Any Target Can be Offense: Adversarial Example Generation via Generalized Latent Infection

Youheng Sun[1⋆], Shengming Yuan[1⋆], Xuanhan Wang[2⋆⋆], Lianli Gao[1], and Jingkuan Song[1]

[1] Center for Future Media, University of Electronic Science and Technology of China
[2] Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China
youheng.sun@std.uestc.edu.cn, shengming.yuan@outlook.com, wxuanhan@hotmail.com, lianli.gao@uestc.edu.cn, jingkuan.song@gmail.com

Abstract. Targeted adversarial attack, which aims to mislead a model to recognize any image as a target object by imperceptible perturbations, has become a mainstream tool for vulnerability assessment of deep neural networks (DNNs). Since existing targeted attackers only learn to attack known target classes, they cannot generalize well to unknown classes. To tackle this issue, we propose **G**eneralized **A**dversarial attac**KER** (**GAKer**), which is able to construct adversarial examples to any target class. The core idea behind GAKer is to craft a latently infected representation during adversarial example generation. To this end, the extracted latent representations of the target object are first injected into intermediate features of an input image in an adversarial generator. Then, the generator is optimized to ensure visual consistency with the input image while being close to the target object in the feature space. Since the GAKer is class-agnostic yet model-agnostic, it can be regarded as a general tool that not only reveals the vulnerability of more DNNs but also identifies deficiencies of DNNs in a wider range of classes. Extensive experiments have demonstrated the effectiveness of our proposed method in generating adversarial examples for both known and unknown classes. Notably, compared with other generative methods, our method achieves an approximately **14.13%** higher attack success rate for unknown classes and an approximately **4.23%** higher success rate for known classes. Our code is available in `https://github.com/VL-Group/GAKer`.

Keywords: Targeted Adversarial Attack · Generator-based Attack · Black-box attack · Unknown Classes

1 Introduction

Deep neural networks (DNNs) have significantly advanced the field of artificial intelligence, achieving remarkable success in various domains, including image

⋆ Equal contribution.
⋆⋆ Corresponding author

recognition [11], natural language processing [34,36], and AIGC [12,29]. Despite great success, DNNs have been shown to be significantly vulnerable to adversarial attacks [7,8,23,31], which misleads DNNs to fail by using adversarial examples, i.e., adding human-imperceptible perturbations into clean images. Thus, it is of great importance to understand the mechanism behind DNNs and design effective assessment methods [7, 8, 23, 31, 45] to identify deficiencies of DNNs before deploying them in security-sensitive applications.

In terms of the way of attacking, adversarial attacks generally can be divided into two categories. The first one is the untargeted attack [3, 4, 7, 15, 43, 45], where the goal of attackers is to fail DNNs. The second one is the targeted attack [5, 21, 44, 46], where attackers not only fail DNNs but also mislead them to recognize an image as the pre-specific target. Since the targeted attack with high flexibility poses a severe threat to security-sensitive applications, it has become a mainstream tool for vulnerability assessment of DNNs. Therefore, in this work, we focus on the study of targeted attacks.

Among the target attack methods, two technical branches exist for adversarial example generation. The first one is the iteration-based framework, which produces an adversarial example of each clean image in an iterative manner. This framework has shown to be susceptible to overfitting white-box models, and iteration-based strategy leads to a heavy computational overhead [5, 10, 44, 46]. In a different line of this, the second branch is the generator-based framework, which constructs an adversarial example by using a trained generative model and shows a great potential of transferability [5, 10, 44, 46]. However, the generator used in existing methods [21, 24, 25, 39, 46] is trained to adapt adversarial examples to known target classes only. As depicted in Fig. 1a and Fig. 1b, the trained generator is responsible for either only one class or a set of known classes. When it comes to an unknown target class (*i.e.*, the class not seen during training), previous methods are not capable of generating relatively adversarial examples unless retraining the generator, thus limiting the comprehensive assessment of DNNs. To tackle this, one straightforward solution is to train a generator to adapt a vast number of target classes with a large-scale dataset (*e.g.*, ImageNet-21k). However, as demonstrated in [44], the attack success rate of an adversarial example significantly degrades as the number of known classes increases. Hence, one question arises: *how to design a generalized yet efficient assessment in which vulnerability of DNNs can be evaluated by any target classes?*

