Composite Adversarial Attacks

Xiaofeng Mao[1], Yuefeng Chen[1], Shuhui Wang[2*], Hang Su[3], Yuan He[1], Hui Xue [1]

[1] Alibaba Group, [2] Inst. of Comput. Tech., CAS
[3] Tsinghua University
{mxf164419, yuefeng.chenyf}@alibaba-inc.com, wangshuhui@ict.ac.cn, suhangss@mail.tsinghua.edu.cn

Abstract

Adversarial attack is a technique for deceiving Machine Learning (ML) models, which provides a way to evaluate the adversarial robustness. In practice, attack algorithms are artificially selected and tuned by human experts to break a ML system. However, manual selection of attackers tends to be sub-optimal, leading to a mistakenly assessment of model security. In this paper, a new procedure called Composite Adversarial Attack (CAA) is proposed for automatically searching the best combination of attack algorithms and their hyper-parameters from a candidate pool of **32 base attackers**. We design a search space where attack policy is represented as an attacking sequence, *i.e.*, the output of the previous attacker is used as the initialization input for successors. Multi-objective NSGA-II genetic algorithm is adopted for finding the strongest attack policy with minimum complexity. The experimental result shows CAA beats 10 top attackers on 11 diverse defenses with less elapsed time (**6 $\times$ faster than AutoAttack**), and achieves the new state-of-the-art on l_∞, l_2 and unrestricted adversarial attacks.

Introduction

DNNs are vulnerable towards adversarial attacks, which aim to fool a well trained model by producing imperceptibly perturbed examples. This serious security implication quickly attracted a lot of attention from the machine learning community. With in-depth study of adversarial examples, a lot of attack algorithms are proposed to validate the adversarial robustness. Meanwhile, several open source toolboxes, such as Cleverhans (Papernot et al. 2016), FoolBox (Rauber, Brendel, and Bethge 2017) or AdverTorch (Ding, Wang, and Jin 2019), are developed and integrating most existing attack algorithms. All of them provide user friendly interface for attacking a model conveniently and quickly.

However, even if well-designed toolboxes are developed, it still needs a lot of user experience or manual tuning of hyper-parameters for attacking a model, especially when we do not know the details of the defense mechanism. This user-dependent characteristic also makes it hard for instrumentalization of adversarial attacks. On the other hand, manually selecting attackers is somewhat tendentious and sub-optimal. It may arouse a mistakenly assessment of model security, *e.g.*, a well-known false sense of security is gradient obfuscation, which leads to illusory defense for gradient-based attacks.

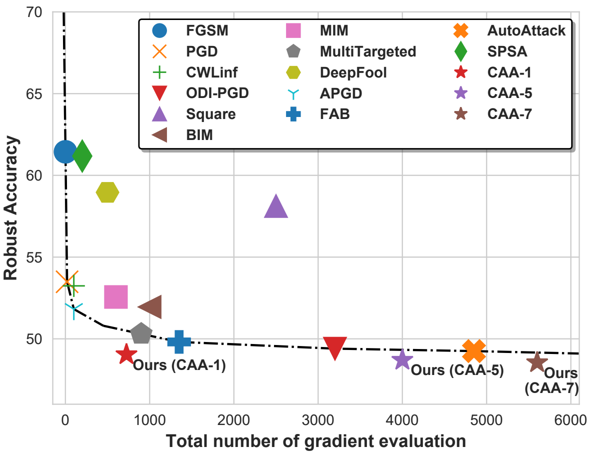


Figure 1: Comparison between CAA and state-of-the-art attackers on CIFAR-10 adversarial training model. CAA-n represents the CAA attack with n restarts. Our method achieves the best attack performance only with a small number of gradient evaluation.

In order to perform more comprehensive and stronger attacks, we first propose to automate the attack process by searching an effective attack policy from a collection of attack algorithms. We name this procedure as Composite Adversarial Attacks (CAA). To demonstrate the key idea of CAA, an example is presented in Fig. 2. Suppose that there are two candidate ways of attacks, *i.e.*, Spatial attack (Engstrom et al. 2019) and FGSM attack (Goodfellow, Shlens, and Szegedy 2014), the goal is choosing one or more of them to compose a stronger attack policy. In Fig. 2 (b), the simplest way is selecting the best single attack as the final policy. However, single attacker is always not strong and generalized enough as shown in previous works (Tramèr and Boneh 2019). A more promising solution (Croce and Hein 2020) is to find multiple attackers, and then ensemble them by choosing the best output constantly that can successfully fool the model (Fig. 2 (c)). Although higher attack success

[*] Shuhui Wang is the corresponding author.

rate can be obtained, the ensemble attack only provides the output level aggregation, without considering the complementarity among different attacking mechanisms.

In our composite adversarial attack, we define an attack policy as the serial connection of attackers, in which the output of previous attacker is used as the initialization input for successors. In Fig. 2 (d), four possible permutations can be generated by two attackers. By using a search algorithm to find the best permutation, we show that an FGSM attack following Spatial attack can achieve 26% higher error rate than ensemble of the two attacks. The advantage of our policy lies in two aspects: 1) By introducing identity attack, CAA can skip any single attack in the sequence and produce more sub-policies. We ensemble all of them during attack to cover all the possible sub-policies. Therefore, our CAA is the more generalized formulation and can represent both single attack and ensemble attack. 2) A strong attack can be produced via progressive steps. Previous works (Suya et al. 2020) have found that some starting points close to the decision boundary are better than the original seeds for optimizing the attacks. Similarly in CAA, we use preceding attackers to create an example far enough from the original seed and close enough to the boundary, such that subsequent attacks are easier to find a stronger adversarial example.

