Superclass Adversarial Attack

Soichiro Kumano [1] **Hiroshi Kera** [2] **Toshihiko Yamasaki** [1]

Abstract

Adversarial attacks have only focused on changing the predictions of the classifier, but their danger greatly depends on how the class is mistaken. For example, when an autonomous driving system mistakes a Persian cat for a Siamese cat, it is hardly a problem. However, if it mistakes a cat for a 120km/h minimum speed sign, serious problems can arise. As a stepping stone to more threatening adversarial attacks, we consider the superclass adversarial attack, which causes misclassification of not only fine classes, but also superclasses. We conducted the first comprehensive analysis of superclass adversarial attacks (an existing and 19 new methods) in terms of accuracy, speed, and stability, and identified several strategies to achieve better performance. Although this study is aimed at superclass misclassification, the findings can be applied to other problem settings involving multiple classes, such as top-k and multi-label classification attacks.

1. Introduction

Adversarial attacks aim to fool a classifier to misclassify (Szegedy et al., 2014; Goodfellow et al., 2015; Sabour et al., 2016; Moosavi-Dezfooli et al., 2016; Papernot et al., 2016; Kurakin et al., 2017b; Carlini & Wagner, 2017; Moosavi-Dezfooli et al., 2017; Chen et al., 2018; Madry et al., 2018; Dong et al., 2018; Athalye et al., 2018a; Su et al., 2019; Croce & Hein, 2020a;b; Andriushchenko et al., 2020). These attacks appear not only in the electronic world, but also in the physical world, and create a serious risk for systems with deep neural networks (DNNs) (Sharif et al., 2016; Kurakin et al., 2017a; Eykholt et al., 2018; Athalye et al., 2018b; Sharif et al., 2019; Thys et al., 2019; Adv, 2022). However, prior studies have not evaluated which classes should be targeted for more harmful attacks, and thus not all the classification errors by the attacks are risky for DNN-based systems. For example, if an attack on a DNN-based autonomous driving system causes that system to mistake a Persian cat for a Siamese cat, the error is trivial. However, if the cat is instead mistaken for a 120km/h minimum speed sign, serious accidents may occur. Therefore, attacks that cause erroneous classification may have widely varying risk levels.

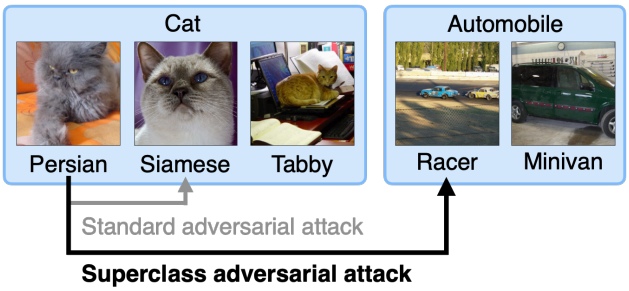

Figure 1. Superclass adversarial attacks result in misclassification of not only fine classes, but also superclasses. Adversarial examples generated by a superclass attack can be misclassified into semantically distant classes, which may cause serious problems.

In this study, to explore adversarial attacks on a more threatening level, we consider the superclass adversarial attack (SAA). Superclass adversarial examples (SAEs) generated by SAAs are misclassified to different superclasses. When superclasses are defined by semantic similarities, SAEs are more semantically distant from the original class than standard adversarial examples. For example, given a sample of Persian cat, its SAE will be misclassified as a sedan or minivan, not as a nearby class such as a Siamese or tabby cat (Figure 1)[1]. Thus, SAEs cause more serious misclassification for DNNs than standard adversarial attacks. We conducted a first comprehensive analysis of SAAs, including suited attack frameworks, efficient loss functions, and the inner mechanisms of SAAs, which have not been extensively researched in prior studies.

We first show that standard adversarial attacks cause misclassification between similar classes, typically within the same superclasses (*e.g.*, from a Persian cat to a Siamese cat), and thus are often not effective threats in real-world systems. Next, we reveal that the existing iterative sequence-based SAA (Ma et al., 2021) is not practical in terms of time efficiency, and we propose an alternative SAA. The iterative sequence-based SAA repeats a targeted attack to each

[1]Note that this study is valid for any superclass configuration. There are other settings besides visual similarities, such as dangerousness.

fine class until success, which is highly time-intensive. In contrast, our proposed iterative sort-based SAA leverages logit-based sort and exhibits faster performance without loss of accuracy. As a practical compromise, we also incorporate early-stopping into the iterative sort-based SAA. Instead of repeating targeted attacks, we propose an efficient non-targeted attack that aims to increase the loss for classes within the same superclass. We investigate 15 different types of such losses, including the simple sum of cross-entropy (CE) loss. Because the success rate of gradient-based adversarial attacks (*e.g.*, projected gradient descent (PGD) (Madry et al., 2018)) strongly depends on the loss function (Carlini & Wagner, 2017; Croce & Hein, 2020a), it is important to verify multiple loss functions. The proposed 15 loss functions are categorized into max-, sum-, and LogSumExp (LSE)-based methods. The max-based method decreases the maximum probability in the original superclass at each step, which gradually creates a decrease in the superclass's overall probability. As methods that can be run in a single step (*e.g.*, fast gradient sign method (FGSM) (Goodfellow et al., 2015)), we propose sum- and LSE-based methods, which are constructed by the sum and LSE of fine class logits, probability, or CE, respectively. These methods calculate the gradients of multiple classes, and decrease multiple probabilities in an original superclass in a single step. In the results of comprehensive experiments for these non-iterative methods, we have identified that certain variants of LSE- and sum-based methods yield stronger attacks in single- and multi-step, respectively. We also found that loss based on CE in non-iterative SAAs does not work well due to conflicts between multiple CE and an accelerated gradient vanishing problem. These findings are not limited to SAAs, and are applicable to general attacks involving multiple classes, including top-k and multi-label classification attacks. Our contributions are summarized as follows.

- We conducted a comprehensive analysis of SAAs, which have received little focus in literature.
- We propose an iterative sort-based SAA, which is significantly faster than the existing method, while being comparably strong. In addition, we propose the implementation of early-stopping in the sort-based method as a realistic compromise.
- We propose non-iterative SAAs with 15 loss functions based on maximum, sum, and LSE strategies. Comprehensive evaluations show that a variant of the LSE-based method in a single-step, and that of the sum-based method in multi-steps, yield the highest success rates and efficiency in non-iterative methods.
- We discuss three strategies and several limitations with cross-entropy loss in the context of non-iterative SAAs. These results not only facilitate the future development of SAAs, but can also be applied across various problem settings involving multiple classes.

2. Related Work

FGSM and PGD generate adversarial examples by causing perturbations, which are calculated by gradients of natural images to increase the loss (*e.g.*, CE loss). Such gradient-based attacks are strongly affected by the design of the loss function (Carlini & Wagner, 2017; Croce & Hein, 2020a). Using this study as a baseline, we propose and compare a variety of loss functions for SAAs.

