Diversified Adversarial Attacks based on Conjugate Gradient Method

Keiichiro Yamamura [1] **Haruiki Sato** [1] **Nariaki Tateiwa** [1 2] **Nozomi Hata** [3] **Toru Mitsutake** [1] **Issa Oe** [1]
Hiroki Ishikura [1] **Katsuki Fujisawa** [3]

Abstract

Deep learning models are vulnerable to adversarial examples, and adversarial attacks used to generate such examples have attracted considerable research interest. Although existing methods based on the steepest descent have achieved high attack success rates, ill-conditioned problems occasionally reduce their performance. To address this limitation, we utilize the conjugate gradient (CG) method, which is effective for this type of problem, and propose a novel attack algorithm inspired by the CG method, named the Auto Conjugate Gradient (ACG) attack. The results of large-scale evaluation experiments conducted on the latest robust models show that, for most models, ACG was able to find more adversarial examples with fewer iterations than the existing SOTA algorithm Auto-PGD (APGD). We investigated the difference in search performance between ACG and APGD in terms of diversification and intensification, and define a measure called Diversity Index (DI) to quantify the degree of diversity. From the analysis of the diversity using this index, we show that the more diverse search of the proposed method remarkably improves its attack success rate.

1. Introduction

Deep learning models are effective for various machine learning tasks, and are increasingly being applied to safety-critical tasks such as automated driving. However, deep learning models may misclassify *adversarial examples* (Szegedy et al., 2014; Goodfellow et al., 2015) formed by applying perturbations to their inputs which are too small for the human eye to perceive. Hence, improving the robustness of deep learning models against adversarial examples is crucial for safety-critical tasks. *Adversarial training* (Goodfellow et al., 2015) is an effective method used to create robust models. In adversarial training, adversarial examples are generated and added to the training data. This requires many adversarial examples to be generated quickly for efficient training.

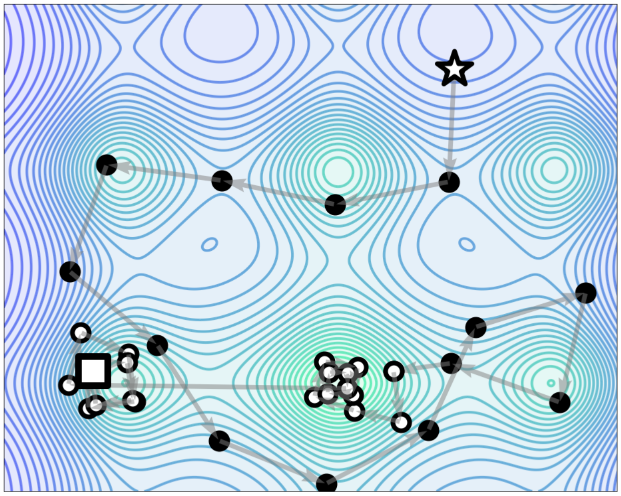


Figure 1. An example of our intended search for a multimodal function. The search finds the global optima owing to an appropriate balance between diversification and intensification. The initial search point is indicated by the white star, and the search ends at the white square. The circles represent the search points, and the black circles indicate that the search has been diversified.

An *adversarial attack* is a method used to generate adversarial examples. A *white-box adversarial attack* assumes that the algorithm can obtain the outputs and gradients of the deep learning models.

The fast gradient sign method (FGSM) (Goodfellow et al., 2015), iterative-FGSM (I-FGSM) (Kurakin et al., 2017), and projected gradient descent (PGD) attack (Madry et al., 2018) use the sign of the steepest gradients at the update. In addition, some attack strategies have successfully improved on the performance of these methods by introducing past search information. Momentum I-FGSM (Dong

[1] Graduate School of Mathematics, Kyushu University, Fukuoka, Japan. [2] Present affiliation is NTT Software Innovation Center, NTT Corporation. [3] Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan.. Correspondence to: Keiichiro Yamamura <keiichiro.yamamura@kyudai.jp>.

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

et al., 2018) and Nesterov I-FGSM (Lin et al., 2020) are based on the momentum method, which simultaneously considers the current and past gradient to determine the next update. Similarly, Auto-PGD (APGD) (Croce & Hein, 2020b), based on projected gradient descent (PGD) also considers the inertia direction of the search point in the next update. However, searches performed by algorithms based on the steepest gradient descent may be insufficient because the objective function of an adversarial attack is nonconvex, nonlinear, and multimodal. We classify the existing methods according to their settings in Appendix A. The relevant literature on adversarial examples has been summarized at the URL provided below.[1].

In this study, a new white-box adversarial attack called an Auto Conjugate Gradient (ACG) attack is proposed based on the conjugate gradient (CG) method. The CG method is a well-known algorithm for systems of linear equations and is also used in nonlinear optimization. It updates the search point in more diverse directions compared to the steepest direction, and can be predicted to search extensively (see Figure 1 and Appendix B).

To the best of our knowledge, our proposed method is the first adversarial attack with a high performance based on the CG. We compared our ACG with APGD, a SOTA white-box adversarial attack, on 64 robust models listed in RobustBench (Croce et al., 2021). The results show that the attack success rate (ASR) of ACG was much higher than that of APGD, with the exception of only a single model (see Tables 2 and 3). Surprisingly, ACG with 1 restart (100 iterations) performed better than APGD with 5 restarts (5 × 100 iterations) against approximately three-fourths of all robust models, although the execution time per iteration of ACG and APGD was almost equal. We thoroughly analyzed the factors involved in the improved attack performance of ACG. Compared to APGD, it may be observed from the results that the movement of the search points in ACG was large, and the attacked class was varied more often during the search.

Moreover, we analyze ACG and APGD for *diversification and intensification*. Diversification and intensification have received considerable attention in the field of metaheuristics (ČrepinšekMatej et al., 2013), where the objective function is generally nonconvex and multimodal, similar to deep neural networks. To control the algorithms properly, some studies on metaheuristics (Cheng et al., 2014; Morales-Castañeda et al., 2020) have attempted to quantify the balance between diversification and intensification. However, to the best of our knowledge, no such methods have been proposed for gradient-based iterative searches such as adversarial attacks. Therefore, we propose *Diversity Index* (DI) to quantify the degree of the diversity of the search points and analyze adversarial attacks (see Section 5.1). Compared to APGD, we demonstrate that ACG can search more extensively by an analysis of the DI .

The contributions of this study are summarized as follows.

- We propose a new adversarial attack called ACG. In a large-scale experiment on 64 robust models, the ASR of ACG overwhelmingly outperformed that of APGD, a SOTA adversarial attack, except for a single model (see Section 4). The ASR of ACG with 1 restart (100 iterations) is generally better than that of APGD with 5 restarts (5 × 100 iterations).
- We propose a metric DI to quantify the degree of diversity and intensity of the search points of gradient-based iterative search algorithms. The DI measure was evaluated, and the results indicated that the search performed by ACG was more diversified than that of APGD (see Section 5).

Our code is available at the URL given below[2].

2. Preliminaries

2.1. Problem Settings

Let the locally differentiable function $g : \mathcal{D} \subseteq \mathbb{R}^m \to \mathbb{R}^K$ be a K-classifier that classifies by $\underset{k=1,\dots,K}{\arg\max}(g_k(\cdot))$, and let $\boldsymbol{x}_{\text{orig}} \in \mathcal{D}$ be a point classified as class c by g. Given the distance function $d(\cdot,\cdot)$ and $\varepsilon > 0$, the feasible region $\mathcal{S}$ in an adversarial attack is defined as $\mathcal{S} = \{\boldsymbol{x} \in \mathcal{D} \,|\, d(\boldsymbol{x}_{\text{orig}}, \boldsymbol{x}) \leq \varepsilon\}$. We then define an adversarial example as $\boldsymbol{x}_{\text{adv}} \in \mathcal{D}$, which satisfies

$$\underset{k=1,\dots,K}{\arg\max}\, g_k(\boldsymbol{x}_{\text{adv}}) \neq c,\ d(\boldsymbol{x}_{\text{orig}}, \boldsymbol{x}_{\text{adv}}) \leq \varepsilon. \tag{1}$$

Let L be the objective function to search for $\boldsymbol{x}_{\text{adv}}$. The adversarial attack can be formulated as follows.

$$\max_{\boldsymbol{x}\in\mathcal{D}} L(g(\boldsymbol{x}), c) \quad \text{s.t.} \quad d(\boldsymbol{x}_{\text{orig}}, \boldsymbol{x}) \leq \varepsilon. \tag{2}$$

The above formulation renders $\boldsymbol{x}$ less discriminative to the class c by g. In classifiers that apply image classification, the Euclidean distance $d(\boldsymbol{v}, \boldsymbol{w}) := \|\boldsymbol{v} - \boldsymbol{w}\|_2$, the uniform distance $d(\boldsymbol{v}, \boldsymbol{w}) := \|\boldsymbol{v} - \boldsymbol{w}\|_\infty$, and $\mathcal{D} = [0, 1]^m$ are often used. We refer to adversarial attacks that use the uniform distance as l_∞ attack.

2.2. PGD Attack

The PGD method is effective for solving the problem (2). Given $f : \mathbb{R}^m \to \mathbb{R}$ and the formulation $\max_{\boldsymbol{x}\in\mathcal{S}} f(\boldsymbol{x})$, let

[1] https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

[2] https://github.com/yamamura-k/ACG

the iterations in PGD be $k = 1, \ldots, N_{\text{iter}}$, where $\boldsymbol{x}^{(k+1)} = P_{\mathcal{S}}(\boldsymbol{x}^{(k)} + \eta^{(k)} \nabla f(\boldsymbol{x}^{(k)}))$, in which $\eta^{(k)}$ is the step size and $P_{\mathcal{S}}$ is the projection onto the feasible region $\mathcal{S}$. APGD adds a momentum update method to PGD. Let $\boldsymbol{\delta}^{(k)}$ be the update direction for each iteration (e.g., $\nabla f(\boldsymbol{x}^{(k)})$ for a uniform distance case). The update rules of APGD in a single iteration, including the momentum term, are defined as follows.

$$\boldsymbol{z}'^{(k+1)} = \boldsymbol{x}^{(k)} + \eta^{(k)} \sigma(\boldsymbol{\delta}^{(k)}) \tag{3}$$

$$\boldsymbol{z}^{(k+1)} = P_{\mathcal{S}}(\boldsymbol{z}'^{(k)}) \tag{4}$$

$$\begin{aligned} \boldsymbol{x}^{(k+1)} = &P_{\mathcal{S}}(\boldsymbol{x}^{(k)} + \alpha(\boldsymbol{z}^{(k+1)} - \boldsymbol{x}^{(k)}) + \\ &(1-\alpha)(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)})), \end{aligned} \tag{5}$$

where $\sigma(\cdot)$ is a type of normalization and α is a coefficient representing the strength of the momentum term, and $\alpha = 0.75$ is used in APGD.

2.3. General Conjugate Gradient Method

The conjugate gradient method (CG) was developed to solve linear equations and subsequently extended to the minimization of strictly convex quadratic and general nonlinear functions. Most existing works on CG methods have considered unconstrained optimization problems, but CG can be applied to constrained problems by using projection. Given an initial point $\boldsymbol{x}^{(0)}$, the initial conjugate gradient $\boldsymbol{s}^{(0)}$ is set to $\mathbf{0}$, and the k-th search point $\boldsymbol{x}^{(k)}$ and conjugate gradient $\boldsymbol{s}^{(k)}$ are updated as $\boldsymbol{s}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)}) + \beta^{(k)} \boldsymbol{s}^{(k-1)}, \eta^{(k)} = \arg\min\{f(\boldsymbol{x}^{(k)} + \eta \boldsymbol{s}^{(k)}) \mid \eta \geq 0\}, \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \eta^{(k)} \boldsymbol{s}^{(k)}$ where $k \geq 1$ and $\beta^{(k)}$ is a parameter calculated from past search information. The step size $\eta^{(k)}$ is usually determined by a linear search to satisfy some conditions such as the Wolfe conditions, because solving $\arg\min\{f(\boldsymbol{x}^{(k)} + \eta \boldsymbol{s}^{(k)}) \mid \eta \geq 0\}$ exactly is difficult.

Consider the problem of minimizing the strictly convex quadratic function $f(x) = \boldsymbol{x}^T A \boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x}$, where A is a positive definite matrix and $\boldsymbol{x} \in \mathbb{R}^n$. In this case, the coefficient $\beta^{(k)}$ is $\frac{\langle A\boldsymbol{s}^{(k-1)}, -\nabla f(\boldsymbol{x}^{(k)})\rangle}{\langle A\boldsymbol{s}^{(k-1)}, \boldsymbol{s}^{(k-1)}\rangle}$. When the objective function is a strictly convex quadratic function, CG is known to be able to find the global solution in less than n iterations under an exact linear search.

