Appropriate Balance of Diversification and Intensification Improves Performance and Efficiency of Adversarial Attacks

Anonymous authors
Paper under double-blind review

Abstract

Recently, adversarial attacks that generate adversarial examples by optimizing a multimodal function with many local optimums have attracted considerable research attention. Quick convergence to a nearby local optimum (intensification) and fast enumeration of multiple different local optima (diversification) are important to construct strong attacks. Most existing white-box attacks that use the model's gradient enumerate multiple local optima based on multi-restart; however, our experiments suggest that the ability of diversification based on multi-restart is limited. To tackle this problem, we propose the multi-directions/objectives (MDO) strategy, which uses multiple search directions and objective functions for diversification. Efficient Diversified Attack, a combination of MDO and multi-target strategies, showed further diversification performance, resulting in better performance than recently proposed attacks against around 88% of 41 CNN-based robust models and 100% of 10 more advanced models, including transformer-based architecture. These results suggest a relationship between attack performances and a balance of diversification and intensification, which is beneficial to constructing more potent attacks.

1 Introduction

Deep neural networks (DNNs) have demonstrated excellent performance in several applications. However, DNNs are known to misclassify adversarial examples generated by tiny perturbing inputs that are imperceptible to humans (Szegedy et al., 2014). Vulnerabilities caused by adversarial examples can have fatal consequences, especially in safety-critical applications such as automated driving (Gupta et al., 2021), facial recognition (Adjabi et al., 2020), and cybersecurity (Liu et al., 2022b). Adversarial training (Madry et al., 2018) is one of the most effective defenses, which uses adversarial examples during training. Generating many adversarial examples faster is beneficial for the security of DNNs because adversarial training requires many adversarial examples.

Adversarial attacks optimize a challenging nonconvex nonlinear function to find adversarial examples. We focus on white-box attacks that use gradient-based optimization algorithms, assuming access to the outputs and gradients of the DNN. Higher objective function values increase misclassification chances, creating adversarial example candidates out of local optima. The objective function of this problem is multimodal because its maximization involves a complex DNN. Because a multimodal function has a myriad of local optima, quick convergence to a nearby local optimum and fast enumeration of multiple different local optima are important. These are referred to as intensification and diversification, respectively (Glover & Samorani, 2019). Many existing gradient-based attacks are considered to achieve some degree of intensification because the objective value can be improved by moving in the gradient direction within a neighborhood. Whereas many existing attacks diversify the search based on the multi-restart (Dong et al., 2013; Madry et al., 2018; Croce & Hein, 2020b; Liu et al., 2022c), few studies have examined other diversification strategies. Therefore, further research on diversification is needed.

Our experiments using diversification indices in Section 3.1 suggest that the multi-restart strategy does not yield high diversification performance, and different local solutions may be enumerated using different search directions and objective functions. Inspired by this observation, we propose the multi-directions/objectives

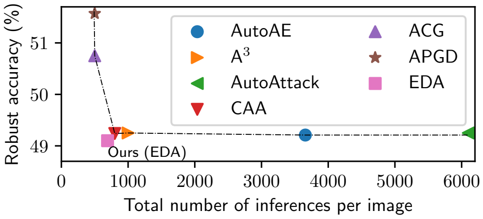


Figure 1: EDA vs. recently proposed attacks on adversarially trained CIFAR-10 model Engstrom et al. (2019).

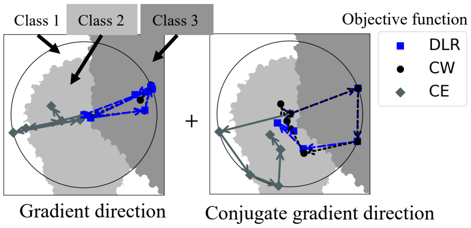


Figure 2: Illustration of the MDO strategy with two search directions and three objective functions on a toy model. The different search directions/objective functions search for different regions.

(MDO) strategy that uses multiple search directions and objective functions. To implement the MDO strategy, we propose the Automated Diversified Selection (ADS) and a search framework called the MDO framework in Section 3.2 and Section 3.3, respectively. ADS selects search directions and objective functions based on the Diversity Index (DI) (Yamamura et al., 2022) to enhance the diversification. The MDO framework explicitly considers the diversification and intensification phases to balance them appropriately. Figure 2 is a toy example of an attack using the MDO strategy.

The attack with an appropriate balance of diversification and intensification is expected to be strong and fast. Based on this, we evaluated the proposed methods with robust accuracy, computation cost, and DI. The experimental results in Sections 4.3 and 4.4 suggest that ADS contributes to search diversification, and the MDO framework realizes an appropriate balance of diversification and intensification, resulting in a promising attack performance. As shown in Section 4.2, the MDO strategy found adversarial examples for some inputs in less time than the multi-target (MT) strategy (Gowal et al., 2019), one of the promising attacks. These results imply that the combination of MDO and MT strategies can realize stronger and faster attacks.

Motivated by these results, we experimentally investigated the attack performance of the combination of MDO and MT strategies, called Efficient Diversified Attack (EDA). Experimental results in Section 4.1 show that EDA exhibits higher diversification performance than the attack using the MDO strategy alone, resulting in better attack performance in less computation time than the recently proposed attacks. Given the difference in robust accuracy among recently proposed attacks, EDA exhibits sufficiently large improvements in robust accuracy and runtime. The above experimental results suggest that appropriately enhancing diversification leads to higher attack performance. The major contributions of this study are summarized below.

1. **Multi-directions/objectives (MDO)** strategy and its implementation: A novel search strategy using multiple search directions and objective functions, realized by Automated Diversified Selection (ADS) and the MDO framework. ADS contributes to diversification, and the MDO framework realizes the proper balance of diversification and intensification.
2. **Efficient Diversified Attack (EDA)**: A faster and stronger attack using MDO and MT strategies. EDA showed the best attack performance among recent attacks against around 88% of 41 CNN-based robust models trained on three representative datasets and 100% of 10 more advanced models, including transformer-based architecture, trained on ImageNet Russakovsky et al. (2015).

2 Preliminaries

2.1 Problem settings

Let $g : D \to \mathbb{R}^C$ be a locally differentiable C-classifier, $\boldsymbol{x}_{\mathrm{org}} \in D$ be a point with c as the correct label, and $d : D \times D \to \mathbb{R}$ be a distance function. Given $\varepsilon > 0$, the feasible region $\mathcal{S}$ is defined as the set of points $\boldsymbol{x} \in D$ that are within a distance of ε from $\boldsymbol{x}_{\mathrm{org}}$, i.e., $\mathcal{S} := \{\boldsymbol{x} \in D \mid d(\boldsymbol{x}, \boldsymbol{x}_{\mathrm{org}}) \leq \varepsilon\}$. Then, we define an adversarial example as $\boldsymbol{x}_{\text{adv}} \in \mathcal{S}$ satisfying $\arg\max_{i=1,\dots,C} g_i(\boldsymbol{x}_{\text{adv}}) \neq c$. The following expression is a formulation of one type of adversarial attack, the untargeted attack, where the attacker does not specify the misclassification target.

$$\text{find } \boldsymbol{x} \in \mathcal{S} \text{ s.t. } \max_{i=1,\dots C} g_i(\boldsymbol{x}) - g_c(\boldsymbol{x}) > 0 \tag{1}$$

Problem 1 is solved through maximization of the objective function $L(g(\boldsymbol{x}), c)$ within the feasible region $\mathcal{S}$. This maximization aims to reduce the probability that $\boldsymbol{x}$ is classified in class c by g. Therefore, $\boldsymbol{x}$ with a high objective value $L(g(\boldsymbol{x}), c)$ is more likely to be misclassified by g. When $d(\boldsymbol{x}_{\text{adv}}, \boldsymbol{x}_{\text{org}})$ is small, the norm of the adversarial perturbation is also small. The targeted attack aims at maximizing the probability that $\boldsymbol{x}_{\text{adv}}$ is classified in a particular class $t \neq c$ by solving $\max_{\boldsymbol{x}\in\mathcal{S}} L\left(g(\boldsymbol{x}), c, t\right)$. For adversarial attacks on image classifiers, $D = [0,1]^n$ and $d(\boldsymbol{v}, \boldsymbol{w}) := \|\boldsymbol{v} - \boldsymbol{w}\|_p, (p = 2, \infty)$ is typically used. This study focuses on the untargeted attack on image classifiers using $d(\boldsymbol{v}, \boldsymbol{w}) := \|\boldsymbol{v} - \boldsymbol{w}\|_\infty$, referred to as ℓ_∞ attacks.

2.2 Related work

General white-box attacks. In the white-box attack, the initial point sampling $\mathtt{x}$ determines $\boldsymbol{x}^{(0)}$ first. Then, the step size update rule η and the search direction $\boldsymbol{\delta} = \boldsymbol{\delta}(L)$ updates the step size $\eta^{(k)}$ and the moving direction $\boldsymbol{\delta}^{(k)}$, respectively. Subsequently, the search point $\boldsymbol{x}^{(k+1)}$ is updated by the following formula.

$$\boldsymbol{x}^{(0)} \leftarrow \text{sampled by } \mathtt{x},\ \boldsymbol{x}^{(k+1)} \leftarrow P_{\mathcal{S}}\left(\boldsymbol{x}^{(k)} + \eta^{(k)}\boldsymbol{\delta}^{(k)}\right), \tag{2}$$

where k is the iteration, and $P_{\mathcal{S}}$ is a projection onto $\mathcal{S}$. The moving direction $\boldsymbol{\delta}^{(k)}$ is usually computed based on the gradient $\nabla L(g(\boldsymbol{x}^{(k)}), c)$. According to equation 2, attack methods are characterized by the initial point sampling $\mathtt{x}$, step size update rule η, search direction $\boldsymbol{\delta}$, and objective function L. The tuple $a = (\mathtt{x}, \eta, \boldsymbol{\delta}, L)$ is referred to as an attack a. Algorithm 4 in Appendix A.1 provides the procedure of the attack a.

Commonly used attack components. Projected Gradient Descent (PGD) (Madry et al., 2018) is a fundamental white-box attack. PGD uses a fixed step size (η_{fix}) and moves to the normalized gradient direction ($\boldsymbol{\delta}_{\text{PGD}}$). Auto-PGD (APGD) (Croce & Hein, 2020b) is a variant of PGD using a heuristic (η_{APGD}) for updating step size and moving to the momentum direction ($\boldsymbol{\delta}_{\text{APGD}}$). In addition, some studies use cosine annealing ($\eta_{\cos}$) (Loshchilov & Hutter, 2017) for updating step size. Gradual reduction of step size, such as η_{APGD} and $\eta_{\cos}$, showed better results than fixed step size. Auto-Conjugate Gradient attack (ACG) (Yamamura et al., 2022) uses η_{APGD} and moves to the normalized conjugate gradient direction ($\boldsymbol{\delta}_{\text{ACG}}$). Whereas the sort of steepest directions, such as $\boldsymbol{\delta}_{\text{PGD}}$ and $\boldsymbol{\delta}_{\text{APGD}}$, are suitable for intensification, the conjugate gradient-based direction is suitable for diversification. For the initial point, uniform sampling from $\mathcal{S}$ or input points ($\mathtt{x}_{\text{org}}$) is usually used. Assuming the many restarts, Output Diversified Sampling (ODS, $\mathtt{x}_{\text{ODS}}$) and its variant, which considers the output diversity of the threat model (Tashiro et al., 2020; Liu et al., 2022c), outperformed naive random sampling. For the objective functions, cross-entropy (CE) loss (Goodfellow et al., 2015) (L_{CE}) and margin-based losses such as CW loss (Carlini & Wagner, 2017) (L_{CW}), a variation of CW loss scaled by the softmax function (L_{SCW}), and Difference of Logits Ratio (DLR) loss (Croce & Hein, 2020b) (L_{DLR}) are often used. We denote the targeted version of these losses as L^T, e.g., L^T_{CE} for targeted CE loss. See Appendices A.2 and A.3 for mathematical formulas of attack components.

Robustness evaluation. The robustness of DNNs is usually evaluated using AutoAttack (AA) Croce & Hein (2020b), combining four different attacks, including the MultiTargeted (MT) Gowal et al. (2019) strategy. MT strategy runs targeted attacks on each of multiple target classes and thus requires higher computation costs. The high computational cost of AA has motivated the research community to pursue faster attacks for adversarial training and robustness evaluation (Gao et al., 2022; Xu et al., 2022). Composite Adversarial Attack (CAA) (Mao et al., 2021) and AutoAE (Liu et al., 2023b) combine multiple attacks by solving an additional optimization problem, requiring the pre-execution of candidate methods on relatively large samples. Consequently, they are still computationally expensive. Adaptive Auto Attack (A^3) Liu et al. (2022c) demonstrated state-of-the-art (SOTA) attack performance in less computation time by improving the initial point sampling and discarding the hard-to-attack images. A^3 is based on PGD with CW loss, which uses a single search direction and objective function. Some studies have investigated how to switch either the search direction or the objective function based on case studies to improve attack performance (Yamamura et al., 2022; Antoniou et al., 2022). However, to the best of our knowledge, this is the first study that investigates the combination of multiple search directions and objective functions based on search diversification. See Appendix A.4 for more information.

3 Multi-directions/objectives strategy

3.1 Motivation

We hypothesize that diversification contributes to attack performance. This section empirically demonstrates that attacks with multiple search directions ($\boldsymbol{\delta}$) and objective functions (L) can achieve more efficient diversification than attacks with a single $\boldsymbol{\delta}$ and L. Four search directions and seven objective functions are used in this experiment. $N_{\max}$ iterations of attacks are performed from R initial points for each pair of search direction and objective function. The search point with the highest CW loss (L_{CW}) value is collected at the end of the attack, which starts at each initial point. The obtained set of these search points is denoted as $X^*(\boldsymbol{x}_i, a, R)$. We focus on this set of search points to analyze the characteristics of the attack using a single $\boldsymbol{\delta}$ and L. For this experiment, we used 10,000 images from CIFAR-10 (Krizhevsky et al., 2009) as test samples and attacked the robust model proposed by Sehwag et al. (2022). See Appendix C for more information and results.

Notation. Let $\mathcal{L} = \{L_{\text{CE}}, L_{\text{CW}}, L_{\text{SCW}}, L_{\text{DLR}}\} \cup \{L_{\text{G-DLR},q} \mid q = 4, 5, 6\}$ be a set of objective functions, and $\mathcal{D} = \{\boldsymbol{\delta}_{\text{PGD}}, \boldsymbol{\delta}_{\text{APGD}}, \boldsymbol{\delta}_{\text{ACG}}, \boldsymbol{\delta}_{\text{Nes}}\}$ be a set of search directions. We have proposed $\boldsymbol{\delta}_{\text{Nes}}$ and $L_{\text{G-DLR},q}$ in this study. $\boldsymbol{\delta}_{\text{Nes}}$ is the search direction of Nesterov's accelerated gradient (NAG) (Nesterov, 2004) normalized by the sign function to accommodate ℓ_∞ attacks. We refer to $L_{\text{G-DLR},q}$ as generalized-DLR (G-DLR) loss with the denominator of DLR loss extended from $g_{\pi_1}(\boldsymbol{x}) - g_{\pi_3}(\boldsymbol{x})$ to $g_{\pi_1}(\boldsymbol{x}) - g_{\pi_q}(\boldsymbol{x})$; $\pi_q \in Y$ denotes the class label that has the q-th largest value of $g(\boldsymbol{x})$. Mathematical expressions of $\boldsymbol{\delta}_{\text{Nes}}$ and $L_{\text{G-DLR},q}$ can be found in Appendix B. Given an initial point sampling $\mathtt{x}$ and a step size update rule η, we define a set of attacks as $\mathcal{A}(\mathtt{x}, \eta) = \{a = (\mathtt{x}, \eta, \boldsymbol{\delta}, L) \mid \boldsymbol{\delta} \in \mathcal{D}, L \in \mathcal{L}\}$. Let $(\boldsymbol{x}_i)_{a,r}^{(k)}$ be a search point in the k-th iteration of the attack a, starting at the r-th initial point. Then, let $(\boldsymbol{x}_i)_{a,r}^*$ be the search point with the highest L_{CW} value obtained by the attack a from the r-th initial point, and $X^*(\boldsymbol{x}_i, a, R) = \left\{(\boldsymbol{x}_i)_{a,r}^* \mid r = 1, 2, \dots, R\right\}$. In the following paragraphs, we analyze the characteristics of attacks $a \in \mathcal{A}(\mathtt{x}_{\text{ODS}}, \eta_{\cos})$, which use single $\boldsymbol{\delta}$ and L, based on the experimental results with $N_{\max} = 30$ and $R = 10$.

Limited diversification ability of attacks using single $\boldsymbol{\delta}$ and L. We quantify the diversity of $X^*(\boldsymbol{x}_i, a, R)$ using DI to reveal the diversification ability of the attack a, which uses a single $\boldsymbol{\delta}$ and L. DI quantifies the degree of density of any point set as a value between 0 and 1 based on the global clustering coefficient of a graph. DI tends to be small when the point set forms clusters. Figure 3 shows the violin plot of $DI\left(X^*(\boldsymbol{x}_i, a, R), M\right)$ for 10,000 images and all attacks a. M denotes the size of the feasible region. This study used the same value of M as in Yamamura et al. (2022). The mean and standard deviation of the first, second, and third quartiles were 0.190 ± 0.019, 0.223 ± 0.023, and 0.269 ± 0.033, respectively. Figure 3 and these DI values suggest that the diversity of the best point set $X^*(\boldsymbol{x}_i, a, R)$ is relatively low. Thus, the diversification ability of a seems to be limited.

Attacks using multiple $\boldsymbol{\delta}$ and L can lead to efficient diversification. Figure 4 shows the best point set $X^*(\boldsymbol{x}_i, a, R)$ embedded in a two-dimensional space using Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018). Dimensionality reduction methods, such as UMAP, preserve the maximum possible distance information in high-dimensional spaces as possible. Figure 4 (a) depicts the points obtained by the attack using the same L in the same color. Figure 4 (a) shows that sets of best search points obtained from searches with different L tend to form different clusters. Figure 4 (b) depicts the points obtained by the attack using the same $\boldsymbol{\delta}$ in the same color. Similarly, sets of best search points obtained by the attack using different $\boldsymbol{\delta}$ also tend to form different clusters. Based on these observations, it is possible to efficiently search for different local solutions using different $\boldsymbol{\delta}$ and L, or an appropriate combination of both.

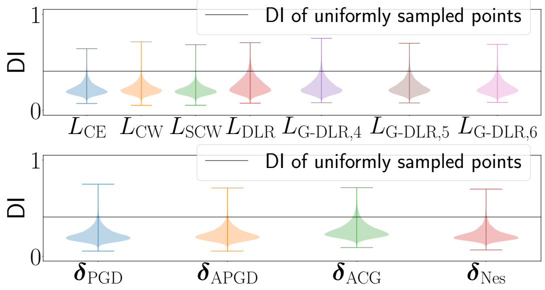


Figure 3: Violin plot of $DI(X^*(\boldsymbol{x}, a, 10), M)$. The upper/lower figure represents the attack with the same objective function/search direction, respectively.

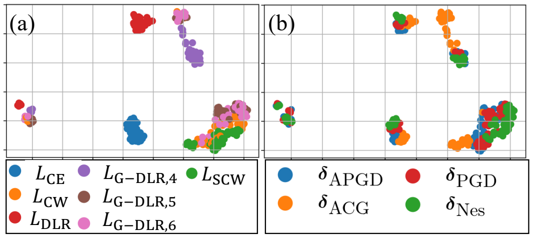


Figure 4: 2D visualization of $X^*(\boldsymbol{x}, a, 10)$ using UMAP. Points of the same color in (a)/(b) represent points obtained using the same objective function/search direction, respectively.

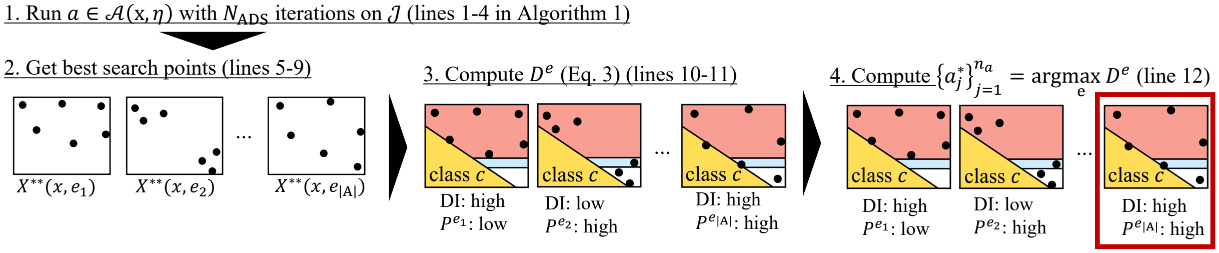

1. Run $a \in \mathcal{A}(\mathrm{x}, \eta)$ with N_{ADS} iterations on $\mathcal{J}$ (lines 1-4 in Algorithm 1)
2. Get best search points (lines 5-9)
$X^{**}(x, e_1)$
$X^{**}(x, e_2)$
...
$X^{**}(x, e_{|A|})$
3. Compute D^e (Eq. 3) (lines 10-11)
class c
DI: high
P^{e_1}: low
class c
DI: low
P^{e_2}: high
...
class c
DI: high
$P^{e_{|A|}}$: high
4. Compute $\{a_j^*\}_{j=1}^{n_a} = \operatorname{argmax}_e D^e$ (line 12)
class c
DI: high
P^{e_1}: low
class c
DI: low
P^{e_2}: high
...
class c
DI: high
$P^{e_{|A|}}$: high

Figure 5: The procedure of ADS. The black circle represents a single search point, and DI tends to be smaller when the search points form clusters. Regions of different colors represent different classification classes, and the more search points are distributed around multiple regions, the higher the degree of diversification in the output space, i.e., the value of P_i^e.

3.2 Automated Diversified Selection

The analyses in Section 3.1 suggest that the MDO strategy can efficiently search for different local solutions. However, this strategy is ineffective unless the combinations of $\boldsymbol{\delta}$ and L are properly determined because different attacks may search similar regions. To address this issue, we propose Automated Diversified Selection (ADS), which selects the combinations of search directions and objective functions based on the degree of diversification in input and output spaces. Algorithm 1 and Figure 5 show the ADS procedure.