The standard non-targeted adversarial attack does not distinguish the class to be misclassified. This can lead to attacks that pose little threat to real systems. In contrast, an SAA causes the misclassification of not only the fine class, but also the superclass. To SAEs, Ma *et al.* performed targeted attacks sequentially on the fine classes in different superclasses (Ma et al., 2018). This method is very slow due to its structure consisting of repeated targeted attacks. We performed a comprehensive analysis of SAAs, including the existing method, its modifications, and novel methods, in terms of accuracy and time consumption.

Top-k and multi-label classification attacks are also considered as adversarial attacks involving multiple classes (Song et al., 2018; Zhou et al., 2020; Tursynbek et al., 2022; Zhang & Wu, 2020; Hu et al., 2021; Song et al., 2018; Zhou et al., 2020; Ghamizi et al., 2021; Melacci et al., 2021; Yang et al., 2021b;a; Zhou et al., 2021). In Appendix A1, these two attacks are discussed in detail.

3. Methods

This section describes existing and our proposed SAA methods, as summarized in Table 1.

3.1. Notations and Definitions

We denote the set of images by $\mathcal{X}$, and the set of fine classes by $\mathcal{Y} := \{1, \dots, K\}$. A map $S : \mathcal{Y} \to \mathcal{P}(\mathcal{Y})$ accepts a fine class as input, and returns its super class, where $\mathcal{P}(\mathcal{Y})$ is the power set of $\mathcal{Y}$. Extending our notation, we denote the set of superclasses by $\mathcal{S}(\mathcal{Y}) := \{\mathcal{S}(i) \mid i \in \mathcal{Y}\}$. Note that we assume that any two different superclasses are disjoint (*i.e.*, for any $S_1, S_2 \in \mathcal{S}(\mathcal{Y})$ with $S_1 \neq S_2$, it holds that $S_1 \cap S_2 = \emptyset$). The logit and probability of input $\mathbf{x}$ with respect to the fine class $i \in \mathcal{Y}$ are denoted by $l_i(\mathbf{x})$ and $p_i(\mathbf{x})$, respectively. When the input is not of our interest, we write $l_i := l_i(\mathbf{x})$ and $p_i := p_i(\mathbf{x})$ for simplicity. The prediction of classifier $f : \mathcal{X} \to \mathcal{Y}$ for $\mathbf{x} \in \mathcal{X}$ is $f(\mathbf{x}) := \operatorname{argmax}_{i \in \mathcal{Y}} l_i(\mathbf{x})$. For simplicity, we define $\mathcal{S}^-(y) := \mathcal{Y} \backslash \mathcal{S}(y)$ and $\pi(A) := \operatorname{argmax}_{i \in \mathcal{A}} l_i$ for

Table 1. Comparison table of standard non-targeted attacks and SAAs. Our proposed SAAs are indicated in bold. Loss $\ell(\mathbf{x}, y)$ is used in Equation (1). Iterative SAAs repeat targeted attacks until success, whereas non-iterative SAAs perform a single non-targeted attack.

ATTACK TYPE			LOSS TYPE	LOSS $\ell(\mathbf{x}, y)$
STANDARD NON-TARGETED ATTACK			CE	$-\ln p_y$
			CW (CARLINI & WAGNER, 2017)	$-l_y + l_{\pi(\mathcal{Y}\setminus\{y\})}$
			PROB-CW	$-p_y + p_{\pi(\mathcal{Y}\setminus\{y\})}$
			WEIGHTED-CW	$-p_y l_y + p_{\pi(\mathcal{Y}\setminus\{y\})} l_{\pi(\mathcal{Y}\setminus\{y\})}$
SAA	ITERATIVE	SEQUENTIAL	CE (MA ET AL., 2021)	$\ln p_i \ \left(i \in \mathcal{S}^-(y)\right)$
		SORTED	**CE**	$\ln p_i \ \left(i \in \left[\mathcal{S}^-(y)\right]^k\right)$
			CW	$l_i - l_{\pi(\mathcal{Y}\setminus\{i\})} \ \left(i \in \left[\mathcal{S}^-(y)\right]^k\right)$
			PROB-CW	$p_i - p_{\pi(\mathcal{Y}\setminus\{i\})} \ \left(i \in \left[\mathcal{S}^-(y)\right]^k\right)$
			WEIGHTED-CW	$p_i l_i - p_{\pi(\mathcal{Y}\setminus\{i\})} l_{\pi(\mathcal{Y}\setminus\{i\})} \ \left(i \in \left[\mathcal{S}^-(y)\right]^k\right)$
	NON-ITERATIVE	**MAX**	**CE**	$-\ln p_{\pi(\mathcal{S}(y))}$
			CW	$-l_{\pi(\mathcal{S}(y))} + l_{\pi\left(\mathcal{S}^-(y)\right)}$
			PROB-CW	$-p_{\pi(\mathcal{S}(y))} + p_{\pi\left(\mathcal{S}^-(y)\right)}$
			WEIGHTED-CW	$-p_{\pi(\mathcal{S}(y))} l_{\pi(\mathcal{S}(y))} + p_{\pi\left(\mathcal{S}^-(y)\right)} l_{\pi\left(\mathcal{S}^-(y)\right)}$
		SUM	**CE**	$-\sum_{i\in\mathcal{S}(y)} \ln p_i$
			LOGIT-CE	$-\ln \frac{\exp\left(\sum_{i\in\mathcal{S}(y)} l_i\right)}{\sum_{I\in\mathcal{S}(\mathcal{Y})} \exp\left(\sum_{i\in I} l_i\right)}$
			PROB-CE	$-\ln \sum_{i\in\mathcal{S}(y)} p_i$
			CW	$-\sum_{i\in\mathcal{S}(y)} l_i + l_{\pi\left(\mathcal{S}^-(y)\right)}$
			PROB-CW	$-\sum_{i\in\mathcal{S}(y)} p_i + p_{\pi\left(\mathcal{S}^-(y)\right)}$
			WEIGHTED-CW	$-\sum_{i\in\mathcal{S}(y)} p_i l_i + p_{\pi\left(\mathcal{S}^-(y)\right)} l_{\pi\left(\mathcal{S}^-(y)\right)}$
		LSE	**CE**	$\mathrm{LSE}_{i\in\mathcal{S}(y)}\left(-\ln p_i\right)$
			PROB-CE	$-\ln \mathrm{LSE}_{i\in\mathcal{S}(y)} p_i$
			CW	$-\mathrm{LSE}_{i\in\mathcal{S}(y)} l_i + l_{\pi\left(\mathcal{S}^-(y)\right)}$
			PROB-CW	$-\mathrm{LSE}_{i\in\mathcal{S}(y)} p_i + p_{\pi\left(\mathcal{S}^-(y)\right)}$
			WEIGHED-CW	$-\mathrm{LSE}_{i\in\mathcal{S}(y)} p_i l_i + p_{\pi\left(\mathcal{S}^-(y)\right)} l_{\pi\left(\mathcal{S}^-(y)\right)}$

$A \subset \mathcal{Y}$. In addition, we denote by $[A]^k$ the subset of $A \subset \mathcal{Y}$ corresponding to the k largest logits.