For nonlinear functions, some formulas have been proposed to calculate $\beta^{(k)}$ have been proposed (for further details, see (Hager & Zhang, 2006)). In this study, we use the following formula for $\beta^{(k)}$ proposed by (Hestenes & Stiefel, 1952), which exhibited the highest ASR in our preliminary experiments (Appendix I).

$$\beta_{HS}^{(k)} = \frac{\langle \nabla f(\boldsymbol{x}^{(k)}), \boldsymbol{y}^{(k-1)} \rangle}{\langle \boldsymbol{s}^{(k-1)}, \boldsymbol{y}^{(k-1)} \rangle}, \tag{6}$$

where $\boldsymbol{y}^{(k-1)} = \nabla f(\boldsymbol{x}^{(k)}) - \nabla f(\boldsymbol{x}^{(k-1)})$. Theoretical convergence studies on CG methods often assume that $\beta^{(k)} \geq 0$ (Hager & Zhang, 2006) and some implementations use $\max\{\beta^{(k)}, 0\}$ instead of $\beta^{(k)}$.

Algorithm 1 ACG

```
Input: $f, \mathcal{S}, \boldsymbol{x}^{(0)}, \eta^{(0)}, N_{\text{iter}}, W = \{w_0, \ldots, w_n\}$
Output: $\boldsymbol{x}_{\text{adv}}$
$\boldsymbol{x}_{\text{adv}} \leftarrow \boldsymbol{x}^{(0)}$; $\beta^{(0)} \leftarrow 0$; $\boldsymbol{s}^{(0)} \leftarrow \nabla f(\boldsymbol{x}^{(0)})$
$\boldsymbol{x}_{\text{pre}} \leftarrow \boldsymbol{x}^{(0)}$; $\boldsymbol{s}_{\text{pre}} \leftarrow \boldsymbol{s}^{(0)}$
for $k = 0$ to $N_{\text{iter}} - 1$ do
    Compute $\boldsymbol{x}^{(k+1)}$ (10)
    if $f(\boldsymbol{x}^{(k+1)}) > f(\boldsymbol{x}_{\text{adv}})$ then
        $\boldsymbol{x}_{\text{adv}} \leftarrow \boldsymbol{x}^{(k+1)}$; $\boldsymbol{x}_{\text{pre}} \leftarrow \boldsymbol{x}^{(k)}$; $\boldsymbol{s}_{\text{pre}} \leftarrow \boldsymbol{s}^{(k)}$
    end if
    $\eta^{(k+1)} \leftarrow \eta^{(k)}$
    if $k \in W$ then
        if Condition (I) or (II) is satisfied then
            $\eta^{(k+1)} \leftarrow \eta^{(k)}/2$;
            $\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}_{\text{adv}}$; $\boldsymbol{x}^{(k)} \leftarrow \boldsymbol{x}_{\text{pre}}$; $\boldsymbol{s}^{(k)} \leftarrow \boldsymbol{s}_{\text{pre}}$
        end if
    end if
    Compute $\beta^{(k+1)}$ (8) and $\boldsymbol{s}^{(k+1)}$ (9)
end for
```

3. Auto Conjugate Gradient (ACG) Attack

We propose the Auto Conjugate Gradient (ACG) attack as a novel adversarial attack inspired by the CG approach. The proposed scheme is summarized in Algorithm 1.

The major differences among ACG, general CG, and APGD are summarized in Table 1. We apply the step size strategy employed in APGD instead of a linear search because the forward propagation is relatively time-consuming. In addition, we do not restrict $\beta \geq 0$, whereas the general CG usually makes β non-negative (see Appendix G).

3.1. ACG Step

To solve the maximization problem as an adversarial attack, we use $-\nabla f(\cdot)$ instead of $\nabla f(\cdot)$ in (6), that is,

$$\boldsymbol{y}^{(k-1)} = \nabla f(\boldsymbol{x}^{(k-1)}) - \nabla f(\boldsymbol{x}^{(k)}), \tag{7}$$

$$\beta_{HS}^{(k)} = \frac{\langle -\nabla f(\boldsymbol{x}^{(k)}), \boldsymbol{y}^{(k-1)} \rangle}{\langle \boldsymbol{s}^{(k-1)}, \boldsymbol{y}^{(k-1)} \rangle}, \tag{8}$$

$$\boldsymbol{s}^{(k)} = \nabla f(\boldsymbol{x}^{(k)}) + \beta_{HS}^{(k)} \boldsymbol{s}^{(k-1)}, \tag{9}$$

$$\boldsymbol{x}^{(k+1)} = P_{\mathcal{S}}\left(\boldsymbol{x}^{(k)} + \eta^{(k)} \cdot \sigma(\boldsymbol{s}^{(k)})\right), \tag{10}$$

where $\sigma(\cdot)$ is a type of normalization. ACG uses a sign function as σ because many previous studies including APGD use it for an l_∞ attack. There is a possibility of division by zero when calculating $\beta_{HS}^{(k)}$, and empirically, it

Table 1. The differences among ACG, General CG, and APGD. $\boldsymbol{\delta}^{(k)}, \sigma, \eta^{(k)}$ refer to the update direction, whether to normalize $\boldsymbol{\delta}^{(k)}$, and the step size strategy, respectively (described in Section 2). $\boldsymbol{s}^{(k)}$ is the CG direction. The "momentum" column refers to whether a momentum term is used in the update.

	$\boldsymbol{\delta}^{(k)}$	momentum	σ	$\eta^{(k)}$
ACG	$\boldsymbol{s}^{(k)}$	-	✓	Section 3.2
General CG	$\boldsymbol{s}^{(k)}$	-	-	linear search
APGD	$\nabla f(\boldsymbol{x}^{(k)})$	✓	✓	Section 3.2

occurs when $\boldsymbol{y}^{(k-1)} = \mathbf{0}$. Therefore, when division by zero is called for, the issue is addressed by setting $\beta^{(k)} = 0$.

3.2. Step Size Selection

We use the same method proposed in APGD to select the step size. The initial step size $\eta^{(0)}$ is set to 2ε, and when the number of iterations reaches the precomputed checkpoint w_j, the step size η is halved if either of the following two conditions are satisfied.

(I) $N_{\text{inc}} < \rho \cdot (w_j - w_{j-1})$,
(II) $\eta^{(w_{j-1})} = \eta^{(w_j)}$ and $f_{\max}^{(w_{j-1})} = f_{\max}^{(w_j)}$,

where $N_{\text{inc}} := \#\{i = w_{j-1}, \cdots, w_j - 1 \mid f(\boldsymbol{x}^{(i+1)}) > f(\boldsymbol{x}^{(i)})\}$ and $f_{\max}^{(k)} := \max\{f(\boldsymbol{x}^{(i)}) \mid i = 1, \cdots, k\}$.

4. Experiments

We investigated the performance of ACG for an l_∞ attack using the robust models listed in RobustBench.

Models and Dataset: We used 64 models, i.e., 42, 17, and 5 models based on the CIFAR-10, CIFAR-100, and ImageNet datasets, respectively. From a validation dataset, we used 10,000 test images for the evaluation when applying the CIFAR-10 and CIFAR-100 datasets, and 5,000 images when using the ImageNet dataset.

Loss Function: We used the CW loss (Carlini & Wagner, 2017) as the objective function. Let c be the correct answer class for input $\boldsymbol{x}$. Then, the CW loss is defined as follows.

$$\mathrm{CW}(\boldsymbol{x}, c) = -g_c(\boldsymbol{x}) + \max_{i \neq c} g_i(\boldsymbol{x}). \tag{11}$$

An attack using CW loss succeeds if we find an adversarial example $\boldsymbol{x}_{\text{adv}}$ that satisfies $\mathrm{CW}(\boldsymbol{x}_{\text{adv}}, c) \geq 0$. The results of experiments using the DLR loss proposed in (Croce & Hein, 2020b) are shown in Appendix F.

Initial Points. In the case of an l_∞ attack, the center of the feasible region (l_∞-ball with a diameter of ε) is defined as $\frac{\boldsymbol{u}+\boldsymbol{\ell}}{2}$, where $\boldsymbol{u} = \min(\boldsymbol{x}_{\text{orig}} + \varepsilon\mathbf{1}, \mathbf{1})$, and $\boldsymbol{\ell} = \max(\boldsymbol{x}_{\text{orig}} - \varepsilon\mathbf{1}, \mathbf{0})$. We referred to one hundred iterations of the update from an initial point as a restart. Following APGD, we applied 5 restarts in the experiments. The initial point of the first restart was the center of the feasible region, whereas the others were determined through sampling from a uniform distribution. For reproducibility of the results, the seed of the random numbers was fixed.

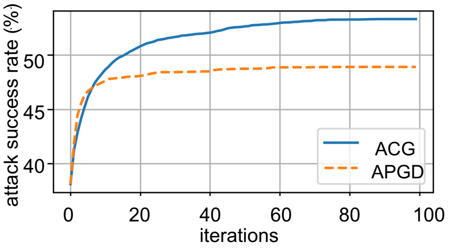


Figure 2. Transitions of the ASR of APGD and ACG. The attack model was (Ding et al., 2020).

4.1. Comparison of ACG and APGD

To evaluate the attack performance under the formulation (2), we conducted experiments to compare the performance of APGD, a SOTA method, with that of ACG. The parameters for the step size selection ρ, checkpoints w, the number of iterations N_{iter}, and the number of restarts were the same as in the study on APGD, i.e., $\rho = 0.75$, $N_{\text{iter}} = 100$, and 5 restarts. The results are summarized in Tables 2 to 3. The columns APGD(N) and ACG(N) show the attack success rate (ASR) of APGD and ACG with N restarts, respectively. From these tables, it may be observed that the ASRs of ACG were higher than those of APGD in all but only 1 of the 64 models. These results indicate that ACG exhibited a higher attack performance, regardless of the datasets or the model's architectures used.

Surprisingly, when comparing APGD(5) and ACG(1), it may be noted that ACG(1) achieved the same or higher ASR for three-fourths of all 64 models with fewer restarts (see Tables 2 to 3). Therefore, we expect that ACG will enable faster attacks with fewer inferences. Details are provided in the next section. Furthermore, ACG(1) does not rely on random numbers to select an initial point because the initial point is the center of the feasible region. In other words, ACG outperforms APGD with only deterministic operations.

Figure 2 shows the transition of the ASR for APGD and ACG. The ASR of APGD rapidly increased and almost converged in the early stages of the search. By contrast, the ASR of ACG continues to increase even at the end of the search. Note that this figure is created based on the data obtained from a model (Ding et al., 2020), whereas the same trend is also observed for other models.

Table 2. The ASR of APGD and ACG for the robust models listed in RobustBench. CIFAR-100 was used as the dataset and $\varepsilon = \frac{8}{255}$. The highest ASR is in bold, and the second is underlined. APGD(N) refers to APGD with N-times the initial point selection. The meanings of the other columns are the same.

CIFAR-100 ($\varepsilon = 8/255$)		Attack Success Rate				
paper	**Architecture**	**APGD(1)**	**ACG(1)**	**APGD(5)**	**ACG(5)**	**diff**
(Addepalli et al., 2021)	PreActResNet-18	72.10	72.12	72.25	**72.47**	0.22
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	70.40	70.63	70.55	**70.86**	0.31
(Rebuffi et al., 2021)	PreActResNet-18	70.86	71.07	70.93	**71.29**	0.36
(Rice et al., 2020)	PreActResNet-18	79.78	80.24	79.99	**80.63**	0.64
(Hendrycks et al., 2019)	WideResNet-28-10	69.28	69.97	69.50	**70.51**	1.01
(Rebuffi et al., 2021)	WideResNet-28-10	66.41	66.87	66.67	**67.27**	0.60
(Addepalli et al., 2021)	WideResNet-34-10	68.53	68.12	**68.74**	68.52	-0.21
(Chen & Lee, 2021)	WideResNet-34-10	68.24	68.42	68.36	**68.77**	0.41
(Cui et al., 2021)	WideResNet-34-10	71.85	72.16	72.15	**72.56**	0.41
(Cui et al., 2021)	WideResNet-34-10	69.63	69.96	69.87	**70.33**	0.46
(Sitawarin et al., 2021)	WideResNet-34-10	73.07	73.64	73.43	**74.27**	0.84
(Wu et al., 2020)	WideResNet-34-10	69.13	69.58	69.32	**70.11**	0.79
(Chen et al., 2021)	WideResNet-34-10	71.76	71.78	71.96	**72.18**	0.22
(Cui et al., 2021)	WideResNet-34-20	68.50	68.75	68.72	**69.13**	0.41
(Gowal et al., 2020)	WideResNet-70-16	61.23	61.67	61.55	**62.19**	0.64
(Gowal et al., 2020)	WideResNet-70-16	68.76	69.13	69.04	**69.43**	0.39
(Rebuffi et al., 2021)	WideResNet-70-16	63.94	64.38	64.17	**64.77**	0.60

4.2. Comparison of execution time

We compared the execution times of APGD and ACG with Intel(R) Xeon(R) Gold 6240R CPU and NVIDIA GeForce RTX 3090 GPU. The execution time was recorded as the time elapsed from the start to the end of the attack. Table 4 shows that APGD with five restarts (APGD(5)) took 22m, 5.88s to attack (Ding et al., 2020), and ACG with five restarts (ACG(5)) took 21m, 15.67s in real time. In addition, APGD with one restart (APGD(1)) took 6m, 45.26s to attack (Ding et al., 2020), and ACG with one restart (ACG(1)) took 6m, 56.78s. From this experiment, the ratio of the execution time of ACG(5) to APGD(5) was about 0.96 and that of ACG(1) to APGD(1) was approximately 1.03. This means the computational cost of ACG is nearly the same as that of APGD. Furthermore, ACG with one restart outperformed APGD with five restarts in terms of the ASR, which show that ACG was able to achieve a higher ASR more than 3 times faster than APGD.

