Towards More Reliable Adversarial Robustness Evaluation via Sequential Difference Maximization

Anonymous authors
Paper under double-blind review

Abstract

Gradient-based adversarial attacks are important methods for evaluating model robustness. However, since the proposal of APGD, it has been difficult for such methods to achieve significant breakthroughs. To achieve such an effect, we first analyze the issue of "high-loss non-adversarial examples" that degrade attack performance in previous methods, and prove that this issue arises from inappropriate objectives for adversarial example generation. Subsequently, we reconstruct the objective as "maximizing the difference between the non-ground-truth label probability upper bound and the ground-truth label probability", and propose a novel and powerful gradient-based attack method named Sequential Difference Maximization (SDM). SDM establishes a three-layer optimization framework of "Cycle-Stage-Step". It adopts the negative probability loss function and the Directional Probability Difference Ratio (DPDR) loss function in the initial and subsequent optimization stages, respectively, and approaches the ideal objective of adversarial example generation via stage-wise sequential optimization. Experiments demonstrate that compared with previous state-of-the-art methods, SDM not only achieves stronger attack performance but also exhibits superior cost-effectiveness.

1 Introduction

Adversarial examples can cause well-trained deep neural networks (DNNs) to make incorrect judgments Goodfellow et al. (2015); Akhtar et al. (2021). Taking image classification tasks as an example, adversarial example attacks (hereafter referred to as adversarial attacks) can induce target models to output misclassification results with high confidence, posing severe threats to downstream tasks such as face recognition and object detection Schwinn et al. (2023); Bartoldson et al. (2024). However, from another perspective, adversarial attacks also profoundly reveal the limitations of DNNs in terms of robustness and essential understanding Croce & Hein (2020a), and are thus commonly employed to evaluate model robustness Liu et al. (2024). This drives researchers to design methods that learn intrinsic, stable features rather than merely superficial statistical patterns.

An important application scenario of adversarial attacks is the white-box setting Baniecki & Biecek (2024). In this scenario, attackers have access to all information of the defender, which represents the most stringent evaluation condition. The representative gradient-based methods, Projected Gradient Descent (PGD) Madry et al. (2018) and its advanced variants (e.g., Auto-PGD, APGD) Croce & Hein (2020a) have almost become the universal benchmark for evaluating model robustness, owing to their standardized perturbation projection constraints, moderate computational cost and strong attack performance Badjie et al. (2025). However, we identify a concealed defect termed "high-loss non-adversarial examples" in these approaches. Specifically, for certain natural examples, the high-loss examples generated by PGD fail to launch effective attacks, whereas low-loss examples generated by other methods can achieve successful attacks (detailed in Sec. 3). This problem essentially stems from the unreasonable objective for adversarial example generation adopted by these methods, which impairs their attack performance and undermines the effectiveness in evaluating the adversarial robustness of models.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To address the above problem, we first reconstruct the generation objective of adversarial examples from "minimizing the ground-truth label probability" to "maximizing the difference between the non-ground-truth label probability upper bound and the ground-truth label probability". In contrast to the former, while the latter is more accurate, its higher nonlinearity and more complex solution space pose greater challenges to optimization. Based on the prior research Liu et al. (2025), we propose the Sequential Difference Maximization (SDM) algorithm to solve this problem. SDM decomposes the solution process into multiple optimization stages: (1) The initial stage aims to minimize the ground-truth label probability and constructs a negative probability loss function accordingly, thereby compressing the solution space to reduce complexity. (2) Subsequent stages introduce the Directional Probability Difference Ratio (DPDR) loss function to adjust the distribution of the non-ground-truth label probabilities, thus elevating their upper bound. Each stage contains multiple identical optimization steps. Within each step, adversarial perturbations are updated along the input-gradient direction of the loss function to maximize the predefined loss value. The optimal solution of one stage serves as the initialization for the next, with all stages optimized in a fixed order to form a single optimization cycle. After each cycle converges, its optimized result is fed as the starting point of the next cycle for further tuning. Overall, SDM establishes a three-level "Cycle-Stage-Step" optimization framework to produce high-performance adversarial examples. Extensive experimental results demonstrate that SDM outperforms previous methods in attack performance and cost-effectiveness. We summarize the main contributions as follows.

- We propose a novel adversarial attack named "SDM", which constructs a three-layer optimization framework comprising "Cycle-Stage-Step". SDM sequentially introduces the negative probability and the DPDR loss functions, and generates high-performance adversarial examples by performing the ascent of different losses.
- We identify the problem of "high-loss non-adversarial examples", which reveals the irrationality in the generation objective of previous attack methods. Based on this, we reconstruct the objective as "maximizing the difference between the non-ground-truth label probability upper bound and the ground-truth label probability".
- Experiments demonstrate that SDM effectively alleviates the problem of high-loss non-adversarial examples. It achieves exceptional performance in attack effectiveness, cost-effectiveness and anti-interference capability, significantly outperforming the prior state-of-the-art methods.

2 Related Work

2.1 Adversarial Example

Let f be a multiclass classifier mapping inputs from $\mathbb{R}^d$ to output scores in $\mathbb{R}^K$ $(K > 2)$. d and K represent the dimension of the input space and the number of labels, respectively. For a natural example $\boldsymbol{x} \in [0,1]^d \subset \mathbb{R}^d$ with the ground-truth label y, the model produces a vector $(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x}))^\top$ as the output score.

Adversarial examples refer to input examples that can actively mislead a target classification model into making incorrect predictions. If no specific direction of the incorrect prediction is designated, they are termed untargeted adversarial examples. As the core research object of this paper, untargeted adversarial examples are directly referred to as "adversarial examples" in subsequent content. An ideal adversarial example $\boldsymbol{x}'$ can be formulated as

$$\underset{k \in \{1,\ldots,K\}}{\arg\max}\ f_k(\boldsymbol{x}') \neq y \quad \text{s.t.} \quad \|\boldsymbol{x}' - \boldsymbol{x}\|_p \leq \epsilon, \tag{1}$$

where ϵ and $\|\cdot\|_p$ represent the perturbation budget and the $\boldsymbol{\ell}_p$-norm constraint, respectively.

2.2 Gradient-Based Adversarial Attack

Gradient-based methods represent the most commonly adopted approach for adversarial attacks. The classic Fast Gradient Sign Method (FGSM) first formalized the attack objective as

$$\min_{\boldsymbol{x}'} \boldsymbol{P}_y \quad \text{s.t.} \quad \|\boldsymbol{x}' - \boldsymbol{x}\|_p \leq \epsilon, \tag{2}$$

where $\boldsymbol{P}_y$ is the probability of the adversarial example $\boldsymbol{x}'$ for the ground-truth label y. Let $\boldsymbol{S}_k = f_k(\boldsymbol{x}')$ $(k = 1, 2, \ldots, K)$, and the probability distribution $\boldsymbol{P}$ of $\boldsymbol{x}'$ can be expressed as

$$\boldsymbol{P} = (\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_K)^\top \in (0,1)^K, \quad \boldsymbol{P}_k = \frac{\mathrm{e}^{\boldsymbol{S}_k}}{\sum_{i=1}^{K} \mathrm{e}^{\boldsymbol{S}_i}}. \tag{3}$$

Compared with Eq. 1, which exhibits high nonlinearity and poses great optimization challenges, Eq. 2 clearly offers a more explicit optimization pathway. Specifically, FGSM first calculates the Cross Entropy (CE) loss between the model's output scores and the ground-truth label, then amplifies the adversarial perturbation along the direction of the partial derivative (gradient) of this loss with respect to the input example, thereby generating an adversarial example. The employed CE loss function is expressed as

$$L_{\mathrm{CE}} = -\sum_{k=1}^{K} y_k \log(\boldsymbol{P}_y). \tag{4}$$

For common hard-label datasets (featuring a single class label per example, e.g., CIFAR and ImageNet, the CE loss function actually degenerates into

$$L_{\mathrm{CE}} = -\log(\boldsymbol{P}_y). \tag{5}$$

It can be observed that, the optimization directions of Loss 5 and Objective 2 are equivalent; maximizing the CE loss is equivalent to minimizing the probability of the ground-truth label.

3 Motivation

As illustrated in Sec. 2, gradient-based attack methods such as FGSM and PGD set the generation objective of adversarial examples as minimizing the probability of the ground-truth label. However, is it truly reasonable?

We argue that this setup is unreasonable. Next, we will elaborate on our viewpoint by discussing an interesting problem termed "high-loss non-adversarial examples".

