Dallas, TX.

Carmon, Y.; Raghunathan, A.; Schmidt, L.; Duchi, J. C.; and Liang, P. 2019. Unlabeled Data Improves Adversarial Robustness. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems, NeurIPS'2019*, 11190–11201. Vancouver, Canada.

Croce, F.; Andriushchenko, M.; and Hein, M. 2019. Provable Robustness of ReLU networks via Maximization of Linear Regions. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, AISTATS'2019*, 2057–2066. Naha, Japan.

Croce, F.; Andriushchenko, M.; Sehwag, V.; Debenedetti, E.; Flammarion, N.; Chiang, M.; Mittal, P.; and Hein, M. 2021. Robustbench: A Standardized Adversarial Robustness Benchmark. In *Proceedings of the NeurIPS Track on Datasets and Benchmarks, NeurIPS Datasets and Benchmarks'2021*.

Croce, F.; and Hein, M. 2020a. Minimally Distorted Adversarial Examples with a Fast Adaptive Boundary Attack. In *Proceedings of the 37th International Conference on Machine Learning, ICML'2020*, 2196–2205. Virtual Event.

Croce, F.; and Hein, M. 2020b. Reliable Evaluation of Adversarial Robustness with An Ensemble of Diverse Parameter-free Attacks. In *Proceedings of the 37th International Conference on Machine Learning, ICML'2020*, 2206–2216. Virtual Event.

Cui, J.; Liu, S.; Wang, L.; and Jia, J. 2021. Learnable Boundary Guided Adversarial Training. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision, ICCV'2021*, 15701–15710. Montreal, Canada.

Dai, Z.; Liu, S.; Tang, K.; and Li, Q. 2022. Saliency Attack: Towards Imperceptible Black-box Adversarial Attack. *arXiv preprint arXiv:2206.01898*.

Ding, G. W.; Sharma, Y.; Lui, K. Y. C.; and Huang, R. 2020. MMA Training: Direct Input Space Margin Maximization through Adversarial Training. In *Proceedings of the 8th International Conference on Learning Representations, ICLR'2020*. Addis Ababa, Ethiopia.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural Architecture Search: A Survey. *Journal of Machine Learning Research*, 20: 55:1–55:21.

Engstrom, L.; Ilyas, A.; Salman, H.; Santurkar, S.; and Tsipras, D. 2019. Robustness (Python Library).

Gowal, S.; Dvijotham, K.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T. A.; and Kohli, P. 2018. On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models. *arXiv preprint:1810.12715*.

Gowal, S.; Qin, C.; Uesato, J.; Mann, T.; and Kohli, P. 2021. Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples. *arXiv preprint:2010.03593*.

Gowal, S.; Uesato, J.; Qin, C.; Huang, P.; Mann, T. A.; and Kohli, P. 2019. An Alternative Surrogate Loss for PGD-based Adversarial Testing. *arXiv preprint:1910.09338*.

Liu, S.; Lu, N.; Chen, C.; and Tang, K. 2022. Efficient Combinatorial Optimization for Word-Level Adversarial Textual Attack. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30: 98–111.

Liu, S.; Tang, K.; and Yao, X. 2019. Automatic Construction of Parallel Portfolios via Explicit Instance Grouping. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI'2019*, 1560–1567. Honolulu, HI.

Liu, S.; Tang, K.; and Yao, X. 2022. Generative Adversarial Construction of Parallel Portfolios. *IEEE Transactions on Cybernetics*, 52(2): 784–795.

Machado, G. R.; Silva, E.; and Goldschmidt, R. R. 2021. Adversarial Machine Learning in Image Classification: A Survey Toward the Defender's Perspective. *ACM Computing Survey*, 55(1): 1–38.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of the 6th International Conference on Learning Representations, ICLR'2018*. Vancouver, Canada.

Mao, X.; Chen, Y.; Wang, S.; Su, H.; He, Y.; and Xue, H. 2021. Composite Adversarial Attacks. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI'2021*, 8884–8892. Virtual Event.

Nemhauser, G. L.; and Wolsey, L. A. 1978. Best Algorithms for Approximating the Maximum of a Submodular Set Function. *Mathematics of Operations Research*, 3(3): 177–188.

Pang, T.; Yang, X.; Dong, Y.; Su, H.; and Zhu, J. 2021. Bag of Tricks for Adversarial Training. In *Proceedings of the 9th International Conference on Learning Representations, ICLR'2021*. Virtual Event.

Pang, T.; Yang, X.; Dong, Y.; Xu, K.; Zhu, J.; and Su, H. 2020. Boosting Adversarial Training with Hypersphere Embedding. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems, NeurIPS'2020*, 7779–7792. Virtual Event.

Qian, C.; Shi, J.; Yu, Y.; and Tang, K. 2017. On Subset Selection with General Cost Constraints. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'2017*, 2613–2619. Melbourne, Australia.