4.3. Variations in the Class Attacked

When we attacked using the CW loss, the search was updated to find inputs classified in class $c' \neq c$. Specifically, c' is $\underset{i \neq c}{\arg\max}\, g_i(\boldsymbol{x})$, and we refer to this c' as the *CW target class* (*CTC*). The value of c' varied during the search, depending on the search point. This section summarizes the CTC results of ACG and APGD attacks based on the results of the CIFAR-10 dataset.

First, CTCs were changed at least once during the search in 42.47% of inputs for ACG and 1.36% for APGD. The average number of times the CTC was switched during the search was 2.14 for ACG and 0.02 for APGD. In APGD, CTC showed almost no change from the initial CTC during the search. In contrast, ACG frequently switched CTCs during the search. Table 5 also summarizes the differences in the final CTCs of ACG and APGD in terms of successful and unsuccessful attacks. As shown in the table, ACG attacked a different class than APGD for 96.14% of the images that failed in APGD but succeeded in ACG. The same trend was observed in the other models, suggesting that ACG increased the ASR by switching CTCs and attacking a different class than APGD.

4.4. Effect of Conjugate Gradient

Tables 2 and 3 shows that ACG achieved a higher ASR value than APGD. Because the update function is the major difference between the two algorithms (see Table 1), we investigated how the conjugate gradient affected the search (see the formula (9)).

The top portion of Figure 3 shows the transitions of the 2-norm between two successive search points $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2$ on APGD and ACG. Regarding the distance moved between the two points, it may be observed that the search points of ACG moved further than those of APGD. Moreover, to investigate the effect of the projection on APGD, we calculated $\frac{\|\boldsymbol{z}^{(k+1)} - \boldsymbol{z}'^{(k+1)}\|_2}{\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2}$ as the ratio of the distance traveled between the two search points, which indicates the amount of update distance wasted by the projection. Although we mainly focused on the numerator $\|\boldsymbol{z}^{(k+1)} - \boldsymbol{z}'^{(k+1)}\|_2$, we divided it by $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2$ to exclude the effect of the difference in step size. The ratio of the distance traveled in 100 iterations is shown at

Table 3. Same as in Table 2, but using CIFAR-10 and ImageNet datasets with $\varepsilon = \frac{8}{255}, \frac{4}{255}$, respectively.

CIFAR-10 ($\varepsilon = 8/255$)		Attack Success Rate				
paper	**Architecture**	**APGD(1)**	**ACG(1)**	**APGD(5)**	**ACG(5)**	**diff**
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	42.33	42.49	42.46	**42.65**	0.19
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	41.51	41.84	41.65	**42.12**	0.47
(Rebuffi et al., 2021)	PreActResNet-18	42.73	43.01	42.91	**43.15**	0.24
(Andriushchenko et al., 2020)	PreActResNet-18	53.55	54.42	53.82	**54.90**	1.08
(Sehwag et al., 2021)	ResNet-18	43.62	44.16	43.91	**44.79**	0.88
(Chen et al., 2020)	ResNet-50	47.95	48.12	48.08	**48.28**	0.20
(Wong et al., 2020)	ResNet-50	54.02	54.75	54.26	**55.44**	1.18
(Engstrom et al., 2019)	ResNet-50	47.69	48.43	48.08	**49.25**	1.17
(Rebuffi et al., 2021)	WideResNet-106-16	34.43	34.70	34.71	**35.03**	0.32
(Carmon et al., 2019)	WideResNet-28-10	39.38	39.68	39.59	**40.03**	0.44
(Gowal et al., 2020)	WideResNet-28-10	36.33	36.63	36.45	**36.90**	0.45
(Hendrycks et al., 2019)	WideResNet-28-10	43.53	43.97	43.82	**44.36**	0.54
(Rade & Moosavi-Dezfooli, 2021)	WideResNet-28-10	38.48	38.62	38.64	**38.87**	0.23
(Rebuffi et al., 2021)	WideResNet-28-10	38.29	38.43	38.47	**38.80**	0.33
(Sehwag et al., 2020)	WideResNet-28-10	41.75	42.07	41.93	**42.41**	0.48
(Sridhar et al., 2021)	WideResNet-28-10	39.27	39.49	39.45	**39.85**	0.40
(Wang et al., 2020)	WideResNet-28-10	41.85	42.12	42.15	**42.57**	0.42
(Wu et al., 2020)	WideResNet-28-10	39.38	39.49	39.56	**39.70**	0.14
(Zhang et al., 2021)	WideResNet-28-10	39.79	39.93	39.98	**40.25**	0.27
(Ding et al., 2020)	WideResNet-28-4	48.73	53.40	49.67	**55.77**	6.10
(Cui et al., 2021)	WideResNet-34-10	46.20	46.42	46.41	**46.90**	0.49
(Huang et al., 2020)	WideResNet-34-10	46.09	46.30	46.19	**46.72**	0.53
(Rade & Moosavi-Dezfooli, 2021)	WideResNet-34-10	36.30	36.57	36.46	**36.83**	0.37
(Sehwag et al., 2021)	WideResNet-34-10	39.32	39.84	39.58	**40.18**	0.60
(Sitawarin et al., 2021)	WideResNet-34-10	46.91	47.58	47.23	**48.02**	0.79
(Wu et al., 2020)	WideResNet-34-10	43.21	43.24	43.36	**43.60**	0.24
(Zhang et al., 2019a)	WideResNet-34-10	52.79	53.46	53.08	**54.15**	1.07
(Zhang et al., 2019b)	WideResNet-34-10	46.44	46.75	46.65	**47.18**	0.53
(Zhang et al., 2020)	WideResNet-34-10	45.54	45.78	45.68	**46.12**	0.44
(Chen et al., 2021)	WideResNet-34-10	47.33	47.47	47.58	**48.00**	0.42
(Sridhar et al., 2021)	WideResNet-34-15	38.75	38.93	38.90	**39.15**	0.25
(Cui et al., 2021)	WideResNet-34-20	45.63	45.91	45.88	**46.23**	0.35
(Gowal et al., 2020)	WideResNet-34-20	42.55	42.72	42.65	**42.86**	0.21
(Pang et al., 2020)	WideResNet-34-20	44.54	44.96	44.75	**45.33**	0.58
(Rice et al., 2020)	WideResNet-34-20	44.69	45.25	44.92	**45.69**	0.77
(Huang et al., 2021)	WideResNet-34-R	36.12	36.34	36.27	**36.76**	0.49
(Huang et al., 2021)	WideResNet-34-R	37.10	37.27	37.33	**37.79**	0.46
(Gowal et al., 2020)	WideResNet-70-16	33.24	33.49	33.42	**33.70**	0.28
(Gowal et al., 2020)	WideResNet-70-16	41.95	42.20	42.12	**42.45**	0.33
(Gowal et al., 2021)	WideResNet-70-16	32.25	32.71	32.57	**33.04**	0.47
(Rebuffi et al., 2021)	WideResNet-70-16	32.28	32.44	32.46	**32.75**	0.29
(Rebuffi et al., 2021)	WideResNet-70-16	34.76	34.95	35.04	**35.27**	0.23
ImageNet ($\varepsilon = 4/255$)						
(Salman et al., 2020)	ResNet-18	72.80	73.34	73.00	**73.72**	0.72
(Salman et al., 2020)	ResNet-50	62.72	63.06	62.86	**63.70**	0.84
(Wong et al., 2020)	ResNet-50	71.58	71.64	71.70	**71.94**	0.24
(Engstrom et al., 2019)	ResNet-50	67.74	68.08	67.86	**68.60**	0.74
(Salman et al., 2020)	WideResNet-50-2	58.88	59.36	58.96	**59.92**	0.96
Summary						
the number of **bold models**		0	0	1	63	
the number of underlined models		1	49	14	0	

Table 4. The average execution time of APGD and ACG. We measure the execution time 5 times with Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz and NVIDIA GeForce RTX 3090 GPUs.

(Ding et al., 2020)	APGD(1)	ACG(1)	APGD(5)	ACG(5)	CPU	RAM	GPU
ASR	48.73	53.40	49.67	55.77	Intel(R) Xeon(R)	786GB	NVIDIA
time	6m45.26s	6m56.78s	22m5.88s	21m15.67s	Gold 6240R		GeForce
ratio	0.97	1	3.18	3.06	×2		RTX 3090×4

Table 5. Average percentage of successful attacks and different CTC. "Success(xx.xx%)" indicates the percentage of images with successful attacks among all attacked images, and "Failure(xx.xx%)" indicates that for images with failed attacks. The "Different CTC" column shows the percentage of the images in which ACG attacks have a different CTC from APGD in the corresponding row.

APGD	ACG	Different CTC
Success(42.33%)	Success(42.27%)	6.18%
	Failure(0.06%)	11.96%
Failure(57.67%)	Success(0.68%)	**96.14%**
	Failure(56.99%)	6.05%

the bottom of Figure 3. APGD exhibited a higher ratio of projection in the distance traveled than ACG. That is, in the update of APGD after moving $\nabla f(\boldsymbol{x}^{(k)})$, the projection returns it to the vicinity of the original point $\boldsymbol{x}^{(k-1)}$. Therefore, we can see that ACG moves more than APGD owing to the introduction of the conjugate direction.

5. Search Diversity Analysis of ACG

The experimental results described in Section 4 show that ACG exhibited a higher ASR than APGD. In addition, the Euclidean distance between the successive two points of ACG was larger than that of APGD (see Section 4.4), which suggests that the search of ACG was more diverse. In this section, we verify this hypothesis. While a search is intensified when the distance between successive two points is small (see the white points in Figure 1), the search is not always diversified even if the successive two points are distant (see bottom of Figure 4). We then regard the search as intensified (diversified) when the search points (do not) form clusters and propose an index that measures the degree of diversification by utilizing a global clustering coefficient.

5.1. Definition of Diversity Index

The *global clustering coefficient* (Kemper, 2010) represents the strength of the connections between the nodes of a graph, and is often used in complex network analysis (Tabak et al., 2014; Said et al., 2018). To apply the global clustering coefficient to our analysis, we consider a graph whose nodes are search points.

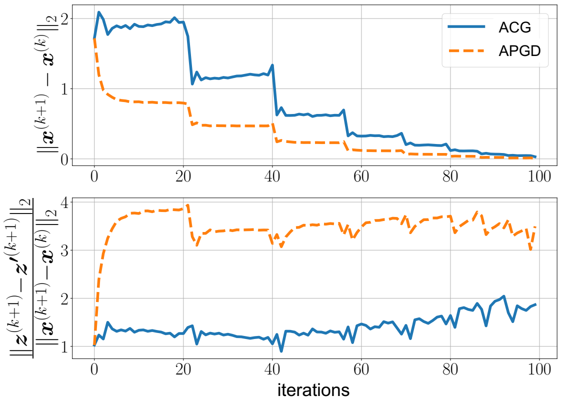


Figure 3. Effects of the projection on the feasible region and moving distances of APGD and ACG. The upper part shows the 2-norm of two successive points. The lower part shows the ratio of the 2-norm of the two consecutive points to the 2-norm of the projection onto the feasible region. Of note, $\boldsymbol{z}^{(k+1)} = \boldsymbol{x}^{(k+1)}$ for ACG.