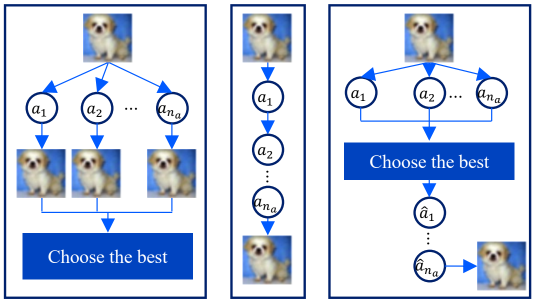
First, N_{ADS} iterations of attack candidates $a \in \mathcal{A}(\mathrm{x}, \eta)$ are executed on the image set $\mathcal{J} \subset D$ (lines 1-4 in Algorithm 1). In this study, $\mathcal{J}$ is 1% of the images uniformly sampled from all the test samples. Subsequently, the set of best search points $X^*(\boldsymbol{x}_i, a, 1)$ and class labels $\Pi_i^a = \left\{\pi(\boldsymbol{x}_i)^{(k)} \mid k = 1, \ldots, N_{\mathrm{ADS}}\right\} \subset Y$ are obtained, where $\pi(\boldsymbol{x}_i)^{(k)} = \arg\max_{q \neq c_i} g_q((\boldsymbol{x}_i)_{a,1}^{(k)})$ is a class label with the highest prediction probability excluding the correct label c_i.

The observations in Section 3.1 suggest that the degree of diversification may be greatly reduced when the selected attacks employ the same objective function. Therefore, a set of the candidate combinations of n_a attacks is defined as $A = \left\{\{a_j\}_{j=1}^{n_a} \subset \mathcal{A}(\mathrm{x}, \eta) \mid |\{L_{a_j}\}_{j=1}^{n_a}| = n_a\right\}$ so that each attack uses a different L (line 5 in Algorithm 1). Let $X^{**}(\boldsymbol{x}_i, e)$ be the set of search points with the highest L_{CW} values obtained by the attacks in $e = \{a_j\}_{j=1}^{n_a} \subset \mathcal{A}(\mathrm{x}, \eta)$ to image $\boldsymbol{x}_i \in D$. The weighted average of the DI is calculated for all $e \in A$ to quantify the diversity of the best point set $X^{**}(\boldsymbol{x}_i, e)$ as follows:

$$D^e = \frac{1}{|\mathcal{J}|} \sum_{i=1}^{|\mathcal{J}|} P_i^e \cdot DI(X^{**}(\boldsymbol{x}_i, e), M), \tag{3}$$

Algorithm 1 Automated Diversified Selection (ADS)

```
Require: $\mathcal{J}$: small samples, $\mathcal{A}(\mathtt{x}, \eta)$: candidate attacks, $N_{\mathrm{ADS}}$: maximum iterations, $P_{\mathcal{S}}$: projection, $g$: DNN, $M$: the size of feasible region, $n_a$: number of attacks to select
Ensure: $\{(\boldsymbol{\delta}_{a_j^*}, L_{a_j^*})\}_{j=1}^{n_a}$: The pairs of search direction and objective function
1:  for $a = (\mathtt{x}, \eta, \boldsymbol{\delta}, L) \in \mathcal{A}(\mathtt{x}, \eta)$ do
2:      /* Run the attack $a$ following Algorithm 4 */
3:      $X^a, \Pi^a \leftarrow \mathrm{Attack}(a, N_{\mathrm{ADS}}, P_{\mathcal{S}}, g, \mathcal{J})$
4:  end for
5:  $A \leftarrow \{e \subset \mathcal{A}(\mathtt{x}, \eta) \mid |e| = n_a \wedge |\{L_{a_j}\}_{j=1}^{n_a}| = n_a\}$
6:  for $e \in A$ do
7:      for $i = 1, \ldots, |\mathcal{J}|$ do
8:          $X^{**}(\boldsymbol{x}_i, e) \leftarrow \cup_{a \in e} X_i^a$, $P_i^e \leftarrow |\cup_{a \in e} \Pi_i^a|$
9:      end for
10:     Compute $D^e$ by equation 3
11: end for
12: Get $\{(\boldsymbol{\delta}_{a_j^*}, L_{a_j^*})\}_{j=1}^{n_a}$ from $\{a_j^*\}_{j=1}^{n_a} = \arg\max_{e \in A} D^e$.
```



(a) Ensemble (b) Composite (c) MDO Framework

Figure 6: Illustration of the ensemble, composite, and the MDO framework.

where $P_i^e = |\cup_{a \in e} \Pi_i^a|$ is the number of types of classification labels with the highest prediction probability excluding the correct label (lines 6-11 in Algorithm 1). A high DI indicates a high diversity of $X^{**}(\boldsymbol{x}_i, e)$, and a high P_i^e indicates a high output diversity. Finally, ADS outputs $\{(\boldsymbol{\delta}_{a_j^*}, L_{a_j^*})\}_{j=1}^{n_a}$ as the appropriate combinations of $\boldsymbol{\delta}$ and L, where $\{a_j^*\}_{j=1}^{n_a} = \arg\max_{e \in A(\mathtt{x}, \eta)} D^e$ (line 12 in Algorithm 1).

3.3 Search framework for the MDO strategy

Considering the difference in diversification/intensification performance between PGD and CG-based search directions reported by Yamamura et al. (2022), we propose a search framework called the MDO framework consisting of diversification and intensification phases. The diversification phase aims to search a wide area based on an ensemble, and the intensification phase, based on a composite, aims to search the nearby area around the best point obtained during the diversification phase in more detail to improve the objective value. Figure 6 illustrates the procedure of the ensemble, composite, and MDO framework. The pseudocode of the MDO framework is described in Algorithm 2 . Assume that the total number of the MDO framework iterations is $N \times n_a$. Inspired by the APGD step size control, the diversification phase takes $\lceil 0.41N \rceil \times n_a$ iterations, and the intensification phase uses the remaining iterations.

Diversification phase. The diversification phase uses large step sizes to search a broader area. First, ADS is executed with a fixed step size of 2ε and initial point sampling $\mathtt{x}$ to determine the pairs $\{(\boldsymbol{\delta}_{a_j^*}, L_{a_j^*})\}_{j=1}^{n_a}$, where $(a_1^*, a_2^*, \ldots, a_{n_a}^*) \subset \mathcal{A}(\mathtt{x}_{\mathrm{init}}, \eta_{\mathrm{fix}})$ (line 4 in Algorithm 2). For each pair of $\boldsymbol{\delta}_{a_j^*}$ and $L_{a_j^*}$, $N_1 = \lceil 0.41N \rceil$ iterations of the attack are executed with an initial step size of 2ε starting at an initial point selected by $\mathtt{x}_{\mathrm{init}}$ (lines 6-12 in Algorithm 2). The step size is updated by η_{APGD}. When the total iteration is N_{iter}, APGD step size control allocates at least $\lceil 0.22 N_{iter} \rceil$ iterations with a step size of 2ε and $\lceil (0.41 - 0.22) N_{iter} \rceil = \lceil 0.19 N_{iter} \rceil$ iterations with a step size of 2ε or ε. Inspired by this, the diversification phase uses the total iterations of N_1 for each attack and the checkpoints of $W = \{\lceil 0.22 \times N \rceil\}$ for step size control η_{APGD} to achieve diversification with large step sizes. Then, move to the intensification phase.

Intensification phase. The intensification phase takes the solution with the highest L_{CW} value found by the diversification phase as the initial point and searches for different local solutions within a range not far from the initial point. In APGD step size control, the step size is set to $\varepsilon/2$ after the search with a step size of ε. Based on this, the initial step size is set to $\varepsilon/2$. First, ADS is executed with a fixed step size of $\eta = \varepsilon/2$ to determine the pairs $\{(\boldsymbol{\delta}_{\hat{a}_j}, L_{\hat{a}_j})\}_{j=1}^{n_a}$ (line 16 in Algorithm 2), where $(\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{n_a}) \subset \mathcal{A}(\mathtt{x}_{\mathrm{best}}, \eta_{\mathrm{fix}})$, and $\mathtt{x}_{\mathrm{best}}$ denotes the initial point sampling that uses the solution with the highest L_{CW} value as the initial point.

Algorithm 2 MDO framework

```
Require: $\mathcal{I}$: test samples, $g$: DNN, $\varepsilon$: allowed perturbation size, $n_a$: number of pairs to select, $N_{\mathrm{ADS}}$: search iterations in ADS, $M$: the size of the feasible region, $N$: search iterations per attack
Ensure: $X^{\mathrm{advs}}$: adversarial examples
 1: $X^{\mathrm{advs}} \leftarrow \emptyset$
 2: /* Diversification phase */
 3: $\mathcal{J} \leftarrow$ uniformly sampled 1% images of $\mathcal{I}$.
 4: $\{(\boldsymbol{\delta}_{a_j^*}, L_{a_j^*})\}_{j=1}^{n_a} \leftarrow \mathrm{ADS}\,(\mathcal{J}, \mathcal{A}(\mathtt{x}_{\mathrm{init}}, \eta_{\mathrm{fix}}), N_{\mathrm{ADS}}, P_{\mathcal{S}}, g, M, n_a)$
 5: $N_1 \leftarrow \lceil 0.41N \rceil, N_2 \leftarrow N - N_1$
 6: for $j = 1, \ldots, n_a$ do
 7:     $a' \leftarrow (\mathtt{x}_{\mathrm{init}}, \eta_{\mathrm{APGD}}, \boldsymbol{\delta}_{a_j^*}, L_{a_j^*})$
 8:     /* Run the attack $a'$ following Algorithm 4 */
 9:     $X^{a'}, \Pi^{a'} \leftarrow \mathrm{Attack}(a', N_1, P_{\mathcal{S}}, g, \mathcal{I})$
10:     Update $X^{\mathrm{advs}}$ by $X^{a'}$
11:     Update $\mathcal{I}$ by excluding images that succeeded in the attack.
12: end for
13: /* Intensification phase */
14: $\mathcal{I}_{sub} \leftarrow \mathcal{I}$
15: $\mathcal{J}' \leftarrow$ uniformly sampled 1% images of $\mathcal{I}$.
16: $\{(\boldsymbol{\delta}_{\hat{a}_j}, L_{\hat{a}_j})\}_{j=1}^{n_a} \leftarrow \mathrm{ADS}\,\big(\mathcal{J}', \mathcal{A}(\mathtt{x}_{\mathrm{best}}, \eta_{\mathrm{fix}}), N_{\mathrm{ADS}}, P_{\mathcal{S}}, g, M, n_a\big)$
17: for $j = 1, \ldots, n_a$ do
18:     $a^l \leftarrow (\mathtt{x}_{\mathrm{best}}, \eta_{\mathrm{cos}}, \boldsymbol{\delta}_{\hat{a}_j}, L_{\hat{a}_j})$ {Use $\eta_{\mathrm{APGD}}$ instead of $\eta_{\mathrm{cos}}$ when $\boldsymbol{\delta}_{\hat{a}_j} = \boldsymbol{\delta}_{\mathrm{ACG}}$}
19:     /* Run the attack $a^l$ following Algorithm 4 */
20:     $X^{a^l}, \Pi^{a^l} \leftarrow \mathrm{Attack}(a^l, N_2, P_{\mathcal{S}}, g, \mathcal{I}_{sub})$
21:     Update $X^{\mathrm{advs}}$ by $X^{a^l}$.
22:     /* Extract images whose highest CW loss value is close to 0 for further intensification. */
23:     $\mathcal{I}_{sub} \leftarrow \mathcal{I}_{sub} \cap \big\{\boldsymbol{x}_i \in \mathcal{I} \mid -0.05 \le L_{\mathrm{CW}}(g((\boldsymbol{x}_i)^*_{a^l,1}), c_i) \le 0\big\}$
24:     $X^{a^l}, \Pi^{a^l} \leftarrow \mathrm{Attack}(a^l, N_2, P_{\mathcal{S}}, g, \mathcal{I}_{sub})$
25:     Update $X^{\mathrm{advs}}$ by $X^{a^l}$.
26: end for
```

For each pair of $(\boldsymbol{\delta}_{\hat{a}_j}, L_{\hat{a}_j})$, $N_2 = N - N_1$ iterative searches are performed with the initial point determined by $\mathtt{x}_{\mathrm{best}}$, and the initial step size of $\varepsilon/2$, using step size update rule η_{cos} (lines 17-25 in Algorithm 2). Note that η_{APGD} is used to update the step size when the search direction $\boldsymbol{\delta}_{\hat{a}_j}$ is equal to $\boldsymbol{\delta}_{\mathrm{ACG}}$, aiming for a better intensification performance. Subsequently, the same searches are performed on the images whose highest L_{CW} values are greater than or equal to -0.05 to accelerate the intensification (lines 22-24 in Algorithm 2).

3.4 Efficient Diversified Attack

The comparisons of the MDO framework and $\mathrm{MT}_{\mathrm{cos}}$ in Section 4.2 suggest that combining MDO and MT strategies would lead to a faster and stronger attack. This result motivates us to propose Efficient Diversified Attack (EDA), a combination of MDO and MT strategies. We used the MDO framework for the MDO strategy and the targeted attack $a^T = (\mathtt{x}_{\mathrm{init}}, \eta_{\mathrm{APGD}}, \boldsymbol{\delta}_{\mathrm{GD}}, L^T_{\mathrm{CW}})$ with N iterations and a target class T selected based on a small-scale search for the multi-target strategy. See Appendix F for validation of the target selection scheme. The pseudocode of EDA is described in Algorithm 3. EDA executes the MDO framework first (lines 1-4 in Algorithm 3). Subsequently, the target class T is selected based on the small-scale search (lines 6-12 in Algorithm 3). Finally, the targeted attack a^T is executed (lines 15-16 in Algorithm 3).

4 Experiments

The efficacy of the proposed methods was examined through a series of experiments involving an ℓ_∞ attack against ℓ_∞ defense models listed in RobustBench (Croce et al., 2021).

Algorithm 3 Efficient Diversified Attack (EDA)

```
**Require:** $\mathcal{I}$: test set, $g$: DNN, $\varepsilon$: allowed perturbation size, $n_a$: number of pairs to select, $K$: number of target classes, $N_{\mathrm{ADS}}$: search iterations in ADS, $M$: the size of the feasible region, $N$: search iterations per attack $a$, $N_s$: number of iterations in small-scale search
**Ensure:** $X^{\mathrm{advs}}$: Adversarial examples
$X^{\mathrm{advs}} \leftarrow \emptyset$
/* Run the MDO framework (Algorithm 2) */
$X^{\mathrm{advs}} \leftarrow$ the MDO framework$(\mathcal{I}, g, \varepsilon, n_a, N_{\mathrm{ADS}}, M, N)$
Update $\mathcal{I}$ by excluding images that succeeded in the attack.
/* Select target class based on small-scale search */
$T \leftarrow \{0\}^{|\mathcal{I}|}$
$a^t \leftarrow (\mathtt{x}_{\mathrm{init}}, \eta_{\cos}, \boldsymbol{\delta}_{\mathrm{GD}}, L^T_{\mathrm{CW}})$
**for** $t = 0, \dots K$ **do**
    /* Run targeted attack with the target class $\pi_t$ */
    $X^{a^t}, \Pi^{a^t} \leftarrow$ Attack$(a^t, N_s, P_{\mathcal{S}}, g, \mathcal{I})$
    Update $T_i$ for images $\boldsymbol{x}_i$ that record the highest loss value
**end for**
$a^T \leftarrow (\mathtt{x}_{\mathrm{init}}, \eta_{\mathrm{APGD}}, \boldsymbol{\delta}_{\mathrm{GD}}, L^T_{\mathrm{CW}})$
/* Run targeted attack with the target $T_i$ */
$X^{a^T}, \Pi^{a^T} \leftarrow$ Attack$(a^T, N, P_{\mathcal{S}}, g, \mathcal{I})$
Update $X^{\mathrm{advs}}$ by $X^{a^T}$.
```

Table 1: Comparison in robust accuracy. The lower the robust accuracy, the higher the attack performance. The lowest robust accuracy is in **bold**. The "#query" row shows the number of queries in the worst-case per image. The robust accuracy of AA is the value reported by RobustBench. MDO represents the MDO framework. OOM (Out of Memory) indicates that execution could not be completed due to a memory error.

Dataset	No.	Defense	clean	AA	ACG	$\mathrm{MT}_{\cos}$	CAA	AutoAE	A^3	MDO	EDA
#query				6.1k	500	504	800	3.6k	1k	502.24	802.24
CIFAR-10	1	(Sehwag et al., 2022)	84.59	55.54	56.19	55.54	55.52	55.51	55.53	55.58	**55.49**
	2	(Carmon et al., 2019)	89.69	59.53	60.10	59.54	59.50	59.47	59.44	59.46	**59.40**
($\varepsilon = 8/255$)	3	(Rebuffi et al., 2021)	88.54	64.25	64.80	64.28	64.23	**64.19**	64.24	64.32	64.20
CIFAR-100	4	(Rice et al., 2020)	53.83	18.95	19.48	18.99	18.96	18.91	18.89	18.97	**18.88**
	5	(Sitawarin et al., 2021)	62.82	24.57	25.69	24.55	24.56	24.52	24.56	24.65	**24.50**
($\varepsilon = 8/255$)	6	(Gowal et al., 2020)	69.15	36.88	37.84	36.95	36.90	36.86	36.87	36.96	**36.81**
ImageNet	7	(Salman et al., 2020)	52.92	25.32	26.40	25.24	25.27	OOM	25.22	25.22	**25.11**
	8	(Engstrom et al., 2019)	62.56	29.22	31.54	29.34	29.41	OOM	29.32	29.20	**29.01**
($\varepsilon = 4/255$)	9	(Wong et al., 2020)	55.62	26.24	28.46	26.40	26.56	OOM	26.42	26.22	**26.12**
Summary (# **bold** / 41 models)				0	0	0	0	2	3	0	36

Dataset and models. We used 41 models and 21 different defenses[1], including 25 models trained on CIFAR-10, 11 on CIFAR-100 (Krizhevsky et al., 2009), and five on ImageNet (Russakovsky et al., 2015). We also used 10 additional models trained on ImageNet with three different defenses to test the EDA's performance against more advanced models, including transformer-based architectures. We performed ℓ_∞ attacks on 10,000 images with $\varepsilon = 8/255$ for CIFAR-10 and CIFAR-100 models and on 5,000 images with $\varepsilon = 4/255$ for ImageNet models. The test images were sampled in the same way as in RobustBench. The text presents results against nine representative models for CNN-based models. Complete results for CNN-based models are described in Appendix D.

Computer specification. The experiments were conducted with two types of CPUs and a single type of GPU. The CPUs used in the experiments were Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz and Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz. The GPU used in the experiments is NVIDIA GeForce RTX 3090.

---

[1] The used models are publicly available as of robustbench v1.1.

Table 2: The comparison in the robust accuracy against robust models trained on ImageNet, including transformer-based architectures. The robust accuracy of AA is the value reported in RobustBench. The lowest robust accuracy is in bold. "sec (ratio)" columns show the runtime in seconds and the ratio of runtime to EDA.

Defense	Model	clean	AA	CAA		A^3		EDA	
		acc	acc	acc	sec (ratio)	acc	sec (ratio)	acc	sec (ratio)
(Liu et al., 2023a)	ConvNeXt-B	76.02	55.82	55.86	20,244 (1.4)	55.84	37,437 (2.7)	**55.78**	14,073 (1.0)
(Liu et al., 2023a)	Swin-B	76.16	56.16	56.14	25,913 (1.9)	56.06	20,620 (1.5)	**56.04**	13,988 (1.0)
(Debenedetti et al., 2023)	XCiT-L12	73.76	47.60	47.70	25,196 (2.0)	47.54	20,044 (1.6)	**47.44**	12,705 (1.0)
(Debenedetti et al., 2023)	XCiT-M12	74.04	45.24	45.44	13,187 (1.7)	45.22	11,067 (1.4)	**45.14**	7,884 (1.0)
(Debenedetti et al., 2023)	XCiT-S12	72.34	41.78	41.88	8,690 (1.6)	41.74	6,560 (1.2)	**41.60**	5,548 (1.0)
(Singh et al., 2023)	ConvNeXt-B+ConvStem	75.90	56.14	56.12	21,650 (1.5)	56.14	35,422 (2.5)	**56.02**	14,399 (1.0)
(Singh et al., 2023)	ConvNeXt-S+ConvStem	74.10	52.42	52.44	14,184 (1.3)	52.38	24,512 (2.3)	**52.30**	10,601 (1.0)
(Singh et al., 2023)	ConvNeXt-T+ConvStem	72.72	49.46	49.42	8,444 (1.2)	49.44	19,391 (2.7)	**49.38**	7,311 (1.0)
(Singh et al., 2023)	ViT-B+ConvStem	76.30	54.66	54.72	22,262 (1.9)	55.02	17,141 (1.4)	**54.62**	11,962 (1.0)
(Singh et al., 2023)	ViT-S+ConvStem	72.56	48.08	48.18	8,912 (1.7)	48.08	5,928 (1.1)	**48.02**	5,355 (1.0)

When we compared the performance of each attack on model A and dataset B, all compared attacks were run on the same device. The runtime comparison is thus fair.

Hyperparameters. The parameters of the ADS are $n_a = 5$ and $N_{\text{ADS}} = 4$, which are the number of pairs of $\boldsymbol{\delta}$ and L and the number of iterations, respectively. These parameters were determined based on small-scale experiments in Appendix D.1. The parameter of the MDO framework is $N = 100$. The total number of the MDO framework iterations is $n_a \times N = 500$. EDA has parameters $N_s = 10$ and K in addition to ADS and the MDO framework parameters. The experiments used $K = 9, 14,$ and 20 for CIFAR-10, CIFAR-100, and ImageNet, respectively. Unless otherwise noted, the initial point sampling ($\mathtt{x}_{\text{init}}$) in ADS, the diversification phase, and the targeted attack was Prediction Aware Sampling ($\mathtt{x}_{\text{PAS}}$, PAS), a variant of ODS described in Appendix E.