While adversarial examples are misclassified between different fine classes, superclass adversarial examples (SAEs) are misclassified between the fine classes of different superclasses. Formally, an SAE is defined as follows:

Definition 3.1 (superclass adversarial example)**.** A sample $\mathbf{x}^* \in \mathcal{X}$ is called a superclass adversarial example of an image $\mathbf{x} \in \mathcal{X}$ (with label $y \in \mathcal{Y}$) if it satisfies $f(\mathbf{x}^*) \notin \mathcal{S}(y)$ and $d(\mathbf{x}, \mathbf{x}^*) \leq \epsilon$, where $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ and $\epsilon > 0$ denote a distance function and constraint, respectively.

When $\mathcal{S}(y) = \{y\}$, SAEs are reduced to standard adversarial examples. We also refer to the methods of generating SAEs as superclass adversarial attacks (SAAs).

3.2. Iterative SAA

An iterative SAA runs the targeted attack on every fine class $i \in \mathcal{S}^-(y)$ that is not included in the original superclass $\mathcal{S}(y)$ until success. Ma *et al.* were the first to propose iterative SAAs (Ma et al., 2021). Because they did not describe a selection method for the order of targeted fine classes to attack, we assume that they used the ascending order of the fine class index. We refer to this method as the iterative sequence-based method (worst-case targeted attack in their study). In this paper, we propose a new iterative SAA called the iterative sort-based method, which selects target fine classes in the descending order of the logit $l_i(\mathbf{x})$ $(i \in \mathcal{S}^-(y))$. Because empirically, the larger the original logit is, the more likely the targeted attack will succeed, the sort-based method performs faster than the sequence-based method in almost all cases. Even in the worst case, the sort-based method attacks all targeted classes, and thus does not lose the attack success rate compared to the sequence-based method.

Although the sort-based method exhibits faster performance than the sequence-based method, it is still slower than a standard attack due to the nature of repeated targeted attacks. Therefore, we propose to incorporate early-stopping into

the sort-based method as a practical compromise. Instead of attacking all possible targeted fine classes $i \in \mathcal{S}^-(y)$, the modified method would attack only the $k \in \mathbb{N}$ fine classes $i \in [\mathcal{S}^-(y)]^k$, whose logit $l_i(\mathbf{x})$ ($i \in \mathcal{S}^-(y)$) is the largest. Although larger values of k improve accuracy, they also increase the time cost. In this study, we use CE, CW, prob-CW (see below), and weighted-CW (see below) losses as loss functions in PGD.

3.3. Non-Iterative SAA

The non-iterative method is an extension of the standard non-targeted attack framework for the SAA. This method is considerably faster than the iterative methods, as it does not repeat attacks. The foundation of the non-iterative method is to reduce the probability of all fine classes in the original superclass $\mathcal{S}(y)$. In this paper, we introduce three different strategies for the non-iterative method: max, sum, and LSE.

We briefly introduce a common framework for gradient-based standard attacks and SAAs. Gradient-based attacks generate the adversarial example $\mathbf{x}^*$ by updating the image $\mathbf{x} \in \mathcal{X}$ in the fine class $y \in \mathcal{Y}$ to increase the loss $\ell(\mathbf{x}, y)$ as follows:

$$\mathbf{x}^* := \Pi_{d(\mathbf{x},\mathbf{x}^*)\leq\epsilon}\{\mathbf{x} + \alpha \cdot \boldsymbol{\nabla}_{\mathbf{x}}\ell(\mathbf{x}, y)\}, \tag{1}$$

where $\Pi_{d(\mathbf{x},\mathbf{x}^*)\leq\epsilon} : \mathcal{X} \to \mathcal{X}$ is a projection operator, $\alpha > 0$ is the step size, and $\epsilon > 0$ is the distance constraint. Equation (1) indicates that the loss function's design determines the nature of the attack. It is established that attack success rate varies significantly depending on the loss function (Carlini & Wagner, 2017; Croce & Hein, 2020a). In particular, the CE and CW functions (Carlini & Wagner, 2017) are common loss functions in adversarial attacks. These functions are expressed as follows:

$$\ell_{\text{CE}}(\mathbf{x}, y) := -\ln p_y. \tag{2}$$
$$\ell_{\text{CW}}(\mathbf{x}, y) := -l_y + l_{\pi(\mathcal{Y}\setminus\{y\})}. \tag{3}$$

In addition, we propose the prob-CW and weighted-CW as new loss function baselines, defined as follows:

$$\ell_{\text{PCW}}(\mathbf{x}, y) := -p_y + p_{\pi(\mathcal{Y}\setminus\{y\})}, \tag{4}$$
$$\ell_{\text{WCW}}(\mathbf{x}, y) := -p_y l_y + p_{\pi(\mathcal{Y}\setminus\{y\})} l_{\pi(\mathcal{Y}\setminus\{y\})}. \tag{5}$$

We extend CE, CW, prob-CW, and weighted-CW to the loss functions in max-, sum-, and LSE-based methods for SAAs (Table 1). Henceforth, the combination of the max-based method and CE loss is referred to as max (CE). Other combinations are similarly named (*e.g.*, sum (CW) and LSE (prob-CW)). Although all of these loss functions appear to have similar roles, we show in Section 4 that they produce different results in terms of robust accuracy.

Max-Based Method. In each step, the max-based method takes $\pi(\mathcal{S}(y))$, the class with the highest logit among all fine classes in the original superclass $\mathcal{S}(y)$, and decreases its probability. This is considered relatively easy to optimize, as one does not need to consider the gradients of multiple classes simultaneously.

Sum-Based Method. The sum-based method sums the values (CE, logits, probability, or products of logit and probability) of all fine classes $i \in \mathcal{S}(y)$ in the original superclass $\mathcal{S}(y)$. Note that the second terms of extended CW, prob-CW, and weighted-CW loss (17, 18, and 19-th rows in Table 1), which are incentives to increase the probability of a fine class $i \in \mathcal{S}^-(y)$ in a different superclass, should not use multiple class values. This is because there is no advantage in SAA to increasing the probability of multiple fine classes. Furthermore, the sum- and LSE-based methods offer several ways to calculate extended CE loss. Sum (logit-CE) calculates the probability in CE from the logit of the superclass and the softmax function. The logit of superclass is defined as $\sum_{i\in I} l_i$ for superclass $I \in \mathcal{S}(\mathcal{Y})$ (Table 1). Sum (prob-CE) defines the probability of a superclass as $\sum_{i\in I} p_i$ for superclass $I \in \mathcal{S}(\mathcal{Y})$ (Table 1).