Given the set of search points X, we define a graph $G_X(\theta) := (X, E(\theta))$, where $E(\theta) = \{(\boldsymbol{v}, \boldsymbol{w}) \in X \times X \mid \|\boldsymbol{v} - \boldsymbol{w}\|_2 \leq \theta\}$. Let $C(G)$ be the global clustering coefficient of a graph G, and let $h(\theta; X)$ be $1 - C(G_X(\theta))$. Note that $0 \leq C(G_X(\theta)) \leq 1$ from the definition of the global clustering coefficient (see Section K.A). Of note, when $G_X(\theta)$ is disjoint union of some complete graphs, $C(G_X(\theta)) = 1$ and thus $h(\theta; X) = 0$. Similarly, $h(\theta; X)$ takes low value when $G_X(\theta)$ exhibits clusters. When the cluster structure of the search point is apparent, the cluster structure appears in $G_X(\theta)$ even for a small θ. Therefore, we can quantify the cluster structure of the search points by the transition of h with the change of θ.

Hence, we define the *Diversity Index* (DI) as the average of $h(\theta; X)$ for θ to quantify the diversity of the search points, as given below.

$$\mathrm{DI}(X, M) := \frac{1}{M} \int_0^M h(\theta; X)\, d\theta,$$

where $M = \sup\{\|\boldsymbol{x} - \boldsymbol{y}\|_2 \mid \boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}\}$ is the size of the feasible region. Because we consider an adversarial attack

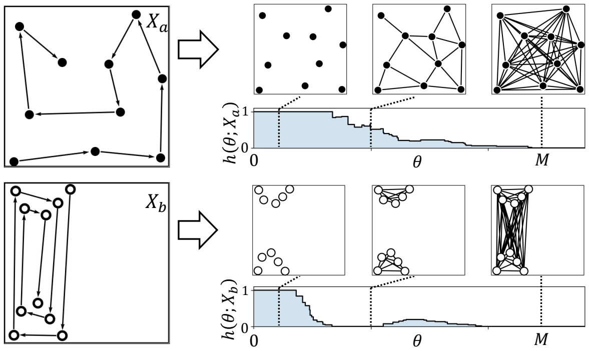


Figure 4. Example of the relationship between the diversity of search points and the global clustering coefficients. The white and black circles are the same as in Figure 1. The left side shows the movements of the search points X. The right side shows the transitions of $h(\theta; X)$, and three graphs for several thresholds θ.

with an l_∞-ball constraint, we obtain $M = \|\boldsymbol{u}-\boldsymbol{\ell}\|_2$. When X and M are obvious, we simply denote $\mathrm{DI}(X, M)$ as DI. From the definition of DI, we obtain $0 \leq \mathrm{DI} \leq 1$.

In Figure 4, examples of $h(\theta; X)$ are shown for two sets of search points, X_a (upper) and X_b (lower). Because X_b includes two clusters, whereas X_a does not, we regard X_a as an example of a diverse search and X_b as an intense search. On the right side of Figure 4, it may be observed that $h(\theta; X_a) \geq h(\theta; X_b)$ for most $\theta \in [0, M]$ and thus $\mathrm{DI}(X_a, M) > \mathrm{DI}(X_b, M)$, which reflects the diversity of the search points. In other words, $G_{X_b}(\theta)$ was more clustered than $G_{X_a}(\theta)$ for most θ.

As shown in Figure 4, the following relationship holds between DI and the diversity of the search points: when DI is small (large), the points in X (do not) form some clusters. That is, an intensive (diverse) search is conducted. Below, we use DI to analyze and discuss APGD and ACG.

5.2. Comparison on the behavior of DI

We considered the difference between the conjugate gradient and the gradient in the momentum direction using DI. In the following analyses, we use $X^{(k)} := \left\{\boldsymbol{x}^{(k-9)}, \cdots, \boldsymbol{x}^{(k)}\right\}$ as a set of search points at k-th iteration where $\boldsymbol{x}^{(i)}$ is the i-th search point. The diversity of the search and the transition of the search trends may be observed from DI calculated for the latest 10 search points. Figure 5 shows the transitions of DI in each algorithm for the models whose results exhibited the most significant differences in the ASR for each dataset. From Figure 5, it may be observed that the DIs during the search for ACG were larger than those obtained in the search by APGD. This suggests that ACG moves to a wide variety of different points and conducts more diverse searches. In particular, when the step size is large, ACG extensively explores the feasible region. By contrast, APGD exhibited a small DI value from the early iterations during the search. Section 4.4 shows that the movement distance of the search point accompanied with an update of APGD was shorter than that of ACG. In addition, the results in this section show that the movement distance in the latest 10 search points of APGD was limited in comparison to that of ACG.

These results demonstrate that ACG conducted a more diverse search than APGD, which is one possible reason for the difference of CTC and the higher ASR of ACG. We believe that utilizing DI to control the balance of diversification and intensification may be considered a promising approach to further improve ASR.

5.3. Comparison of APGD and ACG on the ability to diversification and intensification

In the previous section, we observed that ACG conducts a more diverse search than APGD. In this section, we investigate how the diverse search for ACG contributes to the recorded increase in ASR. We used the CIFAR-10 as the dataset and a WideResNet-28-4 model trained using the method proposed in (Ding et al., 2020), and CW loss as the objective function. APGD utilizes the gradient direction (GD) as the update direction. ACG and APGD utilize the same step-size selection strategy, which halves the step size from 2ε when either condition (I) or (II) is satisfied.

We use GD-to-CG (η) and CG-to-GD (η) to indicate that the update direction was changed at a step size of η from GD to CG and CG to GD, respectively. For example, GD-to-CG($\eta/4$) searches in GD when $\eta^{(k)} \geq \varepsilon/2$, and in CG when $\eta^{(k)} \leq \varepsilon/4$. In Figure 6, we observe that DI initially increased when switching from GD to CG, but did not increase when switching toward the end of the search. In addition, we found that the DI decreased when switching from CG to GD, indicating that the diversification decreased. By contrast, DI did not decrease in the latter half of the search, indicating that diversification did not significantly decrease.

Table 6 shows the ASR results for 100 iterations and 5 restarts when switching from GD to CG or CG to GD at each step size. It may be observed from the table entry of CG-to-GD, we can see that the ASR when using CG in the early stage was higher than that of ACG. In summary, the early stage of ACG performs a more diversified search than that of APGD, and the diversification in the early phase of ACG contributed to the higher ASR of ACG in comparison to that of APGD.

6. Conclusion

In this study, we have proposed ACG using conjugate directions inspired by the CG method. We have conducted extensive experiments to evaluate the performance of the

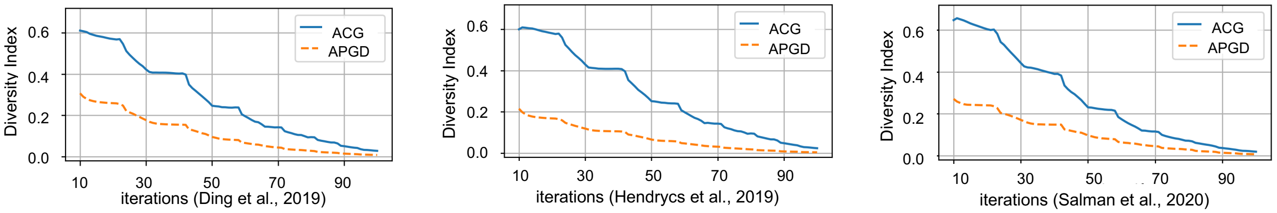


Figure 5. Transition of the average DI over 10,000 images using three models.

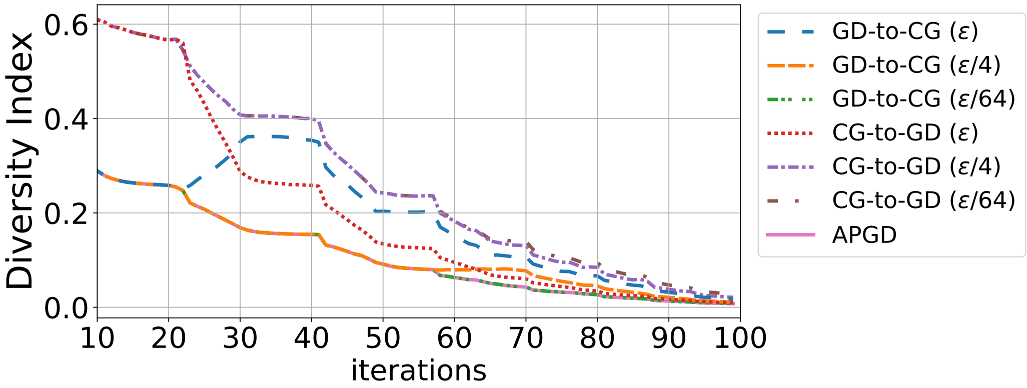


Figure 6. The transitions of DI of the switching update direction from GD and CG averaged over 10,000 images.

Table 6. ASR when switching from GD to CG (GD-to-CG) or CG to GD (CG-to-GD) at the specific step size.

step size	ε	$\varepsilon/2$	$\varepsilon/4$	$\varepsilon/8$	$\varepsilon/16$	$\varepsilon/32$	$\varepsilon/64$	APGD	ACG
GD-to-CG	50.85	49.70	49.66	49.66	49.67	49.67	49.67	49.67	55.77
CG-to-GD	55.77	55.84	55.80	55.80	55.78	55.77	55.77		

proposed approach on a total of 64 models listed in RobustBench. We found that ACG significantly improved the ASR compared to APGD in all but one of the models. In particular, ACG with one restart outperformed the ASR of APGD with 5 restarts for many of the models. This indicates that ACG may be promising in that it achieves a high ASR with only deterministic operations. In addition, ACG frequently switched the attacked class during the search and succeeded in attacking owing to its attacked class being different from APGD, whereas APGD rarely switched the attacked class. This result implies that varying the attacked class contributed significantly to the improvement in terms of ASR exhibited by ACG.

To analyze the difference in the search performance between ACG and APGD, we have proposed the Diversity Index (DI), which measures the degree of diversification of a search. DI is calculated based on the global clustering coefficients on a graph the nodes of which are the latest among several search points. A higher DI indicates that corresponding search points are sparsely distributed. According to the analyses using DI, one of the reasons for the higher ASR of ACG is that the CG direction exhibited greater diversification, particularly in the early phase of the search. DI may also be considered a valuable metric in analyzing other attack algorithms. In the future, we expect that an algorithm designed to efficiently control diversification and intensification using DI would be effective.

Acknowledgement

This research project was supported by the Japan Science and Technology Agency (JST), the Core Research of Evolutionary Science and Technology (CREST), the Center of Innovation Science and Technology based Radical Innovation and Entrepreneurship Program (COI Program), JSPS KAKENHI Grant Number JP20H04.

References

Addepalli, S., Jain, S., Sriramanan, G., Khare, S., and Babu, R. V. Towards achieving adversarial robustness beyond perceptual limits. *ICML 2021 Workshop*, 2021.

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pp. 484–501. Springer, 2020.

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

Carmon, Y., Raghunathan, A., Schmidt, L., Liang, P., and Duchi, J. C. Unlabeled Data Improves Adversarial Robustness. *Advances in Neural Information Processing Systems*, 32, may 2019. ISSN 10495258.

Chen, E.-C. and Lee, C.-R. Ltd: Low temperature distillation for robust adversarial training. *CoRR*, abs/2111.02331, 11 2021.

Chen, J., Cheng, Y., Gan, Z., Gu, Q., and Liu, J. Efficient robust training via backward smoothing, 2021.

Chen, T., Liu, S., Chang, S., Cheng, Y., Amini, L., and Wang, Z. Adversarial Robustness: From Self-Supervised Pre-Training to Fine-Tuning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 696–705, mar 2020. ISSN 10636919. doi: 10.1109/CVPR42600.2020.00078.

Cheng, S., Shi, Y., Qin, Q., Zhang, Q., and Bai, R. Population Diversity Maintenance In Brain Storm Optimization Algorithm. *Journal of Artificial Intelligence and Soft Computing Research*, 4(2):83–97, 2014. ISSN 2083-2567. doi: 10.1515/jaiscr-2015-0001.

ČrepinšekMatej, LiuShih-Hsi, and MernikMarjan. Exploration and exploitation in evolutionary algorithms. *ACM Computing Surveys (CSUR)*, 45(3):33, jul 2013. doi: 10.1145/2480741.2480752.

Croce, F. and Hein, M. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pp. 2196–2205. PMLR, 2020a.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pp. 2206–2216. PMLR, 2020b.

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. Robustbench: a standardized adversarial robustness benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Cui, J., Liu, S., Wang, L., and Jia, J. Learnable boundary guided adversarial training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15721–15730, October 2021.