4.1 Comparison with recent attacks

The comparative experiments of EDA with the standard version of AA, CAA, AutoAE, and A^3 were conducted to investigate the performance of EDA. The parameters of the existing methods were the default values in their official codes. In addition, MT$_{\text{cos}}$, a step size variant of MT-PGD (Gowal et al., 2019), was compared with the MDO framework to analyze the characteristics of the MDO strategy. In our notation, MT$_{\text{cos}}$ is expressed as $(\mathtt{x}_{\text{org}}, \eta_{\cos}, \boldsymbol{\delta}_{\text{GD}}, L^T_{\text{CW}})$. The parameters of MT$_{\text{cos}}$ were the number of target classes, $K = 9$, and iterations per target class, $N_T = 56$. The MDO framework and MT$_{\text{cos}}$ thus spent almost the same number of queries.

EDA showed SOTA performance in less runtime. The summary in Table 1 shows that the attack performance of EDA exceeds that of recently proposed methods for around 88% of 41 CNN-based robust models. We could not finish AutoAE execution against ImageNet models due to the out-of-memory error. As shown in Figure 1, EDA showed lower robust accuracy with fewer numbers of worst-case queries than AA, CAA, AutoAE, and A^3. In addition, Figure 7 shows that EDA is 1.07-2.28, 1.46-2.87, and 18.74-38.61 times faster than A^3, CAA, and AutoAE on average. A^3 stops the search before the given query budget, depending on the models. Because of this, A^3 spent less time than EDA for some models, specifically for CIFAR-10 models. The differences in robust accuracy among CAA, AutoAE, and A^3 are approximately 0.05%. Considering these advances in robust accuracy by recent attacks, EDA's performance improvements, around 2 to 30 times speed-up and approximately 0.01 to 0.21% improvement in robust accuracy, are sufficiently large. As shown in Table 2, EDA also demonstrated better performance against 100% of 10 advanced models, including transformer-based architectures. We discuss the complexity of the compared methods in the number of queries in Appendix D.2 in detail. In the following analyses of EDA, we mainly focus on the comparison with A^3, which is sufficiently fast and strong among baseline methods.

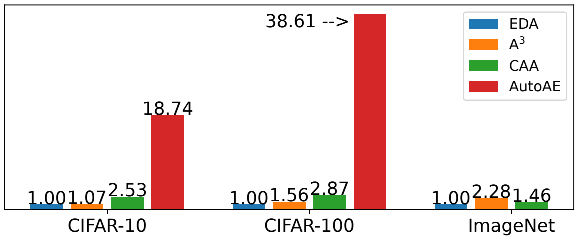


Figure 7: The average ratio of computation time to EDA.

Table 3: Comparison in robust accuracy for transfer setting. "A→B" indicates the transfer attack from model A to model B.

Transfer	clean	AA	CAA	AutoAE	A^3	EDA
Semi→MMA	84.36	71.82	73.15	73.24	70.46	**68.14**
Semi→LBGAT	88.22	71.90	74.76	74.66	73.46	**70.29**
MMA→Semi	89.69	**83.23**	85.01	84.85	85.16	83.68
MMA→LBGAT	88.22	**79.77**	82.16	81.83	82.42	80.32
LBGAT→Semi	89.69	**76.17**	80.24	80.32	79.94	77.24
LBGAT→MMA	84.36	70.16	72.07	72.09	71.28	**69.43**
Best known robust accuracy				LBGAT	MMA	Semi
in RobustBench leaderboard				52.86	41.44	59.53

EDA showed higher transferability. We chose three recent models trained on CIFAR-10, including Semi (Carmon et al., 2019), MMA (Ding et al., 2020), and LBGAT (Cui et al., 2021). Table 3 shows that EDA tends to exhibit the highest performance among baseline methods in the transfer scenarios from more robust models to less robust models. EDA showed the second-best performance in the other scenarios.

The appropriate balance of diversification and intensification is one of the reasons for EDA's high performance. According to Figure 9, EDA and MDO framework showed higher DI values than A^3. The point set with a higher value of DI is less likely to form clusters. This suggests that EDA searches a larger area than A^3 with almost the same or shorter runtime, resulting in EDA's high attack performance. Whereas A^3 uses a single $\boldsymbol{\delta}$ and L, EDA and MDO framework use multiple $\boldsymbol{\delta}$ and L. Thus, this result suggests that the attack with multiple $\boldsymbol{\delta}$ and L has a higher diversification ability than the attack with a single $\boldsymbol{\delta}$ and L. Although gradient-based attacks can achieve some degree of intensification by adjusting step size, the diversification and intensification are trade-offs given the computation budget. Figure 9 shows that $\mathrm{MT}_{\cos}$ showed higher DI than EDA. However, $\mathrm{MT}_{\cos}$ showed lower attack performance than EDA, as shown in Table 1. This result suggests that $\mathrm{MT}_{\cos}$ performs an insufficient intensification, and EDA has an appropriate balance between diversification and intensification.

4.2 Analyses of the MDO strategy

The combination of MDO and MT strategies would lead to an efficient attack. Figure 8 shows the difference between the number of queries spent by $\mathrm{MT}_{\cos}$ and the MDO framework for images that both succeeded in attack. The images that $\mathrm{MT}_{\cos}$ and the MDO framework spent the same number of queries were excluded. The positive value means $\mathrm{MT}_{\cos}$ spent more queries than the MDO framework. Figure 8 indicates the presence of images that can be successfully attacked with fewer queries by $\mathrm{MT}_{\cos}$/the MDO framework but require more queries by the MDO framework/$\mathrm{MT}_{\cos}$.

The MDO strategy mainly contributed to the attack performance of EDA. We examined the ratio of adversarial examples generated only by the MDO framework, the targeted attack a^T, and both methods to entire images that EDA succeeded but A^3 failed. In the case of CIFAR-10, the percentages of adversarial examples generated only by the MDO framework, a^T, and both methods are 58.98%, 6.71%, and 34.31%, respectively. These values are averages over the 25 models trained on CIFAR-10. Similarly, the percentages are 35.02%, 8.28%, and 56.70% for CIFAR-100 and 25.90%, 15.51%, and 58.59% for ImageNet. Here, the percentages in CIFAR-100 are averages over 11 models and those in ImageNet over five. This analysis suggests that the MDO strategy contributes to the attack performance more than the MT strategy, and it can find adversarial examples that are difficult to find for existing attacks. In addition, EDA showed larger improvements in robust accuracy in half the runtime of A^3 for models trained on ImageNet, and the MDO framework showed promising results for the same models. These results indicate that the MDO strategy may have advantages in the attack for models trained on ImageNet, which is more practical regarding image size and the number of classification classes.

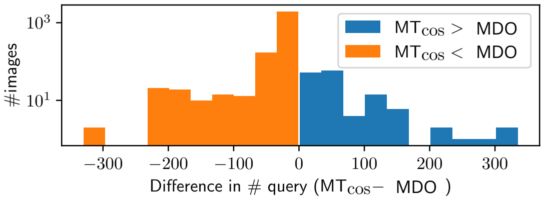


Figure 8: The difference between $\mathrm{MT}_{\mathrm{cos}}$ and the MDO framework in #queries to find adversarial examples. The model is (Carmon et al., 2019)

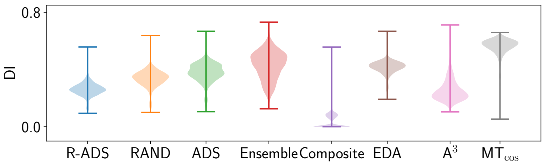


Figure 9: Violin plot of DI. The model is (Carmon et al., 2019)

Table 4: Ablation study of ADS.

	the MDO framework		
No.	ADS	R-ADS	RAND
1	**55.58**	**55.58**	55.61
2	**59.46**	59.53	59.56
3	64.32	64.54	**64.23**
4	**18.97**	18.99	18.98
5	**24.65**	24.68	24.71
6	**36.96**	37.53	37.05
7	**25.22**	25.44	**25.22**
8	**29.20**	29.26	29.56
9	**26.22**	26.36	26.26

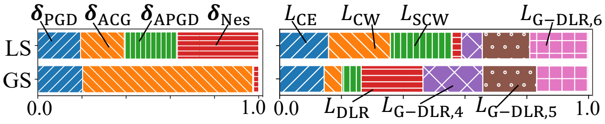


Figure 10: The ratio of $\boldsymbol{\delta}$ and L selected by ADS. LS and GS refer to the intensification and diversification phases, respectively.

4.3 Analyses of ADS

The combination of $\boldsymbol{\delta}$ and L selected by ADS brings a higher degree of diversification. We compared the attack performance of the MDO framework with three selection algorithms, including ADS, Reverse ADS (R-ADS), which finds the pairs minimizing equation 3, and uniform sampling (RAND). ADS in Figure 9 and Table 4 represents the MDO framework with the selection algorithm ADS, and the same is applied to R-ADS and uniform sampling. Figure 9 shows that the DI of the best search points obtained by the MDO framework with ADS tends to be higher than that obtained by the MDO framework with R-ADS and uniform sampling, indicating ADS may select the pairs that enhance the diversification. According to Table 4, the MDO framework with ADS showed lower robust accuracy than that with R-ADS and uniform sampling. In addition, the MDO framework with R-ADS showed significantly higher robust accuracy on several models. Although the performance difference between ADS and uniform sampling is smaller than between ADS and R-ADS, ADS can select the appropriate pair more stably than uniform sampling. These results indicate that ADS selected the pairs that diversify the search more, leading to stronger attacks.

The percentage of $\boldsymbol{\delta}$ and L selected by ADS. As shown in Figure 10, during the diversification phase, ADS tended to choose DLR loss and the search direction of ACG. In the intensification phase, CW loss and directions other than ACG were more likely to be selected. Yamamura et al. (2022) reported that ACG showed superior diversification ability compared to APGD. In addition, ACG showed better attack performance with DLR loss, whereas APGD showed better performance with CW loss. These experimental results suggest that ADS selected appropriate pairs in both the diversification and intensification phases. Similar trends were observed regardless of the dataset.

The role of D^e in the intensification phase. The correlation between D^e and the highest CW loss value was checked to investigate the influence of ADS on the intensification phase. The results show that D^e and the highest CW loss value are positively correlated, with correlation coefficients of 0.57 on average. As discussed in Section 4.4, attack performance is significantly degraded if ADS is not performed in the intensification phase. These experimental results suggest that ADS can select appropriate pairs for diversification and intensification. We further discuss the individual influence of the two terms, P^e and DI, that appear in D^e calculation on the overall ADS search performance in Appendix G.3.

Table 5: Ablation study of the MDO framework.

No.	Ensemble	Composite	MDO framework	MDO framework without ADS in the intensification phase
1	55.68	55.90	**55.58**	62.80
2	59.70	59.59	**59.46**	69.25
3	64.53	64.43	**64.32**	70.71
4	19.08	19.12	**18.97**	24.40
5	24.74	24.83	**24.65**	32.51
6	37.19	37.17	**36.96**	43.87
7	25.46	25.48	**25.22**	29.76
8	29.64	29.68	**29.20**	36.74
9	26.84	26.78	**26.22**	30.66

4.4 Ablation study of the MDO framework

Comparison with an ensemble and composite. We executed n_a attacks selected by ADS using the MDO framework, ensemble, and composite. The ensemble and composite executed the attacks $a_j = (\mathtt{x}_{\text{PAS}}, \eta_{\cos}, \boldsymbol{\delta}_{a_j^*}, L_{a_j^*})$ for $j = 1, \ldots, n_a$. The initial step size and number of iterations of each a_j were set to 2ε and $N = 100$, respectively. Table 5 shows that the attack performance of the MDO framework is higher than that of the ensemble and composite in most cases. Table 1 also shows that the attack performance of the MDO framework is higher than AA for several models in fewer queries. The violin plot for DI in Figure 9 shows that the degree of diversification is higher for the ensemble, the MDO framework, and the composite, in descending order. These experimental results indicate that the MDO framework is one of the effective implementations of the MDO strategy because it can achieve an appropriate balance of diversification and intensification than ensemble and composite.

Influence of ADS in the intensification phase. Table 5 shows the robust accuracy of the MDO framework with and without ADS execution prior to the intensification phase. The results show that the attack performance significantly drops without ADS prior to the intensification phase. This suggests that the pairs suitable for diversification and intensification differ from each other. The EDA's performance may be further improved by considering the execution order of the attacks with selected $\boldsymbol{\delta}$ and L. However, determining the execution order requires that $(n_a!)^2$ permutations be considered, which is computationally expensive. The experimental results suggest that EDA achieves a satisfactory trade-off between the computation cost and the attack performance.

4.5 Additional results and ablations

Appendix D.5 describes the analysis of EDA for the model of Ding et al. (2020), which showed different trends. The analysis using the Euclid distance-based measure in Appendix D.4 showed similar trends to DI-based analysis. Appendix G shows the result of the ablation studies, including the hyperparameter sensitivity of EDA. EDA is expected to be robust to hyperparameter settings to some extent. As described in Appendix H, EDA also showed better performance than AA and A^3 for randomized defenses.

5 Conclusion

This study empirically confirmed that different local solutions can be efficiently enumerated using various search directions and objective functions. Based on this observation, we have proposed the MDO strategy and its implementation, including ADS and the MDO framework. The experiments on robust models, including 41 CNN-based and 10 more advanced architectures, have demonstrated that the MDO strategy realized by the MDO framework has a higher diversification ability. In addition, EDA, a combination of MDO and MT strategy, showed higher attack performance than recently proposed attacks. Though more appropriate indices may exist, these results suggest that the attack designed based on the DI shows an appropriate balance of diversification and intensification, resulting in a stronger attack.

Broader Impact Statement

Deep neural networks (DNNs) are known to be vulnerable to adversarial examples. A promising defense mechanism to address this vulnerability is adversarial training, where training is performed using adversarial examples. Many adversarial examples generated by strong attack methods are required to produce robust models through adversarial training. Therefore, developing strong and fast adversarial attacks helps improve the robustness of DNNs. The EDA, which is one of the main proposals of this research, can generate a larger number of adversarial examples in a shorter time and can be used for both the robustness evaluation of defense methods and data generation in adversarial training. Thus, this research significantly contributes to the security of DNNs. The positive impact of this research is twofold. First, we can make DNNs more robust through adversarial training using the data generated by the strong attack method EDA. Second, we can more accurately evaluate the robustness of the models. The potential negative impact of this research includes possible attacks by malicious users on systems containing DNNs. However, EDA is a white-box attack that assumes the accessibility of model gradients. In addition, it is difficult to access the model gradients involved in a real system. EDA is thus unsuitable for attacking a real system. As described above, research benefits are more significant than the potential negative effects. This study helps to improve the robustness of DNNs, allowing them to be more safely applied to a broader range of applications, including safety-critical applications.

References

Sravanti Addepalli, Samyak Jain, and Venkatesh Babu Radhakrishnan. Efficient and effective augmentation strategy for adversarial training. In *NeurIPS*, 2022.

Insaf Adjabi, Abdeldjalil Ouahabi, Amir Benzaoui, and Abdelmalik Taleb-Ahmed. Past, present, and future of face recognition: A review. *Electronics*, 9(8):1188, 2020.

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. In *NeurIPS*, 2020.

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In *ECCV*, 2020.

Nikolaos Antoniou, Efthymios Georgiou, and Alexandros Potamianos. Alternating objectives generates stronger pgd-based adversarial attacks. *CoRR*, abs/2212.07992, 2022.

Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *SP*, 2017.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C. Duchi, and Percy Liang. Unlabeled data improves adversarial robustness. In *NeurIPS*, 2019.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, 2020a.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020b.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *NeurIPS*, 2021.

Jiequan Cui, Shu Liu, Liwei Wang, and Jiaya Jia. Learnable boundary guided adversarial training. In *ICCV*, 2021.

Edoardo Debenedetti, Vikash Sehwag, and Prateek Mittal. A light recipe to train robust vision transformers. In *SaTML*, 2023.

Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. MMA training: Direct input space margin maximization through adversarial training. In *ICLR*, 2020.

Minjing Dong, Xinghao Chen, Yunhe Wang, and Chang Xu. Random normalization aggregation for adversarial defense. In *NeurIPS*, 2022.

Xingye Dong, Ping Chen, Houkuan Huang, and Maciek Nowak. A multi-restart iterated local search algorithm for the permutation flow shop problem minimizing total flow time. *Computers & Operations Research*, 40(2):627–632, 2013.

Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019.

Yonggan Fu, Qixuan Yu, Meng Li, Vikas Chandra, and Yingyan Lin. Double-win quant: Aggressively winning robustness of quantized deep neural networks via random precision training and inference. In *ICML*, 2021.

Ruize Gao, Jiongxiao Wang, Kaiwen Zhou, Feng Liu, Binghui Xie, Gang Niu, Bo Han, and James Cheng. Fast and reliable evaluation of adversarial robustness with minimum-margin attack. In *ICML*, 2022.

Fred Glover and Michele Samorani. Intensification, diversification and learning in metaheuristic optimization. *Journal of Heuristics*, 2019.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy A. Mann, and Pushmeet Kohli. An alternative surrogate loss for pgd-based adversarial testing. *CoRR*, abs/1910.09338, 2019.

Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy A. Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *CoRR*, abs/2010.03593, 2020.

Abhishek Gupta, Alagan Anpalagan, Ling Guan, and Ahmed Shaharyar Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *ICML*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *ICLR*, 2020.

Chang Liu, Yinpeng Dong, Wenzhao Xiang, Xiao Yang, Hang Su, Jun Zhu, Yuefeng Chen, Yuan He, Hui Xue, and Shibao Zheng. A comprehensive study on robustness of image classification models: Benchmarking and rethinking. *arXiv preprint arXiv:2302.14301*, 2023a.

Jiabao Liu, Qixiang Zhang, Kanghua Mo, Xiaoyu Xiang, Jin Li, Debin Cheng, Rui Gao, Beishui Liu, Kongyang Chen, and Guanjie Wei. An efficient adversarial example generation algorithm based on an accelerated gradient iterative fast gradient. *Comput. Stand. Interfaces*, 82:103612, 2022a.

Ruofan Liu, Yun Lin, Xianglin Yang, Siang Hwee Ng, Dinil Mon Divakaran, and Jin Song Dong. Inferring phishing intention via webpage appearance and dynamics: A deep vision based approach. In *USENIX Security*, 2022b.

Shengcai Liu, Fu Peng, and Ke Tang. Reliable robustness evaluation via automatically constructed attack ensembles. In *AAAI*, 2023b.

Ye Liu, Yaya Cheng, Lianli Gao, Xianglong Liu, Qilong Zhang, and Jingkuan Song. Practical evaluation of adversarial robustness via adaptive auto attack. In *CVPR*, 2022c.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Xiaofeng Mao, Yuefeng Chen, Shuhui Wang, Hang Su, Yuan He, and Hui Xue. Composite adversarial attacks. In *AAAI*, 2021.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: uniform manifold approximation and projection. *J. Open Source Softw.*, 3(29):861, 2018.

Yurii Nesterov. *Introductory Lectures on Convex Optimization.* Springer New York, NY, 2004.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Tianyu Pang, Xiao Yang, Yinpeng Dong, Taufik Xu, Jun Zhu, and Hang Su. Boosting adversarial training with hypersphere embedding. In *NeurIPS*, 2020.

Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, 2000.

Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A. Mann. Data augmentation can improve robustness. In *NeurIPS*, 2021.

Leslie Rice, Eric Wong, and J. Zico Kolter. Overfitting in adversarially robust deep learning. In *ICML*, 2020.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? In *NeurIPS*, 2020.

Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. HYDRA: pruning adversarially robust neural networks. In *NeurIPS*, 2020.

Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In *ICLR*, 2022.

Naman D Singh, Francesco Croce, and Matthias Hein. Revisiting adversarial training for imagenet: Architectures, training and generalization across threat models. *arXiv preprint arXiv:2303.01870*, 2023.

Chawin Sitawarin, Supriyo Chakraborty, and David A. Wagner. SAT: improving adversarial training via curriculum-based loss smoothing. In *AISec@CCS*, 2021.

Kaustubh Sridhar, Oleg Sokolsky, Insup Lee, and James Weimer. Improving neural network robustness via persistency of excitation. In *ACC*, 2022.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

Yusuke Tashiro, Yang Song, and Stefano Ermon. Diversity can be transferred: Output diversification for white- and black-box attacks. In *NeurIPS*, 2020.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *ICLR*, 2020.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020.

Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. In *NeurIPS*, 2020.

Zhuoer Xu, Guanghui Zhu, Changhua Meng, shiwen cui, Zhenzhe Ying, Weiqiang Wang, Ming GU, and Yihua Huang. A2: Efficient automated attacker for boosting adversarial training. In *NeurIPS*, 2022.

Keiichiro Yamamura, Haruki Sato, Nariaki Tateiwa, Nozomi Hata, Toru Mitsutake, Issa Oe, Hiroki Ishikura, and Katsuki Fujisawa. Diversified adversarial attacks based on conjugate gradient method. In *ICML*, 2022.

Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. In *NeurIPS*, 2019.

Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan S. Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *ICML*, 2020.

Appendix

This supplementary material provides additional information as follows.

1. The summary of abbreviations and mathematical notations defined in the main text (Tables 6 and 7).
2. More information about related work (Appendix A).
3. The proposed search direction and objective function (Appendix B).
4. Additional results of the analysis in Section 3.1 (Appendix C).
5. Complete results of the experiments in Section 4 (Appendix D).
6. The details of Prediction Aware Sampling (Appendix E).
7. The details of the targeted attack used in EDA (Appendix F).
8. Ablation study of EDA (Appendix G).
9. Experiments on randomized defenses (Appendix H).
10. Limitations and assumptions (Appendix I).