To answer the above question, we study a more practical paradigm. As shown in Fig. 1c, any target object could be a good offense, and an adversarial example can be constructed from any target regardless of whether it is a known class or not. To achieve such generalization capability, we argue that extracting the major component of an object is the key to adversarial example generation. Motivated by this, we propose **G**eneralized **A**dversarial attac**KER**, termed (**GAKer**). The core idea behind the GAKer is to contaminate the latent representation of a clean image with the major component of the target object. To equip GAKer with the capability of latent infection, it jointly utilizes the latent representation of a clean image and the major component of the target object to

generate the adversarial example. Then, the adversarial example generated from GAKer is optimized to remain visually consistent with the clean image, but the corresponding major component is dominated by the target object. Once trained, the GAKer has the ability to replace major components between two images without depending on specific class or targeted DNN, thus improving the generalization capability of DNN assessment. Comprehensive evaluations across diverse DNNs, encompassing standard models, adversarially trained models, and vision-language foundation models, reveal that the proposed GAKer can effectively generate high-quality adversarial examples regardless of target classes. This demonstrates GAKer's generalization ability, making it a valuable tool for the adversarial robustness assessment of DNNs. In summary, three contributions are highlighted:

- We propose a novel **G**eneralized **A**dversarial attac**KER**, which is a general assessment tool since it can generate adversarial examples from any object. Without satisfying the visual appearance of an image, it can mislead any DNNs by latently changing the major components of an image.
- To our knowledge, this work, for the first time, explores the problem of generalized target adversarial attack. Our study reveals that changing the major components of an object is the key to the generalized assessment of DNNs.
- Extensive experiments conducted on a wide range of DNNs demonstrate the generalizability of our proposed method for vulnerability assessment, especially under the setting of any targeted class. Particularly, our method has increased the targeted attack success rate by approximately 14.13% for unknown classes and by approximately 4.23% for known classes compared with other generator-based approaches.

2 Related Work

2.1 Iterative Methods

Since the discovery of adversarial examples, most iterative methods are proposed, which utilize model gradients to iteratively add adversarial perturbations to specified images. These methods are mainly categorized as gradient-based optimization and input transformation. The gradient-based optimization aims to circumvent poor local optima by employing optimization techniques. MI-FGSM [3] and NI-FGSM [17] introduce momentum and Nesterov accelerated gradient into the iterative attack process to enhance black-box transferability, respectively. PI-FGSM [6] introduces patch-wise perturbations to better cover the discriminative region. VMI-FGSM [37] tunes the current gradient with the gradient variance from the neighborhood. RAP [26] advocates injecting worst-case perturbations at each step of the optimization procedure rather than minimizing the loss of individual adversarial points. The input transformation methods also increase adversarial transferability by preventing overfitting to the surrogate