Ding, G. W., Sharma, Y., Lui, K. Y. C., and Huang, R. MMA training: Direct input space margin maximization through adversarial training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

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

Engstrom, L., Ilyas, A., Salman, H., Santurkar, S., and Tsipras, D. Robustness (python library), 2019.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Gowal, S., Uesato, J., Qin, C., Huang, P., Mann, T. A., and Kohli, P. An alternative surrogate loss for pgd-based adversarial testing. *CoRR*, abs/1910.09338, 2019. URL `http://arxiv.org/abs/1910.09338`.

Gowal, S., Qin, C., Uesato, J., Mann, T., and Kohli, P. Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples. *CoRR*, abs/2010.03593, oct 2020.

Gowal, S., Rebuffi, S.-A., Wiles, O., Stimberg, F., Calian, D. A., and Mann, T. Improving robustness using generated data. In *NeurIPS*, 2021.

Hager, W. W. W. and Zhang, H. A Survey of Nonlinear Conjugate Gradient Methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

Hendrycks, D., Lee, K., and Mazeika, M. Using pre-training can improve model robustness and uncertainty. *Proceedings of the International Conference on Machine Learning*, pp. 2712–2721, 5 2019. ISSN 2640-3498.

Hestenes, M. and Stiefel, E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409, 1952. ISSN 0091-0635. doi: 10.6028/jres.049.044.

Huang, H., Wang, Y., Erfani, S. M., Gu, Q., Bailey, J., and Ma, X. Exploring architectural ingredients of adversarially robust deep neural networks. In *NeurIPS*, 2021.

Huang, L., Zhang, C., and Zhang, H. Self-Adaptive Training: beyond Empirical Risk Minimization. *Advances in Neural Information Processing Systems*, 2020-December, feb 2020. ISSN 10495258.

Kemper, A. Valuation of Network Effects in Software Markets. 2010. doi: 10.1007/978-3-7908-2367-7.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *ICLR Workshop*, 2017.

Lin, J., Song, C., He, K., Wang, L., and Hopcroft, J. E. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *International Conference on Learning Representations*, 2020.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Morales-Castañeda, B., Zaldívar, D., Cuevas, E., Fausto, F., and Rodríguez, A. A better balance in metaheuristic algorithms: Does it exist? *Swarm and Evolutionary Computation*, 54:100671, may 2020. ISSN 22106502. doi: 10.1016/j.swevo.2020.100671.

Pang, T., Yang, X., Dong, Y., Xu, K., Zhu, J., and Su, H. Boosting Adversarial Training with Hypersphere Embedding. *Advances in Neural Information Processing Systems*, 2020-December, feb 2020. ISSN 10495258.

Rade, R. and Moosavi-Dezfooli, S.-M. Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., and Mann, T. A. Fixing data augmentation to improve adversarial robustness. *CoRR*, abs/2103.01946, 2021.

Rice, L., Wong, E., and Kolter, Z. Overfitting in adversarially robust deep learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8093–8104. PMLR, 13–18 Jul 2020.

Said, A., Abbasi, R. A., Maqbool, O., Daud, A., and Aljohani, N. R. CC-GA: A clustering coefficient based genetic algorithm for detecting communities in social networks. *Applied Soft Computing*, 63:59–70, feb 2018. ISSN 1568-4946. doi: 10.1016/J.ASOC.2017.11.014.

Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., and Madry, A. Do adversarially robust imagenet models transfer better? *CoRR*, abs/2007.08489, 2020.

Sehwag, V., Wang, S., Mittal, P., and Jana, S. HYDRA: Pruning Adversarially Robust Neural Networks. *Advances in Neural Information Processing Systems*, 2020-December, feb 2020. ISSN 10495258.

Sehwag, V., Mahloujifar, S., Handina, T., Dai, S., Xiang, C., Chiang, M., and Mittal, P. Robust learning meets generative models: Can proxy distributions improve adversarial robustness?, 2021.

Sitawarin, C., Chakraborty, S., and Wagner, D. Sat: Improving adversarial training via curriculum-based loss smoothing. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, AISec '21, pp. 25–36, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450386579. doi: 10.1145/3474369.3486878.

Sridhar, K., Sokolsky, O., Lee, I., and Weimer, J. Improving Neural Network Robustness via Persistency of Excitation. jun 2021.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Tabak, B. M., Takami, M., Rocha, J. M., Cajueiro, D. O., and Souza, S. R. Directed clustering coefficient as a measure of systemic risk in complex banking networks. *Physica A: Statistical Mechanics and its Applications*, 394:211–216, jan 2014. ISSN 0378-4371. doi: 10.1016/J.PHYSA.2013.09.010.

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2020.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Wu, D., Xia, S. T., and Wang, Y. Adversarial Weight Perturbation Helps Robust Generalization. *Advances in Neural Information Processing Systems*, 2020-December, apr 2020. ISSN 10495258.

Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z., and Yuille, A. L. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Yao, Z., Gholami, A., Xu, P., Keutzer, K., and Mahoney, M. W. Trust region based adversarial attack on neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Zhang, D., Zhang, T., Lu, Y., Zhu, Z., and Dong, B. You Only Propagate Once: Accelerating Adversarial Training via Maximal Principle. *Advances in Neural Information Processing Systems*, 32, may 2019a. ISSN 10495258.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. Theoretically Principled Trade-off between Robustness and Accuracy. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:12907–12929, jan 2019b.

Zhang, J., Xu, X., Han, B., Niu, G., Cui, L., Sugiyama, M., and Kankanhalli, M. Attacks which do not kill training make adversarial learning stronger. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11278–11287. PMLR, 13–18 Jul 2020.

Zhang, J., Zhu, J., Niu, G., Han, B., Sugiyama, M., and Kankanhalli, M. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations*, 2021.

Zheng, T., Chen, C., and Ren, K. Distributionally adversarial attack. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):2253–2260, Jul. 2019. doi: 10.1609/aaai.v33i01.33012253.

Appendix

In this appendix, we provide the following additional information regarding the experimental results and the behavior of the proposed approach during the search for ACG.

- An example of desirable search behavior. (Appendix B)
- Transitions of the best objective values and the effect of increasing the iterations. (Appendix C)
- The effect of random restarts on the performance of ACG. (Appendix D)
- A discussion of the diversification and intensification of APGD search. (Appendix E)
- The effect of the objective function on the performance of ACG. (Appendix F)
- The effect of the operation conducted to render β nonnegative, which is known to perform better for a general nonlinear optimization. (Appendix G)
- Specifications of the computational environments used in our experiments. (Appendix H)
- Differences in the ASR among different formulas used to calculate β. (Appendix I)
- Differences in the search behavior of APGD and ACG for the model in which the ASR of ACG was lower than that of APGD and the other models. (Appendix J)
- A definition of DI on arbitrary bounded distance spaces. (Appendix K)
- The relationship between distributions of the point clouds and DI. (Appendix L)

A. Related works

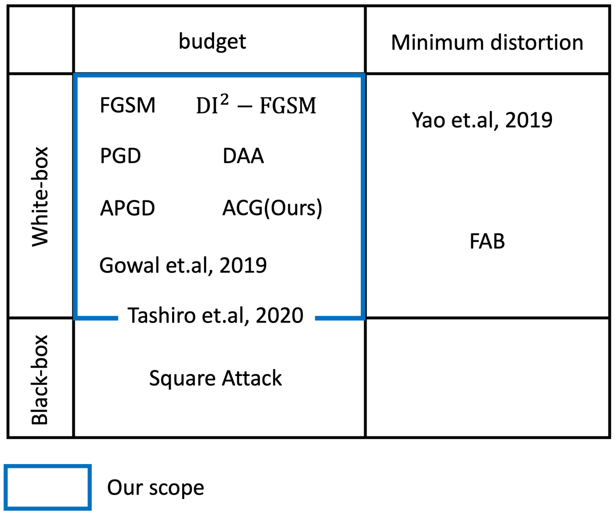
We classify some previous works to clarify the position of the proposed method. Figure 7 shows an overview.

"Budget" is a type of adversarial attacks using the formulation described in Section 2. (Gowal et al., 2019; Xie et al., 2019; Zheng et al., 2019) are examples of this type of attacks.

"Minimum distortion" is another type of adversarial attack which generates minimally distorted adversarial examples by minimizing the norm of adversarial perturbation. Formally, this problem is described as follows.

$$\min_{x\in\mathcal{D}} d(x_{\text{orig}}, x) \;\text{ s.t. }\; \operatorname*{arg\,max}_{k=1,\dots K}\; g_k(x) \neq c$$

. (Yao et al., 2019; Croce & Hein, 2020a) are examples of this type of attacks.



	budget	Minimum distortion
White-box	FGSM DI^2 − FGSM PGD DAA APGD ACG(Ours) Gowal et.al, 2019 Tashiro et.al, 2020	Yao et.al, 2019 FAB
Black-box	Square Attack	

Our scope

Figure 7. Classification of previous works.

In the case of white-box attacks, we can assume that all information about the model to be attacked is known the attacker, including its weights and gradients.

In the case of black-box attacks, only use the output of target models can be used. The proposed method, ACG, is a white-box attack that uses a budget formulation.

B. Examples of Diversification and Intensification

In this section, we detail the balance between diversification and intensification that we aim to realize.

Most white-box adversarial attacks are formulated as optimization problems where the objective function is nonlinear, nonconvex, and multimodal. We observe the diversification and intensification of the search to find the optimal solution by using the following multimodal function.

$$\begin{aligned} f(x,y) = &-10\exp\left(-0.2\sqrt{\frac{1}{2}(x^4+y^2)}\right) \\ &+\exp\left(\frac{1}{2}\left(\cos(2\pi x)+\cos(2\pi y)\right)\right). \end{aligned} \tag{12}$$

Figure 8 shows examples of an intensified search, a diversified search, and an appropriate search exhibiting the proper balance of diversification and intensification. Six local solutions may be observed from the contour lines of the function in the figure, of which the local solution at position "E" is the global optimum.

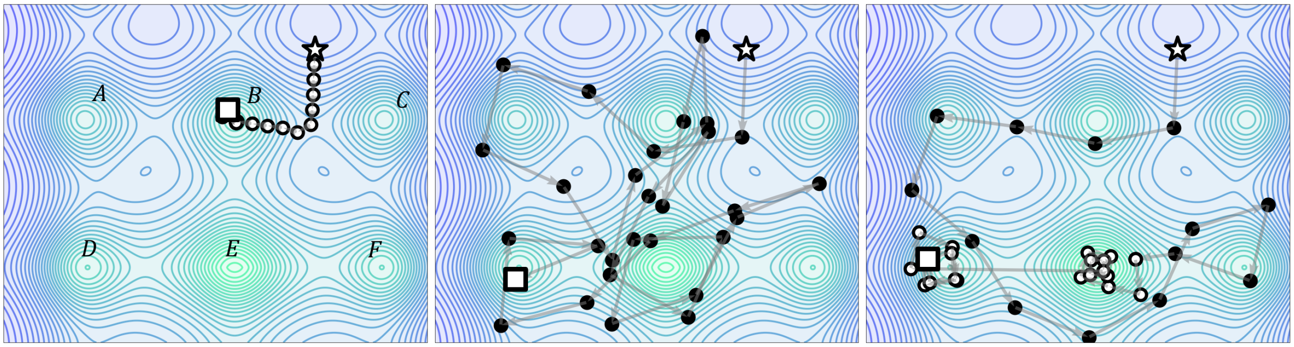


Figure 8. Examples of searches for multimodal functions. The initial search point is shown by the white star, and the search ended at a white square. The circles represent the search points. The black circle indicates that the search was diversified, and the white circle indicates an intense search. "A" to "F" on the left figure refers to the local optima, and "E" shows the global optima. The examples on the left and center are those of failed searches that were not able to find the global optima owing to a lack of diversification or intensification, respectively. The example on the right shows a search that has successfully found the global optima owing to an appropriate balance of diversification and intensification, which is our aim in the present work.

In the figure on the left, the search (white circles) was intensified in the local solution near the initial point. In the middle figure, the search (black circles) was diversified even when reaching the neighborhood of the local solution. Therefore, it appears to be impossible to reach an optimal solution if the search is excessively intensified or diversified. In the figure on the right, it may be observed that the search changed from a focus on diversification to a focus on intensification and that the search reached the neighbor of the global optima through a diversified search (black circles) and converged to the optimal solution through an intensive search (white circles). Thus, for multimodal functions, a balance between diversification and intensification of the search is necessary when searching an entire feasible region to finding an optimum. The right side of Figure 6 shows our intended search, in which the balance between diversification and intensification is properly controlled.