Algorithm 4 Basic white-box Attacks without multi-restart

```
Require: $a = (\mathrm{x}, \eta, \boldsymbol{\delta}, L)$: an attack, $N$: maximum iteration, $P_{\mathcal{S}}$: a projection function, $g$: DNN, $\mathcal{I}$: test samples
Ensure: $X^a$: a set of best search points, $\Pi^a$: a set of classification labels with the highest probabilities except for the correct classification class
for $i = 1, \dots, |\mathcal{I}|$ do
    $\Pi_i^a \leftarrow \emptyset, \boldsymbol{x}_{\mathrm{org}} \leftarrow \boldsymbol{x}_i \in \mathcal{I}, c_i \in Y \leftarrow$ correct classification label corresponding to $\boldsymbol{x}_i$
    $\boldsymbol{x}^{(0)} \leftarrow$ initialize by $\mathrm{x}$, $\boldsymbol{x}_i^{adv} \leftarrow \boldsymbol{x}^{(0)}, f_i^{best} \leftarrow L_{\mathrm{CW}}(g(\boldsymbol{x}^{(0)}), c_i)$
    for $k = 0, \dots, N-1$ do
        Update $\eta^{(k)}$ and $\boldsymbol{\delta}^{(k)}$ by update rule $\eta$ and search direction $\boldsymbol{\delta}$.
        $\boldsymbol{x}^{(k+1)} \leftarrow P_{\mathcal{S}}\left(\boldsymbol{x}^{(k)} + \eta^{(k)} \cdot \boldsymbol{\delta}^{(k)}\right)$
        Update $\boldsymbol{x}_i^{adv}$ and $f_i^{best}$ {$\boldsymbol{x}_i^{adv}$ is equivalent to $(\boldsymbol{x}_i)_{a,1}^*$ in the main paper.}
        $\Pi_i^a \leftarrow \Pi_i^a \cup \{\arg\max_{j \neq c_i} g_j(\boldsymbol{x}^{(k+1)})\}$
    end for
    $X^*(\boldsymbol{x}_i, a, 1) \leftarrow \{\boldsymbol{x}_i^{adv}\}$
    $X_i^a \leftarrow X^*(\boldsymbol{x}_i, a, 1)$
end for
```

Table 6: Summary of abbreviations

Existing attack techniques	
PGD	Projected Gradient Descent
MT-PGD	MultiTargeted-PGD
APGD	Auto-PGD
ACG	Auto Conjugate Gradient attack
AA	AutoAttack
CAA	Composite Adversarial Attack
A^3	Adaptive Auto Attack
CE	Cross-entropy
DLR	Difference of Logit Ratio
ODS	Output Diversified Sampling
DI	Diversity Index
Proposed methods	
G-DLR	Generalized-DLR
NAG	Nesterov's accelerated gradient
PAS	Prediction Aware Sampling
MDO	Multi-directions/objectives
ADS	Automated Diversified Sampling
MDO framework	Search framework for the MDO strategy
EDA	Efficient Diversified Attack
Others	
DNN	Deep neural network
SOTA	State-of-the-art
MT	Multi-target
UMAP	Uniform Manifold Approximation and Projection

A More information about related work

A.1 Procedure of basic white-box attacks

This section provides a detailed explanation of basic white-box attacks introduced in Section 2 in the main paper. We characterize the basic white-box attacks by the tuple $a = (\mathtt{x}, \eta, \boldsymbol{\delta}, L)$, where $\mathtt{x}$ is the initial point sampling, η is the step size update rule, $\boldsymbol{\delta}$ is the search direction, and L is the objective function for the attack. We refer to the tuple a as an attack a. Algorithm 4 shows the procedure of the basic white-box attacks $a = (\mathtt{x}, \eta, \boldsymbol{\delta}, L)$ without multi-restart. We assume that the basic white-box attack a returns a set of best search points X^a, a set of best CW loss values F^a, and a set of classification labels with the highest probability except for the correct classification labels during the search Π^a. The basic white-box attacks iteratively update the search point $\boldsymbol{x}^{(k)}$ as lines 5 and 6 in Algorithm 4 to search for adversarial examples. At each iterations k, the best search point $\boldsymbol{x}_i^{adv}$ is updated if $L(g(\boldsymbol{x}_i^{adv}), c) \leq L(g(\boldsymbol{x}^{(k)}), c)$. After the N iterations of updates, the attack procedure is finished. For the attacks with a multi-restart strategy, Algorithm 4 is repeated from different initial points.

A.2 Search directions

Projected Gradient Descent Projected Gradient Descent (PGD) (Madry et al., 2018) is the most fundamental adversarial attack based on the steepest gradient descent. The search direction of PGD is computed as follows:

$$\boldsymbol{\delta}_{\mathrm{PGD}}^{(k)} = \mathrm{sign}\left(\nabla L(g(\boldsymbol{x}^{(k)}), c)\right) \tag{4}$$

Table 7: Summary of mathematical notations

Defined in Section 2.1	
$g : D \to \mathbb{R}^C$	Locally differentiable C-classifier.
$\boldsymbol{x}_{\text{org}} \in D$	Original input.
$c \in Y = \{1, 2, \ldots, C\}$	The correct classification label of $\boldsymbol{x}_{\text{org}}$.
$d : D \times D \to \mathbb{R}$	A distance function.
$\mathcal{S}$	The feasible region.
$L : \mathbb{R}^C \times Y \to \mathbb{R}$	Objective function (untargeted attack).
$L^T : \mathbb{R}^C \times Y \times Y \to \mathbb{R}$	Objective function (targeted attack).
Defined in Section 2.2	
$P_{\mathcal{S}} : D \to \mathcal{S}$	Projection onto the feasible region $\mathcal{S}$.
$\mathtt{x}$	Initial point sampling.
η	Step size update rule.
$\boldsymbol{\delta}$	Search direction/Update formula.
$\boldsymbol{x}^{(k)} \in D$	Search point at iteration k.
$\eta^{(k)} \in \mathbb{R}$	Step size at iteration k.
$\boldsymbol{\delta}^{(k)} \in \mathbb{R}^n$	Search direction at iteration k.
$a = (\mathtt{x}, \eta, \boldsymbol{\delta}, L)$	An attack.
Defined in Section 3.1	
$\pi_q \in Y$	The class label that has the q-th largest value of $g(\boldsymbol{x})$.
$\mathcal{D}$	A set of update formulas.
$\mathcal{L}$	A set of objective functions.
$\mathcal{A}(\mathtt{x}, \eta)$	A set of attacks with initial point sampling $\mathtt{x}$ and step size update rule η.
$(\boldsymbol{x})_{a,r}^{(k)} \in D$	A search point in the k-th iteration of an attack a starting at the r-th initial point.
$(\boldsymbol{x})_{a,r}^{*} = \underset{k=1,\ldots,N_{\max}}{\arg\max}\ L_{\text{CW}}(g((\boldsymbol{x})_{a,r}^{(k)}), c)$	The best search point found by an attack a starting at r-th initial point.
$X^*(\boldsymbol{x}_i, a, R) = \left\{(\boldsymbol{x}_i)_{a,r}^{*} \mid r = 1, \ldots, R\right\}$	The set of best search points found by an attack a with R restarts.
$DI(X, M)$	Diversity Index of a point set X. M is the size of feasible region.
Defined in Section 3.2	
$e = \{a_j\}_{j=1}^{n_a}$	A combination of attacks.
$X^{**}(\boldsymbol{x}_i, e) = \cup_{a \in e} X^*(\boldsymbol{x}_i, a, 1)$	The set of best search points found by a combination of attacks.

Also, Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) updates towards the same direction.

Auto-PGD Auto-PGD (APGD) (Croce & Hein, 2020b) is a PGD variant that adjusts the step size and updates towards momentum direction in addition to the PGD's search direction. The search direction of APGD is defined as follows.

$$\boldsymbol{z}^{(k)} = P_{\mathcal{S}}\left(\boldsymbol{x}^{(k)} + \eta^{(k)} \boldsymbol{\delta}_{\text{PGD}}^{(k)}\right), \tag{5}$$

$$\boldsymbol{\delta}_{\text{APGD}}^{(k)} = \alpha(\boldsymbol{z}^{(k)} - \boldsymbol{x}^{(k)}) + (1 - \alpha)(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}), \tag{6}$$

where α is a coefficient of momentum term. APGD uses $\alpha = 1$ for the first iteration and $\alpha = 0.75$ for the remaining iterations. The step size $\eta^{(k)}$ is halved if the following conditions are satisfied at the $w_j \in W$ iteration, with the initial value $\eta^{(0)} = 2\varepsilon$.

1. $\displaystyle\sum_{i=w_{j-1}}^{w_j-1} \mathbf{1}_{L\left(g(\boldsymbol{x}^{(i+1)}),c\right)>L\left(g(\boldsymbol{x}^{(i)}),c\right)} < \rho \cdot (w_j - w_{j-1})$
2. $L_{\max}\left(g(\boldsymbol{x}^{(w_{j-1})}), c\right) = L_{\max}\left(g(\boldsymbol{x}^{(w_j)}), c\right)$ and $\eta^{(w_{j-1})} = \eta^{(w_j)}$

The sequence of checkpoints W is computed based on the following gradual equation depending on the total number of iterations N_{iter}. $p_0 = 0, p_1 = 0.22, p_{j+1} = p_j + \max\{p_j - p_{j-1} - 0.03, 0.06\}, w_j = \lceil p_j N_{iter} \rceil$. In our notation, η_{APGD} denotes this step size updating rule.

Auto-Conjugate Gradient attack Auto-Conjugate Gradient (ACG) attack (Yamamura et al., 2022) is inspired by the Conjugate Gradient method for nonlinear optimization problems. ACG performs a more diversified search than the attacks based on the steepest descent. The search direction of ACG is as follows.

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

$$\beta^{(k)} = -\frac{\nabla L(\boldsymbol{x}^{(k)}, c)^T \boldsymbol{y}^{(k)}}{(\boldsymbol{y}^{(k)})^T \boldsymbol{\delta}_{\mathrm{ACG}}^{(k-1)}} \tag{8}$$

$$\boldsymbol{\delta}_{\mathrm{ACG}}^{(k)} = \nabla L(\boldsymbol{x}^{(k)}, c) + \beta^{(k)} \boldsymbol{\delta}_{\mathrm{ACG}}^{(k-1)} \tag{9}$$

Moreover, Yamamura et al. (2022) proposed the *Diversity Index(DI)* to quantify the degree of diversification during the attacks. DI is defined as

$$\mathrm{DI}(X, M) := \frac{1}{M} \int_0^M h(\theta; X)\ d\theta, \tag{10}$$

where $M = \sup\{\|\boldsymbol{x} - \boldsymbol{y}\|_2 \mid \boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}\}$ is the size of the feasible region, X is the set of search points, and $h(\theta; X)$ is the function of θ based on the global clustering coefficient of the graph $G(X, \theta) = (X, E(\theta)) = (X, \{(u, v) \mid \boldsymbol{u}, \boldsymbol{v} \in X, \|u - v\|_2 \leq \theta\})$.

A.3 Objective functions

Cross-entropy loss The untargeted version of cross-entropy (CE) loss is defined as follows.

$$L_{\mathrm{CE}}(g(\boldsymbol{x}), c) = -g_c(\boldsymbol{x}) + \log\left(\sum_{j \neq c} \exp\left(g_j(\boldsymbol{x})\right)\right) \tag{11}$$

Also, the targeted version of CE loss is defined as follows.

$$L_{\mathrm{CE}}^{T}(g(\boldsymbol{x}), c, t) = g_t(\boldsymbol{x}) - \log\left(\sum_{j \neq t} \exp\left(g_j(\boldsymbol{x})\right)\right), \tag{12}$$

where t denotes the target label of misclassification. CE loss is known to be sensitive to the scaling of the logit, i.e., the attack performance significantly varies depending on the scaling of the logit (Carlini & Wagner, 2017; Croce & Hein, 2020b).

CW loss The untargeted version of CW loss is defined as follows.

$$L_{\mathrm{CW}}(g(\boldsymbol{x}), c) = \max_{j \neq c} g_j(\boldsymbol{x}) - g_c(\boldsymbol{x}) \tag{13}$$

Table 8: Characteristics of PGD-like attacks, AA, A^3, and EDA. multi-L denotes the diversification using multiple objective functions, and multi-$\boldsymbol{\delta}$ denotes the diversification using multiple search directions.

Attacks	Diversification				Intensification	Runtime
	multi-L	multi-$\boldsymbol{\delta}$	multi-restart	multi-target		
PGD-like	-	-	✓	-	✓	short
AA	✓	-	✓	✓	✓	long
CAA	✓	-	✓	-	✓	long
AutoAE	✓	-	✓	-	✓	long
A^3	-	-	✓	-	✓	short
EDA	✓	✓	✓	✓	✓	short

Also, the targeted version of CW loss is defined as follows.

$$L^T_{\mathrm{CW}}(g(\boldsymbol{x}), c, t) = g_t(\boldsymbol{x}) - g_c(\boldsymbol{x}), \tag{14}$$

where t denotes the target label of misclassification.

Difference of Logit Ratio loss The untargeted version of the Difference of Logit Ratio (DLR) loss is defined as follows.

$$L_{\mathrm{DLR}}(g(\boldsymbol{x}), c) = \frac{\max_{j \neq c} g_j(\boldsymbol{x}) - g_c(\boldsymbol{x})}{g_{\pi_1} - g_{\pi_3}}, \tag{15}$$

where π_q denotes the classification label with q-th highest value in $g(\boldsymbol{x})$. Also, the targeted version of DLR loss is defined as follows.

$$L^T_{\mathrm{DLR}}(g(\boldsymbol{x}), c, t) = \frac{g_t(\boldsymbol{x}) - g_c(\boldsymbol{x})}{g_{\pi_1} - (g_{\pi_3}(\boldsymbol{x}) + g_{\pi_4}(\boldsymbol{x}))/2}, \tag{16}$$

where t denotes the target label of misclassification.

A.4 Comparison of existing attacks and EDA

Table 8 summarizes the characteristics of PGD-like attacks, Auto Attack (AA) (Croce & Hein, 2020b), Composite Adversarial Attack (CAA) Mao et al. (2021), AutoAE Liu et al. (2023b), Adaptive Auto Attack (A^3) (Liu et al., 2022c), and EDA in terms of diversification, intensification, and computational cost. The attacks in Table 8 perform well in intensification because they include gradient-based attacks with appropriate step size management. Although PGD-like attacks have several variations, this section describes the representative one in Table 8. PGD-like attacks and A^3 use multi-restart for diversification, and both attacks spend a relatively short computational time. However, A^3 outperforms PGD-like attacks because A^3 uses better initial point sampling. AA considers multiple objective functions and multi-target attacks for diversification in addition to multi-restart. Although AA achieves a high attack success rate, AA is computationally expensive because AA consists of four attacks, including APGD with untargeted CE loss, APGD with targeted DLR loss, FAB attack (Croce & Hein, 2020a), and square attack (Andriushchenko et al., 2020). CAA and AutoAE also combine multiple attacks by solving an additional optimization problem. CAA executes MultiTargeted (MT) attack Gowal et al. (2019) and CW attack Carlini & Wagner (2017) as a composite of attacks. AutoAE runs APGD with CE loss, APGD with DLR loss, FAB, and MT attack as an ensemble. In the sense that AA, CAA, and AutoAE use different types of attacks, we can consider that they employ the diversification strategy based on multi-$\boldsymbol{\delta}$. However, these methods use only a single search direction to solve (2) in the white-box setting. Therefore, we do not consider AA an attack with a diversification strategy based on multi-$\boldsymbol{\delta}$. In contrast, EDA, the proposed attack, uses all diversification strategies listed in Table 8 and achieves a higher attack success rate in a short computation time.

B The proposed search direction and objective function

B.1 Search direction inspired by Nesterov's Accelerated Gradient

Although some attacks were inspired by Nesterov's accelerated gradient (Nesterov, 2004), most of them apply constant value to the coefficient of momentum (Lin et al., 2020; Liu et al., 2022a). However, the original Nesterov's accelerated gradient method determines the coefficient of momentum term by solving the quadratic equations. So then we try to adopt Nesterov's accelerated gradient to ℓ_∞ attacks. Mathematically, $\boldsymbol{\delta}_{\text{Nes}}$ is computed by the following equations.

$$\rho^{(k)} \text{ is a positive solution of } (\rho^{(k)})^2 = (1-\rho^{(k)})(\rho^{(k-1)})^2 \tag{17}$$

$$\gamma^{(k)} \leftarrow \frac{\rho^{(k-1)}\left(\rho^{(k-1)}-1\right)}{\rho^{(k)}+(\rho^{(k-1)})^2} \tag{18}$$

$$\tilde{\boldsymbol{x}}^{(k)} \leftarrow \boldsymbol{x}^{(k)} + \gamma^{(k)}\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right) \tag{19}$$

$$\boldsymbol{\delta}_{\text{Nes}}^{(k+1)} \leftarrow \operatorname{sign}\left(\nabla L(g(\tilde{\boldsymbol{x}}^{(k)}), c)\right) \tag{20}$$

B.2 Generalized-DLR loss

We generalize DLR loss by extending the denominator of DLR loss from $g_{\pi_1}(\boldsymbol{x}) - g_{\pi_3}(\boldsymbol{x})$ to $g_{\pi_1}(\boldsymbol{x}) - g_{\pi_q}(\boldsymbol{x})$. $\pi_q \in Y$ denotes the class label that has the q-th largest value of $g(\boldsymbol{x})$. More precisely, $L_{\text{G-DLR},q}$ is defined as

$$L_{\text{G-DLR},q}(g(\boldsymbol{x}), c) = -\frac{g_c(\boldsymbol{x}) - g_{\pi_2}(\boldsymbol{x})}{g_{\pi_1}(\boldsymbol{x}) - g_{\pi_q}(\boldsymbol{x})}. \tag{21}$$

The motivation for proposing generalized-DLR loss is to increase diversity in the output space. For the search points with larger values of G-DLR loss, the denominator/numerator will be smaller/larger value, respectively. The small value of denominator $g_{\pi_1}(\boldsymbol{x}) - g_{\pi_q}(\boldsymbol{x})$ means that the values of $g_{\pi_1}(\boldsymbol{x}), g_{\pi_2}(\boldsymbol{x}), \dots g_{\pi_q}(\boldsymbol{x})$ are close. Therefore, the large and small relationships between the predicted probability of classification classes are expected to be easier to variate than in the case of DLR loss.

C Additional results of the analysis in section 3.1

C.1 Diversity index for the set of best search points

The main paper only includes the results for the model proposed by Sehwag et al. (2022). This section describes the violin plot of DI for several models (Andriushchenko & Flammarion, 2020; Sehwag et al., 2022; Engstrom et al., 2019; Salman et al., 2020). Figure 11 shows the violin plot of DI obtained by attacking the robust models. According to Figure 11, the best point sets obtained by attacks with a single search direction and objective function have similar DI value trends.

C.2 Visualization of the best search points via UMAP

This section describes the 2D visualization of the best search points using UMAP for several models (Andriushchenko & Flammarion, 2020; Sehwag et al., 2022; Engstrom et al., 2019; Salman et al., 2020). Figure 12 shows the 2D visualization of the best search points obtained by attacking the robust models trained on CIFAR-10. Figure 13 shows the 2D visualization of the best search points obtained by attacking the robust models trained on ImageNet. According to Figures 12 and 13, the best point sets obtained by attacks with different search directions and objective functions tend to form different clusters. The points determined to belong to the same cluster due to clustering using the X-means (Pelleg & Moore, 2000) are also plotted close together in the visualization using UMAP. These results suggest that 2D visualizations using UMAP are expected to reflect the actual distribution of search points.

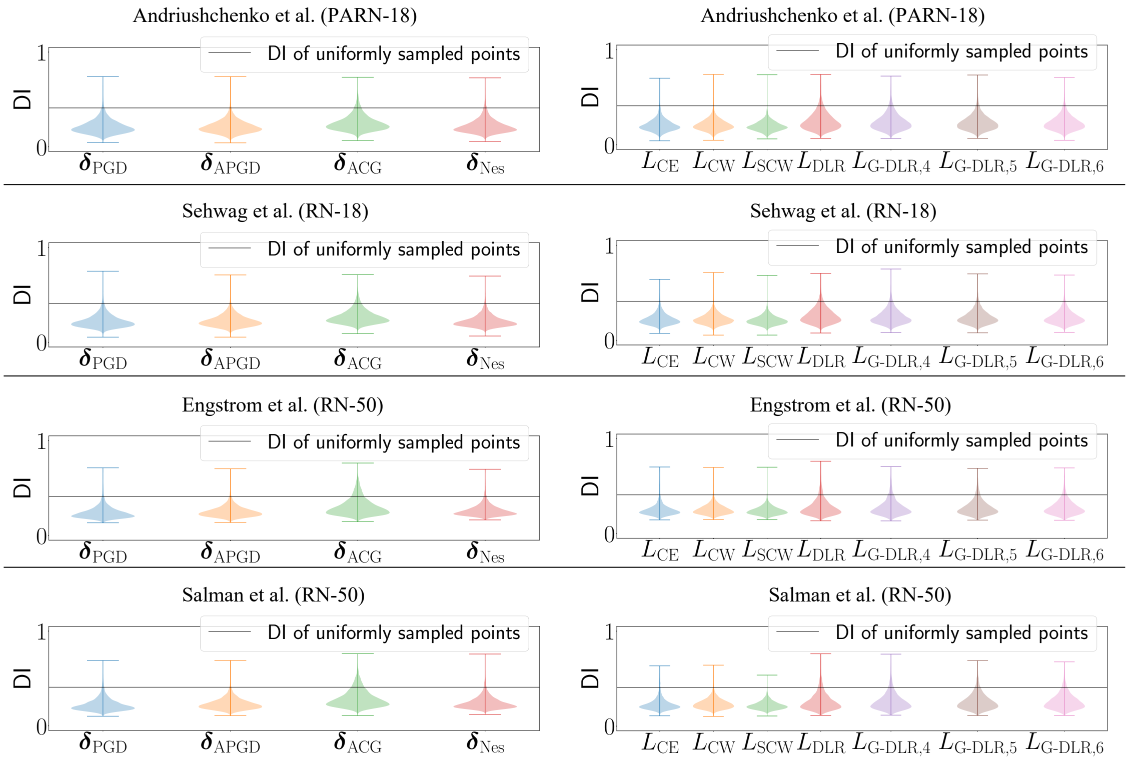


Figure 11: Violin plot of DI obtained by attacking the models proposed by (Sehwag et al., 2022; Andriushchenko & Flammarion, 2020) for CIFAR-10 and (Engstrom et al., 2019; Salman et al., 2020) for ImageNet.