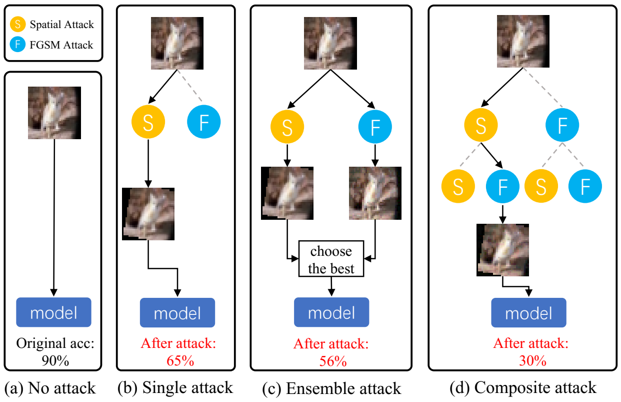
Specifically, CAA is implemented with a search space containing several choices and orders of attack operations. For each attack operation, there are two hyper-parameters, *i.e.*, magnitude ϵ and iteration steps t. We adopt NSGA-II genetic algorithm (Deb et al. 2002) to find the best attack policy which can break through the target model with highest success rate but have the minimal complexity. Extensive experiments show that CAA achieves excellent improvements in two use cases: 1) CAA can be applied directly on the target model of interest to find the best attack policy (CAA$_{dic}$) and 2) learned policies can keep high success rate transferred to attack multiple model architectures, under different tasks (CAA$_{sub}$). We evaluate CAA$_{dic}$ and CAA$_{sub}$ on 11 recent proposed defenses on l_∞, l_2 and unrestricted setting. The result shows our composite adversarial attack achieves the new state-of-the-art in white-box scenario, with a significant reduction of attack time cost.

Preliminaries and Related Work

Adversarial Attacks

Definition and Notation Let $\mathcal{F} : x \in [0,1]^D \rightarrow z \in \mathbb{R}^K$ be a K-class image classifier, where x is an input image in the D-dimensional image space, z represents the logits. Suppose $\mathcal{F}$ is well performed and correctly classify x as its ground truth label y. The purpose of adversarial attacks is to find an adversarial example x_{adv} which is close to the original x under a certain distance metric but causes misclassification of the model: $\mathcal{F}(x_{adv}) \neq y$.

Regular Adversarial Examples Regular adversarial examples are with limited magnitude of perturbations, which is always achieved by bounding the perturbations within the ϵ-radius l_p-ball around the input x. It can be formed by $\mathcal{F}(x_{adv}) \neq y \; s.t. \; \|x_{adv} - x\|_p \leq \lambda$. Fast Gradient Sign Method (FGSM) is a classic l_∞ adversarial attack approach



(a) No attack (b) Single attack (c) Ensemble attack (d) Composite attack

Figure 2: Illustration of single attack, ensemble attack and our composite attack. S and F denote Spatial and FGSM attack, respectively.

performing single step update on the original sample x along the direction of the gradient of loss function. There are many improved versions of FGSM using momentum based multi-steps optimization (Dong et al. 2018), or random initialization of perturbations (Madry et al. 2017). l_2-based attacks such as DDNL2 (Rony et al. 2019) and C&W (Carlini and Wagner 2017) find x_{adv} which has the minimal l_2 distance to the its original examples. l_1-based attackers guarantee the sparsity of the perturbation, such as EAD (Chen et al. 2017). However, l_1-based attacks are not commonly used in practical attack setting. Therefore, we have not implemented CAA under l_1 constraint in this paper.

Unrestricted Adversarial Examples Unrestricted adversarial example is a new type of adversarial example which is not restricted to small norm bounded perturbations. In this case, the attacker might change an input significantly without changing the semantics. (Brown et al. 2018) first introduces the concept of unrestricted adversarial examples and raises a two-player unrestricted attack&defense contest. Recently, there are lots of works aiming to construct such a stronger unrestricted attack using generative models (Song et al. 2018) or spatial transforms (Engstrom et al. 2019). In this paper, we also implement unrestricted CAA with the largest search space (19 attackers). We found that even applying very simple base attackers to form the search space, the policy searched by our CAA still yields surprising attack ability at unrestricted setting.

Automated Machine Learning

Our approach is inspired by recent advances in AutoML and its sub-directions such as Neural Architecture Search (NAS) and Hyper-Parameter Optimization (HPO). In AutoML, search algorithms are used for choice of algorithm, feature pre-processing steps and hyper-parameters automatically. Another similar direction is AutoAugment (Cubuk et al. 2018), which automatically searches for improved data augmentation policies. These automation technologies not only make people get rid of the tedious process of algorithm fine-tuning, but also greatly improve the effectiveness and efficiency of the learning system. In this work, we adopt some search technologies in AutoML, and demonstrate that searching better algorithms and parameters also help for the adversarial attacks.