LSE-Based Method. The LSE-based method uses the LogSumExp (LSE) function. Note that the loss function of LSE (logit-CE) can be rearranged to that of sum (prob-CE); thus, these two are equivalent. This function differs from the sum function in that the larger elements are given a higher weighting. For example, for the first term of LSE (CW), the gradients of logits are weighted as $\sum_{i\in\mathcal{S}(y)} \frac{\exp(l_i)}{\sum_{i\in\mathcal{S}(y)}\exp(l_i)} \boldsymbol{\nabla}_{\mathbf{x}} l_i$; in contrast, they are not weighted in sum (CW) as $\sum_{i\in\mathcal{S}(y)} \boldsymbol{\nabla}_{\mathbf{x}} l_i$. This ensures strong attacks on classes with large probabilities and gives little consideration to classes with small probabilities. Note that other sum-based methods (prob-CE, prob-CW, and weighted-CW) weigh the gradients differently.

4. Experimental Results

In this paper, superclass accuracy is defined as the ratio of data for which the predicted superclass matches the ground truth. We used CIFAR-100 (Krizhevsky, 2009) and 5,000 images extracted from ImageNet (ILSVRC2012) (Deng et al., 2009). The superclasses were constructed by visual similarity between the fine classes. Superclass configurations followed (Krizhevsky, 2009; Ima). The average number of elements in each superclass was 5.00 in CIFAR-100 and 1.79 in ImageNet. As classifiers, two WideResNet (Zagoruyko & Komodakis, 2016) were used, standard trained $\mathcal{M}_{\text{std}}$, and adversarial trained $\mathcal{M}_{\text{adv}}$ for each dataset. Unless stated otherwise, the PGD settings for evaluation were 100 steps, 8/255 epsilons, 2/255 step size, l_∞ norm.

Table 2. Overview table of SAAs. Values represent superclass accuracy (%) and runtime (minutes).

ATTACK TYPE	METHOD	CIFAR-100		IMAGENET	
		$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$	$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$
N/A	N/A	87.5% (0.00MIN)	76.5% (0.00MIN)	81.7% (0.00MIN)	64.2% (0.00MIN)
STANDARD	NON-TARGETED	26.5% (1.00MIN)	43.1% (2.00MIN)	12.2% (1.00MIN)	26.1% (2.00MIN)
SAA	ITERATIVE	0.00% (0.00MIN)	26.1% (4.00MIN)	0.00% (0.00MIN)	21.5% (5.00MIN)
	NON	0.00% (0.00MIN)	26.3% (1.00MIN)	0.00% (0.00MIN)	21.7% (2.00MIN)

All SAEs terminated upon successful attack. Additional configuration details are described in Appendix A2.

4.1. Overview of SAAs

Superclass accuracy and runtime are summarized in Table 2. The superclass accuracy was extracted from the best results of each method. The hyper-parameter k of the iterative SAA is the minimum among those whose robust accuracy is greater than the best accuracy of the non-iterative SAA. Detailed results are in Tables A1 to A3

Table 2 shows that standard non-targeted attacks cannot cause superclass misclassification when the superclasses are constructed by visual similarities. Compared to CIFAR-100, the difference in superclass accuracy between standard attacks and SAAs is slightly smaller in ImageNet. This is due to the average number of elements in each superclass of ImageNet being smaller than that of CIFAR-100, and even standard attacks tended to cause superclass misclassification. Since the update of adversarial examples was stopped upon successful superclass misclassification, some standard attacks were slower than SAAs.

In contrast to standard non-targeted attacks, SAAs can induce superclass misclassification. Although the iterative method with an appropriate hyper-parameter k can achieve a higher success rate than the non-iterative method, it remains more time-intensive. Therefore, the iterative method is recommended for strong attacks where time is not a concern. Note, however, that we do not recommend that the evaluation for adversarial robustness of classifiers with a large k be the standard in the field (cf. Section 4.2). Because the non-iterative method is fast and achieves high rates of success, it can be employed for real-time attacks and adversarial training that necessitate high speed.

4.2. Comparison of Iterative Method

Table 3 shows the results of iterative SAAs with CE loss. Note that the values in Table 3 are different from those in Table 2, where the best loss function was selected among several. More detailed results for iterative SAAs are listed in Table A2.

For the standard trained models $\mathcal{M}_{\text{std}}$, there are no significant differences in robust accuracy between all methods. This is because the standard trained models are very vulnerable to attacks, which succeed regardless of the first targeted class. Instead, major differences between the methods are seen in the adversarial robust models $\mathcal{M}_{\text{adv}}$. Because these models are robust to attacks, the runtime and accuracy vary depending on how the targeted class is selected. First, we compare the sequence- and sort-based method without early-stopping ($k = |\mathcal{S}^{-}(y)|$). Our proposed sort-based method yields a significant reduction in time without any loss in accuracy. Next, we consider the role of the hyper-parameter k. The sort-based method with early stopping successfully reduces the time significantly while compromising accuracy. Larger values of the hyper-parameter k ensure higher success rates, but necessitate more time. However, there are cases where $k = 5$ can significantly reduce the time with almost equal accuracy to that for $k = |\mathcal{S}^{-}(y)|$.

As a future policy in this field, we do not recommend the iterative method with a large k for evaluation. Large k is very time-consuming and requires high-performance computational resources for multiple evaluations, which would limit the number of researchers and hinder future development (Schwartz et al., 2020). Instead, we recommend the iterative method with a small k, or the non-iterative method.

4.3. Comparison of Non-Iterative Method

Tables 4 and 5 show the superclass accuracy extracted from max (CW), sum (CE), sum (logit-CE), sum (weighted-cw), and LSE (CW) in 1 and 100 steps of PGD, respectively. Because the time taken by each non-iterative method does not vary significantly, it is omitted in Tables 4 and 5. More detailed results of the non-iterative SAAs are listed in Table A3.

Tables 4 and 5 indicate that sum (CE) is inefficient as an SAA. An adversarial attack by CE loss for a class $i \in \mathcal{S}^{-}(y)$ decreases the class probability p_i while increasing the p_j of other classes $j \in \mathcal{Y}\backslash\{i\}$. The roles of increasing and decreasing the probability of each CE loss in sum (CE) are in mutual conflict. Thus, sum (CE) is not a suitable loss function for SAAs.

Tables 4 and 5 also shows that sum (logit-CE) is not preferable for SAAs due to the gradient vanishing problem. When

Table 3. Comparison table of iterative SAAs with CE loss. Values represent accuracy (%) and time (minutes). k is a hyper-parameter for early-stopping. $k = \left|\mathcal{S}^{-}(y)\right|$ indicates that early-stopping is not used, and all target labels can be attacked (cf. Section 3.2).