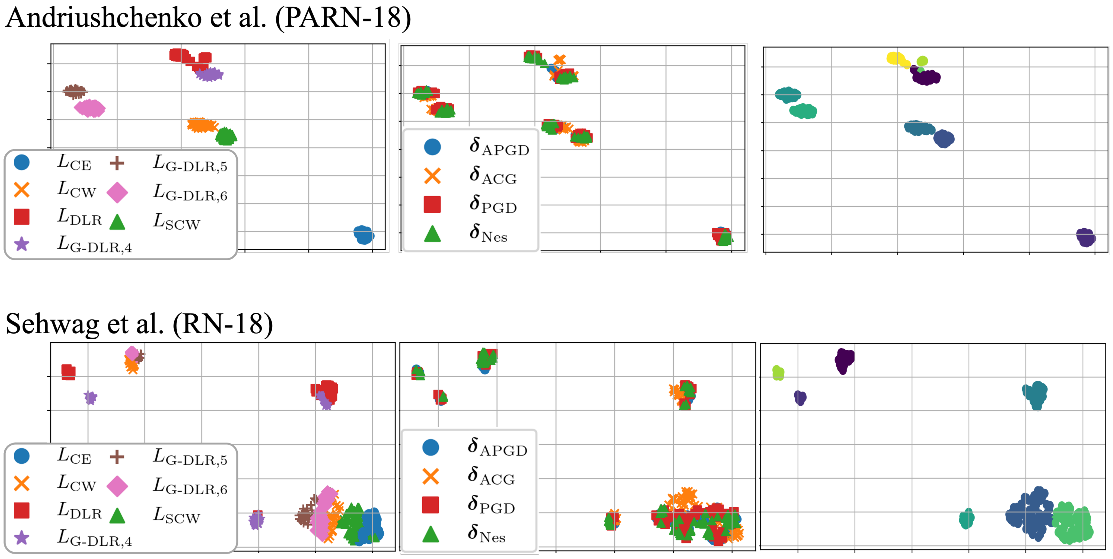


Figure 12: 2D visualization of the best search points obtained by attacking the models proposed by (Sehwag et al., 2022; Andriushchenko & Flammarion, 2020). The dataset is CIFAR-10. The same color in the left/center figure represents points obtained using the same objective function/search direction, respectively. The same color in the right figure shows the points determined by X-means to belong to the same cluster.

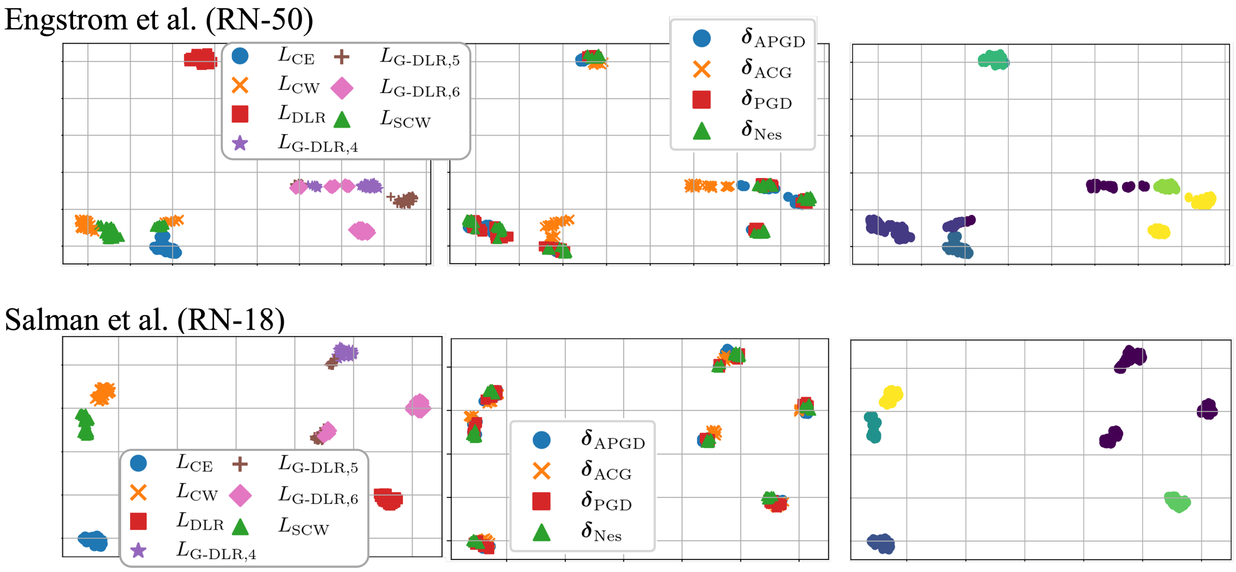


Figure 13: 2D visualization of the best search points obtained by attacking the models proposed by (Engstrom et al., 2019; Salman et al., 2020). The dataset is ImageNet. The same color in the left/center figure represents points obtained using the same objective function/search direction, respectively. The same color in the right figure shows the points determined by X-means to belong to the same cluster.

C.3 The reason why we used UMAP

The objective of the qualitative evaluation using UMAP is to know how the best points obtained by attacks using different objective functions/search directions are distributed and form different clusters. In order to achieve this goal, it is necessary to consider the distance between any two points and the distance between clusters. We have tried quantitative evaluation. However, we finally chose qualitative evaluation using UMAP because quantitative evaluations based on indicators such as objective values or DI are difficult to achieve our objective for the reasons described below. First, in adversarial attacks, distant points may show the same objective value or close points may show very different objective values because the adversarial attack is a maximization problem with many local optimums. Therefore, quantitative evaluation using objective values is considered difficult. In addition, DI cannot consider the distance between clusters because DI shows low values when a point set forms one or more clusters. Another possible evaluation method is clustering, such as k-means, but this is a qualitative evaluation as with UMAP. UMAP is a dimensionality reduction method that preserves the distance information in the original space as much as possible so it can reflect important information, such as the distance between any two points or clusters. Therefore, we think that the qualitative evaluation by UMAP provides convincing results.

D Complete results of the experiments

Table 9 shows the complete results of the experiments in Section 4.1, as described in Table 1. Table 10 shows the runtime of CAA, AutoAE, A^3, and EDA. Table 11 is the complete results of the experiments in Sections 4.3 and 4.4, as described in Table 4. To investigate the stability of the proposed methods, we report the mean and standard deviation from five runs with different random seeds against 41 CNN-based models in Section 4.1 and Appendix G.2. The results in Section 4.1 and Appendix G.2 suggest a stable performance of the proposed methods. Owing to the computation cost, the remaining experiments were conducted with a single fixed random seed. Table 12 shows the quantified degree of diversification of A^3, the MDO framework with ADS, the MDO framework with R-ADS, the MDO framework with uniform sampling, and EDA. Table 13 provides the complete results for the transferability evaluation of EDA described in Table 3.

Figure 14 shows the difference between $\mathrm{MT}_{\mathrm{cos}}$ and the MDO framework in #queries to find adversarial examples for some models. Figure 15 shows the violin plot of DI for several models.

Table 9: Average robust accuracy over five runs. The lowest accuracies are in bold. RN: ResNet, WRN: WideResNet, PARN: PreActResNet, Δ: A^3-EDA. The "EDA/A^3" column is the same as the "ratio" column in Table 1.

CIFAR-10 ($\varepsilon = 8/255$)	model	clean	AA	ACG	$\mathrm{MT}_{\mathrm{cos}}$	CAA	AutoAE	A^3	EDA
Andriushchenko & Flammarion (2020)	PARN-18	79.84	43.93	45.16	43.96	43.93	43.93	43.96±0.00	43.85±0.02
Addepalli et al. (2022)	RN-18	85.71	52.48	52.87	52.48	52.48	52.46	52.46±0.02	52.43±0.03
Sehwag et al. (2022)	RN-18	84.59	55.54	56.19	55.54	55.53	55.51	55.53±0.01	55.49±0.01
Engstrom et al. (2019)	RN-50	87.03	49.25	50.88	49.21	49.24	49.21	49.25±0.02	49.10±0.03
Carmon et al. (2019)	WRN-28-10	89.69	59.53	60.10	59.54	59.50	59.46	59.44±0.01	59.40±0.01
Gowal et al. (2020)	WRN-28-10	89.48	62.80	63.18	62.86	62.78	62.80	62.77±0.01	62.75±0.02
Hendrycks et al. (2019)	WRN-28-10	87.11	54.92	55.71	54.90	54.88	54.81	54.85±0.01	54.77±0.02
Rebuffi et al. (2021)	WRN-28-10	87.33	60.75	61.23	60.80	60.74	60.71	60.72±0.01	60.64±0.01
Sehwag et al. (2020)	WRN-28-10	88.98	57.14	57.68	57.18	57.16	57.10	57.14±0.02	57.03±0.01
Sridhar et al. (2022)	WRN-28-10	89.46	59.66	60.22	59.67	59.60	59.60	59.56±0.01	59.46±0.02
Wang et al. (2020)	WRN-28-10	87.50	56.29	57.45	56.38	56.29	56.23	56.28±0.01	56.15±0.02
Wu et al. (2020)	WRN-28-10	88.25	60.04	60.35	60.06	60.00	59.97	60.02±0.01	59.94±0.01
Ding et al. (2020)	WRN-28-4	84.36	41.44	44.36	41.99	41.67	41.48	41.24±0.06	41.74±0.06
Addepalli et al. (2022)	WRN-34-10	88.71	57.81	58.43	57.77	57.77	57.74	57.73±0.01	57.69±0.02
Sehwag et al. (2022)	WRN-34-10	86.68	60.27	61.09	60.31	60.26	60.26	60.22±0.01	60.18±0.02
Sitawarin et al. (2021)	WRN-34-10	86.84	50.72	51.94	50.75	50.70	50.64	50.69±0.02	50.59±0.01
Zhang et al. (2019)	WRN-34-10	87.20	44.83	45.76	44.69	44.65	44.57	44.63±0.03	44.51±0.02
Zhang et al. (2020)	WRN-34-10	84.52	53.51	54.00	53.52	53.48	53.48	53.46±0.01	53.42±0.02
Sridhar et al. (2022)	WRN-34-15	86.53	60.41	60.93	60.43	60.39	60.35	60.38±0.01	60.32±0.01
Gowal et al. (2020)	WRN-34-20	85.64	56.86	57.15	56.83	56.82	56.85	56.81±0.01	56.79±0.03
Pang et al. (2020)	WRN-34-20	85.14	53.74	54.66	53.71	53.72	53.67	53.69±0.01	53.66±0.01
Rice et al. (2020)	WRN-34-20	85.34	53.42	54.33	53.39	53.36	53.35	53.38±0.01	53.34±0.01
Gowal et al. (2020)	WRN-70-16	85.29	57.20	57.60	57.15	57.16	57.19	57.11±0.01	57.12±0.01
Gowal et al. (2020)	WRN-70-16	91.10	65.88	66.29	65.96	65.87	65.80	65.85±0.01	65.83±0.01
Rebuffi et al. (2021)	WRN-70-16	88.54	64.25	64.80	64.28	64.23	64.19	64.24±0.01	64.20±0.03
CIFAR-100 ($\varepsilon = 8/255$)	model	clean	AA	ACG	$\mathrm{MT}_{\mathrm{cos}}$	CAA	AutoAE	A^3	EDA
Rice et al. (2020)	PARN-18	53.83	18.95	19.48	18.99	18.97	18.91	18.89±0.00	18.88±0.01
Hendrycks et al. (2019)	WRN-28-10	59.23	28.42	29.51	28.43	28.44	28.35	28.32±0.02	28.27±0.02
Rebuffi et al. (2021)	WRN-28-10	62.41	32.06	32.73	32.07	32.05	32.02	32.00±0.02	31.94±0.03
Addepalli et al. (2022)	WRN-34-10	68.75	31.85	32.38	31.80	31.80	31.83	31.81±0.02	31.78±0.01
Cui et al. (2021)	WRN-34-10	60.64	29.33	29.65	28.99	28.99	28.86	28.84±0.02	28.83±0.02
Sitawarin et al. (2021)	WRN-34-10	62.82	24.57	25.69	24.55	24.57	24.52	24.56±0.03	24.50±0.01
Wu et al. (2020)	WRN-34-10	60.38	28.86	29.90	28.86	28.82	28.79	28.79±0.02	28.76±0.01
Cui et al. (2021)	WRN-34-20	62.55	30.20	30.83	30.03	29.99	29.86	29.84±0.01	29.85±0.01
Gowal et al. (2020)	WRN-70-16	60.86	30.03	30.54	30.00	30.00	29.99	29.97±0.01	29.96±0.01
Gowal et al. (2020)	WRN-70-16	69.15	36.88	37.84	36.95	36.90	36.86	36.87±0.02	36.81±0.01
Rebuffi et al. (2021)	WRN-70-16	63.56	34.64	35.30	34.68	34.63	34.62	34.62±0.01	34.55±0.01
ImageNet ($\varepsilon = 4/255$)	model	clean	AA	ACG	$\mathrm{MT}_{\mathrm{cos}}$	CAA	AutoAE	A^3	EDA
Salman et al. (2020)	RN-18	52.92	25.32	26.40	25.24	25.28	OOM	25.22±0.03	25.11±0.02
Engstrom et al. (2019)	RN-50	62.56	29.22	31.54	29.34	29.41	OOM	29.32±0.05	29.01±0.01
Salman et al. (2020)	RN-50	64.02	34.96	36.26	34.68	34.77	OOM	34.75±0.04	34.52±0.02
Wong et al. (2020)	RN-50	55.62	26.24	28.46	26.40	26.57	OOM	26.42±0.04	26.12±0.10
Salman et al. (2020)	WRN-50-2	68.46	38.14	40.24	38.22	38.23	OOM	38.26±0.02	38.03±0.02

Table 10: The computation time in seconds and ratio of runtime to EDA.

CIFAR-10	model	clean	EDA		A^3		CAA		AutoAE	
($\varepsilon = 8/255$)		acc	sec	ratio	sec	ratio	sec	ratio	sec	ratio
Andriushchenko & Flammarion (2020)	PARN-18	79.84	513±32	1.0	382±2	0.74	999±4	1.95	7,782	15.17
Addepalli et al. (2022)	RN-18	85.71	625±25	1.0	434±2	0.69	1,107±3	1.77	8,503	13.60
Sehwag et al. (2022)	RN-18	84.59	589±36	1.0	1,121±68	1.90	1,104±2	1.87	8,183	13.89
Engstrom et al. (2019)	RN-50	87.03	1,485±78	1.0	1,572±11	1.06	3,672±5	2.47	28,263	19.03
Carmon et al. (2019)	WRN-28-10	89.69	3,316±65	1.0	4,223±4	1.27	8,957±5	2.70	64,382	19.42
Gowal et al. (2020)	WRN-28-10	89.48	4,557±112	1.0	3,841±13	0.84	11,358±8	2.49	80,517	17.67
Hendrycks et al. (2019)	WRN-28-10	87.11	3,121±45	1.0	2,719±50	0.87	8,455±7	2.71	62,232	19.94
Rebuffi et al. (2021)	WRN-28-10	87.33	4,459±74	1.0	3,928±30	0.88	11,225±11	2.52	79,293	17.78
Sehwag et al. (2020)	WRN-28-10	88.98	3,255±64	1.0	2,662±50	0.82	8,761±7	2.69	64,273	19.75
Sridhar et al. (2022)	WRN-28-10	89.46	3,355±55	1.0	3,245±119	0.97	8,768±8	2.61	64,021	19.08
Wang et al. (2020)	WRN-28-10	87.50	3,285±78	1.0	2,732±4	0.83	8,688±7	2.64	63,662	19.38
Wu et al. (2020)	WRN-28-10	88.25	3,502±39	1.0	3,273±8	0.93	8,920±8	2.55	64,568	18.44
Ding et al. (2020)	WRN-28-4	84.36	695±29	1.0	2,017±90	2.90	1,591±2	2.29	12,828	18.46
Addepalli et al. (2022)	WRN-34-10	88.71	4,225±109	1.0	3,926±7	0.93	11,183±8	2.65	80,337	19.01
Sehwag et al. (2022)	WRN-34-10	86.68	4,172±18	1.0	3,858±6	0.92	11,067±14	2.65	79,526	19.06
Sitawarin et al. (2021)	WRN-34-10	86.84	3,591±103	1.0	3,845±22	1.07	10,675±5	2.97	79,519	22.14
Zhang et al. (2019)	WRN-34-10	87.20	3,219±32	1.0	3,500±12	1.09	10,395±8	3.23	79,064	24.56
Zhang et al. (2020)	WRN-34-10	84.52	3,936±134	1.0	3,912±16	0.99	10,593±7	2.69	76,724	19.49
Sridhar et al. (2022)	WRN-34-15	86.53	7,785±116	1.0	6,805±14	0.87	20,170±9	2.59	14,9077	19.15
Gowal et al. (2020)	WRN-34-20	85.64	14,693±284	1.0	13,463±38	0.92	40,072±21	2.73	301,256	20.50
Pang et al. (2020)	WRN-34-20	85.14	18,775±241	1.0	12,436±21	0.66	31,712±25	1.69	242,493	12.92
Rice et al. (2020)	WRN-34-20	85.34	11,255±377	1.0	12,290±5	1.09	32,478±55	2.89	244,587	21.73
Gowal et al. (2020)	WRN-70-16	85.29	21,790±430	1.0	26,587±1,032	1.22	60,431±71	2.77	451,566	20.72
Gowal et al. (2020)	WRN-70-16	91.10	24,885±371	1.0	29,544±252	1.19	63,761±54	2.56	468,883	18.84
Rebuffi et al. (2021)	WRN-70-16	88.54	24,652±479	1.0	29,075±887	1.18	63,433±35	2.57	459,409	18.64
CIFAR-100	model	clean	EDA		A^3		CAA		AutoAE	
($\varepsilon = 8/255$)		acc	sec	ratio	sec	ratio	sec	ratio	sec	ratio
Rice et al. (2020)	PARN-18	53.83	497±74	1.0	1,531±924	3.08	1,062±130	2.14	14,409	28.99
Hendrycks et al. (2019)	WRN-28-10	59.23	1,981±38	1.0	2,684±10	1.35	5,508±57	2.78	75,410	38.07
Rebuffi et al. (2021)	WRN-28-10	62.41	2,701±86	1.0	3,044±8	1.13	7,494±60	2.77	98,909	36.62
Addepalli et al. (2022)	WRN-34-10	68.75	2,792±95	1.0	3,046±17	1.09	8,045±83	2.88	109,690	39.29
Cui et al. (2021)	WRN-34-10	60.64	3,075±27	1.0	3,002±7	0.98	7,121±76	2.32	95,323	31.00
Sitawarin et al. (2021)	WRN-34-10	62.82	1,985±40	1.0	4,935±137	2.49	7,081±86	3.57	99,096	49.92
Wu et al. (2020)	WRN-34-10	60.38	2,360±31	1.0	3,258±58	1.38	7,073±66	3.00	96,679	40.97
Cui et al. (2021)	WRN-34-20	62.55	8,027±197	1.0	9,798±8	1.22	22,894±265	2.85	313,505	39.06
Gowal et al. (2020)	WRN-70-16	60.86	13,060±348	1.0	21,452±701	1.64	42,325±393	3.24	563,747	43.17
Gowal et al. (2020)	WRN-70-16	69.15	15,641±467	1.0	22,423±1,969	1.43	49,029±476	3.13	633,166	40.48
Rebuffi et al. (2021)	WRN-70-16	63.56	15,474±404	1.0	21,546±190	1.39	45,074±428	2.91	575,450	37.19
ImageNet	model	clean	EDA		A^3		CAA		AutoAE	
($\varepsilon = 4/255$)		acc	sec	ratio	sec	ratio	sec	ratio	sec	ratio
Salman et al. (2020)	RN-18	52.92	1,667±119	1.0	2,937±10	1.76	1,511±13	0.91	OOM	OOM
Engstrom et al. (2019)	RN-50	62.56	3,159±136	1.0	9,380±188	2.97	4,776±16	1.51	OOM	OOM
Salman et al. (2020)	RN-50	64.02	3,525±302	1.0	9,989±234	2.83	4,997±27	1.42	OOM	OOM
Wong et al. (2020)	RN-50	55.62	4,459±110	1.0	8,472±194	1.90	7,320±31	1.64	OOM	OOM
Salman et al. (2020)	WRN-50-2	68.46	5,102±119	1.0	9,886±120	1.94	9,267±79	1.82	OOM	OOM

Table 11: Comparisons with MDO framework(ADS), MDO framework(R-ADS), and MDO framework(RAND) in robust accuracy to validate ADS. The abbreviations are the same as those used in the main text. The lowest robust accuracies are in bold.