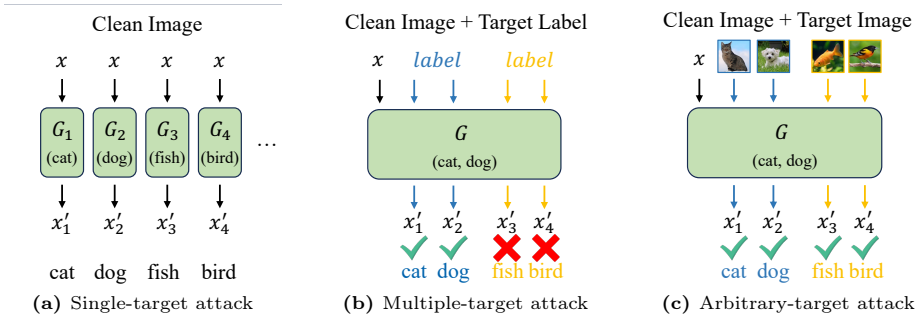

Fig. 1: The inference process of different generator-based targeted attacks. In the center of each scenario is the generator $\mathcal{G}$ that takes the model inputs at the top and aims to produce adversarial examples $\boldsymbol{x}'$ at the bottom. The classes indicated within $\mathcal{G}$ (cat, dog, fish, bird) are the training classes. Blue lines denote known classes that are encountered during inference, and yellow lines denote unknown classes that were not present in the training data. Sub-figure **(a)** depicts a single-target attack where each generator is specialized for one class, thus can only attack that specific class. Sub-figure **(b)** demonstrates a multiple-target attack where the generator $\mathcal{G}$ takes a source image $\boldsymbol{x}$ and known target labels (*e.g.*, cat, dog) to create their adversarial examples $\boldsymbol{x}'$, but it fails to attack labels unknown to the training (*e.g.*, fish, bird). Sub-figure **(c)** represents an arbitrary-target attack where $\mathcal{G}$ can utilize target images to craft adversarial examples capable of misleading the classifier into known and unknown classes (*e.g.*, fish, bird), highlighting the generalization capability of this approach.

model. DI-FGSM [43] applies various input transformations to the clean images. SIT [38] applies a random image transformation onto each image block to generate a diverse set of images for gradient calculation. SU [40] introduces a feature similarity loss to encourage universal learned perturbations by maximizing the similarity between the global adversarial perturbation and randomly cropped local regions.

2.2 Generative Methods

Another branch of targeted attacks utilizes generators to craft adversarial examples. Compared with iterative-based attacks, generator-based attacks have several characteristics [8]: high efficiency with just a single model-forward pass at test time and superior generalizability through learning the target distribution rather than class-boundary information [21]. Thus, many generator-based targeted attack methods are proposed, which can divided into single-target and multi-target generator attacks. Notably, this work focuses on scenarios where black-box models are entirely inaccessible, so query-based generator attacks which requiring extensive querying are not within the scope of discussion.

Single-target Generative Methods: Early generative targeted attacks employed a single generator to attack a specific target, primarily aiming to enhance transferability across various models. Pourseed [25] proposes a gen-

erator capable of producing image diagnostic and image-dependent perturbations for targeted attacks. Naseer [22] introduced a relativistic training objective to mislead networks trained on completely different domains. Furthermore, Naseer [21] matches the perturbed image distribution with that of the target class. TTAA [39] captures the distribution information of the target class from both label-wise and feature-wise perspectives to generate highly transferable targeted adversarial examples.

Multi-target Generative Methods: However, generator-based methods are confined to single-target attacks, which require training a generative model for each target class, resulting in considerable computational costs and inefficiency. Recently, exemplified by the introduction of the Multi-target Adversarial Network (MAN) framework [10], have revolutionized this landscape. MAN represents a paradigm shift by enabling the generation of adversarial examples across multiple target classes through a unified training process. Yang *et al.* [44] introduce a significant contribution by leveraging a hierarchical generative network. Through this design, they are able to train 20 generators, each trained for 50 target classes, thereby covering all 1000 classes in the ImageNet dataset. Gao *et al.* [5] proposes a generative targeted attack strategy named Easy Sample Matching Attack (ESMA), which exhibits a higher success rate for targeted attacks through generating perturbations towards High-Sample-Density-Regions of the target class.

2.3 Adversarial Defenses

A primary class of defense methods processes adversarial images to break the perturbations. For instance, Guo *et al.* [9] introduces several techniques for input transformation, such as JPEG compression [9], to mitigate adversarial perturbations. R&P [41] employs random resizing and padding to reduce adversarial effects. HGD [16] develops a high-level representation guided denoiser to diminish the impact of adversarial disturbances. ComDefend [14] proposes an end-to-end image compression model to defend against adversarial examples. NRP [20] trains a neural representation purifier model that removes adversarial perturbations using automatically derived supervision. Another approach enhances resilience to attacks by incorporating adversarial examples into the training phase. For instance, Tramèr *et al.* [35] bolster black-box robustness by utilizing adversarial examples generated from unrelated models. Similarly, Xie *et al.* [42] integrate feature denoising modules trained on adversarial examples to develop robustness in the white-box model.