Composite Adversarial Attacks

Problem Formulation

Assume that we have an annotated dataset $\{X, Y\}$ and a collection of attack algorithms with some unknown hyper-parameters. In this paper, each attack algorithm is regard as an operation $\mathcal{A} : x \in [0,1]^D \rightarrow x_{adv} \in [0,1]^D$, which transforms the input x to adversarial one x_{adv} on the image space. $\mathcal{A}$ has various choices under different attack settings. For example, in white-box adversarial attack, $\mathcal{A}$ directly optimizes a perturbation δ within the ϵ-radius ball around the input x, for maximizing the classification error:

$$\mathcal{A}(x, \mathcal{F}; \epsilon) = \underset{x+\delta}{\arg\max}\, L(\mathcal{F}(x+\delta), y) \;\; s.t. \left\|\delta\right\|_p \leq \epsilon, \quad (1)$$

where L typically refers to the cross-entropy loss, $\left\|\cdot\right\|_p$ presents the l_p-norm and ϵ is the bound of the l_p-norm. ϵ can be viewed as a hyper-parameter of $\mathcal{A}$. Besides, there are many other attack settings, *e.g.*, black-box attack (Uesato et al. 2018; Andriushchenko et al. 2019), unrestricted adversarial attack (Brown et al. 2018), *etc.* We represent them as the attack operation $\mathcal{A}$ in a unified way.

Suppose we have a set of base attack operations, presented as $\mathbb{A} = \{\mathcal{A}_1, \mathcal{A}_2 ... \mathcal{A}_k\}$, where k is the total number of attack operations. The goal of composite adversarial attack is to automate the adversarial attack process by searching for the best composition of attack operations in $\mathbb{A}$ and hyper-parameters of each operation, to achieve more general and powerful attacks. In this work, we only consider two most common hyper-parameters in attack algorithms: 1) the attack magnitude ϵ (also equivalent to the maximal l_p-norm of the perturbation) and 2) optimization steps t for the attack. To limit the search scope of two hyper-parameters, two intervals are given: $\epsilon \in [0, \epsilon_{max}]$ and $t \in [0, t_{max}]$, where ϵ_{max} and t_{max} are the max magnitude and iteration of each attack predefined by users. In this paper, we do not search for the attack step-size, as it is relevant to optimization step t. Instead, all attacks that require step-size parameter (*e.g.*, PGD) are modified to step-size-free version based on previous method (Croce and Hein 2020). Accordingly, the step-size can be changed adaptively based on the optimization step. Then we can define the policy s as the composition of various attacks, which consists of N consecutive attack operations:

$$s : \mathcal{A}_N^s(\mathcal{A}_2^s(\mathcal{A}_1^s(x, \mathcal{F}; \epsilon_{s_1}, t_{s_1}), \mathcal{F}; \epsilon_{s_2}, t_{s_2}))..., \mathcal{F}; \epsilon_{s_N}, t_{s_N}), \quad (2)$$

where $\{\mathcal{A}_n^s \in \mathbb{A} | n = 1, ..., N\}$ is the sampled attacker from $\mathbb{A}$ separately and $\{\{\epsilon_{s_n}, t_{s_n}\} | n = 1, ..., N\}$ is the hyper-parameter of each attack. With the combination of different attack operations and hyper-parameters, we can obtain thousands of possible policies.

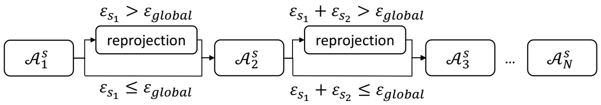

$\varepsilon_{s_1} > \varepsilon_{global}$
$\varepsilon_{s_1} + \varepsilon_{s_2} > \varepsilon_{global}$
$\mathcal{A}_1^s$
reprojection
$\mathcal{A}_2^s$
reprojection
$\mathcal{A}_3^s$
...
$\mathcal{A}_N^s$
$\varepsilon_{s_1} \leq \varepsilon_{global}$
$\varepsilon_{s_1} + \varepsilon_{s_2} \leq \varepsilon_{global}$

Figure 3: Illustration of re-projection module in composite adversarial attack under l_p-norm constraint.

Constraining l_p-Norm by Re-Projection

The attack policy presented in Eq. 2 is a general form, which has no constraint on the global perturbation. When the attack sequence becomes longer, the computed perturbation of each attack algorithm is accumulated, causing the final perturbation on the original input to be large. To solve this problem, we insert a re-projection module between two consecutive attack algorithms. In Fig. 3, the re-projection module first determines whether the ϵ accumulated on previous attackers is larger than the ϵ_{global} of the policy. If it is, the accumulated perturbation will be clipped or rescaled to make the l_p-norm bounded in ϵ_{global}. With this modification, we can use composite adversarial attacks for any l_p-norm conditions.

Search Objective

Previous works commonly use Attack Success Rate (ASR) or the Robust Accuracy (RA) as the objective to design their algorithms. However, these objectives can be achieved at the expense of more elapsed time. For example, recent proposed works (Gowal et al. 2019; Tashiro 2020) use some tricks such as random restart or multiple targets to get higher success rate, with sacrificing the running efficiency. It makes their algorithms extremely slow (even more time-consuming than some black-box attacks). In this work, we emphasize that a good and strong attacker should be both effective and efficient. To meet this goal, we design our objective with minimizing two terms, *i.e.*, robust accuracy and complexity.