		MDO framework			Ensemble	Composite
CIFAR-10 ($\varepsilon = 8/255$)	model	RAND	R-ADS	ADS	ADS	ADS
(Andriushchenko & Flammarion, 2020)	PARN-18	44.09	44.02	**43.95**	44.12	43.95
(Addepalli et al., 2022)	RN-18	**52.45**	52.83	52.50	52.63	52.65
(Sehwag et al., 2022)	RN-18	55.61	**55.58**	**55.58**	55.68	55.90
(Engstrom et al., 2019)	RN-50	49.60	**49.40**	49.52	49.79	50.04
(Carmon et al., 2019)	WRN-28-10	59.56	59.53	**59.46**	59.70	59.59
(Gowal et al., 2020)	WRN-28-10	**62.82**	62.99	62.83	62.92	62.86
(Hendrycks et al., 2019)	WRN-28-10	54.94	55.02	**54.87**	55.09	55.14
(Rebuffi et al., 2021)	WRN-28-10	**60.67**	60.84	60.77	60.82	60.82
(Sehwag et al., 2020)	WRN-28-10	57.16	57.38	**57.14**	57.43	57.34
(Sridhar et al., 2022)	WRN-28-10	59.63	59.76	**59.59**	59.89	59.85
(Wang et al., 2020)	WRN-28-10	**56.32**	56.50	56.33	56.66	56.87
(Wu et al., 2020)	WRN-28-10	60.02	60.13	**59.99**	60.14	60.13
(Ding et al., 2020)	WRN-28-4	43.54	44.88	**43.24**	43.49	44.11
(Addepalli et al., 2022)	WRN-34-10	57.75	57.87	**57.72**	57.92	57.90
(Sehwag et al., 2022)	WRN-34-10	60.26	60.38	**60.21**	60.52	60.44
(Sitawarin et al., 2021)	WRN-34-10	50.83	50.78	**50.70**	50.98	51.14
(Zhang et al., 2019)	WRN-34-10	44.78	44.70	**44.62**	44.87	44.79
(Zhang et al., 2020)	WRN-34-10	53.51	**53.50**	53.55	53.60	53.52
(Sridhar et al., 2022)	WRN-34-15	60.40	60.46	**60.39**	60.46	60.48
(Gowal et al., 2020)	WRN-34-20	56.86	**56.85**	56.88	56.90	56.84
(Pang et al., 2020)	WRN-34-20	**53.77**	53.85	53.81	54.00	53.96
(Rice et al., 2020)	WRN-34-20	53.47	53.48	**53.42**	53.52	53.50
(Gowal et al., 2020)	WRN-70-16	57.23	57.21	**57.18**	57.27	57.22
(Gowal et al., 2020)	WRN-70-16	65.86	66.02	**65.85**	66.04	65.99
(Rebuffi et al., 2021)	WRN-70-16	**64.23**	64.54	64.32	64.53	64.43
CIFAR-100 ($\varepsilon = 8/255$)						
(Rice et al., 2020)	PARN-18	18.98	18.99	**18.97**	19.08	19.12
(Hendrycks et al., 2019)	WRN-28-10	28.56	28.83	**28.44**	28.61	28.45
(Rebuffi et al., 2021)	WRN-28-10	**32.08**	32.13	**32.08**	32.22	32.18
(Addepalli et al., 2022)	WRN-34-10	31.91	32.23	**31.86**	31.91	31.97
(Cui et al., 2021)	WRN-34-10	**28.97**	29.20	28.99	29.20	29.13
(Sitawarin et al., 2021)	WRN-34-10	24.71	24.68	**24.65**	24.74	24.83
(Wu et al., 2020)	WRN-34-10	28.93	29.46	**28.88**	28.97	28.97
(Cui et al., 2021)	WRN-34-20	30.07	30.35	**30.01**	30.15	30.29
(Gowal et al., 2020)	WRN-70-16	30.06	30.42	**30.05**	30.11	30.18
(Gowal et al., 2020)	WRN-70-16	37.05	37.53	**36.96**	37.19	37.17
(Rebuffi et al., 2021)	WRN-70-16	**34.61**	34.97	34.65	34.88	34.65
ImageNet ($\varepsilon = 4/255$)						
(Salman et al., 2020)	RN-18	**25.22**	25.44	**25.22**	25.46	25.48
(Engstrom et al., 2019)	RN-50	29.56	29.26	**29.20**	29.64	29.68
(Salman et al., 2020)	RN-50	**34.68**	**34.68**	34.84	35.00	34.92
(Wong et al., 2020)	RN-50	26.26	26.36	**26.22**	26.84	26.78
(Salman et al., 2020)	WRN-50-2	38.38	38.54	**38.28**	38.52	38.58

D.1 Hyperparameter determination

The values of n_a, N_{ADS}, and N_s are determined based on the following preliminary experiments, as they should be as small as possible regarding computational cost. The initial step size is determined based on the step size rules of APGD, a powerful heuristic. The step size of $\eta = 2\varepsilon$ allows the initial search to move from

Table 12: The quantified degree of diversification. DI denotes the Diversity Index, and E denotes the metric defined by equation 22. RAND, R-ADS, and ADS represent MDO framework(RAND), MDO framework(R-ADS), and MDO framework(ADS), respectively.

CIFAR-10	Models	A^3		RAND		R-ADS		ADS		EDA	
($\varepsilon = 8/255$)		DI	E	DI	E	DI	E	DI	E	DI	E
Andriushchenko & Flammarion (2020)	PARN-18	0.26±0.09	0.66±0.27	0.30±0.06	0.95±0.20	0.25±0.05	0.73±0.14	0.33±0.07	0.98±0.18	0.36±0.05	1.11±0.15
Addepalli et al. (2022)	RN-18	0.27±0.10	0.69±0.28	0.31±0.07	0.94±0.20	0.23±0.05	0.78±0.11	0.36±0.08	1.02±0.19	0.40±0.06	1.18±0.17
Sehwag et al. (2022)	RN-18	0.27±0.09	0.72±0.28	0.31±0.06	0.93±0.20	0.26±0.05	0.79±0.15	0.34±0.06	0.99±0.19	0.38±0.05	1.15±0.15
Engstrom et al. (2019)	RN-50	0.28±0.09	0.72±0.27	0.33±0.06	0.99±0.19	0.25±0.05	0.76±0.15	0.37±0.07	1.04±0.19	0.39±0.05	1.17±0.16
Carmon et al. (2019)	WRN-28-10	0.26±0.08	0.65±0.25	0.33±0.06	1.00±0.18	0.26±0.05	0.84±0.14	0.38±0.07	1.06±0.19	0.41±0.05	1.22±0.17
Gowal et al. (2020)	WRN-28-10	0.22±0.10	0.58±0.30	0.30±0.07	0.94±0.21	0.20±0.05	0.71±0.11	0.37±0.08	1.04±0.22	0.41±0.06	1.20±0.18
Hendrycks et al. (2019)	WRN-28-10	0.25±0.09	0.64±0.27	0.30±0.06	0.94±0.20	0.22±0.04	0.71±0.12	0.34±0.07	1.01±0.19	0.37±0.05	1.14±0.15
Rebuffi et al. (2021)	WRN-28-10	0.24±0.10	0.63±0.30	0.30±0.06	0.89±0.21	0.24±0.05	0.75±0.12	0.34±0.07	0.97±0.18	0.37±0.05	1.09±0.16
Sehwag et al. (2020)	WRN-28-10	0.25±0.08	0.64±0.26	0.33±0.06	0.99±0.19	0.23±0.04	0.75±0.11	0.38±0.07	1.09±0.20	0.43±0.05	1.24±0.19
Sridhar et al. (2022)	WRN-28-10	0.25±0.09	0.64±0.26	0.33±0.06	1.00±0.18	0.24±0.04	0.76±0.10	0.37±0.07	1.05±0.18	0.41±0.05	1.24±0.16
Wang et al. (2020)	WRN-28-10	0.28±0.08	0.68±0.24	0.31±0.06	0.93±0.18	0.25±0.06	0.79±0.12	0.35±0.07	0.99±0.18	0.35±0.06	1.04±0.15
Wu et al. (2020)	WRN-28-10	0.25±0.09	0.64±0.27	0.32±0.06	0.98±0.19	0.23±0.04	0.76±0.10	0.38±0.07	1.08±0.19	0.43±0.06	1.25±0.16
Ding et al. (2020)	WRN-28-4	0.24±0.10	0.91±0.34	0.33±0.07	0.98±0.22	0.29±0.07	0.91±0.23	0.36±0.08	1.02±0.22	0.38±0.07	1.19±0.16
Addepalli et al. (2022)	WRN-34-10	0.27±0.09	0.67±0.28	0.31±0.06	0.94±0.20	0.24±0.05	0.83±0.13	0.37±0.08	1.04±0.20	0.41±0.06	1.21±0.17
Sehwag et al. (2022)	WRN-34-10	0.25±0.09	0.65±0.26	0.32±0.06	0.96±0.18	0.24±0.04	0.74±0.12	0.37±0.07	1.02±0.18	0.38±0.05	1.15±0.14
Sitawarin et al. (2021)	WRN-34-10	0.27±0.10	0.72±0.30	0.32±0.06	0.99±0.20	0.24±0.05	0.74±0.17	0.33±0.06	1.03±0.20	0.38±0.05	1.18±0.16
Zhang et al. (2019)	WRN-34-10	0.27±0.11	0.73±0.31	0.30±0.06	0.96±0.21	0.23±0.05	0.70±0.16	0.32±0.07	1.00±0.21	0.37±0.05	1.18±0.17
Zhang et al. (2020)	WRN-34-10	0.25±0.09	0.62±0.26	0.31±0.06	0.95±0.20	0.25±0.05	0.82±0.17	0.33±0.07	0.99±0.20	0.39±0.06	1.21±0.16
Sridhar et al. (2022)	WRN-34-15	0.23±0.08	0.57±0.24	0.31±0.07	0.96±0.19	0.24±0.05	0.78±0.11	0.37±0.08	1.06±0.19	0.41±0.06	1.21±0.16
Gowal et al. (2020)	WRN-34-20	0.21±0.11	0.55±0.32	0.29±0.07	0.95±0.22	0.21±0.05	0.73±0.15	0.34±0.07	1.02±0.21	0.38±0.05	1.19±0.18
Pang et al. (2020)	WRN-34-20	0.24±0.09	0.66±0.30	0.22±0.08	0.80±0.23	0.16±0.07	0.64±0.25	0.25±0.10	0.87±0.21	0.31±0.08	1.06±0.19
Rice et al. (2020)	WRN-34-20	0.24±0.11	0.64±0.32	0.28±0.06	0.91±0.22	0.22±0.05	0.71±0.17	0.33±0.07	0.98±0.21	0.37±0.05	1.14±0.17
Gowal et al. (2020)	WRN-70-16	0.22±0.10	0.56±0.30	0.30±0.06	0.94±0.20	0.21±0.05	0.71±0.12	0.38±0.08	1.04±0.20	0.39±0.05	1.18±0.17
Gowal et al. (2020)	WRN-70-16	0.19±0.10	0.52±0.32	0.28±0.06	0.94±0.22	0.22±0.05	0.76±0.15	0.31±0.07	1.01±0.21	0.38±0.05	1.19±0.16
Rebuffi et al. (2021)	WRN-70-16	0.23±0.09	0.59±0.29	0.30±0.06	0.88±0.20	0.23±0.05	0.74±0.12	0.34±0.07	0.96±0.17	0.37±0.05	1.09±0.16
CIFAR-100	Models	A^3		RAND		R-ADS		ADS		EDA	
($\varepsilon = 8/255$)		DI	E	DI	E	DI	E	DI	E	DI	E
Rice et al. (2020)	PARN-18	0.34±0.13	0.83±0.31	0.32±0.06	0.97±0.20	0.25±0.06	0.78±0.18	0.35±0.07	1.02±0.21	0.39±0.06	1.15±0.20
Hendrycks et al. (2019)	WRN-28-10	0.27±0.11	0.71±0.30	0.29±0.07	0.94±0.22	0.22±0.06	0.75±0.16	0.31±0.08	0.97±0.21	0.36±0.06	1.13±0.18
Rebuffi et al. (2021)	WRN-28-10	0.32±0.15	0.81±0.36	0.29±0.07	0.94±0.23	0.25±0.06	0.82±0.17	0.35±0.08	1.02±0.22	0.41±0.06	1.20±0.20
Addepalli et al. (2022)	WRN-34-10	0.34±0.13	0.84±0.30	0.31±0.07	0.97±0.21	0.24±0.06	0.86±0.13	0.36±0.09	1.04±0.21	0.42±0.06	1.21±0.18
Cui et al. (2021)	WRN-34-10	0.27±0.10	0.67±0.27	0.27±0.08	0.89±0.22	0.23±0.08	0.82±0.14	0.30±0.09	0.97±0.19	0.34±0.07	1.04±0.17
Sitawarin et al. (2021)	WRN-34-10	0.31±0.12	0.79±0.30	0.32±0.06	0.98±0.20	0.24±0.05	0.75±0.17	0.35±0.07	1.02±0.21	0.38±0.06	1.14±0.20
Wu et al. (2020)	WRN-34-10	0.28±0.12	0.73±0.32	0.29±0.07	0.95±0.22	0.24±0.05	0.79±0.15	0.32±0.08	0.99±0.21	0.37±0.06	1.16±0.18
Cui et al. (2021)	WRN-34-20	0.27±0.09	0.66±0.27	0.29±0.08	0.92±0.21	0.22±0.06	0.75±0.13	0.32±0.09	0.99±0.21	0.35±0.07	1.07±0.18
Gowal et al. (2020)	WRN-70-16	0.27±0.13	0.69±0.34	0.28±0.07	0.93±0.22	0.19±0.05	0.66±0.13	0.33±0.08	0.98±0.23	0.37±0.06	1.19±0.17
Gowal et al. (2020)	WRN-70-16	0.31±0.14	0.79±0.33	0.31±0.07	0.94±0.22	0.23±0.06	0.73±0.14	0.33±0.08	1.01±0.21	0.40±0.06	1.21±0.17
Rebuffi et al. (2021)	WRN-70-16	0.29±0.14	0.76±0.36	0.30±0.07	0.93±0.23	0.21±0.06	0.74±0.14	0.34±0.08	0.99±0.20	0.40±0.05	1.19±0.19
ImageNet	Models	A^3		RAND		R-ADS		ADS		EDA	
($\varepsilon = 4/255$)		DI	E	DI	E	DI	E	DI	E	DI	E
Salman et al. (2020)	RN-18	0.26±0.07	2.60±0.91	0.34±0.07	3.52±0.69	0.28±0.05	2.83±0.42	0.38±0.08	3.66±0.69	0.43±0.04	4.37±0.66
Engstrom et al. (2019)	RN-50	0.29±0.06	2.79±0.83	0.38±0.05	3.69±0.58	0.31±0.06	3.18±0.51	0.40±0.06	3.88±0.61	0.44±0.04	4.30±0.64
Salman et al. (2020)	RN-50	0.26±0.06	2.61±0.90	0.35±0.06	3.61±0.65	0.30±0.06	3.03±0.47	0.37±0.07	3.70±0.63	0.42±0.04	4.43±0.53
Wong et al. (2020)	RN-50	0.27±0.07	3.45±1.12	0.36±0.06	4.59±0.82	0.30±0.06	3.76±0.58	0.38±0.07	4.81±0.80	0.43±0.05	5.43±0.89
Salman et al. (2020)	WRN-50-2	0.27±0.06	2.63±0.82	0.37±0.06	3.69±0.60	0.34±0.06	3.53±0.58	0.38±0.05	3.80±0.61	0.43±0.04	4.22±0.65

one end of the feasible region to the other, thus allowing a broader search. The parameters 0.22 and 0.41, which determine the allocation of the number of iterations for the diversification phase and the intensification phase, are inspired by the checkpoints in the APGD's step size update. N_1 is the number of iterations to be searched with a step size of 2ε to ε in APGD. The experiments in Yamamura et al. (2022) suggest that the CG diversification performance is well achieved by moving in the CG direction according to this iteration allocation and step size assignment. Therefore, we chose these values for step sizes, N_1, and N_2.

Preliminary experiments to determine hyperparameters of ADS We conducted preliminary experiments on the following five models to determine the hyperparameters of ADS. 1. ResNet-18 (Sehwag et al., 2022), 2. WideResNet-28-10 (Gowal et al., 2020), 3. PreActResNet-18 (Rice et al., 2020), 4. WideResNet-34-10 (Sitawarin et al., 2021), 5. ResNet-50 (Engstrom et al., 2019). These numbers correspond to the "No." column in Table 14.

Table 13: Comparison in robust accuracy for transfer setting. "source→target" indicates that the transfer attack from the source to the target model. AutoAE was executed only with CIFAR10 because of its long execution time.

source→target	clean acc (target model)	robust acc (in RobustBench)	AA	CAA	AutoAE	A^3	EDA
CIFAR-10							
(Carmon et al., 2019)→(Ding et al., 2020)	84.36	41.44	71.82	73.15	73.24	70.46	**68.14**
(Carmon et al., 2019)→(Cui et al., 2021)	88.22	52.86	71.90	74.76	74.66	73.46	**70.29**
(Ding et al., 2020)→(Carmon et al., 2019)	89.69	59.53	**83.23**	85.01	84.85	85.16	83.68
(Ding et al., 2020)→(Cui et al., 2021)	88.22	52.86	**79.77**	82.16	81.83	82.42	80.32
(Cui et al., 2021)→(Carmon et al., 2019)	89.69	59.53	**76.17**	80.24	80.32	79.94	77.24
(Cui et al., 2021)→(Ding et al., 2020)	84.36	41.44	70.16	72.07	72.09	71.28	**69.43**
CIFAR-100							
Rice et al. (2020)→Cui et al. (2021)	62.55	30.20	**56.42**	58.72	-	58.36	58.10
Rice et al. (2020)→Rebuffi et al. (2021)	63.56	34.64	**58.28**	60.45	-	59.74	59.55
Cui et al. (2021)→Rice et al. (2020)	53.83	18.95	44.51	45.87	-	44.50	**42.86**
Cui et al. (2021)→Rebuffi et al. (2021)	63.56	34.64	**53.58**	56.24	-	55.35	54.02
Rebuffi et al. (2021)→Rice et al. (2020)	53.83	18.95	44.44	45.64	-	43.51	**40.84**
Rebuffi et al. (2021)→Cui et al. (2021)	62.55	30.20	50.15	51.98	-	51.26	**48.45**
ImageNet							
Salman et al. (2020)→Engstrom et al. (2019)	62.52	29.22	53.74	55.88	-	55.20	**52.76**
Salman et al. (2020)→Wong et al. (2020)	55.64	26.24	47.22	47.90	-	46.40	**45.12**
Engstrom et al. (2019)→Salman et al. (2020)	68.46	38.14	**63.62**	65.34	-	64.68	64.46
Engstrom et al. (2019)→Wong et al. (2020)	55.64	26.24	**44.64**	46.34	-	46.06	44.92
Wong et al. (2020)→Salman et al. (2020)	68.46	38.14	**65.36**	66.08	-	65.56	65.42
Wong et al. (2020)→Engstrom et al. (2019)	62.52	29.22	**56.40**	58.02	-	57.80	56.72

Table 14: Results of the preliminary experiments to determine the hyperparameters of ADS. The robust accuracy obtained by the MDO framework is described. The default parameters are in bold.

		N_{ADS}				n_a			
Dataset	No.	3	**4**	5	10	3	4	**5**	6
CIFAR-10	1	55.56	55.58	55.65	55.67	55.63	55.66	55.58	55.57
CIFAR-10	2	56.83	56.80	56.84	56.81	56.90	56.83	56.80	56.78
CIFAR-100	3	19.01	18.87	19.03	19.01	19.16	19.10	18.87	18.90
CIFAR-100	4	24.66	24.56	24.61	24.59	24.61	24.78	24.56	24.56
ImageNet	5	29.40	29.24	29.26	29.30	29.28	29.34	29.24	29.14

D.2 Comparison in computation cost based on the number of queries

We compare the number of queries because the bottleneck in an adversarial attack is forward/backward (queries). In attack selection, CAA requires $KNt\times\#$ samples $= 60t\times$ #samples $\geq 60\times$ #samples, where $K = 20$ is the population size, $N = 3$ is the policy length, $t \geq 1$ is the number of iterations for the candidate attacks, and #samples = 4,000 for CIFAR-10 and 1000 for ImageNet. In summary, attack selection in CAA requires more than 240,000 queries for CIFAR-10 and 60,000 for ImageNet. For attack selection by ADS, $2|A|N_{ADS}\times$ #samples $= 112\times$ #samples queries are required, where $|A| = 28$ is the number of candidates, $N_{ADS} = 4$ is the number of iterations for candidates, and #samples=100 for CIFAR-10/100 and 50 for ImageNet. ADS thus requires 11,200 queries for CIFAR-10 and 5,600 queries for ImageNet.

For the entire attack procedure, the standard AA requires queries of 6100× #images. CAA requires at least $60t\times$#samples queries. AutoAE runs 32 iterations of APGD with CE loss, 63 iterations of APGD with DLR loss, 160 iterations of FAB, and 378 iterations of MultiTargeted attack with nine target classes. According to the official implementation of AutoAE, the MultiTargeted attack runs 378 iterations for each target class. Therefore, AutoAE requires $(32+63+160+378\times 9)\times$#images $= 3,657\times$#images queries for the adversarial

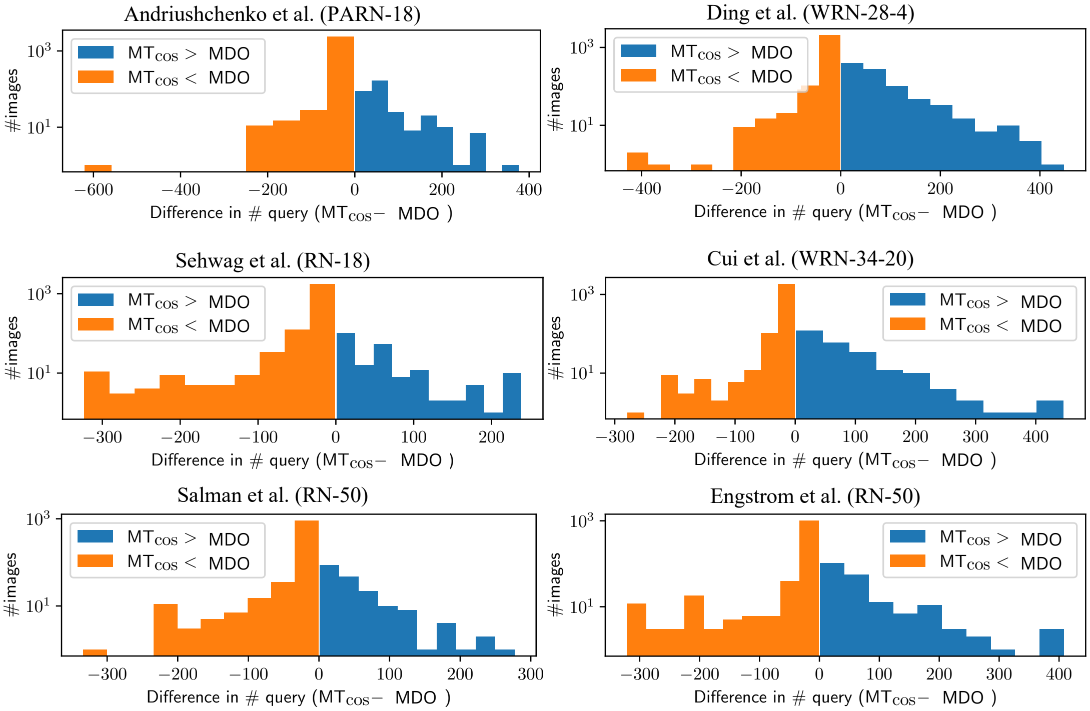


Figure 14: The difference between $\mathrm{MT}_{\cos}$ and the MDO framework in #queries to find adversarial examples for some models. The attacked models are Andriushchenko & Flammarion (2020), Sehwag et al. (2022), and Ding et al. (2020) for CIFAR-10, Cui et al. (2021) for CIFAR-100, and Salman et al. (2020) and Engstrom et al. (2019) for ImageNet.

attack. EDA requires $n_a \times (N_1 + N_2 + N_3) \times$# images= $5 \times 100 \times$#images queries for the MDO framework, and $K \times N_s + N_4 \times$# images= 190-300×#images queries for the targeted attack a^t. Therefore, EDA requires 692.24 to 802.24×# images queries in total. We compared the runtime of EDA with that of A^3 because A^3 automatically terminates its search before the query limit.