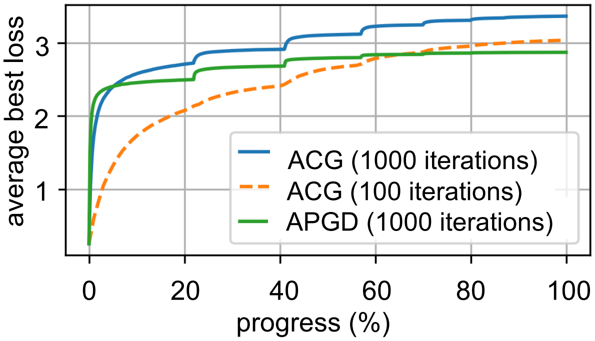


Figure 9. Transitions of the best objective values averaged over 10,000 images. Here, "progress" means the percentage of the iterations to the total iterations.

C. ACG vs. APGD: Comparison of the best objective values

From Figure 9, we predicted that ACG (100 iterations) could still improve the objective value because it improves the best objective value to a relatively large degree even at the end of the search (dashed line, orange). We then investigated the search performance of ACG in terms of the best objective values. Figure 9 shows the transitions of the best loss of ACG after 100 iterations, and those of APGD and ACG after 1000 iterations. The x-axis represents the percentage of the iterations to N_{iter}. The best loss at iteration k is defined as $\text{best loss}^{(k)} = \max_{i \leq k} L\left(g(\boldsymbol{x}^{(i)}), c\right)$. As expected, ACG (1000 iterations) exhibited higher objective values than ACG (100 iterations), whereas ACG (1000 iterations) improved on the best objective values even at the end of the search. By contrast, the improvement of the best objective values of APGD (1000 iterations) was small at the end of the attack compared to ACG (100 iterations, 1000 iterations). To summarize these results, ACG may be considered a more promising algorithm than APGD because, in contrast to APGD, it can significantly improve on its best loss even at the end of the search.

D. Effects of random restarts for the performance of ACG

In this section, we compare the results of APGD with 5 restarts for 100 iterations, ACG with 5 restarts for 100 iterations, and ACG with 1 restart for 500 iterations. Tables 9 to 11 show that ACG(5), which is randomized by the restarts, and ACG-500iter, which is deterministic, achieved almost the same ASR. This result suggests that the search diversity of ACG does not depend on the random sampling of the initial points, but that the update direction itself has the property of a diverse search compared to APGD.

E. Diversification and Intensification of APGD Search

Herein, we discuss the diversification and intensification of APGD. APGD is a PGD-based adversarial attack that achieves a higher ASR than previous SOTA methods by gradually reducing the step size, as shown in Section 3.2. APGD introduced this step size reduction to gradually switch from exploration to exploitation. Note that "exploration" and "exploitation" are synonymous with "diversification" and "intensification" as we used the terms in this work. However, to the best of out knowledge, whether APGD switches from diversification to intensification has not been verified.

Figure 5 in Section 5.2 show that the DIs of APGD gradually decreased during the search for 100 iterations. These results indicate that APGD can switch from diversification to intensification in terms of DI, and thus the motivation for the step size selection was achieved. However, the DI value of APGD was smaller than that of ACG. We discuss the reason for this based on the results described in Section 4.4. From the image at the top of Figure 3, it may be observed that the projected distance of the PGD-based updated search point to the feasible region was larger than that of the CG-based search point. This means that with APGD, the search points are close to the boundary of the region and are updated toward the outside of the boundary. Because the PGD-based method tends to update toward the same local optimum, doing so is natural when there is a local optimum outside the feasible region. As a result, a small distance was induced between two successive search points (as shown at the bottom of Figure 3), dense search points, and small DI values.

Because ACG, which exhibited a more variable DI, delivers a better performance than APGD, we expect that higher-performing adversarial attacks will be developed in future research via sophisticated control of the DI transition.

F. DLR loss

In this section, we compare the ASR under the experimental setup of Section 4; however, we used the DLR loss proposed in (Croce & Hein, 2020b) as the objective function instead of the CW loss in Section 4. Tables 12 and 13 show that the ASR of ACG was higher than that of APGD for all 64 models when the objective function was the DLR loss.

Table 7. The average percentage of the iterations in which β_{HS}^{+}=0 among all 495 iterations. The CIFAR-10 dataset was used for this experiment, and the diameter of the feasible region ε was $8/255$.

paper	**Ratio** of β_{HS}^{+} is 0
(Ding et al., 2020)	40.59%
(Carmon et al., 2019)	30.41%
(Rebuffi et al., 2021)	38.57%

G. Analysis of effect of the assumption of $\beta_{\mathrm{HS}} \geq 0$

When discussing the convergence of the conjugate gradient method, it has occasionally been assumed that $\beta_{\mathrm{HS}} \geq 0$ (Hager & Zhang, 2006). In addition, some studies have suggested that it is better to assume $\beta_{\mathrm{H}S} \geq 0$ is preferable in practice. In this section, we investigate the effect of assuming $\beta_{\mathrm{H}S} \geq 0$ by comparing the behavior of ACG without any assumptions on $\beta_{\mathrm{H}S}$. To render $\beta_{\mathrm{H}S}$ be nonnegative, we can obtain the following.

$$\beta_{\mathrm{H}S}^{+} := \max(\beta_{\mathrm{H}S}, 0). \tag{13}$$

We call ACG using $\beta_{\mathrm{H}S}^{+}$ as ACG^{+}, as determined using (13). Table 8 show that the ASR of ACG^{+} was equal to or less than that of APGD, suggesting a decrease in the search performance of ACG^{+}. In addition, the percentage of iterations where $\beta_{\mathrm{HS}}^{+} = 0$ was 30% to 40% of all 495 iterations, excluding iteration 0, where the steepest descent occured(see Table 7). That is, ACG^{+} updated in the same way as APGD without a momentum update method once every three iterations. In addition, from Figure 10, it may be observed that ACG^{+} diversified the search less than ACG. These results show that operations that make β_{HS} nonnegative, such as (13), are unsuitable for this problem. Applying the operation to ACG limits the diversification performance, which is one of the strengths of ACG.

H. Experiment Environments

The computational environments for our experiments, such as the CPU and GPU specifications and RAM capacity, are provided in Table 15. More information is also provided in the source codes.

I. Experimental results of the evaluation of the representative β formulas

In this section, among the formulas used to determine β proposed in prior works, we verified the effectiveness of seven representative formulas are verified through secondary experiments. Based on these experiments, we chose the formula for our approach. The formulas we verified in

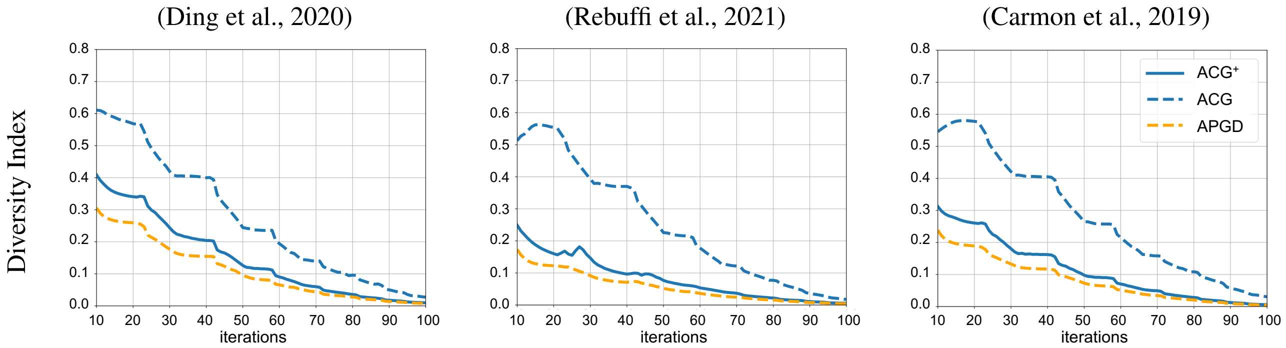


Figure 10. Comparison of DI of ACG$^+$, ACG, and APGD. The CIFAR-10 dataset was used for this comparison.

Table 8. ASR of ACG$^+$ and APGD: The dataset is CIFAR-10, and $\varepsilon = \frac{8}{255}$.

CIFAR-10 ($\varepsilon = 8/255$)		Attack Success Rate				
paper	**Architecture**	**APGD(1)**	**ACG$^+$(1)**	**APGD(5)**	**ACG$^+$(5)**	**diff**
(Rebuffi et al., 2021)	PreActResNet-18	42.73	42.71	**42.91**	42.88	-0.03
(Carmon et al., 2019)	WideResNet-28-10	39.38	39.28	**39.59**	39.51	-0.08
(Ding et al., 2020)	WideResNet-28-4	48.73	49.66	49.67	**50.68**	1.01

this experiment are given as follows.

- FR: $\beta_{FR}^{(k)} = \dfrac{\|\nabla f(\boldsymbol{x}^{(k)})\|_2^2}{\|f(\boldsymbol{x}^{(k-1)})\|_2^2}$.
- PR: $\beta_{PR}^{(k)} = \dfrac{\langle \nabla f(\boldsymbol{x}^{(k)}), \boldsymbol{y}^{(k-1)}\rangle}{\|f(\boldsymbol{x}^{(k-1)})\|_2^2}$.
- HS: $\beta_{HS}^{(k)} = -\dfrac{\langle \nabla f(\boldsymbol{x}^{(k)}), \boldsymbol{y}^{(k-1)}\rangle}{\langle \boldsymbol{s}^{(k-1)}, \boldsymbol{y}^{(k-1)}\rangle}$.
- DY: $\beta_{DY}^{(k)} = \dfrac{\|\nabla f(\boldsymbol{x}^{(k)})\|_2^2}{\langle \boldsymbol{s}^{(k-1)}, \boldsymbol{y}^{(k-1)}\rangle}$.
- HZ: $\beta_{HZ}^{(k)} = \dfrac{\left\langle \boldsymbol{y}^{(k-1)} - \dfrac{2\boldsymbol{s}^{(k-1)}\|\boldsymbol{y}^{(k-1)}\|_2^2}{\langle \boldsymbol{s}^{(k-1)}, \boldsymbol{y}^{(k-1)}\rangle}, \nabla f(\boldsymbol{x}^{(k)})\right\rangle}{\langle \boldsymbol{s}^{(k-1)}, \boldsymbol{y}^{(k-1)}\rangle}$.
- DL: $\beta_{DL}^{(k)} = -\dfrac{\left\langle \boldsymbol{y}^{(k-1)} - t\boldsymbol{s}^{(k-1)}, \nabla f(\boldsymbol{x}^{(k)})\right\rangle}{\langle \boldsymbol{s}^{(k-1)}, \boldsymbol{y}^{(k-1)}\rangle}$, $t \geq 0$.
- LS: $\beta_{LS}^{(k)} = -\dfrac{\langle \nabla f(\boldsymbol{x}^{(k)}), \boldsymbol{y}^{(k-1)}\rangle}{\left\langle \boldsymbol{s}^{(k-1)}, \nabla f(\boldsymbol{x}^{(k-1)})\right\rangle}$.

We conducted small experiments on only models described in (Ding et al., 2020; Carmon et al., 2019; Rebuffi et al., 2021) using the same experimental setup described in Section 4. The CIFAR-10 dataset was used, and the results are presented in Table 14. Table 14 shows that the ASR of the formulas whose enumerator is $\langle \nabla f(\boldsymbol{x}), \boldsymbol{y}\rangle$ (PR, HS, LS) were higher than those of the formulas whose numerator is $\|\nabla f(\boldsymbol{x})\|_2^2$ (FR, DY), and that the ASR of the HS formula was the highest.

J. Analysis of WideResNet-34-10 (Addepalli et al., 2021)

We analyzed the search behavior of ACG on a WideResNet-34-10 model trained by the method proposed in (Addepalli et al., 2021) using CIFAR-100, the only model in which the ASR of ACG was lower than that of APGD in the experiment using the CW loss (see Table 2). From Figure 11, it may be observed that the DI of ACG of this model was lower than that of ACG on the other models. This means that the search for ACG in this model tended to be more intensified than the attacks on the other two models (Sitawarin et al., 2021; Wu et al., 2020). By contrast, the DI of APGD on this model was higher than that of APGD in the other models, indicating that it tended to be more diversified. These results suggest that the reason why APGD was superior to ACG for this model was because intensification was required to achieve a higher ASR. However, the models requiring intensification for more effective attacks were rare because ACG exhibited a higher ASR than APGD for most of the models in Tables 2 to 3.