For a natural example $\boldsymbol{x}$ with the ground-truth label $y = 4$, $\boldsymbol{x}'_{(1)}$ and $\boldsymbol{x}'_{(2)}$ are two examples generated under the perturbation budget of $\boldsymbol{\ell}_\infty = 8/255$. The output scores and probabilities of the target model for them are shown in Tab. 1.

For example $\boldsymbol{x}'_{(1)}$, the classification score and probability at the predicted label "4" are the highest. Its predicted label matches the ground-truth label, and the attack is thus deemed failed, with a corresponding CE loss of 1.196. For example $\boldsymbol{x}'_{(2)}$, the score and probability at the predicted label "6" are the highest. Its predicted label is different from the ground-truth label, and the attack is therefore judged successful, with a corresponding CE loss of 1.058. It can be clearly observed that for example $\boldsymbol{x}'_{(2)}$, the attack succeeds despite a relatively low CE loss value; whereas for example $\boldsymbol{x}'_{(1)}$, the attack fails even with a relatively higher CE loss value. Example $\boldsymbol{x}'_{(1)}$ corresponds to the "high-loss non-adversarial example" mentioned previously. This is not an isolated case. In App. A, we verify the widespread existence of such examples via a series of empirical analyses.

This problem arises since minimizing the ground-truth class probability cannot ensure it drops below the upper bound of incorrect class probabilities—a necessary condition for valid adversarial examples. To elaborate on this mechanism, we analyze the probability distribution landscapes Li et al. (2018) of six PGD-generated adversarial samples. Experimental setups are detailed in App. B, and visualization results are presented in Fig. 1. The upper

Table 1: Scores and probabilities of two examples generated from a natural example.

In.	Out.	k 1	2	3	4 (y)	5	6	7	8	9	10	Loss	Pred.	Attack
$\boldsymbol{x}'_{(1)}$	$\boldsymbol{S}_k$	0.314	-1.267	-0.126	1.438	0.264	1.036	0.191	-0.118	-0.498	-1.041	1.196	4	Failed
	$\boldsymbol{P}_k$	9.83%	2.02%	6.33%	30.25%	9.35%	20.24%	8.69%	6.38%	4.36%	2.54%			
$\boldsymbol{x}'_{(2)}$	$\boldsymbol{S}_k$	-0.674	-1.434	-0.398	2.864	-0.488	3.367	-0.371	-0.613	-1.421	-0.833	1.058	6	Succ.
	$\boldsymbol{P}_k$	1.01%	0.47%	1.33%	34.74%	1.22%	57.45%	1.37%	1.07%	0.48%	0.86%			

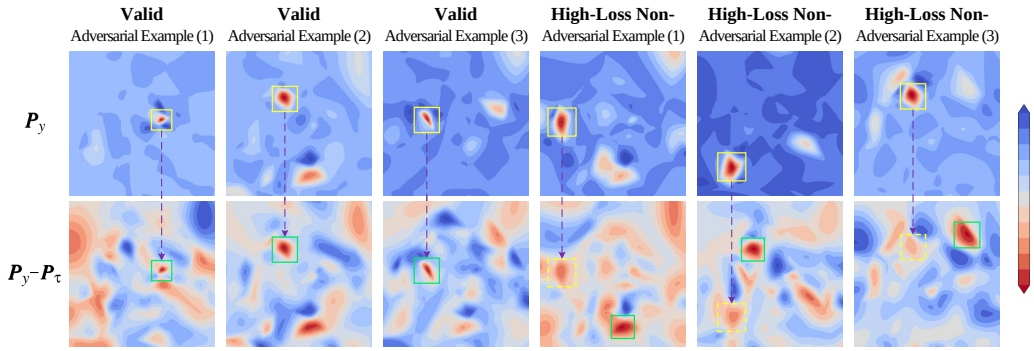

Figure 1: Probability distribution landscapes of valid adversarial examples and high-loss non-adversarial examples.

subplots plot ground-truth probability over perturbation spaces, with yellow boxes marking regions of minimal probability. The lower subplots show probability gaps (ground-truth probability minus the maximum incorrect-class probability), where green boxes highlight minimal-gap regions. Arrows link paired marked regions to align their spatial positions across subplots. For the first three samples, yellow and green boxes overlap: the minima of ground-truth probability and probability gap lie within identical perturbation regions, so optimizing the former directly optimizes the latter. Accordingly, PGD produces successfully attacking adversarial examples. For the remaining three samples, however, the two boxes separate, meaning the two minima occur in distinct regions. Minimizing ground-truth probability thus cannot minimize the gap, and PGD merely yields samples with low ground-truth probability yet failed attacks. As derived in Eq. 5, these samples retain large CE losses, matching the high-loss non-adversarial cases described above.

Inspired by the above discussion, we recognize that a low ground-truth label probability and a high CE loss are not sufficient conditions for generating valid adversarial examples. Both the ground-truth label probability and the non-ground-truth label probabilities must be constrained when generating adversarial examples. Therefore, we reconstruct the generation objective of adversarial examples as "maximizing the difference between the non-ground-truth label probability upper bound and the ground-truth label probability", which aims to achieve successful attacks by constraining all labels. This objective can be formulated as

$$\max_{\boldsymbol{x}'} \{\boldsymbol{P}_\tau - \boldsymbol{P}_y\} \quad \text{s.t.} \quad \|\boldsymbol{x}' - \boldsymbol{x}\|_p \leq \epsilon, \tag{6}$$

where τ denotes the non-ground-truth label with the maximum probability. The reconstructed Objective 6 features two distinct optimization directions: "reducing the ground-truth label probability" and "increasing the non-ground-truth label probability upper bound". App. C proves that these two directions are neither mutually independent nor completely consistent; instead, they exhibit partial correlation. This correlation leads to a dramatic surge in the optimization difficulty for Objective 6. To address this issue, some intuitive solution strategies include directly converting the optimization objective into a loss function (C&W Carlini & Wagner (2017)) or adopting the Difference of Logits Ratio (DLR) loss function (APGD). However, these methods tend to fall into local optima due to the local conflicts between the two optimization directions, resulting in only moderate optimization performance. In the next section, we propose a novel solution method.

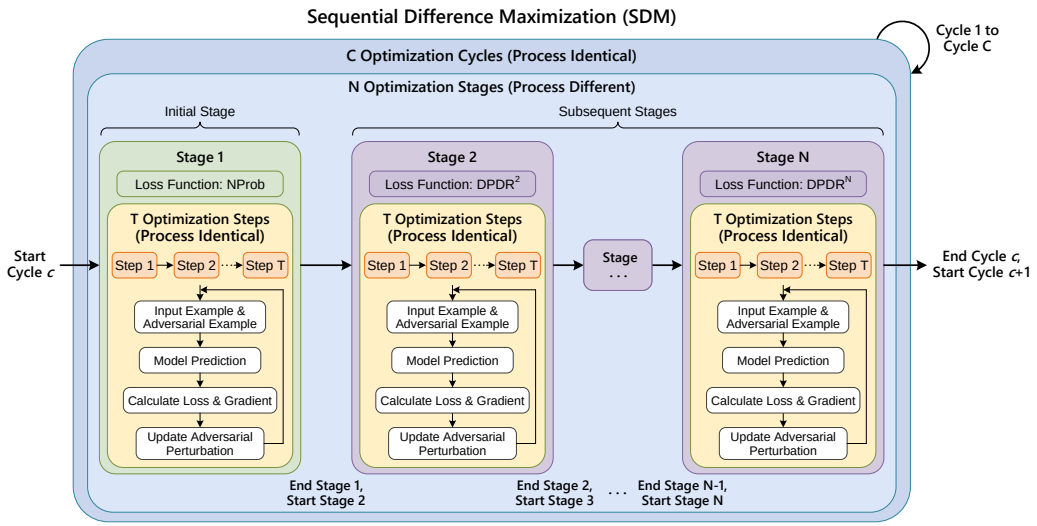

Figure 2: Overall structure of SDM.

4 Methodology

Herein, we propose the Sequential Difference Maximization (SDM) to solve the adversarial example generation problem from Generation Objective 6. Specifically, SDM is a gradient-based multi-stage iterative method, which decomposes the overall optimization objective into multiple optimization sub-objectives. It performs sequential optimization on these sub-objectives, thereby gradually approaching the optimal solution of the overall objective.