D.3 Mathematical definition of the best point sets of A^{3}and EDA

Mathematically, the best point sets of A^3 and EDA are defined as follows. First, the best point set of A^3 is defined as $X^*(\boldsymbol{x}_i, a_{\mathrm{A}^3}, R_i)$, where $a_{\mathrm{A}^3} = (\mathtt{x}_{\mathrm{ADI}}, \eta_{\cos}, \boldsymbol{\delta}_{\mathrm{GD}}, L_{\mathrm{CW}})$, where $\mathtt{x}_{\mathrm{ADI}}$ is Adaptive Direction Initialization (ADI) proposed by Liu et al. (2022c). Subsequently, the best point set of EDA is defined as $X^*(\boldsymbol{x}_i, e^*) \cup X^*(\boldsymbol{x}_i, a^t, 1)$, where $e^* = \left\{(\mathtt{x}, \eta, \boldsymbol{\delta}_a, L_a) \mid a \in \{a_1^* \dots, a_{n_a}^*, \hat{a}_1, \dots, \hat{a}_{n_a}\}\right\}$ and $a^t = (\mathtt{x}_{\mathrm{PAS}}, \eta_{\mathrm{APGD}}, \boldsymbol{\delta}_{\mathrm{GD}}, L_{\mathrm{CW}}^T)$.

D.4 Analysis of EDA using an index based on Euclid distance

DI takes small values when the point set forms a cluster, even if the Euclidean distance between any two points is large. Therefore, quantification by DI and quantification based on the Euclidean distance between points in the point set may have different characteristics. Therefore, in this section, to compare the diversification performance from a different perspective than DI, we consider quantifying the degree of diversification of the best point set based on the average value of the Euclidean distance between the centroid of the point set X and all points in the point set X. Mathematically, the average Euclidean distance between all points in a

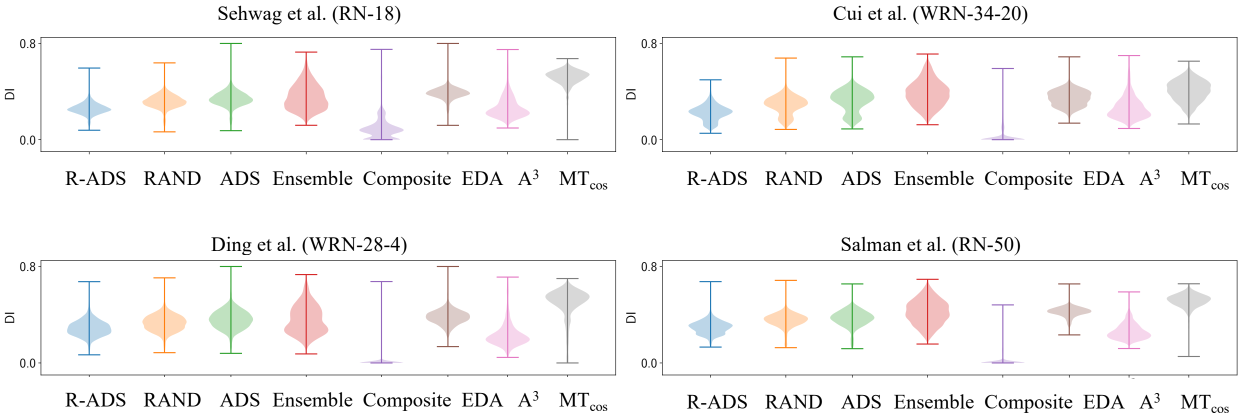


Figure 15: Violin plot of DI. The attacked models are Sehwag et al. (2022) and Ding et al. (2020) for CIFAR-10, Cui et al. (2021) for CIFAR-100, and Salman et al. (2020) for ImageNet.

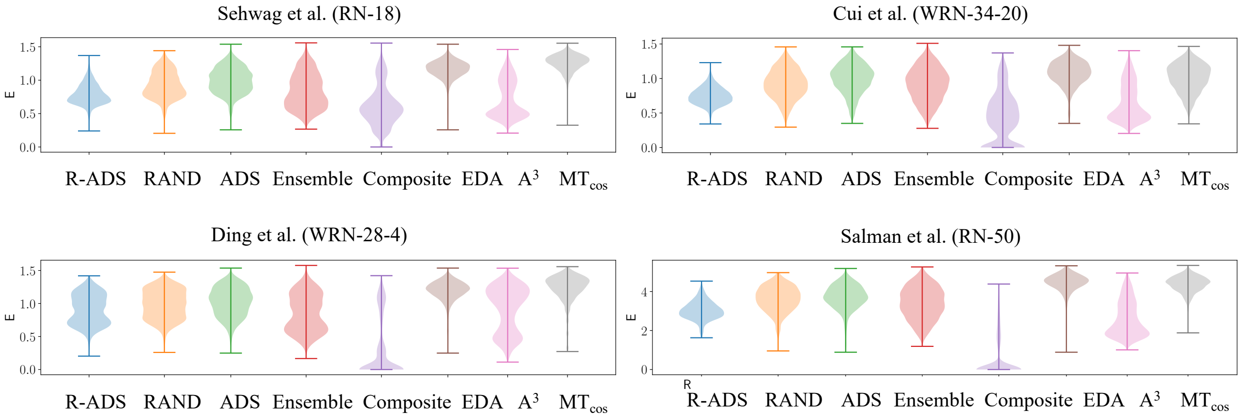


Figure 16: Violin plot of equation 22. The attacked models are Sehwag et al. (2022) and Ding et al. (2020) for CIFAR-10, Cui et al. (2021) for CIFAR-100, and Salman et al. (2020) for ImageNet.

point set X is defined as

$$E(X) = \frac{1}{|X|} \sum_{x \in X} \|x - \bar{x}\|_2, \tag{22}$$

where $\bar{x}$ is the centroid of the point set X, defined as $\bar{x} = \frac{1}{|X|} \sum_{x \in X} x$. As shown in Figure 16, the value of equation 22 tends to be larger for EDA than for A^3 in most models where EDA has higher attack performance than A^3. This difference is more pronounced than the difference in DI. While EDA shows a similar trend for all models, A^3 shows a different trend in the value of equation 22 for some models. For example, as shown in Figure 16, the value of equation 22 for A^3 tends to be larger for the model proposed by Ding et al. (2020) than for the other models. Given the high attack performance of A^3 against these models, this suggests that the A^3 diversification strategy may be more effective for these models.

D.5 Analysis of EDA for the model proposed by Ding et al.

The attack performance of EDA is significantly lower for the model proposed by Ding et al. (2020) compared to A^3. This section discusses the reasons for this regarding diversification performance and computation

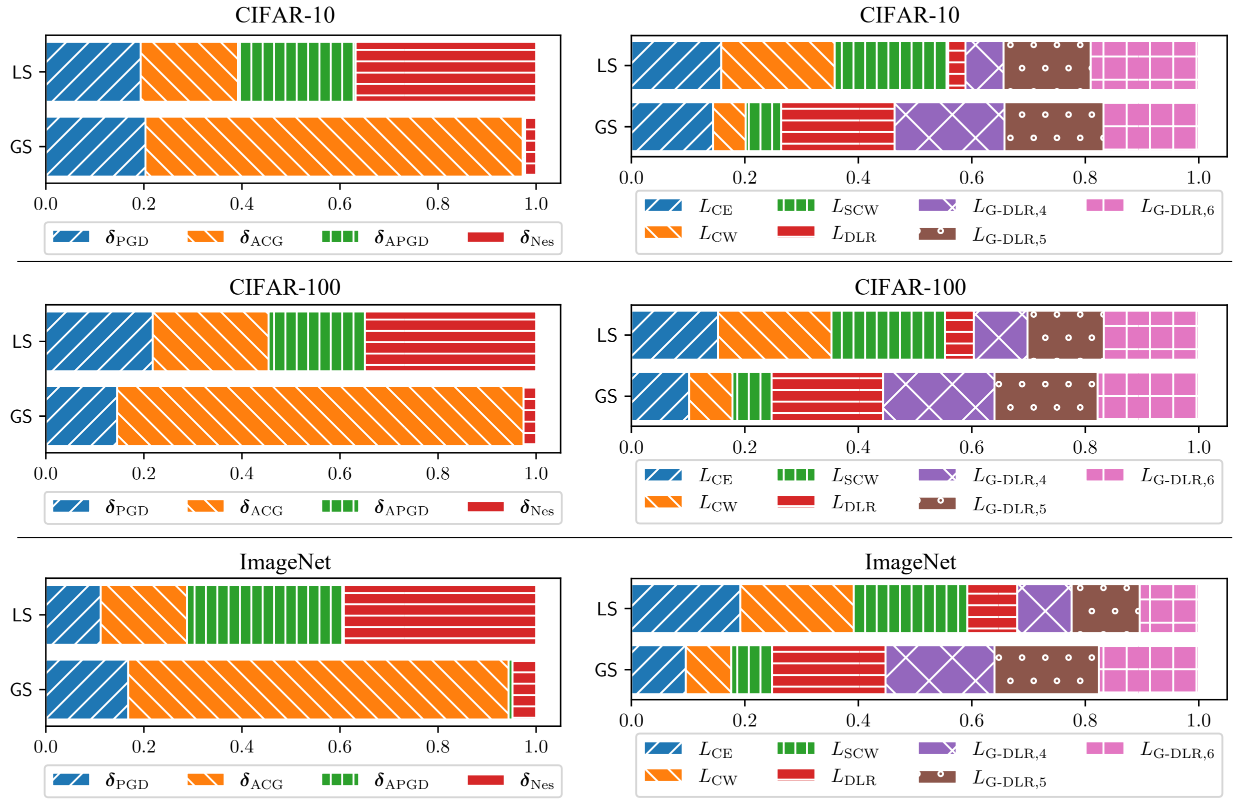


Figure 17: The average ratio of times each search direction and objective function used by EDA. LS and GS refer to the intensification phase and diversification phase, respectively.

time. As described in the main text, the value of DI for the best point set tends to be higher for EDA and lower for A^3, similar to the results for other models. On the other hand, the analysis in the previous section shows that for the model proposed by Ding et al. (2020), the value of equation 22 for the best point set obtained by A^3 tends to take larger values than the results for the other models. In addition, a comparison of the computation time for EDA and A^3 shows that A^3 takes more than three times longer than EDA. The above comparison suggests that the A^3 can perform better diversification for the model than for other models. In summary, setting a longer computation time and increasing the number of multi-restart are considered particularly effective in improving the attack performance for the model proposed by Ding et al. (2020).

D.6 Trends of search directions and objective functions selected by ADS

Figure 17 is a bar chart displaying the ratio of times each search direction and objective function used by EDA. Figure 17 shows that $\boldsymbol{\delta}_{\text{ACG}}$ and $L_{\text{G-DLR},q}$ are frequently used in the diversification phase, and $\boldsymbol{\delta}_{\text{Nes}}$ is rarely used. In the intensification phase, $L_{\text{CW}}, L_{\text{SCW}}$, and $\boldsymbol{\delta}_{\text{Nes}}$ were more likely to be selected. This trend may reflect ACG's high diversification performance and NAG's high intensification performance. The potential reasons for these trends are: 1. P_i^e and DI play different roles from each other, 2. the ACG's search direction may be similar to the steepest for small step sizes, and 3. the difference between Nesterov's acceleration gradient direction and gradient direction.

The role of P_i^e and DI. The P_i^e measures the degree of diversification in the output space during the search. A pair with the largest P_i^e is expected to show a high diversity in the output space. In addition, from Yamamura et al. (2022), it can be assumed that the ACG direction increases P_i^e, while the steepest-like direction does not. From the above, it is considered that the pair with the maximum P_i^e is likely to include the ACG direction. DI measures the diversity of the best point set obtained by the search. In our use case, DI represents the dissimilarity between the best points. That is, we expect that pairs with the largest DI are more likely to enumerate dissimilar solutions. Intuitively, updates in diverse directions contribute to the enumeration of dissimilar solutions. Given that the search direction is gradient-dependent, the pair with the largest DI is likely to include a variety of objective functions and update formulas.

ACG's search direction may be similar to the steepest for small step sizes. The reason for this is as follows. According to equation 9, $\boldsymbol{s}^{(k)}$ is close to the gradient $\nabla L(g(\boldsymbol{x}^{(k)}), c)$ when $\beta^{(k)}$ is close to 0. From equation 8, $\langle \nabla L(g(\boldsymbol{x}^{(k)}), c), \boldsymbol{y}^{(k-1)} \rangle$ is the numerator of $\beta^{(k)}$. Therefore, as $\boldsymbol{y}^{(k-1)}$ approaches $\mathbf{0}$, $\beta^{(k)}$ also approaches 0. When the step size is small, $\|x^{(k)} - x^{(k-1)}\|$ is also small, so $\boldsymbol{y}^{(k-1)}$ is likely to be close to $\mathbf{0}$. As a result, the ACG's direction and the steepest direction may be similar. The experiments conducted by Yamamura et al. (2022) also support this claim.

The difference between Nesterov's acceleration gradient direction and gradient direction. Nesterov's accelerated gradient (NAG) method updates the search point using the gradient of the point moved from the current search point to the momentum direction. Assuming that the objective function is multimodal, the gradient at the current point is unlikely to be similar to NAG's search direction. Thus, if the objective function is multimodal, a search in the NAG's direction may find different local solutions from that in the gradient direction.

E Prediction Aware Sampling (PAS)

Motivation The hypothesis behind PAS is that starting the search with an initial point near multiple decision boundaries increases the likelihood of finding an adversarial example. When maximizing the inner product of a random vector and logit, as ODS does, the distance to the decision boundary may be farther away than the initial point. However, when moving in the direction where the predicted probability for the correct class is as small as possible, the initial point is more likely to be closer to the decision boundary than the original point. We hypothesized that the attack's success rate could be improved by starting the search at a point closer to the decision boundary.

Prediction Aware Sampling One promising initial point sampling is ODS, which considers diversification in the output space. However, there is room for improvement because its sampling does not consider image-specific information. Based on the idea that the randomly sampled initial point close to decision boundaries makes the attacks easier to succeed, we propose Prediction-Aware Sampling (PAS), a variant of ODS. PAS maximizes the following function in the same way as ODS to sample the initial point.

$$v(\boldsymbol{w}, g, \boldsymbol{x}) = \boldsymbol{w}^T g(\boldsymbol{x}) \times \exp\left(-g_c(\boldsymbol{x})\right), \quad \left(\boldsymbol{w} \sim U\left(-1, 1\right)^C\right) \tag{23}$$

PAS samples the initial point by repeating the following updates for N_{PAS} iterations.

$$\boldsymbol{x} \leftarrow P_{\mathcal{S}}\left(\boldsymbol{x} + \eta_{\text{PAS}} \ \text{sign}\left(\frac{\nabla_{\boldsymbol{x}} v(\boldsymbol{w}, g, \boldsymbol{x})}{\|\nabla_{\boldsymbol{x}} v(\boldsymbol{w}, g, \boldsymbol{x})\|_2}\right)\right) \tag{24}$$

Same as ODS, PAS used $N_{\text{PAS}} = 2$ and $\eta_{\text{PAS}} = \varepsilon$. Intuitively, maximizing equation 23 means moving the initial point closer to the decision boundary by reducing the prediction probability of the correct class c and, at the same time, moving the logit $g(\boldsymbol{x})$ closer to the random vector $\boldsymbol{w}$.

Experiments To test our hypothesis, we compared the success rate for each class in targeted attacks with nine target classes, using the input point, the point sampled by ODS, and the point sampled by PAS as initial points. In our notation, we compared the attack performance of $(\mathtt{x}, \eta_{\cos}, \boldsymbol{\delta}_{\text{GD}}, L_{\text{CW}}^T), \mathtt{x} \in \{\mathtt{x}_{\text{org}}, \mathtt{x}_{\text{ODS}}, \mathtt{x}_{\text{PAS}}\}$ with 100 iterations for each target class and initial step size of 2ε. The number of target classes K was set to 9. The following five models were used in the experiments. 1. ResNet-18 (Sehwag et al., 2022), 2. WideResNet-28-10 (Gowal et al., 2020), 3. PreActResNet-18 (Rice et al., 2020), 4. WideResNet-34-10 (Sitawarin et al., 2021), 5. ResNet-50 (Engstrom et al., 2019). These numbers correspond to the "No."

column in Table 15. The experimental results in Table 15 show that the attack with PAS can achieve higher attack success rates for many target classes than other initial point selections. The experimental results support our hypothesis that PAS brings the starting point closer to the decision boundary, resulting in a more successful attack. As described in Appendix G.2, the ablation results for the initial point of the EDA also indicate that the PAS contributes to the attack performance of the EDA and the MDO framework.

F Targeted attack in EDA

Motivation The motivation for using a targeted attack is to efficiently diversify the most likely prediction class of the adversarial example away from the correct class (diversification in the output space). CW loss and DLR loss are objective functions that generate adversarial examples misclassified into the class with the highest prediction probability among classes other than the correct class. In other words, they attempt to generate an adversarial example misclassified into the class whose decision boundary is closest to the current point. However, it is difficult to approach the decision boundary when the gradient is zero, even if the distance to the decision boundary is close because the gradient-based attack moves in the direction of the gradient. In addition, Yamamura et al. (2022) reported that in the steepest gradient-based attacks, the class with the highest prediction probability among classes other than the correct class hardly changes during the search. Considering these factors, diversification in the output space could be effective, especially for attacks with untargeted losses. Some existing research also supports the effectiveness of diversification in the output space (Tashiro et al., 2020; Gowal et al., 2019).

Target selection Although a multi-target attack shows high performance by achieving diversification in the output space, existing methods (Gowal et al., 2019; Croce & Hein, 2020b) are computationally expensive because they assign an equal number of iterations to each target. To reduce the computational cost of the multi-target attack, we propose Target Selection (TS), which estimates the easiest target class to attack based on a small-scale search. TS estimates the easiest target class to attack based on a small-scale search to reduce the computational cost of the multi-target attack. The objective is to reduce the computational cost by focusing the number of iterations on the selected target. The procedure of TS is as follows: (1) Upper K classes with large logit values of initial points are selected as target candidates. (2) Targeted attacks are performed for N_s iterations for each target candidate. (3) The output is the target candidate T with the highest objective function value.

Hyperparameters The parameters of the targeted attack are the number of candidate targets in target selection $K = 9, 14, 20$, the number of iterations $N_s = 10$, and the number of iterations in targeted attack $N = 100$. We chose $N = 100$ because the number of iterations per targeted attack in AA is set to 100, which achieves a reasonable trade-off between computational cost and attack performance.

Experiments To investigate the validity of this target selection, we compare the attack performance of EDA+, which executes a normal targeted attack after the MDO framework, with that of EDA. In this experiment, we performed $(\mathbf{x}_{\mathrm{PAS}}, \eta_{\cos}, \boldsymbol{\delta}_{\mathrm{GD}}, L_{\mathrm{CW}}^{T})$ with 100 iterations for each target class and the initial step size of 2ε. The following five models were used in the experiments. 1. ResNet-18 (Sehwag et al., 2022), 2. WideResNet-28-10 (Gowal et al., 2020), 3. PreActResNet-18 (Rice et al., 2020), 4. WideResNet-34-10 (Sitawarin et al., 2021), 5. ResNet-50 (Engstrom et al., 2019). These numbers correspond to the "No." column in Table 16. Although the normal targeted attack requires $K \times N = 100K$ queries, our targeted attack scheme selects a single target class and thus requires $N_s \times K + N = 10K + 100$ queries per image. Given the parameter of $K = 9, 14, 20 \geq 2$, our targeted attack scheme requires fewer queries than the normal targeted attack. Considering the results described in Table 16, our target selection may reduce runtime without significantly degrading the attack performance of EDA.

Table 15: Validation of PAS. The lowest robust accuracy is in bold.

Dataset	No.	target	input	ODS	PAS
CIFAR-10	1	1	57.41	57.40	**57.29**
		2	65.33	64.82	**64.28**
		3	67.85	66.98	**66.27**
		4	69.06	68.21	**67.52**
		5	69.88	68.77	**67.81**
		6	69.66	68.59	**67.88**
		7	69.11	68.37	**67.59**
		8	68.08	67.47	**67.10**
		9	67.32	66.84	**66.46**
CIFAR-10	2	1	**57.52**	57.54	57.57
		2	65.93	65.79	**65.33**
		3	68.88	68.63	**68.15**
		4	70.62	70.34	**69.33**
		5	71.41	70.95	**70.00**
		6	72.37	71.82	**70.71**
		7	72.57	71.99	**70.80**
		8	73.09	72.49	**71.31**
		9	72.86	72.28	**71.08**
CIFAR-100	3	1	20.37	20.39	**20.34**
		2	24.71	24.67	**24.46**
		3	26.65	26.64	**26.14**
		4	27.70	27.49	**26.99**
		5	28.37	28.36	**27.70**
		6	29.02	28.98	**28.33**
		7	29.55	29.32	**28.62**
		8	29.83	29.68	**28.95**
		9	29.72	29.55	**28.94**
CIFAR-100	4	1	27.25	27.22	**27.17**
		2	31.87	31.74	**31.50**
		3	33.70	33.73	**33.18**
		4	34.92	34.72	**34.17**
		5	36.00	35.91	**35.13**
		6	36.55	36.50	**35.58**
		7	36.75	36.56	**35.87**
		8	37.34	37.20	**36.39**
		9	37.81	37.54	**36.73**
ImageNet	5	1	32.66	32.62	**32.42**
		2	38.04	37.98	**37.68**
		3	39.64	39.64	**39.22**
		4	41.50	41.34	**40.74**
		5	41.68	41.48	**41.00**
		6	42.82	42.80	**42.20**
		7	43.10	42.92	**42.46**
		8	43.22	43.02	**42.52**
		9	43.20	43.12	**42.58**

Table 16: Validation of targeted attack

Dataset	No.	EDA	EDA+
CIFAR-10	1	**55.49**	**55.49**
CIFAR-10	2	**56.75**	56.78
CIFAR-100	3	**18.86**	18.87
CIFAR-100	4	**24.47**	24.48
ImageNet	5	**29.00**	29.02

Table 17: Ablation study of EDA. The default parameter is in bold.

		N_s			# sampled images			
Dataset	No.	5	**10**	15	**1%**	3%	5%	7%
CIFAR-10	1	55.59	55.49	55.49	55.49	55.47	55.48	55.50
CIFAR-10	2	56.73	56.76	56.76	56.76	56.81	56.81	56.79
CIFAR-100	3	18.85	18.86	18.86	18.86	18.87	18.88	18.87
CIFAR-100	4	24.46	24.47	24.48	24.47	24.49	24.49	24.47
ImageNet	5	29.04	29.04	29.04	29.04	29.04	29.08	29.04

Table 19: Ablation for the targeted attack. The default setting is in bold.