3 Methodology

3.1 Problem Formulation

Formally, let $\boldsymbol{x}_s$ denote a clean image, and y is the corresponding label. $\mathcal{F}_\phi(\cdot)$ is the classifier with parameters ϕ. The targeted attack aims to mislead the classifier

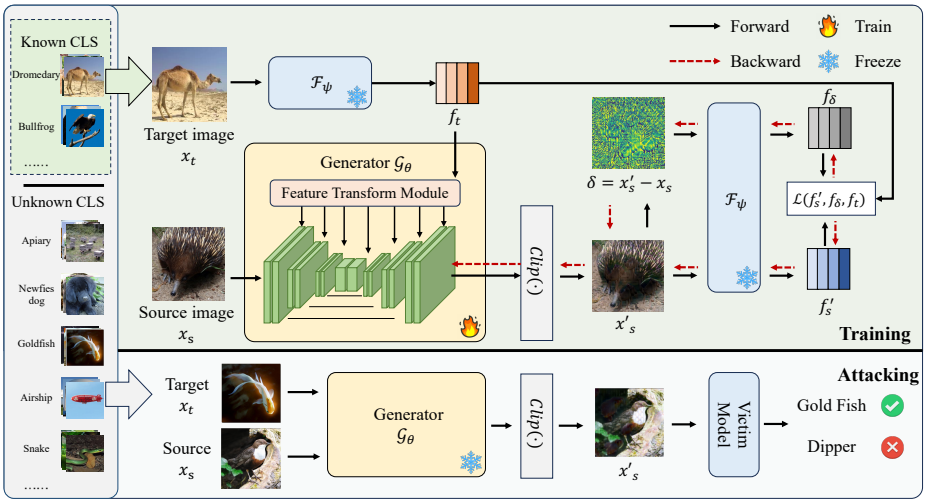

Fig. 2: The pipeline of our GAKer. We propose a generator-based method, GAKer, that can achieve attacks even when targeting classes unseen during training. During the training phase, we extract target features f_t through a frozen $\mathcal{F}_\psi$ and inject them into the generator $\mathcal{G}_\theta$, then use $Clip(\cdot)$ to constrain x'_s within the perturbation budget. $\mathcal{G}_\theta$ aims to minimize the cosine similarity between f'_s and f_t, as well as between f_δ and f_t. Due to our training strategy built on the feature distribution independent of the training classes, our generator can generate adversarial examples x'_s for unknown classes to attack the victim model.

to output a specific target class from an adversarial example $\boldsymbol{x}'_s$ corresponding to the clean image $\boldsymbol{x}_s$, as formulated as Eq. 1.

$$\mathcal{F}_\phi(\boldsymbol{x}'_s) = y^t, \qquad s.t.\ \|\boldsymbol{x}'_s - \boldsymbol{x}_s\|_\infty \leq \epsilon \tag{1}$$

where y^t represents a specific target class and it is often constrained as one of known classes, ϵ is the perturbation constraint.

In terms of arbitrary-target attack, it releases the constraint of the specific target class and aims to construct adversarial examples from any target regardless of whether it is from known classes or not. Given an arbitrary target image $\boldsymbol{x}_t$, the adversarial example generation is formulated as Eq. 2:

$$\boldsymbol{x}'_s = \min(\boldsymbol{x}_s + \epsilon, \max(\mathcal{G}(\boldsymbol{x}_s, \boldsymbol{x}_t), \boldsymbol{x}_s - \epsilon)), \tag{2}$$

where $\mathcal{G}_i$ denotes a trained adversarial generator. Compared with the conventional targeted attack, the arbitrary-target attack is more challenging since it requires a generator with a strong generalization capability.