METHOD	k	CIFAR-100		IMAGENET	
		$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$	$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$
SORTED	1	0.00% (0.00MIN)	28.2% (1.00MIN)	0.00% (0.00MIN)	27.5% (2.00MIN)
	3	0.00% (0.00MIN)	26.1% (4.00MIN)	0.00% (0.00MIN)	22.5% (6.00MIN)
	5	0.00% (0.00MIN)	25.7% (7.00MIN)	0.00% (0.00MIN)	21.9% (9.00MIN)
	$\left\|\mathcal{S}^{-}(y)\right\|$	0.00% (0.00MIN)	25.4% (138MIN)	0.00% (0.00MIN)	20.6% (153×10^{1}MIN)
SEQ. (MA ET AL., 2021)	$\left\|\mathcal{S}^{-}(y)\right\|$	0.00% (0.00MIN)	25.4% (170MIN)	0.00% (0.00MIN)	20.6% (203×10^{1}MIN)

the probability of the targeted class is close enough to 1, the CE loss and its gradients approach 0, *i.e.*, which causes gradient vanishing. Sum (logit-CE) calculates the probability by summing all logits within the original superclass and applying the softmax function. When the superclass is composed of visual similarities, the logits l_i $(i \in \mathcal{S}(y))$ tend to be positive, and their sum $\sum_{i\in\mathcal{S}(y)} l_i$ tends to be greater than the original logit l_y. In this case, the probability in the sum (logit-CE) $\frac{\exp\left(\sum_{i\in\mathcal{S}(y)} l_i\right)}{\sum_{I\in\mathcal{S}(\mathcal{Y})}\exp\left(\sum_{i\in I} l_i\right)}$ approaches 1 faster than the original probability p_y, *i.e.*, which accelerate gradient vanishing. Therefore, sum (logit-CE) is inefficient.

In a single-step, the LSE (CW) is the best for SAAs. Max (CW) does not work well with single-step because it reduces the only probability of the fine class $\pi(\mathcal{S}(y))$ and cannot reduce the superclass's overall probability. In contrast, LSE- and sum-based methods incorporate the values of all fine classes into the loss function, and can decrease multiple probabilities, *i.e.*, the superclass's overall probability, by single-step. However, because sum (weighted-CW) is somewhat less accurate than max (CW), the incorporation of multiple fine class values is important.

For 100 steps, sum (weighted-CW) method is the best method. Intuitively, sum (weighted-CW) and LSE (CW) may be expected to have similar robust accuracy in terms of summing multiple values of fine classes. However, the differences shown in Tables 4 and 5 indicate that SAA necessitates a careful selection of the loss function in accordance with the number of steps. One reason for the poor accuracy of max (CW) in multi-steps is oscillation. When the maximum probability of one fine class $\pi(\mathcal{S}(y))$ decreases, the probability of another fine class $j \in \mathcal{S}(y)\backslash\{\pi(\mathcal{S}(y))\}$ may increase. The sum- and LSE-based methods always calculate the gradients of all fine classes at each step, and can therefore achieve a higher success rate.

5. Conclusion

In this study, we considered superclass adversarial attacks (SAAs) and superclass adversarial examples (SAEs) as stepping stones to understanding higher-risk adversarial attacks. Our comprehensive analysis of SAAs (an existing and 19 new methods) revealed iterative methods are strong but time-consuming, while non-iterative methods are fast but slightly weaker. The experimental results showed that among non-iterative methods, a variant of LSE- and sum-based methods achieves the highest attack success rate and the best efficiency in single- and multi-step attacks. In addition, we suggested that CE should not be incorporated into loss functions of non-iterative methods because of the conflicts between multiple CEs and the accelerated gradient vanishing problem. Our findings are not only important for the future development of SAAs, but can also be generalized to adversarial attacks with multiple targeted classes. In a future study, we further analyze what in a loss function of non-iterative SAA strongly affects a successful attack. Furthermore, we consider more threatening SAAs with a superclass configuration constructed by danger level rather than visual similarity.

Table 4. Comparison table of non-iterative SAAs in a single step. The best attack success rates are indicated in bold. Values represent superclass accuracy (%).

METHOD	CIFAR-100		IMAGENET	
(1 STEP)	$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$	$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$
MAX (CW)	17.3%	51.8%	7.29%	**47.1%**
SUM (CE)	75.9%	73.2%	53.3%	53.1%
SUM (LOGIT-CE)	43.2%	62.4%	32.1%	58.7%
SUM (W-CW)	24.6%	55.4%	14.8%	48.1%
LSE (CW)	**16.3%**	**51.2%**	**6.97%**	**47.1%**

Table 5. Comparison table of non-iterative SAAs in 100 steps. The description is the same as in Table 4.

METHOD	CIFAR-100		IMAGENET	
(100 STEPS.)	$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$	$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$
MAX (CW)	**0.00%**	28.2%	**0.00%**	24.5%
SUM (CE)	77.2%	72.1%	50.3%	37.2%
SUM (LOGIT-CE)	38.7%	49.6%	19.8%	47.8%
SUM (W-CW)	**0.00%**	**26.3%**	**0.00%**	**21.7%**
LSE (CW)	**0.00%**	27.8%	**0.00%**	24.6%

References

Imagenetsampling. https://github.com/innerlee/ImagenetSampling.

Adversraial fashion. https://adversarialfashion.com/, 2022.

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

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, pp. 274–283, 2018a.

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. Synthesizing robust adversarial examples. In *ICML*, pp. 284–293, 2018b.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *SSP*, pp. 39–57, 2017.

Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C.-J. EAD: elastic-net attacks to deep neural networks via adversarial examples. In *AAAI*, 2018.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, pp. 2206–2216, 2020a.

Croce, F. and Hein, M. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, pp. 2196–2205, 2020b.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255, 2009.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. Boosting adversarial attacks with momentum. In *CVPR*, pp. 9185–9193, 2018.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. Robust physical-world attacks on deep learning visual classification. In *CVPR*, pp. 1625–1634, 2018.

Ghamizi, S., Cordy, M., Papadakis, M., and Le Traon, Y. Evasion attack steganography: Turning vulnerability of machine learning to adversarial attacks into a real-world application. In *ICCV*, pp. 31–40, 2021.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Hu, S., Ke, L., Wang, X., and Lyu, S. Tkml-ap: Adversarial attacks to top-k multi-label learning. In *ICCV*, pp. 7649–7657, 2021.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

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

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. In *ICLR*, 2017b.

Ma, A., Virmaux, A., Scaman, K., and Lu, J. Improving hierarchical adversarial robustness of deep neural networks. *arXiv:2102.09012*, 2021.