4.1 Overall Structure

As Fig. 2 shows, SDM has C identical optimization cycles.

The current cycle takes the solution of the previous cycle as the initial solution, and the algorithm terminates when the cycle count reaches the preset threshold value C. Each cycle contains N different optimization stages internally.

The current stage takes the solution of the previous stage as the initial solution, and terminates the current optimization cycle when the stage count reaches the preset value N. The initial stage adopts the negative probability loss function; subsequent stages adopt the DPDR (see Subsec. 4.3) loss function (not completely consistent, with a hyperparameter related to the stage count). Each stage contains T identical optimization steps internally.

The current step takes the solution of the previous step as the initial solution, and terminates the current optimization stage when the step count reaches the preset value T. Each step amplifies the adversarial example along the gradient direction of the loss (the specific loss function is determined by the current optimization stage) with respect to the input example, aiming to effectively enhance the attack performance.

4.2 Adversarial Example Optimization in a Single Step

For the t-th step, the generated adversarial example is

$$\boldsymbol{x}'_t = \boldsymbol{x} + \text{Proj}\left(\boldsymbol{x}'_{t-1} - \boldsymbol{x} + \alpha \cdot \text{sign}(Grad), -\epsilon, \epsilon\right), \tag{7}$$

where α denotes the step size for adjusting the adversarial perturbation, and $\text{Proj}(\cdot)$ and $\text{sign}(\cdot)$ represent the projection function and sign function, respectively. $Grad$ denotes the gradient of the loss between the output probability distribution and the ground-truth label with respect to the adversarial example, which is expressed as

$$Grad = \nabla_{\boldsymbol{x}'_{t-1}} L\left(\boldsymbol{P}, y\right). \tag{8}$$

Under the $\boldsymbol{\ell}_\infty$-norm constraint, $\mathrm{Proj}(\cdot)$ is formulated as

$$\mathrm{Proj}\left(\boldsymbol{x}, a, b\right) = \max\left(a, \min\left(\boldsymbol{x}, b\right)\right), \tag{9}$$

where the loss function L varies across each stage, which will be elaborated on below.

4.3 Sub-Objectives and Losses for Different Stages

For the initial optimization stage, SDM sets the optimization sub-objective as "reducing the ground-truth label probability", and defines the negative probability loss function L_{NProb}:

$$L_{\mathrm{NProb}} = -\boldsymbol{P}_y. \tag{10}$$

Compared with optimizing the non-ground-truth label probability upper bound first, prioritizing the ground-truth label probability can effectively compress the solution space, thereby facilitating subsequent optimization steps.

For each subsequent stage, SDM sets its optimization sub-objectives to two aspects: (1) maintaining the ground-truth label probability at a low value; (2) reducing the n-th label probability after descending sorting, so as to elevate the non-ground-truth label probability upper bound. To achieve these objectives, this paper proposes a novel loss function termed Directional Probability Difference Ratio (DPDR), whose mathematical expression is:

$$L_{\mathrm{DPDR}}^{(n)} = \frac{\boldsymbol{P}_\tau - \boldsymbol{P}_y}{\phi - \mathrm{sign}\big(\boldsymbol{P}_\tau - \boldsymbol{P}_y\big) \cdot \big(\boldsymbol{P}_\tau - \grave{\boldsymbol{P}}_n - \phi\big) + \zeta}. \tag{11}$$

In Eq. 11, ζ and ϕ are two items devised for engineering design. ζ is a numerical stability constant with a default value of 10^{-10}, which can avoid numerical anomalies caused by the denominator approaching zero. ϕ precisely regulates the upper and lower bounds of the denominator to accelerate the optimization process, which is expressed as

$$\phi = 0.5 \times \max\big(\boldsymbol{P}_\tau - \grave{\boldsymbol{P}}_n\big). \tag{12}$$

The optimization mechanism of the DPDR loss function is described as follows.

(1) When the adversarial attack fails, i.e., the non-ground-truth label probability upper bound is less than the ground-truth label probability ($\boldsymbol{P}_\tau < \boldsymbol{P}_y$), the denominator of the DPDR loss can be simplified to $\boldsymbol{P}_\tau - \grave{\boldsymbol{P}}_n - \phi$. In this case, the numerator is always negative while its denominator is always positive. Increasing the DPDR loss will lead to simultaneous increases in the values of both its numerator and denominator, which ultimately achieves the effect of compressing the n-th largest label probability $\grave{\boldsymbol{P}}_n$ to increase the largest non-ground-truth label probability (i.e., the non-ground-truth label probability upper bound $\boldsymbol{P}_\tau$).

(2) When the adversarial attack succeeds, i.e., the non-ground-truth label probability upper bound is greater than the ground-truth label probability ($\boldsymbol{P}_\tau > \boldsymbol{P}_y$), the denominator of the DPDR loss can be simplified to $\max(\boldsymbol{P}_\tau - \grave{\boldsymbol{P}}_n) - (\boldsymbol{P}_\tau - \grave{\boldsymbol{P}}_n) + \zeta$. Thus, both the numerator and denominator are always positive. Increasing the value of DPDR loss will raise the value of its numerator and reduce the value of its denominator, which ultimately still achieves the effect of compressing the n-th largest label probability $\grave{\boldsymbol{P}}_n$ to increase the non-ground-truth label probability upper bound $\boldsymbol{P}_\tau$.

It can be observed that, due to the adoption of the sign function and bias term in the DPDR loss, it is guaranteed to compress the n-th largest label probability regardless of whether the attack succeeds or not. Through this approach, the DPDR loss gradually increases the non-ground-truth label probability upper bound, thereby achieving consistency and stability of the optimization direction in the process of adversarial example generation.

4.4 Detailed Algorithm Flow

This section summarizes the implementation process of the proposed SDM algorithm.

Algorithm 1 Detailed SDM procedure under $\boldsymbol{\ell}_\infty$-norm.

Input: A natural example $\boldsymbol{x}$ with the ground-truth label y, a perturbation budget ϵ and step size α, the optimization cycles C, the optimization stages N, and the iteration steps T.
Output: The adversarial example $\boldsymbol{x}'$.

Set the initial value of the adversarial example to the natural example $\boldsymbol{x}'_0 = \boldsymbol{x}$;
for $c = 1$ to C do
  for $n = 1$ to N do
    for $t = 1$ to T do
      if n is 1 then
        Set the loss function as $L = L_{\text{NProb}}$;
      else
        Set the loss function as $L = L^{(n)}_{\text{DPDR}}$;
      end if
      Obtain the probability distribution $\boldsymbol{P}$ of the adversarial example $\boldsymbol{x}'_{t-1}$;
      Calculate the loss $L(\boldsymbol{P}, y)$;
      Compute the gradient $Grad = \nabla_{\boldsymbol{x}'_{t-1}} L(\boldsymbol{P}, y)$ of the loss with respect to $\boldsymbol{x}'_{t-1}$;
      Update the adversarial example $\boldsymbol{x}'_t = \boldsymbol{x} + \text{Proj}\left(\boldsymbol{x}'_{t-1} - \boldsymbol{x} + \alpha \cdot \text{sign}(Grad), -\epsilon, \epsilon\right)$;
    end for
    Set the initial value of the adversarial example as $\boldsymbol{x}'_0 = \boldsymbol{x}'_T$.
  end for
end for

(1) In the initial optimization stage (Stage 1), the negative probability loss L_{NProb} is adopted as the loss function, and gradient ascent is performed through multiple iterative steps to find the optimal solution for Stage 1.

(2) Optimization Stage 2 takes the optimal solution of Stage 1 as the initial solution, adopts $L^{(2)}_{\text{DPDR}}$ as the dedicated loss function, and repeats the same multi-step iterative process as in Stage 1 to find the optimal solution for Stage 2. Subsequently, Stage 3 takes the optimal solution of Stage 2 as the initial solution, adopts $L^{(3)}_{\text{DPDR}}$ as the loss function, and continues the aforementioned iterative steps. This process is repeated until Stage N is completed. In this process, SDM gradually reduces the probability values of the 2nd, 3rd, ..., and ultimately the N-th largest label probability through the DPDR loss function, thereby continuously elevating the non-ground-truth label probability upper bound.

(3) Optimization Stages 1 to N are executed sequentially to form a complete optimization cycle. After one cycle, a new cycle is initiated to mitigate blind spots in optimization.