Next we elaborate the two objective terms. The first term RA is the accuracy of the target model on generated adversarial examples. It also reflects the strength of the attackers. As for the second term complexity, we use the number of gradient evaluation as the complexity metric. For a regular attack algorithm, the number of gradient evaluation represents the number of times an attack algorithm computing the gradient of the target model during the attack process, and it equals to the optimization step t typically. Therefore, we can formulate the overall objective function as:

$$\mathcal{L} = -\sum_x [\mathcal{F}(s(x)) \neq y] + \alpha \sum_{i=0}^{N} t_{s_i}, \quad (3)$$

where $s(x)$ represents the output of the attack policy for input x, N is the length of the attack policy, and α is a coefficient to compromise the attack strength and complexity. Then, we can apply a search algorithm to find an optimal attack policy s^* from thousands of possible policies, by minimizing the objective $\mathcal{L}$:

$$s^* = \min_s \mathcal{L}. \quad (4)$$

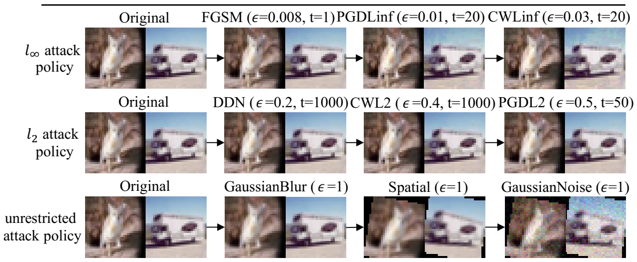

Original
FGSM (ϵ=0.008, t=1)
PGDLinf (ϵ=0.01, t=20)
CWLinf (ϵ=0.03, t=20)
l_∞ attack policy
Original
DDN (ϵ=0.2, t=1000)
CWL2 (ϵ=0.4, t=1000)
PGDL2 (ϵ=0.5, t=50)
l_2 attack policy
Original
GaussianBlur (ϵ=1)
Spatial (ϵ=1)
GaussianNoise (ϵ=1)
unrestricted attack policy

Figure 4: Output visualization of a specific attack policy. For instance, the first row is an l_∞ policy consists of consecutive FGSM, PGD-Linf and CW-Linf attack. Each column presents the output of each attacker component. For unrestricted attacks, ϵ is unlimited, so we set $\epsilon = 1$.

Search Space

Our search space is divided into two parts: 1) searching for choices and orders of attack operations; 2) searching for magnitudes ϵ and steps t of each attack operation. For an attack policy consisting of N base attack operations, the attack operations search forms a problem space of $\|\mathbb{A}\|^N$ possibilities. Additionally, each operation is also associated with their magnitude and step. We discretize the range of magnitudes ϵ and steps t into 8 values (uniform spacing) so that we can simplify the composite adversarial attacks search as a discrete optimization problem. Finally, the overall search space has total size of $(8 \times 8 \times \|\mathbb{A}\|)^N$.

In this paper, three types of policy spaces, *i.e.*, S_{l_∞}, S_{l_2} and $S_{unrestricted}$ are constructed. We implement six l_∞-attackers and six l_2-attackers in space S_{l_∞} and S_{l_2} respectively. In unrestricted case, we use a larger search space with 19 implemented attack algorithms. Besides, all of S_{l_∞}, S_{l_2} and $S_{unrestricted}$ also adopt an IdentityAttack to represent the identity operation. An output visualization of attack policy in each search space is shown in Fig. 4.

Search Strategy

The search strategy plays an important role in finding the best attack policy. In our problem setting, the scale of search space is relatively small. And the cost of policy evaluation is much less than other task such as NAS. This allows us to use some high-performance search algorithms. We compare three widely used methods, *i.e.*, Bayesian Optimization (Snoek, Larochelle, and Adams 2012), Reinforcement Learning (Zoph and Le 2016) and NSGA-II Genetic Algorithm (Deb et al. 2002). The detailed implementation and comparison are shown in Appendix B. Although Bayesian Optimization and Reinforcement Learning are widely considered to be efficient in the field of AutoML, in this problem, we found that they are more time-consuming and slow to converge. In contrast, NSGA-II is faster since there is no need for an additional model optimization processes during the search period. It only needs a few iterations of the population updating to find an optimal solution quickly.

Detailly, NSGA-II needs to maintain a finite set S of all possible policies and a policy evaluation function that maps each policy $s \in S$ onto the set of real numbers $\mathbb{R}$. In this work, we use Eq. 3 as the policy evaluation function. NSGA-II algorithm explores a space of potential attack policies in three steps, namely, a population initialization step that is generating a population P_0 with random policies, an exploration step comprising crossover and mutation of attack policy, and finally an exploitation step that utilizes the hidden useful knowledge stored in the entire history of evaluated policies and find the optimal one. The whole process is shown in Alg. 1. In the remainder of this work, we adopt NSGA-II algorithm for the policy search.

Algorithm 1 Attack policy search using NSGA-II

```
Require: Pool of candidate attackers $\mathbb{A}$; Population size $P$;
Require: Maximum number of generations $G$
$P_0 \leftarrow \emptyset$                ▷ Initialized population with size of $K$
$t \leftarrow 0$
for $i \leftarrow 1$ to $K$ do
    for $j \leftarrow 1$ to $N$ do
        Random sample $\mathcal{A}_j$ from $\mathbb{A}$
        Random sample $\epsilon_j \sim [0, \epsilon_{max}]$, $t_j \sim [0, t_{max}]$
    end for
    $s \leftarrow \mathcal{A}_N(\mathcal{A}_1(x, \mathcal{F}; \epsilon_1, t_1)...), \mathcal{F}; \epsilon_N, t_N)$
    $P_0 \leftarrow P_0 \cup s$
end for
for $t < G$ do          ▷ Run search with Eq. 3 for evaluation
    $P_{t+1} \leftarrow$ NSGA-II($P_t$)          ▷ Update the populations
    $t \leftarrow t + 1$
end for
return best attack policy $s^*$ from $P_t$
```

Experiments

Experiment Setup

In order to validate the performance of our CAA, the searched attack policies on S_{l_∞}, S_{l_2} and $S_{unrestricted}$ are evaluated on 11 open source defense models. We run l_∞ and l_2 attack experiments on CIFAR-10 and ImageNet (Deng et al. 2009) datasets. We perform unrestricted attack on Bird&Bicycle (Brown et al. 2018) datasets. The robust accuracy is recorded as measurement to make comparison with 10 recent top attackers. In the implementation, we take all the results of sub-policies and ensemble them as similar to (Croce and Hein 2020).