3.2 Generalized Adversarial Attacker

In this section, we introduce the details of the proposed **Generalized Adversarial Attacker (GAKer)**. The overall pipeline is depicted in Fig. 2. Given a

target object, the GAKer intends to contaminate latent representations of the clean image by utilizing major components of the target. It is worth noting that the target object can be either a one-hot class label or an image with the visual appearance of the object. Existing methods [5, 10, 44] use the label index or the one-hot label as the condition for targeted adversarial example generation. However, the one-hot label lacks visual characteristics, which leads a trained generator to memory class features, thus limiting the generalization capability. To incorporate richer target information, we use images of the target object as input for generating adversarial examples. Next, we divide classes into known classes and unknown classes $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{Y}}_{\text{known}} \cup \boldsymbol{\mathcal{Y}}_{\text{unknown}}$. Then we select known classes as the training set ($t \sim \boldsymbol{\mathcal{Y}}_{\text{known}}$) and generate an adversarial example using relevant image $\boldsymbol{x}_t$ of the target. The objective function is represented as Eq. 3:

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x}_s \sim \mathcal{X}_s, t \sim \mathcal{Y}_{known})} [\mathcal{L}(\boldsymbol{x}_s, \boldsymbol{x}_t)], \tag{3}$$

where $\mathcal{L}$ is the loss function, $\mathcal{F}_\psi(\cdot)$ is the pretrained feature extractor, and $\mathcal{G}_\theta$ is our arbitrary-target generator (see Sec. 3.3 for detailed architecture). To generalize to unknown target classes during the inference phase, we use cosine distance as the loss function instead of cross-entropy:

$$\mathcal{D}_{cos}(f'_s, f_t) = 1 - \frac{f'_s \cdot f_t}{\|f'_s\|_2 \cdot \|f_t\|_2}, \tag{4}$$

where $f'_s = \mathcal{F}_\psi(\boldsymbol{x}'_s)$, $f_t = \mathcal{F}_\psi(\boldsymbol{x}_t)$. In addition, an identical learning objective is used to constraint the feature of adversarial perturbation $\delta = \boldsymbol{x}'_s - \boldsymbol{x}_s$ and the feature of the target object:

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}_s, \boldsymbol{x}_t) =& \mathcal{D}_{cos}(\mathcal{F}_\psi(\mathcal{G}_\theta(\boldsymbol{x}_s, \boldsymbol{x}_t)), \mathcal{F}_\psi(\boldsymbol{x}_t)) \\ &+ \alpha \mathcal{D}_{cos}(\mathcal{F}_\psi(\mathcal{G}_\theta(\boldsymbol{x}_s, \boldsymbol{x}_t) - \boldsymbol{x}_s), \mathcal{F}_\psi(\boldsymbol{x}_t)), \end{aligned} \tag{5}$$

where $f_\delta = \mathcal{F}_\psi(\delta)$, α denotes the hyper-parameter (see Appendix C for the effect of α). The entire training process is independent of specific target classes, enabling adaptation to unknown classes, including those from different datasets.

3.3 Latent Infection

This section describes how to inject target features into the source image in the latent feature space. There are two major modules in GAKer: the Feature Extractor $\mathcal{F}_\psi$ and the Generator $\mathcal{G}_\theta$, as shown in the Fig. 2. The Feature Extractor is adapted from a pretrained model by removing the classification head. It is specialized for extracting feature vectors from input images. During the training, the weights of this module are frozen. We employ a UNet [5] as the basic architecture of the generator. It is designed to generate adversarial examples by using clean images with features of the target object obtained from the Feature Extractor.

As depicted in Fig. 3, the latent infection involves a two-step process. First, features of the target object are processed through the Feature Transform Module (FTM), which adopts a Linear-GELU-Linear sequence to enhance their representational capacity. Second, these enhanced features are combined with the features of the clean image. Particularly, the Dimension Matching Module (DMM), which consists of a Linear-GELU layer, is used to align dimensions for two features. With the designed architecture, the generator is optimized to extract major components from the target object, and learns to replace the major components of the clean image with that of the target object.

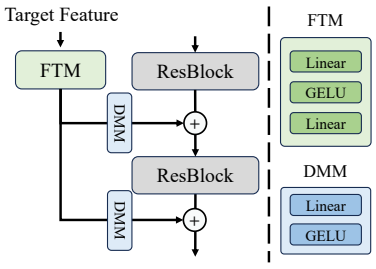

Fig. 3: Schematic diagram of feature insertion into each ResBlock of a UNet. The features are first transformed by the Feature Transform Module (FTM), followed by dimension matching through the Dimension Matching Module (DMM) layers before being integrated into each ResBlock of the UNet.