The SDM algorithm can be adapted to $\boldsymbol{\ell}_\infty$- and $\boldsymbol{\ell}_2$-norm constraints. The detailed procedure under the $\boldsymbol{\ell}_\infty$-norm constraint is shown in Algorithm 1. The procedure under the $\boldsymbol{\ell}_2$-norm constraint is essentially similar to that under the $\boldsymbol{\ell}_\infty$-norm constraint, with the only difference lying in the method of projecting the adversarial example back to the perturbation domain. This projection method has been thoroughly elaborated in detail in App. D.

5 Experiments

5.1 Experimental Setup

Models and Datasets. ResNet-34 (CNN), ViT-B and Swin-L (Transformer) are adopted as victim target models. CIFAR-10, CIFAR-100 Krizhevsky & Hinton (2009), Mini-ImageNet Vinyals et al. (2016) and ImageNet-1K Deng et al. (2009) are used as evaluation datasets.

Baseline Defenses. The defense methods include Vanilla Adversarial Training (VAT) Madry et al. (2018), Misclassification Aware Adversarial Training (MART) Wang et al. (2020), Helper-Based Adversarial Training (HAT) Rade & Moosavi-Dezfooli (2022), and Logit-Oriented Adversarial Training (LOAT) Yin & Ruan (2024).

Baseline Attacks. The attack methods competing with the SDM include PGD, C&W Carlini & Wagner (2017), Fast Adaptive Boundary attack (FAB) Croce & Hein (2020b) and APGD. The specific parameters will be elaborated on in subsequent experiments.

Table 2: ASR (%) of various attack methods on the small-scale datasets.

Constraint	Attack	CIFAR-10				CIFAR-100				Mini-ImageNet			
		VAT	MART	HAT	LOAT	VAT	MART	HAT	LOAT	VAT	MART	HAT	LOAT
Budget: ℓ_∞ =8/255 Iterations: 500	PGD	56.55	48.47	46.91	45.10	77.23	72.53	70.55	68.96	78.34	75.97	73.49	71.14
	C&W	56.28	53.70	50.63	49.28	78.06	77.11	75.96	72.16	80.47	78.96	76.80	74.42
	FAB	57.01	51.62	50.18	47.24	78.49	75.96	74.38	71.05	78.79	76.19	75.37	72.07
	APGD	57.28	49.37	47.66	45.96	78.20	73.20	71.13	69.56	79.15	77.57	74.20	71.85
	SDM	58.79	54.43	52.92	51.04	80.63	77.89	76.76	74.17	83.03	81.46	79.05	77.32
Budget: $\ell_2 = 1.0$ (2.0 for Mini-ImageNet) Iterations: 500	PGD	71.12	60.42	57.38	55.65	81.83	77.08	74.16	72.22	75.51	67.96	65.25	63.25
	C&W	70.26	48.60	45.77	43.71	79.71	72.11	69.12	66.75	53.22	48.55	46.81	45.62
	FAB	71.63	61.23	57.59	55.24	81.62	77.25	72.41	71.09	75.54	67.04	64.56	63.10
	APGD	71.95	62.05	57.92	56.08	82.81	78.18	74.79	72.85	76.52	68.71	65.93	63.83
	SDM	72.64	68.02	65.80	63.75	84.50	82.38	79.50	77.26	80.24	77.14	74.24	71.47

Table 3: ASR (%) of various attack methods on the large-scale dataset (ImageNet-1K).

Attack	ViT-B	Swin-L	XCiT-L12	CLIP-ViT-B/32	ConvNeXt-L	RaWideResNet-101-2
PGD	90.87	67.32	79.61	97.55	67.78	75.03
C&W	87.28	66.00	76.42	98.57	64.04	72.21
FAB	90.95	67.75	78.34	98.29	67.50	75.48
APGD	91.22	68.05	79.85	98.32	68.19	75.31
SDM	92.27	70.31	80.83	99.25	69.44	76.29

5.2 Determination of Hyperparameter Combinations

SDM has four hyperparameters: total iterations Z, optimization cycles C, optimization stages N, and per-stage iterative steps T, satisfying $Z = C \times N \times T$. These parameters affect SDM performance; poorly chosen values impair algorithm effectiveness. Quantitatively analyzing their optimal allocation for maximum efficiency is theoretically intractable. We therefore perform comprehensive experiments to determine optimal hyperparameter matches (App. E), and the representative optimal combinations are summarized as follows.

Iterations:	10	20	50	100	200	500	1000
Cycles:	1	1	2	2	4	4	5
Stages:	5	5	5	5	5	5	5
Steps:	2	4	5	10	10	25	40

5.3 Attack Performance Against Different Defenses

To validate SDM's attack efficacy against various defenses, we conduct evaluations under ℓ_∞ and ℓ_2 constraints. For ℓ_∞ attacks, all baselines (PGD, C&W, FAB, APGD, SDM) use a perturbation budget of 8/255 and step size 2/255. APGD runs 10 restarts with 50 iterations each; other attackers use 500 iterations. For ℓ_2, the budget is 1.0 (2.0 for Mini-ImageNet) with step size 0.2, retaining identical iteration counts. We adopt ResNet-34 as the victim model and VAT, MART, HAT, LOAT as defense baselines, testing on CIFAR-10, CIFAR-100 and Mini-ImageNet (Tab. 2). Under ℓ_∞, SDM achieves average ASRs of 54.30%, 77.36% and 80.22% across the three datasets, exceeding PGD by 5.04%, 5.05%, 5.48% and APGD by 4.23%, 4.34%, 4.52%. Under ℓ_2, its overall average ASR hits 74.75%, outperforming PGD and APGD by 6.26% and 5.44%, respectively.

We further validate SDM on large-scale ImageNet-1K with diverse architectures: ViT-B, Swin-L, XCiT-L12 (Transformers), CLIP-ViT-B/32 (multimodal contrastive model), ConvNeXt-L and RaWideResNet-101-2 (CNNs). Official pre-trained weights ensure fairness: non-robust CLIP weights are from the Transformers library, while robust CNN/Transformer weights are sourced from RobustBench Croce et al. (2021). All attacks run for 100 iterations under $\ell_\infty = 8/255$. As shown in Tab. 3, SDM surpasses PGD, C&W, FAB and APGD by average ASR gaps of 1.71%, 3.98%, 1.68% and 1.24%. In addition, App. F presents the performance evaluation of SDM-based composite methods.

Overall results verify SDM's strong generalization across CNN and Transformer models, consistently delivering stronger adversarial performance than PGD and APGD on both small CIFAR and large ImageNet benchmarks.

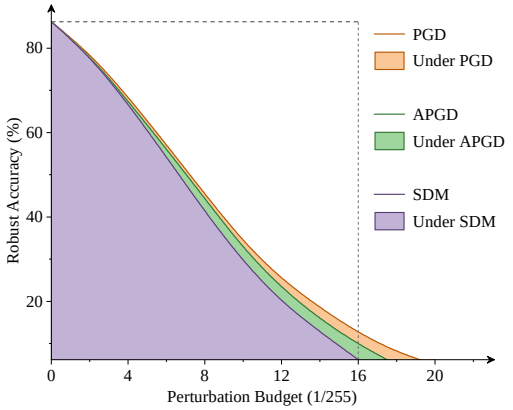

Figure 3: Robustness curves of various attack methods under the ℓ_∞-norm.

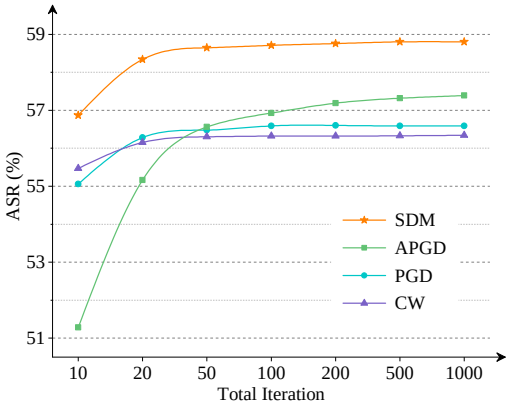

Figure 4: Cost-effectiveness comparison of various attack methods under the ℓ_∞-norm.