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S., Schoenebeck, G., Song, D., Houle, M. E., and Bailey, J. Characterizing adversarial subspaces using local intrinsic dimensionality. In *ICLR*, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Melacci, S., Ciravegna, G., Sotgiu, A., Demontis, A., Biggio, B., Gori, M., and Roli, F. Domain knowledge alleviates adversarial attacks in multi-label classifiers. *PAMI*, 2021.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, pp. 2574–2582, 2016.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In *CVPR*, pp. 1765–1773, 2017.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *SSP*, pp. 372–387, 2016.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pp. 8026–8037, 2019.

Sabour, S., Cao, Y., Faghri, F., and Fleet, D. J. Adversarial manipulation of deep representations. In *ICLR*, 2016.

Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., and Madry, A. Do adversarially robust imagenet models transfer better? *NeurIPS*, 33:3533–3545, 2020.

Schapire, R. E. and Singer, Y. Boostexter: A boosting-based system for text categorization. *ML*, 39(2):135–168, 2000.

Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. Green ai. *CACM*, 63(12):54–63, 2020.

Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *ACM CCS*, pp. 1528–1540, 2016.

Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. A general framework for adversarial examples with objectives. *ACM TOPS*, 22(3):1–30, 2019.

Song, Q., Jin, H., Huang, X., and Hu, X. Multi-label adversarial perturbations. In *ICDM*, pp. 1242–1247. IEEE, 2018.

Su, J., Vargas, D. V., and Sakurai, K. One pixel attack for fooling deep neural networks. *TEVC*, 23(5):828–841, 2019.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *ICLR*, 2014.

Thys, S., Van Ranst, W., and Goedeme, T. Fooling automated surveillance cameras: Adversarial patches to attack person detection. In *CVPR WS*, 2019.

Tursynbek, N., Petiushko, A., and Oseledets, I. Geometry-inspired top-k adversarial perturbations. In *WACV*, pp. 3398–3407, 2022.

Yang, Z., Han, Y., and Zhang, X. Attack transferability characterization for adversarially robust multi-label classification. In *ECML PKDD*, pp. 397–413. Springer, 2021a.

Yang, Z., Han, Y., and Zhang, X. Characterizing the evasion attackability of multi-label classifiers. In *AAAI*, 2021b.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *BMVC*, pp. 1–12, 2016.

Zhang, J., Xu, X., Han, B., Niu, G., Cui, L., Sugiyama, M., and Kankanhalli, M. Attacks which do not kill training make adversarial learning stronger. In *ICML*, pp. 11278–11287, 2020.

Zhang, Z. and Wu, T. Learning ordered top-k adversarial attacks via adversarial distillation. In *CVPR WS*, pp. 776–777, 2020.

Zhou, N., Luo, W., Lin, X., Xu, P., and Zhang, Z. Generating multi-label adversarial examples by linear programming. In *IJCNN*, pp. 1–8. IEEE, 2020.

Zhou, N., Luo, W., Zhang, J., Kong, L., and Zhang, H. Hiding all labels for multi-label images: An empirical study of adversarial examples. In *IJCNN*, pp. 1–8. IEEE, 2021.

A1. Additional Related Work

Section 2 describes studies that relate to standard adversarial attacks and SAAs. In this section, top-k and multi-label classification attacks are discussed as adversarial attacks that involve multiple classes.

Multi-label classification involves assigning multiple classes to a single instance (*e.g.*, an image) (Song et al., 2018; Zhou et al., 2020; Ghamizi et al., 2021; Melacci et al., 2021; Yang et al., 2021b;a; Zhou et al., 2021). Some studies have proposed adversarial attacks in multi-label classification (Song et al., 2018; Zhou et al., 2020) as an extension of the C&W attack or DeepFool (Moosavi-Dezfooli et al., 2016). Song *et al.* leveraged the relationships between labels for multi-label classification attacks (Song et al., 2018) motivated by multi-label ranking loss (Schapire & Singer, 2000). Some studies have demonstrated empirical research of adversarial examples that hide all labels (Zhou et al., 2021), theoretical attackability (Yang et al., 2021b;a), adversarial defense (Melacci et al., 2021), and relationships with steganography (Ghamizi et al., 2021) in the multi-label classification attacks. This study considers single-label classification wherein one class is assigned to each instance. In addition, information concerning the relationships between the classes, which may be useful for attacks, is not used.

Some studies have considered adversarial attacks focusing on the top-k prediction of the classifier (Tursynbek et al., 2022; Zhang & Wu, 2020; Hu et al., 2021). Tursynbek *et al.* considered an attack in which the true label of an image was excluded from the top-k prediction of the classifier (Tursynbek et al., 2022). Zhang *et al.* proposed an attack that replaces the top-k of the classifier with the class specified by an adversary (Zhang & Wu, 2020). Hu *et al.* considered an adversarial attack on top-k multi-label learning, where an instance is associated with a non-empty subset of labels, and the output of the system is the top-k predicted labels (Hu et al., 2021). This study does not focus on top-k predictions, with the goal being to prevent top-1 predictions from being included in the original superclass.

Although some multi-classification and top-k attacks can be applied to SAA with some modifications (Song et al., 2018; Zhou et al., 2020; Tursynbek et al., 2022; Zhang & Wu, 2020; Hu et al., 2021), they were not considered in this study for the following reasons. One of the goals, here, is to compare several loss functions under identical conditions to determine the configuration best suited for SAAs. Therefore, these studies deviated from our experimental settings. Furthermore, one of the nine proposed methods, the sum (CW) method (cf. Section 3.3), is essentially based on the same idea (*i.e.*, idea based on C&W attack) as (Song et al., 2018; Zhang & Wu, 2020; Hu et al., 2021), and can therefore, be used as a substitute. (Song et al., 2018; Zhou et al., 2020; Tursynbek et al., 2022) proposed geometric methods, which may be explored in future work.

A2. Experimental Settings

This section describes the experimental settings in detail.

Datasets. The superclass settings for CIFAR-100 and ImageNet followed (Krizhevsky, 2009) and (Ima), respectively[2]. The number of superclasses given in CIFAR-100 was 20, and each superclass included five (fine) classes. The number of superclasses provided in ImageNet was 558, the average number of elements 1.79, and the standard deviation of the elements 1.57.

Models. For CIFAR-100, we trained WideResNet-40-10 (Zagoruyko & Komodakis, 2016). For ImageNet, we used pretrained WideResNet-50-2 (normally trained model from (Paszke et al., 2019) and adversarially trained one from (Salman et al., 2020)).

CIFAR-100 Training. The number of epochs was 200 and the batch size was 128. The optimizer was a stochastic gradient descent with a learning rate of 0.1, momentum of 0.9, and weight decay of 2e-4, and the learning rate was multiplied by 0.1 at epochs 100 and 150, respectively. In adversarial training, a PGD with 10 steps was used: l_∞ norm, 8/255 constraint of distance ϵ, and 2/255 step size α. As in (Zhang et al., 2020), updates of adversarial examples were stopped when an attack was successful to reduce the training time. Note that adversarial training was conducted using standard adversarial examples for fine class misclassification.