4 Experiment

4.1 Experimental Settings

Datasets. We train our models on the ImageNet training set [2]. Correspondingly, we evaluate the performance on the ImageNet val set.
Networks. We consider several models, including DenseNet-121 (Dense-121) [13], ResNet-50 (Res-50) [11] and VGG-19 as surrogate models. We select various black-box models, i.e., ResNet-152 (Res-152) [11], VGG-19 [28], Inception-v3 (Inc-v3) [30], ViT [32], DeiT [33] and CLIP [27], for testing the transferability of attacks. Additionally, we evaluate the proposed method on defense models, including Inc-v3$_{adv}$, Inc-v3$_{ens3}$, Inc-v3$_{ens4}$, IncRes-v2$_{ens}$ [35], and Large Vision-Language Models (LVLM) such as LLaVA [18, 19] and Qwen-VL [1].
Baselines. For iterative attacks, we compare our method with MI [3] and the advanced method SU [40], which is competitive in target settings. For single-target generative attacks, we choose TTP [21] as the method for comparison with our approach. Specifically, we train multiple TTP models to accomplish multi-target attacks. For multi-target generative attacks, we choose HGN [44] and ESMA [5]. Both of these methods, along with ours, only require training a single generator to perform attacks on multiple targets.
Implementation details. In all experiments, the perturbation constraint ϵ is set to 16, the number of known classes N is set to 200, the α is set to 0.5 and the number of samples in each known class M is 325. We train the generator with an AdamW optimizer for 20 epochs. For the MI method, we set the decay factor μ to 1. For the SU method, we perform the combinational attack of DTMI [3, 43]

Table 1: Targeted Attack Success Rates on Unknown Classes. We report targeted attack success rates (%) of each method and the leftmost model column denotes the substitute model ("*" means white-box attack results).

Model	Attacks	Res-50	Res-152	VGG-19	Dense-121	Inc-v3	ViT	DeiT	CLIP	Avg
	Clean	0.03	0.01	0.00	0.03	0.05	0.00	0.00	0.02	0.02
Res-50	HGN	0.05*	0.14	0.15	0.06	0.04	0.05	0.06	0.10	0.08
	GAKer (Ours)	**41.69***	**23.05**	**26.02**	**23.80**	**5.85**	**1.44**	**4.99**	**4.36**	**16.40**
VGG-19	HGN	0.08	0.10	0.08*	0.10	0.04	0.04	0.04	0.10	0.07
	GAKer (Ours)	**11.05**	**5.41**	**43.25***	**13.33**	**3.20**	**0.51**	**2.55**	**2.28**	**10.20**
Dense-121	HGN	0.06	0.05	0.09	0.05*	0.00	0.03	0.01	0.05	0.04
	GAKer (Ours)	**23.10**	**17.28**	**25.17**	**40.31***	**7.23**	**2.60**	**7.56**	**4.49**	**15.97**

and SU [40]. For both iterative methods, we employ the logit loss and set the number of iterations to 300 steps. For multi-target generative attacks, such as HGN and ESMA, and our method, only one model needs to be trained to achieve attacks on multiple target classes. Detailed training costs and implementation specifics are provided in Appendix A and Appendix B, respectively.

4.2 Main Results

Results on Unknown Classes. Compared with existing generator-based attacks, the best innovation of our method is the ability to attack unknown classes. We select 200 classes as known classes and the remaining 800 classes as unknown classes from ImageNet. Then we train the generator with the known classes and evaluate the targeted attack success rate on unknown classes. Notably, only HGN and our method can be evaluated on unknown classes.