5.4 Attack Performance Under Various Budgets

We quantify the Area Under the Robustness Evaluation Curve (AUREC) Cinà et al. (2025) to evaluate attack performance under various perturbation budgets. AUREC integrates model robust accuracy ($1 - \text{ASR}$) across perturbation budgets; smaller AUREC means stronger attacks. We test four CIFAR-10 models: vanilla GoogLeNet (natural training), VAT-defended ResNet-34, MART-defended WideResNet-28-10, and LOAT-defended TinyViT-21M. Baselines include PGD, FAB, and APGD. Experiments use budgets $[0/255, 16/255]$, step size $2/255$, and 1000 iterations. Fig. 3 plots robustness curves for ResNet-34 as a representative case. It can be found that, SDM's curve lies nearest the origin with the minimal AUREC, confirming its superior average attack performance.

5.5 Cost-Effectiveness Analyses of Attack Methods

We evaluate the cost-effectiveness of SDM and baseline attacks under varying iterations. Experiments are conducted on CIFAR-10 with a VAT-defended ResNet-34. Under the ℓ_∞-norm, all attacks (PGD, C&W, APGD, SDM) adopt a perturbation budget of $8/255$ and a step size of $2/255$, with iterations set to 10, 20, 50, 100, 200, 500, and 1000. For APGD, iterations equal the product of restarts and per-restart steps; the restart number is 5 for iterations ≤ 100 and 10 otherwise. The corresponding results are shown in Fig. 4.

App. G analyzes the computational complexity and runtime of competing methods, revealing their per-step overheads differ by less than 10%. Thus, total iterations can serve as a reliable proxy for overall computational cost. In Fig. 4, APGD and C&W suffer severe performance degradation under limited computational resources (iterations ≤ 20). In contrast, SDM maintains superior performance across all iteration settings, achieving optimal cost-effectiveness. It is well suited for resource-constrained and real-time scenarios, such as edge computing and stream processing.

6 Conclusion

In this paper, we propose a novel gradient-based attack method termed SDM, aiming to alleviate the issue of "high-loss non-adversarial examples" prevalent in prior methods. Compared with existing approaches, SDM adopts a more challenging objective for adversarial example generation. To optimize this objective, SDM progressively approaches the global optimal solution by cyclically applying the negative probability loss and DPDR loss functions across predefined optimization stages. Extensive experiments across four public datasets and ten prevalent models demonstrate that SDM achieves superior attack performance and cost efficiency compared with previous state-of-the-art alternatives, enabling fast and efficient robustness evaluation of various target models under the canonical white-box test setting.

AI use statement

In this work, we used generative AI tools for language polishing and grammar refinement of the manuscript. We have not used generative AI tools for the core research methodology, theoretical analysis, experimental design, data collection, or result interpretation, and these tasks are not applicable to this work. Additionally, we used generative AI tools for improving the clarity and readability of the presentation. We have reviewed all AI-assisted work. Specifically, we manually verified that the AI-polished text did not alter any technical claims, mathematical formulations, or experimental results. All revised sentences were cross-checked against the original content to ensure accuracy and consistency. The final version was carefully read by all authors to confirm that the AI-assisted language improvements did not introduce errors or misrepresentations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

The code is available at https://anonymous.4open.science/r/SDM-9232.

References

Naveed Akhtar, Ajmal Mian, Navid Kardan, and Mubarak Shah. Advances in adversarial attacks and defenses in computer vision: A survey. IEEE Access, 9:155161–155196, 2021. URL https://doi.org/10.1109/ACCESS.2021.3127960.

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIII, volume 12368 of Lecture Notes in Computer Science, pp. 484–501. Springer, 2020. URL https://doi.org/10.1007/978-3-030-58592-1_29.

Bakary Badjie, José Cecílio, and António Casimiro. Adversarial attacks and countermeasures on image classification-based deep learning models in autonomous driving systems: A systematic review. ACM Comput. Surv., 57(1):20:1–20:52, 2025. URL https://doi.org/10.1145/3691625.

Hubert Baniecki and Przemyslaw Biecek. Adversarial attacks and defenses in explainable artificial intelligence: A survey. Inf. Fusion, 107:102303, 2024. URL https://doi.org/10.1016/j.inffus.2024.102303.

Brian R. Bartoldson, James Diffenderfer, Konstantinos Parasyris, and Bhavya Kailkhura. Adversarial robustness limits via scaling-law and human-alignment studies. In Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=HQtTg1try7.

Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017, pp. 39–57. IEEE Computer Society, 2017. URL https://doi.org/10.1109/SP.2017.49.

Antonio Emanuele Cinà, Jérôme Rony, Maura Pintor, Luca Demetrio, Ambra Demontis, Battista Biggio, Ismail Ben Ayed, and Fabio Roli. Attackbench: Evaluating gradient-based attacks for adversarial examples. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 39, pp. 2600–2608, 2025. URL https://doi.org/10.1609/aaai.v39i3.32263.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pp. 2206–2216. PMLR, 2020a. URL http://proceedings.mlr.press/v119/croce20b.html.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pp. 2196–2205. PMLR, 2020b. URL http://proceedings.mlr.press/v119/croce20a.html.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual, 2021. URL https://robustbench.github.io/.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 20-25 June 2009, Miami, Florida, USA, pp. 248–255, 2009. URL https://doi.org/10.1109/CVPR.2009.5206848.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6572.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009. URL https://www.cs.utoronto.ca/~kriz/cifar.html.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pp. 6391–6401, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/a41b3bb3e6b050b6c9067c67f663b915-Abstract.html.

Jiangfan Liu, Yishan Li, Yanming Guo, Yu Liu, Jun Tang, and Ying Nie. Generation and countermeasures of adversarial examples on vision: a survey. Artif. Intell. Rev., 57(8): 199, 2024. URL https://doi.org/10.1007/s10462-024-10841-z.

Xinlei Liu, Tao Hu, Peng Yi, Weitao Han, Jichao Xie, and Baolin Li. Sequential difference maximization: Generating adversarial examples via multi-stage optimization. In Proceedings of the 34th ACM International Conference on Information and Knowledge Management, CIKM 2025, Seoul, Republic of Korea, November 10-14, 2025, pp. 4976–4980. ACM, 2025. URL https://doi.org/10.1145/3746252.3760870.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Rahul Rade and Seyed-Mohsen Moosavi-Dezfooli. Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022. URL https://openreview.net/forum?id=Azh9QBQ4tR7.

Leo Schwinn, René Raab, An Nguyen, and Dario Zanca & Bjoern Eskofier. Exploring misclassifications of robust neural networks to enhance adversarial attacks. Applied Intelligence, 53:19843–19859, 2023. URL https://doi.org/10.1007/s10489-023-04532-5.

J. Michael Steele. The Cauchy-Schwarz master class: An introduction to the art of mathematical inequalities. Higher Education Press, 2022. URL https://doi.org/10.1017/CBO9780511817106.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9, 2015. URL https://doi.org/10.1109/CVPR.2015.7298594.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In Advances in Neural Information Processing Systems (NeurIPS) 2016, December 5-10, 2016, Barcelona, Spain, pp. 3630–3638, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/90e1357833654983612fb05e3ec9148c-Abstract.html.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, 2020. URL https://openreview.net/forum?id=rklOg6EFwS.

Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In Computer Vision – ECCV 2022, pp. 68–85, Cham, 2022. Springer Nature Switzerland. URL https://doi.org/10.1007/978-3-031-19803-8_5.

Xiangyu Yin and Wenjie Ruan. Boosting adversarial training via fisher-rao norm-based regularization. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024, pp. 24544–24553. IEEE, 2024. URL https://doi.org/10.1109/CVPR52733.2024.02317.

A Empirical Analyses of High-Loss Non-Adversarial Examples

High-loss non-adversarial examples are inputs that fail to fool the target model but yield a higher CE loss than successful examples. This issue cannot be identified using only PGD (APGD), as lacking a relative reference. Only by comparing PGD with other attacks and analyzing examples where PGD fails but others succeed can we reveal the existence of these examples. The motivating case in Sec. 3 originated from PGD-C&W comparisons. Now, we design comprehensive controlled experiments to verify this issue.

We use C&W, FAB, and Jitter Schwinn et al. (2023) as baseline attacks, and PGD, APGD, and SDM as candidate attacks. We train 4 models via VAT on CIFAR-10: ResNet-18, GoogLeNet Szegedy et al. (2015), WideResNet-28-10, and TinyViT-21M Wu et al. (2022). For examples (indexed by h) where candidate attacks fail but baseline attacks succeed, we compute their CE loss differences, e.g., $L_{\text{pgd-cw}} = L(\boldsymbol{x}_{\text{pgd}}[h]) - L(\boldsymbol{x}_{\text{cw}}[h])$, with results in Tab. 4.