		δ_{GD}			δ_{CG}		δ_{APGD}
Dataset	No.	$\boldsymbol{L^T_{\text{CW}}}$	L^T_{CE}	L^T_{DLR}	L^T_{CW}	L^T_{DLR}	L^T_{DLR}
CIFAR-10	1	55.49	55.57	55.48	55.50	55.50	55.48
CIFAR-10	2	56.76	56.89	56.81	56.81	56.83	56.79
CIFAR-100	3	18.86	18.96	18.90	18.93	18.94	18.91
CIFAR-100	4	24.47	24.63	24.50	24.53	24.53	24.50
ImageNet	5	29.00	29.18	29.02	29.08	29.06	29.00

G Ablation study

G.1 Hyperparameter sensitivity of EDA

We investigated the impact of hyperparameter values on EDA's performance. In this experiment, the following five models were used. 1. ResNet-18 (Sehwag et al., 2022), 2. WideResNet-28-10 (Gowal et al., 2020), 3. PreActResNet-18 (Rice et al., 2020), 4. WideResNet-34-10 (Sitawarin et al., 2021), 5. ResNet-50 (Engstrom et al., 2019). Tables 17 and 18 show the robust accuracy obtained by EDA with each parameter value. Although the attack performance of the MDO framework is different among different hyperparameters, as described in Table 14, the EDA's performance is stable regardless of the hyperparameter setting. These experimental results imply that the attacks composed by different strategies could be robust to the hyperparameter settings. In addition, we tested the EDA's performance with several targeted attacks. As described in Table 19, the search directions based on the steepest direction performed better than the conjugate gradient-based direction. Also, the margin-based losses showed higher performance than the CE loss.

Table 18: Ablation study of EDA. The default parameter is in bold.

		initial stepsize				N_{ADS}				n_a			
Dataset	No.	$\varepsilon/4$	$\varepsilon/2$	ε	$\boldsymbol{2\varepsilon}$	3	**4**	5	10	3	4	**5**	6
CIFAR-10	1	55.50	55.48	55.51	55.49	55.47	55.49	55.50	55.51	55.50	55.48	55.48	55.49
CIFAR-10	2	56.78	56.78	56.78	56.75	56.78	56.76	56.77	56.76	56.77	56.82	56.76	56.76
CIFAR-100	3	18.92	18.92	18.87	18.86	18.88	18.86	18.89	18.89	18.87	18.90	18.89	18.86
CIFAR-100	4	24.49	24.50	24.50	24.47	24.47	24.47	24.49	24.48	24.48	24.49	24.48	24.47
ImageNet	5	29.10	29.08	29.08	29.00	29.00	29.04	29.02	29.04	29.06	29.04	29.04	29.02

Table 20: Robust accuracy of the ablation study for the MDO framework and EDA with different initial point sampling.

CIFAR-10 ($\varepsilon = 8/255$)	model	MDO framework			EDA		
		$\mathrm{x}_{\mathrm{org}}$	$\mathrm{x}_{\mathrm{ODS}}$	$\mathrm{x}_{\mathrm{PAS}}$	$\mathrm{x}_{\mathrm{org}}$	$\mathrm{x}_{\mathrm{ODS}}$	$\mathrm{x}_{\mathrm{PAS}}$
(Andriushchenko & Flammarion, 2020)	PARN-18	43.96 ± 0.03	44.05 ± 0.07	43.95 ± 0.02	43.85 ± 0.02	43.86 ± 0.01	43.85 ± 0.02
(Addepalli et al., 2022)	RN-18	52.50 ± 0.03	52.66 ± 0.12	52.49 ± 0.03	52.41 ± 0.01	52.45 ± 0.02	52.43 ± 0.03
(Sehwag et al., 2022)	RN-18	55.62 ± 0.03	55.60 ± 0.02	55.56 ± 0.02	55.49 ± 0.01	55.50 ± 0.01	55.49 ± 0.01
(Engstrom et al., 2019)	RN-50	49.40 ± 0.03	49.34 ± 0.06	49.43 ± 0.09	49.11 ± 0.01	49.10 ± 0.01	49.10 ± 0.03
(Carmon et al., 2019)	WRN-28-10	59.49 ± 0.03	59.62 ± 0.03	59.49 ± 0.03	59.40 ± 0.01	59.45 ± 0.02	59.40 ± 0.01
(Gowal et al., 2020)	WRN-28-10	62.84 ± 0.02	62.99 ± 0.05	62.82 ± 0.03	62.75 ± 0.01	62.78 ± 0.02	62.75 ± 0.02
(Hendrycks et al., 2019)	WRN-28-10	54.87 ± 0.03	54.84 ± 0.01	54.85 ± 0.04	54.77 ± 0.01	54.77 ± 0.01	54.77 ± 0.02
(Rebuffi et al., 2021)	WRN-28-10	60.78 ± 0.04	60.83 ± 0.07	60.75 ± 0.03	60.69 ± 0.02	60.69 ± 0.02	60.64 ± 0.01
(Sehwag et al., 2020)	WRN-28-10	57.11 ± 0.03	57.22 ± 0.05	57.11 ± 0.02	57.01 ± 0.02	57.07 ± 0.03	57.03 ± 0.01
(Sridhar et al., 2022)	WRN-28-10	59.62 ± 0.02	59.68 ± 0.10	59.55 ± 0.03	59.48 ± 0.03	59.52 ± 0.03	59.46 ± 0.02
(Wang et al., 2020)	WRN-28-10	56.48 ± 0.06	56.52 ± 0.24	56.29 ± 0.02	56.18 ± 0.01	56.20 ± 0.04	56.15 ± 0.02
(Wu et al., 2020)	WRN-28-10	60.01 ± 0.02	60.11 ± 0.09	60.00 ± 0.02	59.94 ± 0.01	59.97 ± 0.03	59.94 ± 0.01
(Ding et al., 2020)	WRN-28-4	43.33 ± 0.16	43.69 ± 0.13	43.32 ± 0.12	41.70 ± 0.04	41.84 ± 0.03	41.74 ± 0.06
(Addepalli et al., 2022)	WRN-34-10	57.78 ± 0.01	58.01 ± 0.09	57.75 ± 0.03	57.70 ± 0.01	57.74 ± 0.01	57.69 ± 0.02
(Sehwag et al., 2022)	WRN-34-10	60.27 ± 0.02	60.30 ± 0.02	60.27 ± 0.05	60.17 ± 0.01	60.18 ± 0.02	60.18 ± 0.02
(Sitawarin et al., 2021)	WRN-34-10	50.72 ± 0.04	50.69 ± 0.03	50.72 ± 0.03	50.59 ± 0.02	50.60 ± 0.02	50.59 ± 0.01
(Zhang et al., 2019)	WRN-34-10	44.62 ± 0.03	44.65 ± 0.04	44.65 ± 0.06	44.51 ± 0.02	44.52 ± 0.02	44.51 ± 0.02
(Zhang et al., 2020)	WRN-34-10	53.53 ± 0.03	53.56 ± 0.03	53.50 ± 0.03	53.43 ± 0.01	53.44 ± 0.03	53.42 ± 0.02
(Sridhar et al., 2022)	WRN-34-15	60.43 ± 0.02	60.44 ± 0.11	60.38 ± 0.03	60.31 ± 0.02	60.32 ± 0.03	60.32 ± 0.01
(Gowal et al., 2020)	WRN-34-20	56.89 ± 0.02	57.08 ± 0.05	56.86 ± 0.02	56.79 ± 0.01	56.81 ± 0.01	56.79 ± 0.03
(Pang et al., 2020)	WRN-34-20	53.85 ± 0.03	53.80 ± 0.03	53.82 ± 0.03	53.64 ± 0.00	53.67 ± 0.01	53.66 ± 0.01
(Rice et al., 2020)	WRN-34-20	53.47 ± 0.02	53.56 ± 0.09	53.46 ± 0.04	53.33 ± 0.01	53.34 ± 0.01	53.34 ± 0.01
(Gowal et al., 2020)	WRN-70-16	57.21 ± 0.02	57.28 ± 0.05	57.18 ± 0.02	57.12 ± 0.00	57.14 ± 0.01	57.12 ± 0.01
(Gowal et al., 2020)	WRN-70-16	65.92 ± 0.04	65.91 ± 0.05	65.87 ± 0.01	65.81 ± 0.03	65.81 ± 0.02	65.83 ± 0.01
(Rebuffi et al., 2021)	WRN-70-16	64.31 ± 0.04	64.36 ± 0.06	64.30 ± 0.05	64.20 ± 0.01	64.20 ± 0.02	64.20 ± 0.03
CIFAR-100 ($\varepsilon = 8/255$)	model	the MDO framework			EDA		
		$\mathrm{x}_{\mathrm{org}}$	$\mathrm{x}_{\mathrm{ODS}}$	$\mathrm{x}_{\mathrm{PAS}}$	$\mathrm{x}_{\mathrm{org}}$	$\mathrm{x}_{\mathrm{ODS}}$	$\mathrm{x}_{\mathrm{PAS}}$
(Rice et al., 2020)	PARN-18	18.97 ± 0.01	19.04 ± 0.09	18.98 ± 0.01	18.87 ± 0.01	18.86 ± 0.01	18.88 ± 0.01
(Hendrycks et al., 2019)	WRN-28-10	28.46 ± 0.02	28.60 ± 0.13	28.41 ± 0.02	28.28 ± 0.01	28.28 ± 0.02	28.27 ± 0.02
(Rebuffi et al., 2021)	WRN-28-10	32.22 ± 0.03	32.45 ± 0.23	32.07 ± 0.04	31.97 ± 0.01	31.99 ± 0.01	31.94 ± 0.03
(Addepalli et al., 2022)	WRN-34-10	31.90 ± 0.03	32.01 ± 0.16	31.87 ± 0.02	31.78 ± 0.01	31.78 ± 0.01	31.78 ± 0.01
(Cui et al., 2021)	WRN-34-10	29.00 ± 0.04	29.14 ± 0.23	28.96 ± 0.05	28.85 ± 0.02	28.84 ± 0.02	28.83 ± 0.02
(Sitawarin et al., 2021)	WRN-34-10	24.61 ± 0.04	24.67 ± 0.07	24.66 ± 0.03	24.50 ± 0.01	24.48 ± 0.01	24.50 ± 0.01
(Wu et al., 2020)	WRN-34-10	28.95 ± 0.04	29.17 ± 0.19	28.89 ± 0.04	28.76 ± 0.01	28.77 ± 0.01	28.76 ± 0.01
(Cui et al., 2021)	WRN-34-20	30.06 ± 0.02	30.11 ± 0.10	30.00 ± 0.02	29.86 ± 0.01	29.86 ± 0.01	29.85 ± 0.01
(Gowal et al., 2020)	WRN-70-16	30.13 ± 0.03	30.29 ± 0.21	30.07 ± 0.01	29.97 ± 0.01	29.99 ± 0.01	29.96 ± 0.01
(Gowal et al., 2020)	WRN-70-16	37.06 ± 0.04	37.17 ± 0.15	36.96 ± 0.04	36.81 ± 0.01	36.83 ± 0.01	36.81 ± 0.01
(Rebuffi et al., 2021)	WRN-70-16	34.69 ± 0.04	34.81 ± 0.09	34.64 ± 0.03	34.57 ± 0.02	34.58 ± 0.02	34.55 ± 0.01
ImageNet ($\varepsilon = 4/255$)	model	the MDO framework			EDA		
		$\mathrm{x}_{\mathrm{org}}$	$\mathrm{x}_{\mathrm{ODS}}$	$\mathrm{x}_{\mathrm{PAS}}$	$\mathrm{x}_{\mathrm{org}}$	$\mathrm{x}_{\mathrm{ODS}}$	$\mathrm{x}_{\mathrm{PAS}}$
(Salman et al., 2020)	RN-18	25.31 ± 0.05	25.48 ± 0.21	25.25 ± 0.02	25.12 ± 0.02	25.10 ± 0.01	25.11 ± 0.02
(Engstrom et al., 2019)	RN-50	29.29 ± 0.06	29.77 ± 0.48	29.27 ± 0.07	29.01 ± 0.01	29.00 ± 0.02	29.01 ± 0.01
(Salman et al., 2020)	RN-50	34.67 ± 0.09	35.13 ± 0.30	34.73 ± 0.07	34.54 ± 0.03	34.54 ± 0.03	34.52 ± 0.02
(Wong et al., 2020)	RN-50	26.41 ± 0.12	26.71 ± 0.34	26.32 ± 0.13	26.13 ± 0.12	26.14 ± 0.08	26.12 ± 0.10
(Salman et al., 2020)	WRN-50-2	38.42 ± 0.04	39.36 ± 0.10	38.31 ± 0.06	38.04 ± 0.02	38.03 ± 0.03	38.03 ± 0.02

G.2 The impact of initial point sampling on the performance of the MDO framework and EDA

Table 20 shows the robust accuracy of the MDO framework and EDA using different initial point sampling, including original input ($\mathrm{x}_{\mathrm{org}}$), Output Diversified Sampling (ODS, $\mathrm{x}_{\mathrm{ODS}}$), and Prediction Aware Sampling (PAS, $\mathrm{x}_{\mathrm{PAS}}$). the MDO framework and EDA showed slightly better performance using PAS than using the original input. The difference in robust accuracy between the MDO framework using the original input and ODS was larger than that using PAS and the original input. For EDA, the difference in robust accuracy among the three initial point sampling was smaller than that of the MDO framework. These experimental results suggest that initial point sampling does affect attack performance but that differences in the framework and the pairs of objective function and search direction have a greater impact.

G.3 Influence of DI and P^eon ADS's performance

To investigate the influence of two terms that appear in the computation of indicators used in ADS, we executed the MDO framework and EDA with the pairs of search direction and objective function selected using DI, P^e, and D^e. The model under attack is a subset of the 41 CNN-based robust models used in the main text. The allowable perturbation size, test samples, and computing environment are the same as in the experiments described in the text. The results of the experiments are shown in Table 21. DI, P^e, and D^e in Table 21 indicate that the pairs of search direction and objective function were selected using these indicators, respectively. From the experimental results, the highest attack performance is expected when P^e is used to select the pairs. However, focusing on the MDO framework against CIFAR-10 and CIFAR-100 models, DI showed the highest number of models that performed better than the 2nd-best. Thus, the MDO framework is expected to show higher performance when the pairs are selected with DI than with P^e for a wide range of models. The attack performance of the MDO framework with D^e is often intermediate between that with DI and P^e. Thus, we used D^e to select the pairs in the text, considering performance and generalization. The reason for the better performance in selecting the pairs that maximize P^e in EDA can be attributed to its different nature from targeted attacks. Targeted attacks maximize the prediction probability of the target class. Thus, the class with the highest prediction probability, excluding the correct prediction, is unlikely to change during the search. In contrast, the pairs that maximize P^e have a high probability that the class with the highest prediction probability, excluding the correct prediction, will change during the search. The pairs that maximize P^e are expected to be more complementary to the targeted attack than those that maximize DI.

H EDA's performance against randomized defenses

We investigated the efficacy of EDA against two randomized defenses, including Double-Win Quant (DWQ) (Fu et al., 2021) and Random Normalization Aggregation (RNA) (Dong et al., 2022). For the models trained on CIFAR-10/100 and SVHN (Netzer et al., 2011), we used $\varepsilon = 8/255$ and 10,000 test images. For the ImageNet, we used $\varepsilon = 4/255$ and 5,000 images, the same as the RobustBench. We report the results of a single run with a fixed random seed for reproducibility. The compared methods are AutoAttack (AA) and Adaptive Auto Attack (A^3). A^3 showed sufficiently fast and strong performance among recently proposed attacks against CNN-based robust models. For consistency with AA, we calculated the robust accuracy using generated adversarial examples when the attack terminated and compared these values because the model with randomized defense may produce different outputs with each inference. Official implementations of AA and EDA calculate robust accuracy in this way, but the official implementation of A^3 calculates differently. Therefore, we slightly modified the implementation of A^3 to calculate robust accuracy in the same way. The parameters of the compared attacks were the same as in the main text. Table 22 shows the robust accuracy and runtime of the standard version of AA, A^3, and EDA. From Table 22, EDA showed higher attack performance in less computation time. Although these are the results of a single run, EDA is expected to be effective for randomized defenses.

I Limitations and assumptions

This study has the following assumptions and limitations. EDA assumes to be able to access the outputs and gradients of the attacked models. Similar to the Adaptive Auto Attack (A^3), EDA assumes to attack the set of images. These assumptions are reasonable for robustness evaluation and adversarial training. The number of combinations of search directions and objective functions grows exponentially. Thus, the number of search directions and objective functions to be selected should be manageable.

Expected situations in EDA work well This study experimentally verifies the performance of EDA, focusing on image classifiers using deterministic defenses such as adversarial training. Although EDA was tested with images of different resolutions, domains, and models of different architectures, EDA worked in most settings. Therefore, it can be assumed that EDA has some generalization performance for image classifiers using deterministic defenses. Similar to AA, EDA is applicable to the models whose gradients

Table 21: Influence of DI and P^e on ADS's performance. The lowest and second lowest robust accuracy of each attack is in **bold** and underlined, respectively.

defense	model	MDO framework			EDA		
CIFAR-10		DI	P^e	D^e	DI	P^e	D^e
Andriushchenko & Flammarion (2020)	PARN-18	**43.90**	44.00	43.95	**43.85**	**43.85**	43.86
Addepalli et al. (2022)	RN-18	52.52	**52.45**	52.50	52.42	**52.41**	52.43
Sehwag et al. (2022)	RN-18	55.61	55.59	**55.58**	55.50	55.50	**55.49**
Engstrom et al. (2019)	RN-50	49.50	**49.31**	49.52	49.11	**49.09**	49.12
Carmon et al. (2019)	WRN-28-10	59.49	59.51	**59.46**	59.42	59.42	**59.38**
Sehwag et al. (2020)	WRN-28-10	**57.10**	**57.10**	57.14	57.03	**57.01**	57.06
Sridhar et al. (2022)	WRN-28-10	**59.50**	59.54	59.59	**59.45**	59.46	59.47
Wu et al. (2020)	WRN-28-10	60.02	60.04	**59.99**	59.96	59.96	**59.95**
Addepalli et al. (2022)	WRN-34-10	57.73	**57.72**	**57.72**	57.71	**57.66**	57.68
Sitawarin et al. (2021)	WRN-34-10	50.75	50.85	**50.70**	50.62	50.60	**50.58**
Pang et al. (2020)	WRN-34-20	53.81	**53.74**	53.81	**53.65**	**53.65**	**53.65**
Gowal et al. (2020)	WRN-70-16	65.89	**65.80**	65.85	65.83	**65.78**	65.81
# bold		3	6	5	3	7	5
# underlined		6	2	5	5	4	3
CIFAR-100							
Rice et al. (2020)	PARN-18	**18.91**	19.01	18.97	**18.88**	**18.88**	18.89
Addepalli et al. (2022)	WRN-34-10	**31.81**	31.85	31.86	**31.77**	**31.77**	31.78
Cui et al. (2021)	WRN-34-10	**28.93**	28.96	28.99	28.85	28.82	**28.81**
Sitawarin et al. (2021)	WRN-34-10	24.63	**24.59**	24.65	24.49	**24.48**	24.49
Wu et al. (2020)	WRN-34-10	**28.87**	28.90	28.88	28.78	28.78	**28.77**
Gowal et al. (2020)	WRN-70-16	37.03	**36.94**	36.96	36.84	**36.79**	36.80
Rebuffi et al. (2021)	WRN-70-16	34.65	**34.64**	34.65	34.57	**34.54**	34.55
# bold		4	3	0	2	5	2
# underlined		2	2	4	2	2	5
ImageNet							
Salman et al. (2020)	RN-18	25.20	**25.16**	25.22	25.12	**25.06**	25.10
Engstrom et al. (2019)	RN-50	29.20	**29.10**	29.20	**28.98**	29.00	29.00
Salman et al. (2020)	RN-50	34.64	**34.62**	34.84	34.52	**34.50**	34.56
Wong et al. (2020)	RN-50	26.34	**26.10**	26.22	26.10	**25.96**	26.02
Salman et al. (2020)	WRN-50-2	38.32	38.30	**38.28**	38.04	**38.00**	38.02
# bold		0	4	2	1	4	0
# underlined		3	1	1	1	1	4

are available. However, the performance of EDA against models out of image classifiers needs further investigation.

Expected situations in EDA do not work well EDA contains gradient-based attacks as some components. Therefore, EDA's performance may be degraded for models which cause incorrect gradient calculations. As reported by Croce & Hein (2020b), existing techniques like expectation over transformation might help improve the attack performance on these models. Also, the performance of EDA may be degraded when the MDO and MT strategies do not work well.

Table 22: Robust accuracy of randomized defenses. The lowest robust accuracy is in bold.

CIFAR-10 ($\varepsilon = 8/255$)			AA		A^3		EDA	
Defense	Model	clean	acc	sec	acc	sec	acc	sec
RNA	RN-18	84.48	62.13	5,094	65.99	1,342	**53.87**	743
RNA	WRN-32	86.49	64.53	30,560	69.70	16,087	**53.43**	4,764
DWQ	PARN-18	82.11	53.81	6,469	56.63	3,161	**48.00**	897
DWQ	WRN-34	81.56	58.05	46,058	60.21	7,043	**50.32**	5,856
SVHN ($\varepsilon = 8/255$)								
DWQ	PARN-18	81.97	31.90	4,117	38.80	1,802	**28.53**	676
DWQ	WRN-34	88.65	37.20	29,969	43.87	5,438	**35.23**	4,027
CIFAR100 ($\varepsilon = 8/255$)								
RNA	RN-18	56.75	42.78	2,378	47.55	1,921	**30.67**	426
RNA	WRN-32	60.19	41.94	15,296	45.98	17,575	**30.71**	1,851
DWQ	PARN-18	55.96	32.61	3,559	33.72	1,491	**26.51**	607
DWQ	WRN-34	55.94	36.77	25,539	37.82	4,658	**29.64**	3,311
ImageNet ($\varepsilon = 4/255$)								
DWQ	RN-50 (free)	49.44	35.62	2,386	45.08	109	**24.18**	829
DWQ	RN-50 (fgsmrs)	62.18	39.08	21,994	40.72	4,854	**33.72**	3,773