Details of Search Space The candidate pool of CAA consists of 32 attack operations, *i.e.*, six l_∞-attacks, six l_2-attacks, 19 unrestricted attacks and the last IdentityAttack (*i.e.*, identity operation). A detailed summary of implemented attack algorithms is shown in Tab. 1. We borrowed the codes of some algorithms from open source attack toolbox, such as Foolbox (Rauber, Brendel, and Bethge 2017) and Advertorch (Ding, Wang, and Jin 2019). The implementation and citation of each base attacker can be found in Appendix A.

Data Configuration For CIFAR-10, we search for the best policies on a small subset, which contains 4,000 examples

S_{l_∞}	S_{l_2}	$S_{unrestricted}$
MI-LinfAttack MT-LinfAttack FGSMAttack PGD-LinfAttack CW-LinfAttack SPSAAttack IdentityAttack	DDNAttack CW-L2Attack MI-L2Attack PGD-L2Attack MT-L2Attack SquareAttack IdentityAttack	17 CorruptionAttacks SpatialAttack SPSAAttack IdentityAttack

Table 1: The implemented attack algorithms in search space S_{l_∞}, S_{l_2} and $S_{unrestricted}$ respectively.

randomly chosen from the train set. Total 10,000 examples in test set are used for the evaluation of the searched policy. For ImageNet, as the whole validation set is large, we randomly select 1000 images for policy search and 1000 images for evaluation from training and testing database respectively. For Bird&Bicycle, we use all 250 test images for evaluation, and 1000 randomly selected training images for attack policy search.

Summary of Experiments We investigate four cases: 1) BestAttack, searching for best single attacker in candidate pool; 2) EnsAttack, searching for the ensemble of multiple attackers; 3) CAA$_{dic}$, directly searching CAA policy on given datasets; and 4) CAA$_{sub}$, searching by attacking adversarial training CIFAR10 model as substitute and transferred to other models or tasks. For fairness, we compare our method with previous state-of-the-art attackers on 11 collected defense models: Advtrain (Madry et al. 2017), TRADES (Zhang et al. 2019), AdvPT (Hendrycks, Lee, and Mazeika 2019), MMA (Ding et al. 2019), JEM (Grathwohl et al. 2019), PCL (Mustafa et al. 2019), Semi-Adv (Carmon et al. 2019), FD (Xie et al. 2019), AdvFree (Shafahi et al. 2019), TRADESv2[1] and LLR[2]. Next we leverage multiple architectures (VGG16 (Simonyan 2014), ResNet50 (He et al. 2016), Inception (Szegedy et al. 2015)) and datasets (MNIST (LeCun et al. 1998), CIFAR-100, SVHN) to investigate the transferability of CAA in black-box and white-box settings. Finally, we do some ablation study on the effect of different policy search algorithms and the attack policy length N. We also analyse the difference between searched policies of non-target and target attack. Some insights can be found in these ablation experiments.

Comparison With State-of-the-Art

Tab. 2 shows the l_∞-based attack result of four variants, *i.e.*, CAA$_{sub}$, CAA$_{dic}$, EnsAttack and BestAttack on CIFAR-10 dataset. Most works study the model robustness in this setting, so we can collect more defenses for evaluation. The compared attackers are 150-step ODI-PGD with 10 ODI-step and 20 restarts, 100-step PGD&APGD with 10 restarts, FAB and AA. The hyper-parameters of FAB and AA are consistent with the original paper (Croce and Hein 2020). All these attackers have the total number of gradient evaluation (complexity) larger than 1000. In contrast, our CAA$_{sub}$ has lower complexity (800), and breaks the model with a higher error rate. It implies that even a substitute attack policy may have high time efficiency and reliability. Direct search on the task of interest can further improve the performance. From the last row of the table, stronger attack policies are founded by CAA$_{dic}$, with the average decrease of 0.1% on the robust accuracy. We also evaluate two optional schemes in Fig. 2, named BestAttack and EnsAttack. The final searched policy of BestAttack is MT-LinfAttack, which is the strongest attacker in S_{l_∞} case. However, the result shows the best single attacker is not competitive in front of existing methods. EnsAttack searches a policy with an ensemble of MT-Linf, PGD-Linf and CW-Linf attacks. Compared to BestAttack, EnsAttack merges multiple attacks and achieves better results. But it is still worse than CAA policy. It implies that CAA are empirically better than ensemble of attackers. For l_2-based attack on CIFAR-10, our method also yields excellent performance.

The result on ImageNet is shown in Tab. 3. We show that CAA gains greater improvement on ImageNet, compared to CIFAR-10. In particular, CAA$_{sub}$ achieves 38.30% accuracy attacking l_∞ adversarially trained models, with around 2% improvement over state-of-the-art. It implies that CAA is more suitable for attacking complex classification tasks. ImageNet classification has more categories and larger image input size. Also, we found the adversarial examples generated by base attackers are more diverse on ImageNet. For such a complex task, there is more room for the attack strategy design.