The table is interpreted as follows. For the first group "190/0.105", "152/0.087" and "20/0.046", 190 denotes the number of examples where PGD method fails but C&W method succeeds (i.e., $\boldsymbol{x}_{\text{pgd}}[h]$), and 0.105 is the mean value of $L_{\text{pgd-cw}}$; 152 and 20 denote the numbers of such failed examples for APGD and SDM (i.e., $\boldsymbol{x}_{\text{apgd}}[h]$ and $\boldsymbol{x}_{\text{sdm}}[h]$), while 0.087 and 0.046 correspond to the average values of $L_{\text{apgd-cw}}$ and $L_{\text{sdm-cw}}$, respectively.

The average CE loss differences are all positive. Combined with their attack failure, these examples correspond to the high-loss non-adversarial examples. Compared with the high loss of PGD, SDM reduces the failure rate of such examples to approximately 10% of the original level by a smaller loss. This shows that SDM effectively mitigates this issue and improves the attack performance. We summarize as follows: (1) High-loss non-adversarial examples are a relative metric requiring comparison between PGD and other attacks. (2) They widely occur across evaluation settings, with an incidence rate of 1.5%–3%. (3) SDM reduces their occurrence by 90%–95%, substantially alleviating the issue.

B Construction of the Probability Landscape

The detailed procedures for constructing the probability landscape are presented as follows.

Table 4: Number and loss values of high-loss non-adversarial examples in the CIFAR-10 dataset.

Model	Attack	C&W	FAB	Jitter
ResNet-18	PGD	190 / 0.105	213 / 0.201	173 / 0.107
	APGD	152 / 0.087	167 / 0.187	146 / 0.091
	SDM	20 / 0.046	19 / 0.142	20 / 0.048
GoogLeNet	PGD	192 / 0.097	212 / 0.182	180 / 0.106
	APGD	157 / 0.086	163 / 0.164	151 / 0.094
	SDM	19 / 0.040	17 / 0.129	17 / 0.050
WideResNet -28-10	PGD	214 / 0.119	231 / 0.216	208 / 0.118
	APGD	162 / 0.101	184 / 0.192	159 / 0.082
	SDM	18 / 0.041	17 / 0.126	18 / 0.040
TinyViT-21M	PGD	245 / 0.132	256 / 0.241	228 / 0.127
	APGD	183 / 0.117	194 / 0.213	174 / 0.106
	SDM	22 / 0.052	25 / 0.138	23 / 0.057

1. Randomly sample a set of adversarial perturbations within the predefined perturbation budget.
2. Project all adversarial perturbations onto a two-dimensional plane via Principal Component Analysis (PCA), where each perturbation is assigned a corresponding 2D coordinate.
3. Add each adversarial perturbation to a natural example individually. Then feed the perturbed examples into the classification model for prediction, and obtain the probabilities of the ground-truth labels as well as the probability upper bounds for non-ground-truth labels.
4. Calculate the differences based on the above 2D coordinates and probability values to generate two groups of smooth probability distribution landscapes, which intuitively illustrate how the prediction probabilities vary with input perturbations.

C Analysis on Optimization Directions of the Reconstruction Objective

It is known that the probability distribution $\boldsymbol{P}$ of an example can be expressed according to Eq. 3 in the paper as

$$\boldsymbol{P} = \frac{1}{\sum_{i=1}^{K} \mathrm{e}^{\boldsymbol{S}_i}} \left(\mathrm{e}^{\boldsymbol{S}_1}, \ldots, \mathrm{e}^{\boldsymbol{S}_y}, \ldots, \mathrm{e}^{\boldsymbol{S}_\tau}, \ldots, \mathrm{e}^{\boldsymbol{S}_K}\right)^\top. \tag{13}$$

The gradient of $\boldsymbol{P}$ along the optimization direction of "reducing the ground-truth label probability" can be represented in the form of a partial derivative as

$$\nabla_{-\boldsymbol{P}_y} \boldsymbol{P} = -\frac{\partial \boldsymbol{P}}{\partial \boldsymbol{S}_y} \cdot \frac{\mathrm{d}\boldsymbol{S}_y}{\mathrm{d}\boldsymbol{P}_y}. \tag{14}$$

The first term in Eq. 14 can be expressed as

$$-\frac{\partial \boldsymbol{P}}{\partial \boldsymbol{S}_y} = \boldsymbol{P}_y(\boldsymbol{P}_1, \ldots, \boldsymbol{P}_y - 1, \ldots, \boldsymbol{P}_\tau, \ldots, \boldsymbol{P}_K)^\top. \tag{15}$$

The second term in Eq. 14 can be expressed as

$$\frac{\mathrm{d}\boldsymbol{S}_y}{\mathrm{d}\boldsymbol{P}_y} = \frac{1}{\boldsymbol{P}_y\,(1 - \boldsymbol{P}_y)}. \tag{16}$$

Since, the simplified result of Eq. 14 is derived as

$$\nabla_{-\boldsymbol{P}_y}\boldsymbol{P} = \frac{1}{1-\boldsymbol{P}_y}(\boldsymbol{P}_1,\ldots,\boldsymbol{P}_y-1,\ldots,\boldsymbol{P}_\tau,\ldots,\boldsymbol{P}_K)^\top. \tag{17}$$

Similarly, the gradient of $\boldsymbol{P}$ along the optimization direction of "increasing the non-ground-truth label probability upper bound" is expressed in the form of a partial derivative as

$$\nabla_{\boldsymbol{P}_\tau}\boldsymbol{P} = \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{S}_\tau}\cdot\frac{\mathrm{d}\boldsymbol{S}_\tau}{\mathrm{d}\boldsymbol{P}_\tau}. \tag{18}$$

The first term in Eq. 18 can be expressed as

$$\frac{\partial \boldsymbol{P}}{\partial \boldsymbol{S}_\tau} = -\boldsymbol{P}_\tau(\boldsymbol{P}_1,\ldots,\boldsymbol{P}_y,\ldots,\boldsymbol{P}_\tau-1,\ldots,\boldsymbol{P}_K)^\top. \tag{19}$$

The second term in Eq. 18 can be expressed as

$$\frac{\mathrm{d}\boldsymbol{S}_\tau}{\mathrm{d}\boldsymbol{P}_\tau} = \frac{1}{\boldsymbol{P}_\tau\left(1-\boldsymbol{P}_\tau\right)}. \tag{20}$$

Therefore, the simplified result of Eq. 18 is

$$\nabla_{\boldsymbol{P}_\tau}\boldsymbol{P} = \frac{1}{\boldsymbol{P}_\tau-1}(\boldsymbol{P}_1,\ldots,\boldsymbol{P}_y,\ldots,\boldsymbol{P}_\tau-1,\ldots,\boldsymbol{P}_K)^\top. \tag{21}$$

To verify the relationship between the two optimization directions, we calculate the inner product and cosine similarity of the gradients of the probability distribution $\boldsymbol{P}$ along them, which are given in Eq. 22 and Eq. 23, respectively.

$$\nabla_{-\boldsymbol{P}_y}\boldsymbol{P}\cdot\nabla_{\boldsymbol{P}_\tau}\boldsymbol{P} = \frac{1}{\left(1-\boldsymbol{P}_y\right)\left(1-\boldsymbol{P}_\tau\right)}\left(\boldsymbol{P}_y+\boldsymbol{P}_\tau-\Phi\right). \tag{22}$$

$$\cos\left(\nabla_{-\boldsymbol{P}_y}\boldsymbol{P},\nabla_{\boldsymbol{P}_\tau}\boldsymbol{P}\right) = \frac{\boldsymbol{P}_y+\boldsymbol{P}_\tau-\Phi}{\sqrt{\left(\Phi-2\boldsymbol{P}_y+1\right)\left(\Phi-2\boldsymbol{P}_\tau+1\right)}}. \tag{23}$$

In Eq. 22 and Eq. 23, Φ denotes the sum of the squares of the probabilities corresponding to all labels, which is explicitly expressed as follows.

$$\Phi = \sum_{k=1}^{K}{\boldsymbol{P}_k}^2. \tag{24}$$

It can be proven that the aforementioned gradient inner product is always greater than 0 and the cosine similarity is always less than 1, as shown below. Therefore, the two optimization directions in Generation Objective 6 in the paper are neither mutually independent nor completely consistent; instead, they exhibit partial correlation.