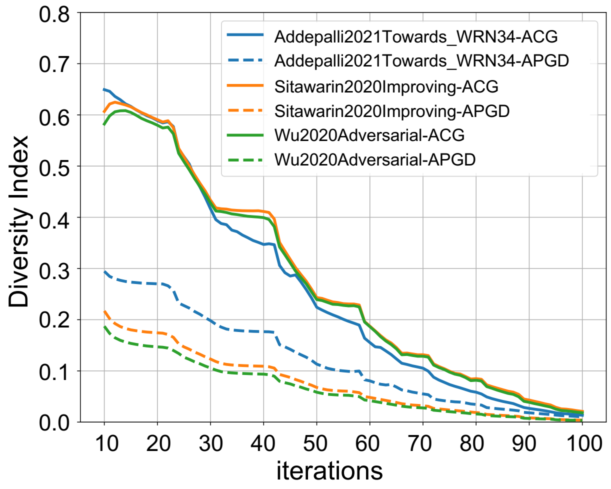


Figure 11. DI of ACG and APGD on three models (Addepalli et al., 2021; Wu et al., 2020; Sitawarin et al., 2021)

K. Generalization of DI

In this section, we generalize the DI defined in Section 5.1 to arbitrary bounded distance spaces. First, we describe the definition of local and global clustering coefficients for undirected graphs and then use the global clustering coefficient to define the DI for arbitrarily bounded distance spaces. Then, using the global clustering coefficient, we define the DI for arbitrary bounded distance spaces.

K.A. Definition of the local clustering coefficient (Kemper, 2010)

Let $G = (V, E)$ be an undirected graph. Let the set of neighbor nodes of node $\mathfrak{v}_i \in V$ be $\mathcal{N}_i := \{\mathfrak{v}_j \in V \mid e_{ij} \in E\}$. Then the local clustering coefficient $C_{\mathfrak{v}_i}(G)$ at node $\mathfrak{v}_i$ of graph G is defined as follows.

$$C_{\mathfrak{v}_i}(G) := \frac{2\#\{e_{jk} \in E \mid e_{ij} \in E \wedge e_{ik} \in E\}}{\#\mathcal{N}_i \times (\#\mathcal{N}_i - 1)} \tag{14}$$

According to this definition, $C_{\mathfrak{v}_i}(G) \in [0, 1]$.

K.B. Definition of the global clustering coefficient (Kemper, 2010)

Let $G = (V, E)$ be an undirected graph, and the global clustering coefficient $C(G)$ of the graph G be defined using the equation (14) as follows.

$$C(G) := \frac{1}{\#V} \sum_{\mathfrak{v} \in V} C_{\mathfrak{v}}(G) \tag{15}$$

In the same manner as $C_{\mathfrak{v}_i}(G)$, $C(G)$ also satisfies $0 \leq C(G) \leq 1$.

K.C. Definition of DI on the general bounded metric spaces

Let (U, d) be a bounded distance space, $\mathcal{V} \subset U$ be its finite subset, and $\theta \in \mathbb{R}$ be the graph of $G_{\mathcal{V}}(\theta) := (\mathcal{V}, E(\theta))$, where $E(\theta) = \{(v, w) \in \mathcal{V} \times \mathcal{V} \mid d(v, w) \leq \theta\}$. In this case, as $M = \sup_{v,w \in U} d(v, w)$, DI is defined as follows.

$$\mathrm{DI}\,(\mathcal{V}, M) := 1 - \frac{1}{M} \int_0^M C(G_{\mathcal{V}}(\theta)) d\theta \tag{16}$$

L. Examples of DI

Figure 12 shows an example of the calculation of DI for point clouds with different distributions. The first row of Figure 12 shows an example of a point cloud and its DI in which the point clouds form a single cluster. The second row shows an example of a point cloud and its DI , in which the point clouds form approximately three to four clusters. The third row shows an example of a point cloud and its DI where most points are distributed on the boundary. The final example is diverse in that it shows a diversity search performed on the boundary. These examples show that DI takes a small value when the point cloud is dense or when the clusters are formed, and DI takes a relatively large value when there are no clusters the elements of which number greater than 2.

M. Evaluating the performance of ACG combined with Auto Attack

We did not compare the ASR of Auto Attack(AA) and ACG directly because we focused on generating many adversarial examples quickly. AA comprises of four different algorithms for adversarial attacks, and it takes much longer than ACG. We considered that evaluating ACG as a component of AA would be more suitable; we therefore constructed AA(ACG-CE) using ACG with untargeted cross-entropy loss instead of APGD with the same loss. From the numerical results shown in Table 16 below, it may be observed that ACG is also a useful as a component of AA. We expect that future work along there lines will adopt an appropriate combination of existing algorithms, including ACG.

Table 9. The ASR of APGD with 5 restarts, ACG with 5 restarts, and ACG with 1 restart for 500 iterations. The CIFAR-10 dataset was used with $\varepsilon = \frac{8}{255}$. The highest ASR is in bold, and the second is underlined. APGD(N) refers to APGD with N-times the initial point selection. The meanings of the other columns are the same. *diff* is the difference between APGD(5) and $\max\{$ACG(5), ACG-500iter$\}$.

CIFAR-10 ($\varepsilon = 8/255$)		Attack Success Rate			
paper	**Architecture**	**APGD(5)**	**ACG(5)**	**ACG-500iter**	**diff**
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	42.46	42.65	**42.67**	0.21
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	41.65	**42.12**	42.07	0.47
(Rebuffi et al., 2021)	PreActResNet-18	42.91	43.15	**43.17**	0.26
(Andriushchenko et al., 2020)	PreActResNet-18	53.82	**54.90**	54.81	1.08
(Sehwag et al., 2021)	ResNet-18	43.91	**44.79**	44.53	0.88
(Chen et al., 2020)	ResNet-50	48.08	48.28	**48.29**	0.21
(Wong et al., 2020)	ResNet-50	54.26	**55.44**	55.34	1.18
(Engstrom et al., 2019)	ResNet-50	48.08	**49.25**	49.08	1.17
(Rebuffi et al., 2021)	WideResNet-106-16	34.71	**35.03**	34.98	0.32
(Carmon et al., 2019)	WideResNet-28-10	39.59	**40.03**	39.98	0.44
(Gowal et al., 2020)	WideResNet-28-10	36.45	36.90	**36.96**	0.51
(Hendrycks et al., 2019)	WideResNet-28-10	43.82	44.36	**44.37**	0.55
(Rade & Moosavi-Dezfooli, 2021)	WideResNet-28-10	38.64	**38.87**	38.79	0.23
(Rebuffi et al., 2021)	WideResNet-28-10	38.47	**38.80**	38.77	0.33
(Sehwag et al., 2020)	WideResNet-28-10	41.93	42.41	**42.48**	0.55
(Sridhar et al., 2021)	WideResNet-28-10	39.45	39.85	**39.88**	0.43
(Wang et al., 2020)	WideResNet-28-10	42.15	**42.57**	42.53	0.42
(Wu et al., 2020)	WideResNet-28-10	39.56	39.70	**39.72**	0.16
(Zhang et al., 2021)	WideResNet-28-10	39.98	**40.25**	40.24	0.27
(Ding et al., 2020)	WideResNet-28-4	49.67	**55.77**	55.32	6.10
(Cui et al., 2021)	WideResNet-34-10	46.41	46.90	**46.90**	0.49
(Huang et al., 2020)	WideResNet-34-10	46.19	**46.72**	46.67	0.53
(Rade & Moosavi-Dezfooli, 2021)	WideResNet-34-10	36.46	**36.83**	36.77	0.37
(Sehwag et al., 2021)	WideResNet-34-10	39.58	40.18	**40.18**	0.60
(Sitawarin et al., 2021)	WideResNet-34-10	47.23	48.02	**48.05**	0.82
(Wu et al., 2020)	WideResNet-34-10	43.36	43.60	**43.62**	0.26
(Zhang et al., 2019a)	WideResNet-34-10	53.08	54.15	**54.17**	1.09
(Zhang et al., 2019b)	WideResNet-34-10	46.65	47.18	**47.19**	0.54
(Zhang et al., 2020)	WideResNet-34-10	45.68	46.12	**46.13**	0.45
(Chen et al., 2021)	WideResNet-34-10	47.58	**48.00**	47.99	0.42
(Sridhar et al., 2021)	WideResNet-34-15	38.90	39.15	**39.22**	0.32
(Cui et al., 2021)	WideResNet-34-20	45.88	**46.23**	46.14	0.35
(Gowal et al., 2020)	WideResNet-34-20	42.65	42.86	**42.91**	0.26
(Pang et al., 2020)	WideResNet-34-20	44.75	45.33	**45.34**	0.59
(Rice et al., 2020)	WideResNet-34-20	44.92	45.69	**45.73**	0.81
(Huang et al., 2021)	WideResNet-34-R	37.33	37.79	**37.90**	0.57
(Huang et al., 2021)	WideResNet-34-R	36.27	36.76	**36.81**	0.54
(Gowal et al., 2020)	WideResNet-70-16	33.42	33.70	**33.82**	0.40
(Gowal et al., 2020)	WideResNet-70-16	42.12	**42.45**	42.40	0.33
(Gowal et al., 2021)	WideResNet-70-16	32.57	**33.04**	32.95	0.47
(Rebuffi et al., 2021)	WideResNet-70-16	35.04	**35.27**	35.19	0.23
(Rebuffi et al., 2021)	WideResNet-70-16	32.46	**32.75**	32.69	0.29

Table 10. The ASR of APGD with 5 restarts, ACG with 5 restarts, and ACG with 1 restart for 500 iterations. The CIFAR-100 dataset with $\varepsilon = \frac{8}{255}$ was used. The highest ASR is in bold, and the second is underlined. APGD(N) refers to APGD with N-times the initial point selection. The meanings of the other columns are the same. *diff* is the difference between APGD(5) and $\max\{$ACG(5), ACG-500iter$\}$.

CIFAR-100 ($\varepsilon = 8/255$)		Attack Success Rate			
paper	**Architecture**	**APGD(5)**	**ACG(5)**	**ACG-500iter**	**diff**
(Addepalli et al., 2021)	PreActResNet-18	72.25	**72.47**	72.36	0.22
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	70.55	**70.86**	70.77	0.31
(Rebuffi et al., 2021)	PreActResNet-18	70.93	**71.29**	71.21	0.36
(Rice et al., 2020)	PreActResNet-18	79.99	**80.63**	80.55	0.64
(Hendrycks et al., 2019)	WideResNet-28-10	69.50	**70.51**	70.44	1.01
(Rebuffi et al., 2021)	WideResNet-28-10	66.67	**67.27**	67.16	0.60
(Addepalli et al., 2021)	WideResNet-34-10	**68.74**	68.52	68.59	-0.15
(Chen & Lee, 2021)	WideResNet-34-10	68.36	**68.77**	68.68	0.41
(Cui et al., 2021)	WideResNet-34-10	69.87	**70.33**	70.32	0.46
(Cui et al., 2021)	WideResNet-34-10	72.15	72.56	**72.64**	0.49
(Sitawarin et al., 2021)	WideResNet-34-10	73.43	**74.27**	74.09	0.84
(Wu et al., 2020)	WideResNet-34-10	69.32	**70.11**	69.97	0.79
(Chen et al., 2021)	WideResNet-34-10	71.96	72.18	**72.19**	0.23
(Cui et al., 2021)	WideResNet-34-20	68.72	**69.13**	69.08	0.41
(Gowal et al., 2020)	WideResNet-70-16	61.55	**62.19**	62.15	0.64
(Gowal et al., 2020)	WideResNet-70-16	69.04	**69.43**	69.35	0.39
(Rebuffi et al., 2021)	WideResNet-70-16	64.17	**64.77**	64.61	0.60

Table 11. The ASR of APGD with 5 restarts, ACG with 5 restarts, and ACG with 1 restart for 500 iterations. The ImageNet dataset with $\varepsilon = \frac{4}{255}$ is used. The highest ASR is in bold, and the second is underlined. APGD(N) refers to APGD with N-times the initial point selection. The meanings of the other columns are the same. *diff* is the difference between APGD(5) and $\max\{$ACG(5), ACG-500iter$\}$.