For unrestricted attack, we choose the benchmark of Bird&Bicycle proposed in *unrestricted adversarial examples contest* (Brown et al. 2018). The top two defense models LLR and TRADESv2 on leaderboard are used for evaluation. For fairness, we only use warm-up attacks in contest as our search space $S_{unrestricted}$, and avoid the attacks that the defense model has never seen before. Both LLR and TRADESv2 get nearly 100% robust accuracy on Corruption, Spatial and SPSA attacks. But after composing these attacks by CAA, the robust accuracy of LLR and TRADESv2 is rapidly dropped to around zero. The result shows that existing unrestricted adversarial defense models are severely overfitting to the single test attackers. In unrestricted attack setting, there is no good defense against our CAA. Therefore, we think there is still a lot of work to do for achieving the truly unrestricted adversarial robustness.

Analysis of Searching Policy We visualize the searched best policy on S_{l_∞}, S_{l_2} and $S_{unrestricted}$ in Tab. 2. The presented policy is searched by attacking adversarially trained model on CIFAR-10 classification task. In all l_∞, l_2 and unrestricted attack scenarios, CAA tends to choose strong attacks. Take policy of S_{l_∞} as an example, CAA chooses the strongest MT-LinfAttack as the first and the second position attack, and abandons the weaker attackers, such as one-step FGSM. Therefore, we think a well selected candidate attacker pool is critical to the performance of CAA. Another foundation is that CAA prefers some policies with the combination of diverse base attackers. It means that a policy

[1] https://github.com/google/unrestricted-adversarial-examples

[2] https://github.com/deepmind/deepmind-research/tree/master/unrestricted_advx

Visualization of CAA$_{sub}$ proxy attack policies	
S_{l_∞}	[('MT-LinfAttack', ϵ=8/255, t=50), ('MT-LinfAttack', ϵ=8/255, t=25), ('CWLinfAttack', ϵ=8/255, t=125)]
S_{l_2}	[('MT-L2Attack', ϵ=0.5, t=100), ('PGD-L2Attack', ϵ=0.4375, t=125), ('DDNAttack', t=1000)]
$S_{unrestricted}$	[('FogAttack', $\epsilon = 1$, $t = 1$), ('FogAttack', $\epsilon = 1$, $t = 1$), ('SPSAAttack', ϵ=16/255, t=100)]

CIFAR-10 - l_∞ - $\epsilon = 8/255$	AdvTrain	TRADES	AdvPT	MMA	JEM	PCL	Semi-Adv	Complexity
PGD (Madry et al. 2017)	51.95	53.47	57.21	50.04	9.21	8.12	61.83	1000
FAB (Croce and Hein 2019)	49.81	51.70	55.27	42.47	62.71	0.71	60.12	1350
APGD (Croce and Hein 2020)	51.27	53.25	56.76	49.88	9.06	7.96	61.29	1000
AA (Croce and Hein 2020)	49.25	51.28	54.92	41.44	8.15	0.28	59.53	4850
ODI-PGD (Tashiro 2020)	49.37	51.29	54.94	41.75	8.62	0.53	59.61	3200
BestAttack on S_{l_∞}	50.12	52.01	55.23	41.85	9.12	0.84	60.74	900
EnsAttack on S_{l_∞}	49.58	51.51	55.02	41.56	8.33	0.73	60.12	800
CAA$_{sub}$ on S_{l_∞}	49.18	51.19	54.82	40.87	7.47	0.0	59.45	**800**
CAA$_{dic}$ on S_{l_∞}	**49.18**	**51.10**	**54.69**	**40.69**	**7.28**	**0.0**	**59.38**	-

Table 2: The table is divided into two parts. The lower part presents the reported RA(%) of l_∞-based attack on diverse CIFAR-10 defenses. Each column presents the result on a specific defense and the last column presents the complexity of the attack algorithm. The upper part of the table presents the best attack policies found by our method.

formed with MI-Linf and PGD-Linf attack always yields little improve, because the difference among them are subtle (with the same principle and objective function). In contrast, in the best policy of S_{l_∞}, CAA selected a more diverse margin loss based CW-Linf attack to assist cross entropy loss based attackers, which promotes the attack performance.

Attack Transferability

We study the transferability of CAA in two scenarios: 1) black-box setting and 2) white-box setting. In black-box setting, we cannot obtain the gradient of the target model. Instead, we use CAA to search a policy on substitute model and generate adversarial examples to attack the target model. In white-box setting, gradient evaluation is allowed, so policy searched on substitute tasks or models are used for generating adversarial examples directly on the target model.

Black-Box Transferability of CAA Here we discuss if CAA can be used for searching black-box transfer attacks. We slightly modify the original CAA to meet this requirement. Specifically, we use an attack policy s to attack substitute model at adversarial example generation stage. Then these adversarial examples is tested on target model. The robust accuracy on target model is regarded as the evaluation score of the policy s. Except for this, the entire search process remains unchanged. We name this variation as CAA$_{trans}$. In the attack transferability experiment, we use three types of models (VGG16, Inceptionv3 and ResNet50) with different architectures, and all of them are defended by standard adversarial training. The result is recorded in Tab. 4. The first column presents the experiment setting. For example, R→V means that we use ResNet50 as substitute model to attack VGG16.