C.1 Proof of the Gradient Inner Product Being Greater Than Zero

The proof that the gradient inner product of the output probability $\boldsymbol{P}$ along the two optimization directions is greater than zero (i.e., $\nabla_{-\boldsymbol{P}_y}\boldsymbol{P}\cdot\nabla_{\boldsymbol{P}_\tau}\boldsymbol{P}>0$) is presented as follows.

It is currently known that $\{\boldsymbol{P}_k\in(0,1)|k=1,\ldots,K\}$, $\sum_{k=1}^{K}\boldsymbol{P}_k=1$, $\Phi=\sum_{k=1}^{K}{\boldsymbol{P}_k}^2$, and $\grave{\boldsymbol{P}}$ is the descending arrangement of $\boldsymbol{P}$, satisfying $\grave{\boldsymbol{P}}_1\geq\grave{\boldsymbol{P}}_2\geq\cdots\geq\grave{\boldsymbol{P}}_K>0$.

Given that $0<\boldsymbol{P}_y,\boldsymbol{P}_\tau<1$, the proof for $\nabla_{-\boldsymbol{P}_y}\boldsymbol{P}\cdot\nabla_{\boldsymbol{P}_\tau}\boldsymbol{P}>0$ can be converted into the proof for $\boldsymbol{P}_y+\boldsymbol{P}_\tau-\Phi>0$. Here, $\boldsymbol{P}_y$ and $\boldsymbol{P}_\tau$ denote the ground-truth label probability and the non-ground-truth label probability upper bound, respectively. Exactly one of $\boldsymbol{P}_y$

and $\boldsymbol{P}_\tau$ is equal to $\grave{\boldsymbol{P}}_1$, thus $\boldsymbol{P}_y + \boldsymbol{P}_\tau > \grave{\boldsymbol{P}}_1$. Therefore, a sufficient condition for satisfying $\boldsymbol{P}_y + \boldsymbol{P}_\tau - \Phi > 0$ is given by

$$\grave{\boldsymbol{P}}_1 - \sum_{k=1}^{K} \grave{\boldsymbol{P}}_k^2 \geq 0. \tag{25}$$

Ineq. 25 can be transformed into

$$\grave{\boldsymbol{P}}_1 - \grave{\boldsymbol{P}}_1^2 \geq \sum_{k=2}^{K} \grave{\boldsymbol{P}}_k^2. \tag{26}$$

According to the Cauchy-Schwarz inequality Steele (2022), the maximum value of the right-hand side of Ineq. 26 is $\grave{\boldsymbol{P}}_1(1 - \grave{\boldsymbol{P}}_1)$ when $\grave{\boldsymbol{P}}_1 \in (0, 0.5)$, and $(1 - \grave{\boldsymbol{P}}_1)^2$ when $\grave{\boldsymbol{P}}_1 \in [0.5, 1)$. Hence, the proof for Ineq. 26 can be further reduced to the proof for Ineq. Set 27.

$$\begin{cases} \grave{\boldsymbol{P}}_1 - \grave{\boldsymbol{P}}_1^2 \geq \grave{\boldsymbol{P}}_1(1 - \grave{\boldsymbol{P}}_1), & \grave{\boldsymbol{P}}_1 \in (0, 0.5) \\ \grave{\boldsymbol{P}}_1 - \grave{\boldsymbol{P}}_1^2 \geq (1 - \grave{\boldsymbol{P}}_1)^2 & , \grave{\boldsymbol{P}}_1 \in [0.5, 1) \end{cases}. \tag{27}$$

When $\grave{\boldsymbol{P}}_1 \in (0, 0.5)$, the first term of Ineq. Set 27 obviously holds. Next, we discuss the case where $\grave{\boldsymbol{P}}_1 \in [0.5, 1)$.

A function $h_1(\grave{\boldsymbol{P}}_1)$ is constructed as follows.

$$h_1(\grave{\boldsymbol{P}}_1) = \grave{\boldsymbol{P}}_1 - \grave{\boldsymbol{P}}_1^2 - (1 - \grave{\boldsymbol{P}}_1)^2. \tag{28}$$

The function $h_1(\grave{\boldsymbol{P}}_1)$ can be transformed into

$$h_1(\grave{\boldsymbol{P}}_1) = -2(\grave{\boldsymbol{P}}_1 - \frac{3}{4})^2 + \frac{1}{8}. \tag{29}$$

When $\grave{\boldsymbol{P}}_1 \in [0.5, 1)$, the function $h_1(\grave{\boldsymbol{P}}_1)$ is always greater than or equal to 0. Thus, it can be proven that Ineq. Set 27 holds for all values within the domain of $\grave{\boldsymbol{P}}_1$, thereby confirming the validity of Ineq. 25. Since Ineq. 25 is a sufficient condition for the original proposition $\nabla_{-\boldsymbol{P}_y}\boldsymbol{P} \cdot \nabla_{\boldsymbol{P}_\tau}\boldsymbol{P} > 0$, the original proposition is true.

C.2 Proof of the Cosine Similarity Being Less Than One

The proof that the cosine similarity of the output probability $\boldsymbol{P}$ along the two optimization directions is less than one (i.e., $\cos\left(\nabla_{-\boldsymbol{P}_y}\boldsymbol{P}, \nabla_{\boldsymbol{P}_\tau}\boldsymbol{P}\right) < 1$) is provided as follows.

A function $h_2(\boldsymbol{P})$ is constructed and expressed as

$$h_2(\boldsymbol{P}) = (\boldsymbol{P}_y + \boldsymbol{P}_\tau - \Phi)^2 - (\Phi - 2\boldsymbol{P}_y + 1)(\Phi - 2\boldsymbol{P}_\tau + 1). \tag{30}$$

The original proposition is then transformed into: proving that $h_2(\boldsymbol{P}) < 0$ holds consistently within the domain of $\boldsymbol{P}$. Given that $\Phi = \sum_{k=1}^{K} \boldsymbol{P}_k{}^2$, the function $h_2(\boldsymbol{P})$ is decomposed into

$$h_2(\boldsymbol{P}) = -\left(\sum_{\substack{k=1 \\ k\neq y, k\neq \tau}}^{K} \boldsymbol{P}_k\right)^2 - 2\sum_{\substack{k=1 \\ k\neq y, k\neq \tau}}^{K} \boldsymbol{P}_k{}^2. \tag{31}$$

Evidently, the value of $h_2(\boldsymbol{P})$ is consistently less than 0 within the domain, thus verifying that the original proposition $\cos\left(\nabla_{-\boldsymbol{P}_y}\boldsymbol{P}, \nabla_{\boldsymbol{P}_\tau}\boldsymbol{P}\right) < 1$ is true.

D Projection Method under L_2-Norm

Under the $\boldsymbol{\ell}_2$-norm constraint, the gradient matrix is first normalized with $\boldsymbol{\ell}_2$ normalization (i.e., Euclidean normalization). This operation scales the gradient matrix to a matrix with a unit norm, namely, making its $\boldsymbol{\ell}_2$-norm equal to 1 while preserving the original direction of the gradient, which is formulated as

$$Grad = \frac{Grad}{\|Grad\|_2 + \zeta}, \tag{32}$$

where ζ (typically set to 10^{-10}) is added to prevent division by zero. Subsequently, the adversarial example is updated along the direction of the unit gradient with a step size of α, and the current adversarial perturbation is generated as

$$\boldsymbol{\delta}_{t-1} = \boldsymbol{x}'_{t-1} + \alpha \cdot \text{sign}(Grad) - \boldsymbol{x}. \tag{33}$$

Then, the $\boldsymbol{\ell}_2$-norm of the current adversarial perturbation is calculated. If it exceeds the predefined adversarial perturbation budget, the perturbation is scaled down to the range strictly within the perturbation budget; if it does not exceed the budget, no additional processing is performed.

$$\boldsymbol{\delta}_t = \boldsymbol{\delta}_{t-1} \cdot \min\left(\frac{\epsilon}{\|\boldsymbol{\delta}_{t-1}\|_2}, 1\right). \tag{34}$$

Finally, the projected adversarial perturbation is added to the original natural example to generate the adversarial example under the $\boldsymbol{\ell}_2$-norm constraint.

$$\boldsymbol{x}'_t = \boldsymbol{x} + \boldsymbol{\delta}_t. \tag{35}$$

E Hyperparameter Selection Experiments

For the hyperparameters used in SDM, the optimization cycle is the top-level parameter. In general, its value should be smaller than the number of optimization stages and the number of iterative steps. Thus, for a total number of iterations of 10, two sets of hyperparameter combinations are configured in this section: ① $(1, 2, 5)$ and ② $(1, 5, 2)$ for C, N, and T respectively. Subsequently, attack evaluations are performed on the defense methods VAT and MART under the CIFAR-10 and CIFAR-100 datasets, with ResNet-34 and WideResNet-28-10 employed as the target models. The evaluation results are presented in Tab. 5, where the columns for VAT and MART represent the ASR of the attack method against the corresponding defense.