Validation. An NVIDIA A100-PCIE-40GB was used for all evaluations. The batch sizes were 2,048 for CIFAR-100 and 512 for ImageNet. In ImageNet, 5,000 images were randomly extracted. The based PGD settings are 100 steps: l_∞ norm,

[2]The configuration details of ImageNet is provided at `https://github.com/s-kumano/imagenet-superclass`.

Table A1. Comparison table of the standard adversarial attacks. The best attack success rates are indicated in bold. Values represent accuracy (%) and time (minutes).

LOSS	CIFAR-100		IMAGENET	
	$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$	$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$
CE	30.1% (1.00MIN)	**43.1% (2.00MIN)**	14.4% (1.00MIN)	**26.1% (2.00MIN)**
CW	34.7% (2.00MIN)	47.6% (2.00MIN)	19.1% (1.00MIN)	32.8% (2.00MIN)
PROB-CW	34.3% (1.00MIN)	46.2% (2.00MIN)	20.7% (1.00MIN)	29.8% (2.00MIN)
WEIGHTED-CW	**26.5% (1.00MIN)**	44.8% (2.00MIN)	**12.2% (1.00MIN)**	29.6% (2.00MIN)

8/255 distance constraint ϵ, 2/255 step size α, with random initialization by a uniform distribution $U(-\epsilon, \epsilon)$. In the iterative method, random initialization was performed for each targeted attack. The updates of adversarial examples were stopped when the attack was successful, that is, when superclass misclassification was achieved.

A3. Experimental Results

Table A1 shows the robust accuracy under the standard adversarial attacks with four loss functions. Robust accuracy varies across loss functions, but attacks with them cause fewer superclass misclassifications than SAAs. These results indicate that standard adversarial attacks cause misclassification between similar fine classes and are not at high risk for DNN-based systems.

Table A2 shows the results of iterative SAAs. The larger the k, the higher is the attack success rate; however, the operation will take a longer time. Note that the results on ImageNet are only a sample of 5,000, and iterative SAAs with large k for all images in ImageNet will take a considerable amount of time. Owing to different random initialization, the superclass accuracy under sort (weighted-CW) with $k = 5$ was larger than that with $k = 4$.

Table A3 shows the results of non-iterative SAAs with one, 10, and 100 steps. In a single step, LSE (CW) achieves a high attack success rate. This is because the gradients of all fine classes in the same superclass are calculated in a single step, and are weighted more efficiently than the other sum- and LSE-based methods. In 100 steps, the sum-based methods were stronger than the other two methods. In particular, the sum (weighted-CW) in 100 steps achieved a high attack success rate in a short time. The sum (CE) and LSE (CE) in 10 steps were weaker than those in a single step. This is attributed to the difference in step size.

Table A2. Comparison table of the iterative SAAs. The best attack success rates are indicated in bold. Values represent accuracy (%) and time (minutes).

k	METHOD	LOSS	CIFAR-100		IMAGENET	
			$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$	$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$
1	SORT	CE	**0.00% (0.00MIN)**	**28.2% (1.00MIN)**	**0.00% (0.00MIN)**	27.5% (2.00MIN)
		CW	**0.00% (0.00MIN)**	28.4% (1.00MIN)	**0.00% (0.00MIN)**	25.1% (2.00MIN)
		PROB-CW	5.13% (0.00MIN)	32.6% (2.00MIN)	3.26% (0.00MIN)	**23.1% (2.00MIN)**
		W-CW	0.06% (0.00MIN)	30.6% (2.00MIN)	**0.00% (0.00MIN)**	23.2% (2.00MIN)
2	SORT	CE	**0.00% (0.00MIN)**	**26.5% (3.00MIN)**	**0.00% (0.00MIN)**	24.0% (4.00MIN)
		CW	**0.00% (0.00MIN)**	26.8% (3.00MIN)	**0.00% (0.00MIN)**	22.8% (4.00MIN)
		PROB-CW	4.10% (0.00MIN)	32.1% (3.00MIN)	2.97% (1.00MIN)	**21.9% (3.00MIN)**
		W-CW	0.06% (0.00MIN)	30.2% (3.00MIN)	**0.00% (0.00MIN)**	22.1% (4.00MIN)
3	SORT	CE	**0.00% (0.00MIN)**	**26.1% (4.00MIN)**	**0.00% (0.00MIN)**	22.5% (6.00MIN)
		CW	**0.00% (0.00MIN)**	26.3% (4.00MIN)	**0.00% (0.00MIN)**	21.8% (5.00MIN)
		PROB-CW	3.68% (1.00MIN)	32.0% (5.00MIN)	2.77% (2.00MIN)	**21.5% (5.00MIN)**
		W-CW	0.06% (0.00MIN)	30.0% (5.00MIN)	**0.00% (0.00MIN)**	21.7% (5.00MIN)
4	SORT	CE	**0.00% (0.00MIN)**	**25.8% (6.00MIN)**	**0.00% (0.00MIN)**	22.2% (7.00MIN)
		CW	**0.00% (0.00MIN)**	26.1% (6.00MIN)	**0.00% (0.00MIN)**	21.4% (7.00MIN)
		PROB-CW	3.39% (1.00MIN)	32.0% (7.00MIN)	2.74% (2.00MIN)	**21.3% (7.00MIN)**
		W-CW	0.03% (1.00MIN)	29.9% (7.00MIN)	**0.00% (0.00MIN)**	21.4% (7.00MIN)
5	SORT	CE	**0.00% (0.00MIN)**	**25.7% (7.00MIN)**	**0.00% (0.00MIN)**	21.9% (9.00MIN)
		CW	**0.00% (0.00MIN)**	25.9% (7.00MIN)	**0.00% (0.00MIN)**	**21.2% (8.00MIN)**
		PROB-CW	3.31% (1.00MIN)	31.9% (9.00MIN)	2.67% (3.00MIN)	**21.2% (8.00MIN)**
		W-CW	0.05% (1.00MIN)	29.8% (8.00MIN)	**0.00% (0.00MIN)**	21.4% (8.00MIN)
$\lvert\mathcal{S}(y)\rvert$	SEQ.	CE	**0.00% (0.00MIN)**	25.4% (171MIN)	**0.00% (0.00MIN)**	20.6% (203×10^1MIN)
	SORT	CE	**0.00% (0.00MIN)**	**25.4% (138MIN)**	**0.00% (0.00MIN)**	**20.6% (153×10^1MIN)**

Table A3. Comparison table of the iterative SAAs. Values represent accuracy (%) and time (minutes). The best attack success rates in each step are indicated in bolded and they in each step and broadly categorized method (max, sum, and LSE) are in underscore.