We show that CAA gains better performance in most transfer attack settings. Especially, it significantly increases the attack strength when VGG16 is used for substitute model, causing the decrease of 3% on target model accuracy. The result suggests that an automatic search process also helps for discovering a more black-box transferable attack policy, not limited to white box scenarios. From the visualization of searched transferable policy in Appendix D, we found that CAA$_{trans}$ does not adopt some "strong" attacks, since such attacks may have poor transferability. Oppositely, attacks like FGSM or MI-Linf attack are chosen as better transferable component in the policy, which explains why CAA$_{trans}$ could improve the attack transferability.

White-Box Transferability of CAA Here we seek to understand if it is possible to transfer attack policies in white-box case, namely, policies searched on substitute tasks or models are used for attacking the target model. A detailed experiment is presented in Appendix C. From the result, we highlight that the searched policies on CIFAR-10 still transfer well to many model architectures and datasets. Therefore, we believe that CAA does not "overfit" to the datasets or model architectures and it indeed finds effective policies that catch the true weakness and can be applied to all kinds of such problems. However, there is no guarantee that the attack policies are transferred across defenses. One empirical practice to improve the transferability across defenses is using stronger and more diverse attack algorithms in candidate pool. A reference is in Tab. 2, by using six strong attackers in S_{l_∞}, CAA$_{sub}$ has achieved satisfactory results on multiple defense models.

Ablations

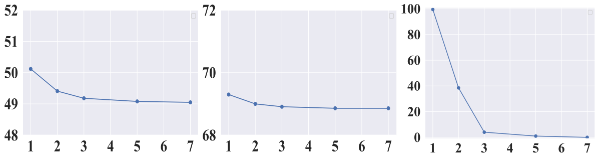
Analysis of the Policy Length N We conduct a series of experiments to explore if a longer policy, which can adopt more and diverse base attackers, exhibits stronger attack ability. We choose five policies with length of 1, 2, 3, 5 and 7. Fig. 5 shows the curve of robust accuracy with the policy length. The CAA equals to find the best base attacker in

CIFAR-10 - l_2 - $\epsilon = 0.5$	AdvTrain	MMA
DDN (Rony et al. 2019)	69.86	66.21
FAB (Croce and Hein 2019)	69.46	66.33
AA (Croce and Hein 2020)	69.26	66.09
CAA$_{sub}$ on S_{l_2}	69.22	65.98
CAA$_{dic}$ on S_{l_2}	**69.20**	**65.95**
ImageNet - l_2 - $\epsilon = 3$	AdvTrain	AdvFree
DDN (Rony et al. 2019)	38.1	34.65
FAB (Croce and Hein 2019)	36.93	34.46
AA (Croce and Hein 2020)	36.3	34.11
CAA$_{sub}$ on S_{l_2}	35.18	33.95
CAA$_{dic}$ on S_{l_2}	**35.07**	**33.89**
ImageNet - l_∞ - $\epsilon = 4/255$	AdvTrain	FD
APGD (Croce and Hein 2020)	42.87	23.18
FAB (Croce and Hein 2019)	41.24	21.86
AA (Croce and Hein 2020)	40.03	21.54
CAA$_{sub}$ on S_{l_∞}	38.30	19.41
CAA$_{dic}$ on S_{l_∞}	**38.21**	**19.27**
Bird&Bicycle - *unrestricted*	LLR	TRADESv2
Common Corruptions	100.0	100.0
Spatial (Engstrom et al. 2019)	100.0	99.5
Boundary (Brendel et al. 2017)	100.0	95.0
SPSA (Uesato et al. 2018)	100.0	100.0
CAA$_{dic}$ on $S_{unrestricted}$	**7.9**	**4.0**

Table 3: RA (%) under l_2 and unrestricted attacks, experimented on ImageNet, CIFAR-10 and Bird&Bicycle datasets.

the candidate pool when $N = 1$, so that the performance is the worst in this case. With the increase of N, the attack policy becomes stronger in all l_∞, l_2 and unrestricted settings. We found that the policy length has the smallest effect on l_2-attack settings. It is reasonable that more base attack just means more optimization steps for l_2-attack. In contrast, N greatly influences the performance on unrestricted attack. The accuracy quickly drops to around zero in unrestricted setting when using a searched attack policy larger than 3.

Different Search Methods Tab. 5 presents the performance and search time of four optimization methods, *i.e.*, Random Search, Bayesian Optimization, Reinforcement Learning and NSGA-II Genetic Algorithm. The detailed im- best one chosen, is regarded as a baseline. Compared to the baseline, all heuristic algorithms find better policies. Although Bayesian Optimization and Reinforcement Learning are widely considered to be efficient in searching of large space, in this problem, we found they are more time-consuming and prone to fall into local optimal. In contrast, NSGA-II finds better policies with the lower cost of 3 GPU/d, and achieves better performance.



$(a)\ l_\infty$ $(b)\ l_2$ $(c)\ unrestricted$

Figure 5: Effect of the policy length on attack performance in l_∞, l_2 and unrestricted settings.

Models	BestAttack	EnsAttack	CAA$_{dic}$
R → V	64.75	63.93	**63.85**
R → I	67.34	**66.05**	66.21
V → R	67.21	65.23	**64.98**
V → I	63.42	61.33	**60.81**
I → R	65.29	64.98	**64.38**
I → V	59.82	58.54	**58.32**

Table 4: Black-box transfer attack results on CIFAR-10. R, V and I represent ResNet, VGG and Inception respectively.