The results show that for a total number of iterations of 10, the average ASR of hyperparameter combination ② is 4.54% higher than that of combination ①. Under the limited conditions, this demonstrates that combination ② $(1, 5, 2)$ is the optimal hyperparameter set for total iterations of 10.

Next, based on the optimal combination $(1, 5, 2)$ for 10 total iterations, three hyperparameter combinations are designed for total iterations of 20: ① $(2, 5, 2)$, ② $(1, 10, 2)$, and ③ $(1, 5, 4)$ for C, N, and T respectively. The experimental setup remains consistent with the above experiments, and the results are shown in Tab. 6.

It can be observed that for total iterations of 20, the average ASR of hyperparameter combination ③ is 0.19% higher than that of combination ① and 0.05% higher than that of combination ②. This indicates that combination ③ $(1, 5, 4)$ is the optimal set for total iterations of 20.

Subsequently, the optimal combination $(1, 5, 4)$ for 20 total iterations is extended to derive hyperparameter sets for total iterations of 50 from three perspectives: increasing the iterative

Table 5: ASR (%) of SDM with various hyperparameter combinations under 10 total iterations.

Combination	CIFAR-10				CIFAR-100			
	ResNet-34		WideResNet-28-10		ResNet-34		WideResNet-28-10	
	VAT	MART	VAT	MART	VAT	MART	VAT	MART
①	55.10	48.22	56.38	51.39	76.41	72.58	77.08	73.34
②	56.87	52.78	58.37	56.62	79.17	76.78	79.89	77.25

Table 6: ASR (%) of SDM with various hyperparameter combinations under 20 total iterations.

Combination	CIFAR-10				CIFAR-100			
	ResNet-34		WideResNet-28-10		ResNet-34		WideResNet-28-10	
	VAT	MART	VAT	MART	VAT	MART	VAT	MART
①	58.10	53.94	59.75	57.97	80.35	77.73	81.13	78.16
②	58.15	54.15	59.85	58.25	80.36	77.98	81.20	78.28
③	58.34	54.21	59.94	58.29	80.40	77.91	81.24	78.32

Table 7: ASR (%) of SDM with various hyperparameter combinations under 50 total iterations.

Combination	CIFAR-10				CIFAR-100			
	ResNet-34		WideResNet-28-10		ResNet-34		WideResNet-28-10	
	VAT	MART	VAT	MART	VAT	MART	VAT	MART
①	58.63	54.35	60.19	58.43	80.55	77.90	81.39	78.35
②	58.57	54.33	60.19	58.42	80.53	77.91	81.38	78.34
③	58.62	54.35	60.19	58.42	80.55	77.91	81.40	78.35
④	58.65	54.36	60.19	58.44	80.57	77.93	81.41	78.37

steps per stage to yield ① $(1, 5, 10)$, increasing the number of optimization stages to yield ② $(1, 10, 5)$, and increasing the number of optimization cycles to yield ③ $(2, 6, 4)$ and ④ $(2, 5, 5)$. The setup is consistent with the previous one, and the results are shown in Tab. 7.

In Tab. 7, the average ASR of hyperparameter combination ④ $(2, 5, 5)$ is 0.02% higher than that of combination ①, 0.03% higher than that of combination ②, and 0.02% higher than that of combination ③. Under the constraint of 50 total iterations, combination ④ exhibits the strongest attack performance and is therefore identified as the optimal hyperparameter set for this iteration count.

In the subsequent experiments, the optimal sets for the total iterations of 100, 200, 500, and 1000 are identified in turn, and the detailed process is not repeated herein. All optimal hyperparameter combinations are collated uniformly in Subsec. 5.2 of the paper.

F Attack Performance of Composite Methods

The integrated attack method AA is composed of three individual attack methods, namely APGD, FAB, and Square Andriushchenko et al. (2020). Despite its strong attack performance, it incurs extremely high computational costs in practice. To further enhance the performance of AA, we replace APGD with SDM to form SDM-AA (the original AA is referred to as APGD-AA). Subsequently, we adopt ResNet-34 and VAT as the base model and defense method, respectively, and evaluate these attack methods under the constraints of $\ell_\infty = 8/255$ and $\ell_2 = 1.0$ (2.0 specifically for Mini-ImageNet). The evaluation results are shown in Tab. 8. It can be observed that the average ASR of SDM is only 0.18% lower than that of APGD-AA, while its computational cost is merely 15%–20% of the latter (practical experience). Moreover, after upgrading with SDM, SDM-AA achieves an average ASR that is 0.44% higher than APGD-AA, providing a new effective avenue for the accurate evaluation of model robustness under sufficient computational resources.

Table 8: ASR (%) of the SDM-based composite methods.

Attack	CIFAR-10		CIFAR-100		Mini-ImageNet	
	VAT	MART	VAT	MART	VAT	MART
APGD	57.28	49.37	78.20	73.20	79.15	77.57
SDM	58.79	54.43	80.63	77.89	83.03	81.46
APGD-AA	58.70	55.10	80.41	78.22	83.20	81.69
SDM-AA	59.02	55.35	80.78	78.69	83.76	82.35

Table 9: Computational complexity of PGD and SDM.

Item	PGD	SDM
Forward	$O(B \cdot C \cdot H \cdot W \cdot L \cdot D)$	$O(B \cdot C \cdot H \cdot W \cdot L \cdot D)$
Backward	$O(B \cdot C \cdot H \cdot W \cdot L \cdot D)$	$O(B \cdot C \cdot H \cdot W \cdot L \cdot D)$
CE Loss	$O(B \cdot K)$	\
DPDR Loss	\	$O(B \cdot K) + O(B \cdot K \cdot \log K)$

Table 10: Practical runtime (ms) of various attack methods.

Attack	CIFAR-10		ImageNet-1K	
	BS = 32	BS = 128	BS = 32	BS = 128
PGD	7.30±0.21	12.24±0.13	92.21±5.23	376.79± 9.86
APGD	8.36±0.31	11.69±0.16	80.10±6.72	356.02±12.34
SDM	8.15±0.26	13.33±0.14	95.02±5.18	385.50±11.75

G Analysis of Computational Complexity

This section supplements Subsec. 5.5 in the paper and conducts a detailed computational complexity analysis. The classic PGD method mainly consists of data forward propagation, gradient backward propagation, and CE loss computation. SDM adopts identical forward and backward propagation as PGD; the primary distinction is that SDM uses the DPDR loss and incorporates probability ranking strategies.

In each iterative computation, let the batch size, input dimension (C, H, W) and class number of the dataset be B, (C, H, W) and K, respectively. The network depth and average channel number are denoted as L and D. The computational complexity of PGD and SDM in each operation module is illustrated in Tab. 9.

It is observed that the DPDR loss introduces an extra ranking-related term $O(B \cdot K \log K)$, which is nearly insignificant when compared to the major computational cost $O(B \cdot C \cdot H \cdot W \cdot L \cdot D)$, as the dominant cost is 5 to 6 orders of magnitude larger. Hence, SDM possesses theoretically comparable computational complexity to PGD. To validate this finding, we perform experiments with 100 iterations under the ℓ_∞ constraint for PGD, APGD and SDM. We carry out evaluations on ResNet-34 trained on CIFAR-10 (32×32) and ViT-B trained on ImageNet-1K, with batch sizes (BS) set to 32 and 128 separately. All experiments are run on an NVIDIA L40S GPU and repeated five times. We record the step-wise runtime in milliseconds, and the statistical results are presented in Tab. 10.

On CIFAR-10, the average runtime of SDM is 9.92% and 7.13% higher than that of PGD and APGD, respectively. On ImageNet-1K, the increased ratios are 2.46% and 8.28%. Although equipped with a more elaborate optimization paradigm, SDM only brings slight additional overheads within the range of 5% to 9%. In addition, we think that such extra consumption is not derived from core logic of the SDM, but mainly results from frequent tensor initialization and index operations in DPDR. In future research, we will focus on minimizing the number of memory I/O operations to further accelerate the algorithm.