Qian, C.; Yu, Y.; Tang, K.; Yao, X.; and Zhou, Z.-H. 2019. Maximizing Submodular or Monotone Approximately Submodular Functions by Multi-objective Evolutionary Algorithms. *Artificial Intelligence*, 275: 279–294.

Qian, C.; Yu, Y.; and Zhou, Z.-H. 2015. Subset Selection by Pareto Optimization. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems, NeurIPS'2015*, 1774–1782. Montreal, Canada.

Rade, R.; and Moosavi-Dezfooli, S.-M. 2021. Helper-based Adversarial Training: Reducing Excessive Margin to Achieve a Better Accuracy vs. Robustness Trade-off. In *ICML'2021 Workshop on Adversarial Machine Learning*.

Rebuffi, S.-A.; Gowal, S.; Calian, D. A.; Stimberg, F.; Wiles, O.; and Mann, T. 2021. Fixing Data Augmentation to Improve Ddversarial Robustness. *arXiv preprint:2103.01946*.

Rice, L.; Wong, E.; and Kolter, J. Z. 2020. Overfitting in Adversarially Robust Deep Learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML'2020*, 8093–8104. Virtual Event.

Rony, J.; Hafemann, L. G.; Oliveira, L. S.; Ayed, I. B.; Sabourin, R.; and Granger, E. 2019. Decoupling Direction and Norm for Efficient Gradient-Based L2 Adversarial Attacks and Defenses. In *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition, CVPR'2019*, 4322–4330. Long Beach, CA.

Ruan, W.; Wu, M.; Sun, Y.; Huang, X.; Kroening, D.; and Kwiatkowska, M. 2019. Global Robustness Evaluation of Deep Neural Networks with Provable Guarantees for the Hamming Distance. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI'2019*, 5944–5952.

Salman, H.; Ilyas, A.; Engstrom, L.; Kapoor, A.; and Madry, A. 2020. Do Adversarially Robust ImageNet Models Transfer Better? In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems, NeurIPS'2020*, 3533–3545. Virtual Event.

Sehwag, V.; Mahloujifar, S.; Handina, T.; Dai, S.; Xiang, C.; Chiang, M.; and Mittal, P. 2022. Robust Learning Meets Generative Models: Can Proxy Distributions Improve Adversarial Robustness? In *Proceedings of the 10th International Conference on Learning Representations, ICLR'2022*. Virtual Event.

Sridhar, K.; Sokolsky, O.; Lee, I.; and Weimer, J. 2022. Improving Neural Network Robustness via Persistency of Excitation. In *Proceedings of the 2022 American Control Conference, ACC'2022*, 1521–1526. Atlanta, GA.

Streeter, M.; and Golovin, D. 2007. An online algorithm for maximizing submodular functions. Technical report, School of Computer Science, Carnegie Mellon University.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing Properties of Neural Networks. In *Proceedings of the 2nd International Conference on Learning Representations, ICLR'2014*. Banff, Canada.

Tang, K.; Liu, S.; Yang, P.; and Yao, X. 2021. Few-Shots Parallel Algorithm Portfolio Construction via Co-Evolution. *IEEE Transactions on Evolutionary Computation*, 25(3): 595–607.

Wang, Y.; Zou, D.; Yi, J.; Bailey, J.; Ma, X.; and Gu, Q. 2020. Improving Adversarial Robustness Requires Revisiting Misclassified Examples. In *Proceedings of the 8th International Conference on Learning Representations, ICLR'2020*. Addis Ababa, Ethiopia.

Wong, E.; Rice, L.; and Kolter, J. Z. 2020. Fast is better than free: Revisiting adversarial training. In *Proceedings of the 8th International Conference on Learning Representations, ICLR'2020*. Addis Ababa, Ethiopia.

Wu, D.; tao Xia, S.; and Wang, Y. 2020. Adversarial Weight Perturbation Helps Robust Generalization. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems, NeurIPS'2020*, 2958–2969. Virtual Event.

Zhang, H.; and Wang, J. 2019. Defense Against Adversarial Attacks Using Feature Scattering-based Adversarial Training. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems, NeurIPS'2019*, 1829–1839. Vancouver, Canada.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E. P.; Ghaoui, L. E.; and Jordan, M. I. 2019. Theoretically Principled Trade-off between Robustness and Accuracy. In *Proceedings of the 36th International Conference on Machine Learning, ICML'2019*, 7472–7482. Long Beach, CA.

Zhang, J.; Zhu, J.; Niu, G.; Han, B.; Sugiyama, M.; and Kankanhalli, M. 2021. Geometry-aware Instance-reweighted Adversarial Training. In *Proceedings of the 9th International Conference on Learning Representations, ICLR'2021*. Virtual Event.