Search Methods	Performance	Search time
Random Search-100	52.09	8 Hours
Reinforcement Learning	51.44	5 GPU/d
Bayesian Optimization	50.02	5 GPU/d
NSGA-II	49.18	3 GPU/d

Table 5: Comparison of different optimization algorithms for the attack policy search.

Target vs. Non-Target Attack Target attack is an application scenario where attackers fool the model to output target label they specified. Otherwise, it is called non-target attack that no target labels are given. We experiment our CAA under target settings in Appendix C. For target attack, CAA searches a policy with less random initialization. It indicates that attackers without a random initialization are more suitable for target setting. Also, compared to margin loss, base attackers with cross entropy loss are favoured by CAA. Policies of CAA also gains improvement in target attack.

Conclusion

We propose an automatic process of learning attack policies formed by a sequence of base attackers for breaking an ML system. By comparing our searched policy with 10 recent attackers on 11 diverse defense, we show that our method achieved better attack success rate with less running time. It empirically demonstrates that searching better algorithms and hyper-parameters also helps for the adversarial attacks.

We think the foremost extension of our work is how to defense attackers which can automatically search for the strongest attack algorithm. From this point of view, we are going to study the adversarial training method based on our CAA in future work.

References

Andriushchenko, M.; Croce, F.; Flammarion, N.; and Hein, M. 2019. Square attack: a query-efficient black-box adversarial attack via random search. *arXiv preprint arXiv:1912.00049* .

Brendel, W.; Rauber, J.; Bethge, M.; and Bethge, M. 2017. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248* .

Brown, T. B.; Carlini, N.; Zhang, C.; Olsson, C.; Christiano, P.; and Goodfellow, I. 2018. Unrestricted adversarial examples. *arXiv preprint arXiv:1809.08352* .

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

Carmon, Y.; Raghunathan, A.; Schmidt, L.; Duchi, J. C.; and Liang, P. S. 2019. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, 11192–11203.

Chen, P.-Y.; Sharma, Y.; Zhang, H.; Yi, J.; and Hsieh, C.-J. 2017. Ead: elastic-net attacks to deep neural networks via adversarial examples. *arXiv preprint arXiv:1709.04114* .

Croce, F.; and Hein, M. 2019. Minimally distorted adversarial examples with a fast adaptive boundary attack. *arXiv preprint arXiv:1907.02044* .

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv preprint arXiv:2003.01690* .

Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2018. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501* .

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6(2): 182–197.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Ding, G. W.; Sharma, Y.; Lui, K. Y. C.; and Huang, R. 2019. Mma training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*.

Ding, Gavin, W.; Wang, L.; and Jin, X. 2019. AdverTorch v0. 1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623* .

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 9185–9193.

Engstrom, L.; Tran, B.; Tsipras, D.; Schmidt, L.; and Madry, A. 2019. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, 1802–1811.

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

Gowal, S.; Uesato, J.; Qin, C.; Huang, P.-S.; Mann, T.; and Kohli, P. 2019. An alternative surrogate loss for pgd-based adversarial testing. *arXiv preprint arXiv:1910.09338* .

Grathwohl, W.; Wang, K.-C.; Jacobsen, J.-H.; Duvenaud, D.; Norouzi, M.; and Swersky, K. 2019. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263* .

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

Hendrycks, D.; Lee, K.; and Mazeika, M. 2019. Using pre-training can improve model robustness and uncertainty. *arXiv preprint arXiv:1901.09960* .

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324.

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

Mustafa, A.; Khan, S.; Hayat, M.; Goecke, R.; Shen, J.; and Shao, L. 2019. Adversarial defense by restricting the hidden space of deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 3385–3394.

Papernot, N.; Faghri, F.; Carlini, N.; Goodfellow, I.; Feinman, R.; Kurakin, A.; Xie, C.; Sharma, Y.; Brown, T.; Roy, A.; et al. 2016. Technical report on the cleverhans v2. 1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768* .

Rauber, J.; Brendel, W.; and Bethge, M. 2017. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131* .

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 IEEE Conference on Computer Vision and Pattern Recognition*, 4322–4330.

Shafahi, A.; Najibi, M.; Ghiasi, M. A.; Xu, Z.; Dickerson, J.; Studer, C.; Davis, L. S.; Taylor, G.; and Goldstein, T. 2019. Adversarial training for free! In *Advances in Neural Information Processing Systems*, 3358–3369.

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

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2951–2959.

Song, Y.; Shu, R.; Kushman, N.; and Ermon, S. 2018. Constructing unrestricted adversarial examples with generative models. In *Advances in Neural Information Processing Systems*, 8312–8323.

Suya, F.; Chi, J.; Evans, D.; and Tian, Y. 2020. Hybrid batch attacks: Finding black-box adversarial examples with limited queries. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 1327–1344.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Tashiro, Y. 2020. Output Diversified Initialization for Adversarial Attacks. *arXiv preprint arXiv:2003.06878* .

Tramèr, F.; and Boneh, D. 2019. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems*, 5866–5876.

Uesato, J.; O'Donoghue, B.; Oord, A. v. d.; and Kohli, P. 2018. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv preprint arXiv:1802.05666* .

Xie, C.; Wu, Y.; Maaten, L. v. d.; Yuille, A. L.; and He, K. 2019. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 501–509.

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. *arXiv preprint arXiv:1901.08573* .

Zoph, B.; and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* .