ImageNet ($\varepsilon = 4/255$)		Attack Success Rate			
paper	**Architecture**	**APGD(5)**	**ACG(5)**	**ACG-500iter**	**diff**
(Salman et al., 2020)	ResNet-18	73.00	**73.72**	73.56	0.72
(Salman et al., 2020)	ResNet-50	62.86	**63.70**	63.54	0.84
(Wong et al., 2020)	ResNet-50	71.70	**71.94**	71.92	0.24
(Engstrom et al., 2019)	ResNet-50	67.86	**68.60**	68.58	0.74
(Salman et al., 2020)	WideResNet-50-2	58.96	**59.92**	59.82	0.96

Table 12. Comparison of the ASR of APGD and ACG with DLR loss as the objective function. Dataset: CIFAR10($\varepsilon = \frac{8}{255}$), ImageNet($\varepsilon = \frac{4}{255}$)

CIFAR-10 ($\varepsilon = 8/255$)			Attack Success Rate		
paper	**Architecture**	**clean acc**	**APGD(5)**	**ACG(5)**	**diff**
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	89.02	41.61	**42.13**	0.52
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	86.86	42.45	**42.72**	0.27
(Rebuffi et al., 2021)	PreActResNet-18	83.53	42.88	**43.22**	0.34
(Rice et al., 2020)	PreActResNet-18	85.34	44.25	**46.06**	1.81
(Andriushchenko et al., 2020)	PreActResNet-18	79.84	52.99	**55.38**	2.39
(Sehwag et al., 2021)	ResNet-18	84.59	43.38	**45.11**	1.73
(Chen et al., 2020)	ResNet-50	86.04	47.82	**48.35**	0.53
(Wong et al., 2020)	ResNet-50	83.34	53.25	**55.72**	2.47
(Engstrom et al., 2019)	ResNet-50	87.03	47.36	**49.82**	2.46
(Rebuffi et al., 2021)	WideResNet-106-16	88.50	34.67	**35.19**	0.52
(Carmon et al., 2019)	WideResNet-28-10	89.69	39.35	**40.14**	0.79
(Gowal et al., 2020)	WideResNet-28-10	89.48	36.29	**37.00**	0.71
(Hendrycks et al., 2019)	WideResNet-28-10	87.11	43.03	**44.75**	1.72
(Rade & Moosavi-Dezfooli, 2021)	WideResNet-28-10	88.16	38.62	**39.00**	0.38
(Rebuffi et al., 2021)	WideResNet-28-10	87.33	38.37	**39.12**	0.75
(Sehwag et al., 2020)	WideResNet-28-10	88.98	41.80	**42.56**	0.76
(Sridhar et al., 2021)	WideResNet-28-10	89.46	39.14	**40.06**	0.92
(Wang et al., 2020)	WideResNet-28-10	87.50	41.52	**43.08**	1.56
(Wu et al., 2020)	WideResNet-28-10	88.25	39.50	**39.86**	0.36
(Zhang et al., 2021)	WideResNet-28-10	89.36	39.79	**40.57**	0.78
(Ding et al., 2020)	WideResNet-28-4	84.36	49.70	**56.00**	6.30
(Cui et al., 2021)	WideResNet-34-10	88.22	44.28	**46.92**	2.64
(Huang et al., 2020)	WideResNet-34-10	83.48	45.76	**46.93**	1.17
(Rade & Moosavi-Dezfooli, 2021)	WideResNet-34-10	91.47	36.40	**36.93**	0.53
(Sehwag et al., 2021)	WideResNet-34-10	86.68	39.22	**40.61**	1.39
(Sitawarin et al., 2021)	WideResNet-34-10	86.84	46.79	**48.78**	1.99
(Wu et al., 2020)	WideResNet-34-10	85.36	43.36	**43.72**	0.36
(Zhang et al., 2019a)	WideResNet-34-10	87.20	52.59	**54.65**	2.06
(Zhang et al., 2019b)	WideResNet-34-10	84.92	46.51	**47.27**	0.76
(Zhang et al., 2020)	WideResNet-34-10	84.52	45.44	**46.26**	0.82
(Chen et al., 2021)	WideResNet-34-10	85.32	47.32	**48.24**	0.92
(Sridhar et al., 2021)	WideResNet-34-15	86.53	38.65	**39.27**	0.62
(Cui et al., 2021)	WideResNet-34-20	88.70	44.68	**46.39**	1.71
(Gowal et al., 2020)	WideResNet-34-20	85.64	42.57	**43.02**	0.45
(Pang et al., 2020)	WideResNet-34-20	85.14	43.98	**45.92**	1.94
(Huang et al., 2021)	WideResNet-34-R	90.56	36.91	**38.04**	1.13
(Huang et al., 2021)	WideResNet-34-R	91.23	35.91	**36.93**	1.02
(Gowal et al., 2020)	WideResNet-70-16	91.10	33.33	**33.91**	0.58
(Gowal et al., 2020)	WideResNet-70-16	85.29	42.04	**42.59**	0.55
(Gowal et al., 2021)	WideResNet-70-16	88.74	32.08	**33.45**	1.37
(Rebuffi et al., 2021)	WideResNet-70-16	88.54	35.02	**35.54**	0.52
(Rebuffi et al., 2021)	WideResNet-70-16	92.23	32.40	**33.13**	0.73
ImageNet ($\varepsilon = 4/255$)					
(Engstrom et al., 2019)	ResNet-50	62.56	67.36	**69.58**	2.22
(Salman et al., 2020)	ResNet-18	52.92	72.78	**74.34**	1.56
(Salman et al., 2020)	WideResNet-50-2	68.46	58.38	**60.90**	2.52
(Wong et al., 2020)	ResNet-50	55.62	71.38	**73.00**	1.62
(Salman et al., 2020)	ResNet-50	64.02	62.40	**64.70**	2.30

Table 13. Comparison of the ASR of APGD and ACG with DLR loss as the objective function. Dataset: CIFAR-100($\varepsilon = \frac{8}{255}$)

CIFAR-100 ($\varepsilon = 8/255$)			Attack Success Rate		
paper	**Architecture**	**clean acc**	**APGD(5)**	**ACG(5)**	**diff**
(Rade & Moosavi-Dezfooli, 2021)	PreActResNet-18	61.50	70.52	**71.03**	0.51
(Wu et al., 2020)	WideResNet-34-10	60.38	68.97	**70.64**	1.67
(Rebuffi et al., 2021)	WideResNet-28-10	62.41	66.64	**67.71**	1.07
(Rebuffi et al., 2021)	WideResNet-70-16	63.56	64.14	**65.07**	0.93
(Chen et al., 2021)	WideResNet-34-10	62.14	71.77	**72.50**	0.73
(Chen & Lee, 2021)	WideResNet-34-10	64.07	68.31	**69.11**	0.80
(Rice et al., 2020)	PreActResNet-18	53.83	79.83	**80.76**	0.93
(Hendrycks et al., 2019)	WideResNet-28-10	59.23	68.37	**70.73**	2.36
(Cui et al., 2021)	WideResNet-34-20	62.55	67.78	**69.47**	1.69
(Rebuffi et al., 2021)	PreActResNet-18	56.87	70.86	**71.42**	0.56
(Sitawarin et al., 2021)	WideResNet-34-10	62.82	72.83	**74.93**	2.10
(Addepalli et al., 2021)	WideResNet-34-10	65.73	68.65	**68.80**	0.15
(Cui et al., 2021)	WideResNet-34-10	60.64	70.92	**72.89**	1.97
(Gowal et al., 2020)	WideResNet-70-16	60.86	69.01	**69.85**	0.84
(Gowal et al., 2020)	WideResNet-70-16	69.15	61.39	**62.73**	1.34
(Addepalli et al., 2021)	PreActResNet-18	62.02	72.19	**72.58**	0.39
(Cui et al., 2021)	WideResNet-34-10	70.25	69.35	**70.72**	1.37

Table 14. The experimental result of the representative seven formulas to calculate β for ACG.

CIFAR-10 ($\varepsilon = 8/255$)		Attack Success Rate						
paper	**Architecture**	**FR**	**PR**	**HS**	**DY**	**HZ**	**DL**	**LS**
(Ding et al., 2020)	WideResNet-28-4	48.88	52.78	**55.77**	48.08	49.98	44.87	52.05
(Carmon et al., 2019)	WideResNet-28-10	39.03	39.55	**40.03**	35.43	39.05	33.70	39.56
(Rebuffi et al., 2021)	PreActResNet-18	42.75	42.88	**43.15**	40.68	42.50	40.20	42.90

Table 15. Computational environments: Information on the five computers used in the experiments are shown.

Machine	No.1 & No.2	No.3 & No.4	No.5
CPU	Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz $\times 2$	Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz $\times 2$	Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz $\times 2$
GPU	NVIDIA GeForce RTX 3090 $\times 4$		
RAM	768GB		256GB

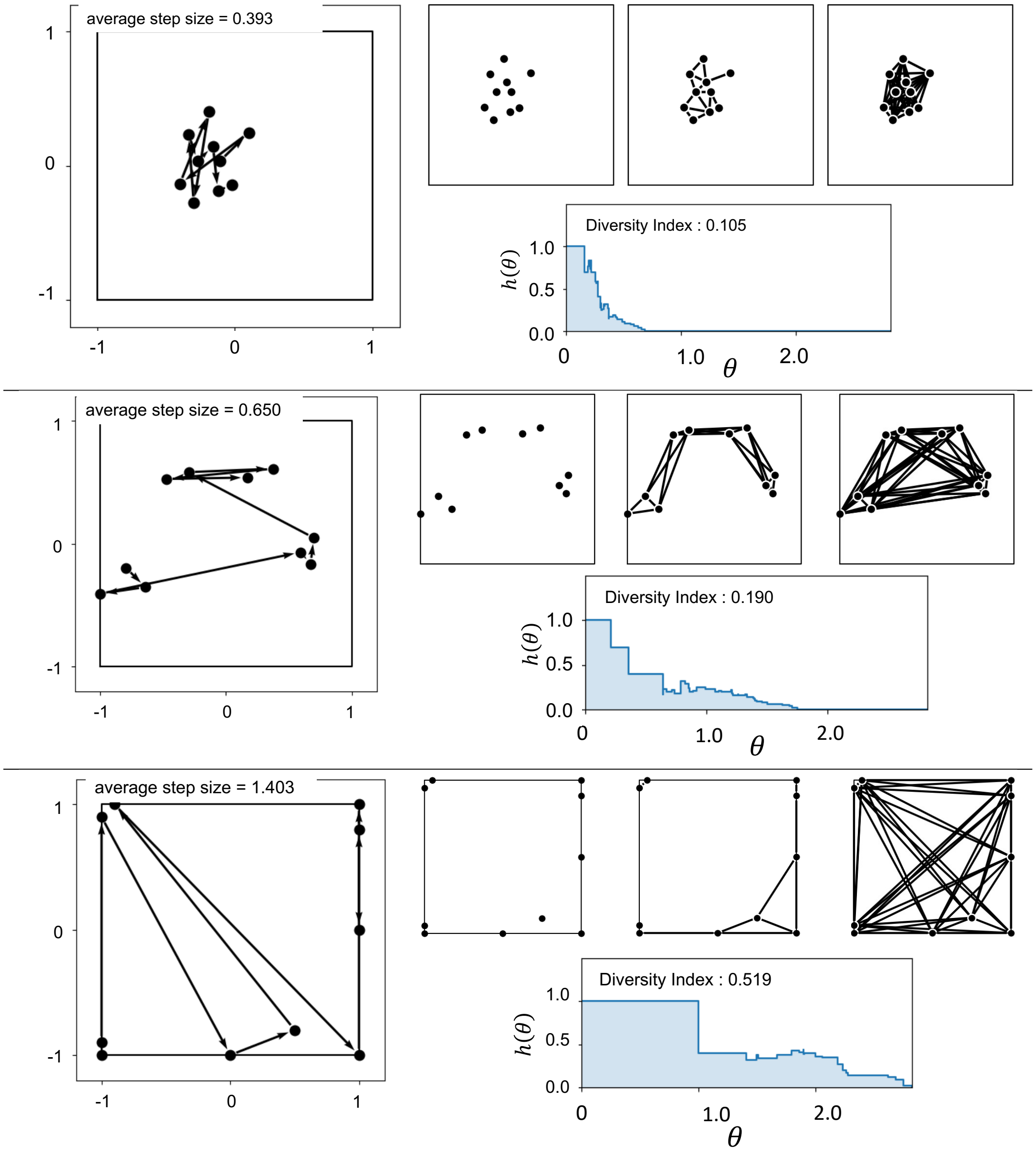


Figure 12. Additional examples of point clouds and its DIs.

Table 16. The ASR of AutoAttack(Reported) and AutoAttack combined with ACG with CE loss (AA(ACG-CE)).

paper	**Architecture**	AA(ACG-CE)	Reported
(Ding et al., 2020)	WideResNet-28-4	**58.60**	58.56
(Carmon et al., 2019)	WideResNet-28-10	**40.48**	40.47
(Andriushchenko et al., 2020)	WideResNet-18	**56.07**	**56.07**
(Rebuffi et al., 2021)	PreActResNet-18	**43.34**	**43.34**
CIFAR 100			
(Rice et al., 2020)	PreActResNet-18	81.02	**81.05**
ImageNet			
(Engstrom et al., 2019)	ResNet-50	70.76	**70.78**
(Wong et al., 2020)	ResNet-50	**73.80**	73.76
(Salman et al., 2020)	ResNet-50	**65.36**	65.04