STEPS	METHOD	LOSS	CIFAR-100		IMAGENET	
			$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$	$\mathcal{M}_{\text{STD}}$	$\mathcal{M}_{\text{ADV}}$
1	MAX	CE	32.1% (0.00MIN)	57.0% (0.00MIN)	21.9% (0.00MIN)	49.9% (0.00MIN)
		CW	17.3% (0.00MIN)	51.8% (0.00MIN)	7.29% (0.00MIN)	**47.1% (0.00MIN)**
		PROB-CW	30.3% (0.00MIN)	55.2% (0.00MIN)	20.2% (0.00MIN)	48.0% (0.00MIN)
		WEIGHTED-CW	27.8% (0.00MIN)	57.3% (0.00MIN)	17.7% (0.00MIN)	48.3% (0.00MIN)
	SUM	CE	75.9% (0.00MIN)	73.2% (0.00MIN)	53.3% (0.00MIN)	53.1% (0.00MIN)
		LOGIT-CE	43.2% (0.00MIN)	62.4% (0.00MIN)	32.1% (0.00MIN)	58.7% (0.00MIN)
		PROB-CE	21.6% (0.00MIN)	52.6% (0.00MIN)	12.5% (0.00MIN)	49.0% (0.00MIN)
		CW	19.5% (0.00MIN)	57.4% (0.00MIN)	12.8% (0.00MIN)	50.6% (0.00MIN)
		PROB-CW	20.5% (0.00MIN)	52.0% (0.00MIN)	11.8% (0.00MIN)	47.8% (0.00MIN)
		WEIGHTED-CW	24.6% (0.00MIN)	55.4% (0.00MIN)	14.8% (0.00MIN)	48.1% (0.00MIN)
	LSE	CE	77.8% (0.00MIN)	75.2% (0.00MIN)	55.9% (0.00MIN)	56.0% (0.00MIN)
		PROB-CE	32.3% (0.00MIN)	55.0% (0.00MIN)	20.9% (0.00MIN)	49.1% (0.00MIN)
		CW	**16.3% (0.00MIN)**	**51.2% (0.00MIN)**	**6.97% (0.00MIN)**	**47.1% (0.00MIN)**
		PROB-CW	27.6% (0.00MIN)	52.9% (0.00MIN)	17.6% (0.00MIN)	47.7% (0.00MIN)
		WEIGHTED-CW	27.5% (0.00MIN)	57.0% (0.00MIN)	17.6% (0.00MIN)	48.1% (0.00MIN)
10	MAX	CE	7.72% (0.00MIN)	40.7% (0.00MIN)	3.53% (0.00MIN)	31.9% (0.00MIN)
		CW	**0.10% (0.00MIN)**	33.5% (0.00MIN)	**0.00% (0.00MIN)**	29.9% (0.00MIN)
		PROB-CW	6.58% (0.00MIN)	38.2% (0.00MIN)	3.28% (0.00MIN)	28.7% (0.00MIN)
		WEIGHTED-CW	1.15% (0.00MIN)	38.3% (0.00MIN)	**0.00% (0.00MIN)**	28.8% (0.00MIN)
	SUM	CE	77.2% (0.00MIN)	72.1% (0.00MIN)	50.5% (0.00MIN)	41.5% (0.00MIN)
		LOGIT-CE	38.7% (0.00MIN)	52.3% (0.00MIN)	20.0% (0.00MIN)	51.4% (0.00MIN)
		PROB-CE	6.84% (0.00MIN)	33.1% (0.00MIN)	4.52% (0.00MIN)	31.0% (0.00MIN)
		CW	0.20% (0.00MIN)	40.9% (0.00MIN)	**0.00% (0.00MIN)**	35.0% (0.00MIN)
		PROB-CW	5.90% (0.00MIN)	**32.7% (0.00MIN)**	3.95% (0.00MIN)	**28.4% (0.00MIN)**
		WEIGHTED-CW	4.00% (0.00MIN)	34.6% (0.00MIN)	**0.00% (0.00MIN)**	28.8% (0.00MIN)
	LSE	CE	79.4% (0.00MIN)	74.5% (0.00MIN)	53.2% (0.00MIN)	47.2% (0.00MIN)
		PROB-CE	11.8% (0.00MIN)	36.8% (0.00MIN)	4.72% (0.00MIN)	31.0% (0.00MIN)
		CW	**0.10% (0.00MIN)**	33.1% (0.00MIN)	**0.00% (0.00MIN)**	30.2% (0.00MIN)
		PROB-CW	7.15% (0.00MIN)	34.6% (0.00MIN)	3.70% (0.00MIN)	29.1% (0.00MIN)
		WEIGHTED-CW	0.83% (0.00MIN)	37.3% (0.00MIN)	**0.00% (0.00MIN)**	28.7% (0.00MIN)
100	MAX	CE	5.70% (0.00MIN)	34.6% (2.00MIN)	3.51% (0.00MIN)	24.8% (2.00MIN)
		CW	**0.00% (0.00MIN)**	28.2% (1.00MIN)	**0.00% (0.00MIN)**	24.5% (2.00MIN)
		PROB-CW	4.98% (0.00MIN)	32.3% (1.00MIN)	3.26% (0.00MIN)	22.2% (2.00MIN)
		WEIGHTED-CW	0.03% (0.00MIN)	30.0% (1.00MIN)	**0.00% (0.00MIN)**	21.9% (2.00MIN)
	SUM	CE	77.2% (4.00MIN)	72.1% (3.00MIN)	50.3% (3.00MIN)	37.2% (3.00MIN)
		LOGIT-CE	38.7% (2.00MIN)	49.6% (2.00MIN)	19.8% (1.00MIN)	47.8% (3.00MIN)
		PROB-CE	6.61% (0.00MIN)	27.1% (1.00MIN)	4.52% (0.00MIN)	23.7% (2.00MIN)
		CW	**0.00% (0.00MIN)**	35.4% (2.00MIN)	**0.00% (0.00MIN)**	29.0% (2.00MIN)
		PROB-CW	5.75% (0.00MIN)	27.1% (1.00MIN)	3.95% (0.00MIN)	21.8% (2.00MIN)
		WEIGHTED-CW	**0.00% (0.00MIN)**	**26.3% (1.00MIN)**	**0.00% (0.00MIN)**	**21.7% (2.00MIN)**
	LSE	CE	79.4% (4.00MIN)	74.5% (4.00MIN)	53.0% (3.00MIN)	43.2% (3.00MIN)
		PROB-CE	10.6% (0.00MIN)	30.2% (1.00MIN)	4.67% (0.00MIN)	23.7% (2.00MIN)
		CW	**0.00% (0.00MIN)**	27.8% (1.00MIN)	**0.00% (0.00MIN)**	24.6% (2.00MIN)
		PROB-CW	6.36% (0.00MIN)	28.5% (1.00MIN)	3.68% (0.00MIN)	22.9% (2.00MIN)
		WEIGHTED-CW	0.01% (0.00MIN)	28.7% (1.00MIN)	**0.00% (0.00MIN)**	21.8% (2.00MIN)