As shown in Tab. 1, our method significantly outperforms HGN, highlighting the superior transferability of our approach. For instance, with a substitute model of ResNet-50 and a black-box model of VGG-19, our method achieves a success rate of **41.69%** on unknown classes, while HGN only achieves 0.05%. This result underscores the limitation of existing methods in generating targeted adversarial examples for unknown classes, while our method demonstrates effective generalization to such classes. Separate average results on all unknown classes can be found in Appendix D.

Results on Known Classes. For evaluation on known classes, we compare our method with state-of-the-art iterative-based method (SU), single-target generator-based attacks (TTP), and multi-target generator-based attacks (HGN, ESMA). All multi-target generator-based attacks (HGN, ESMA, and our method) are trained on the same 200 classes. Due to the TTP method requiring training a model for each target class, the cost of training 200 models is prohibitively high. Therefore, we randomly select 10 classes from the 200 classes and train 10 TTP models separately for each substitute model (TTP-10). We then test our method on the same 10 classes (GAKer-10).

Table 2 shows the targeted attack success rates on known classes for each method. Compared with the iterative-based method SU, our method achieves

Table 2: Targeted Transfer Success Rates on Known Classes. We report targeted attack success rates (%) of each method and the leftmost model column denotes the substitute model ("*" means white-box attack results).

Model	Attacks	Res-50	Res-152	VGG-19	Dense-121	Inc-v3	ViT	DeiT	CLIP	Avg
	Clean	0.02	0.02	0.01	0.03	0.03	0.02	0.02	0.01	0.02
Res-50	MI	99.35*	16.55	4.10	12.10	0.65	0.60	0.65	0.15	16.77
	SU	**99.45***	83.15	75.60	81.15	14.95	8.15	21.80	6.90	48.89
	HGN	87.06*	55.68	52.09	64.59	24.09	16.36	29.35	6.71	41.99
	ESMA	95.60*	83.22	81.98	**82.54**	**40.14**	**28.01**	**55.45**	**24.08**	**61.37**
	GAKer (Ours)	96.61*	**83.36**	**82.20**	81.95	34.27	20.84	50.31	20.13	58.71
	TTP-10	97.80*	77.80	73.00	79.00	44.00	**33.10**	44.00	19.40	58.51
	GAKer-10 (Ours)	**98.10***	**88.20**	**90.60**	**86.00**	**45.20**	28.70	**62.00**	**24.50**	**65.41**
VGG-19	MI	3.10	1.40	96.85*	3.60	0.30	0.20	0.55	0.35	13.29
	SU	26.35	14.29	96.90*	26.05	4.10	2.20	5.89	4.60	22.55
	HGN	28.50	16.65	90.15*	33.05	6.95	**6.00**	13.50	2.95	24.72
	ESMA	27.30	17.85	95.39*	35.80	4.85	5.35	**16.70**	**6.12**	26.17
	GAKer (Ours)	**30.00**	**18.02**	**97.61***	**33.38**	**10.76**	3.36	11.65	4.75	**26.22**
	TTP-10	**50.10**	**37.50**	**98.20***	**48.80**	**17.20**	**13.40**	**20.00**	**10.60**	**36.98**
	GAKer-10 (Ours)	32.00	16.50	93.80*	32.70	12.00	3.05	10.05	4.00	25.51
Dense-121	MI	10.25	5.85	5.15	**99.55***	1.00	0.65	1.20	0.50	15.52
	SU	65.20	49.70	59.90	**99.55***	14.90	7.65	19.35	6.00	40.28
	HGN	53.48	40.15	50.67	90.11*	23.32	13.92	22.35	5.00	37.38
	ESMA	43.01	34.74	42.61	81.52*	11.74	9.89	23.47	6.24	31.65
	GAKer (Ours)	**70.26**	**61.12**	**69.83**	90.91*	**30.81**	**22.15**	**43.81**	**16.41**	**50.66**
	TTP-10	**86.70**	**76.50**	82.80	**97.90***	**49.20**	**44.10**	57.00	**24.60**	**64.85**
	GAKer-10 (Ours)	74.40	70.20	**82.89**	93.20*	42.80	28.99	**60.00**	23.10	59.45

higher targeted attack success rates on black-box models. For example, if the substitute model is Dense-121, our method performs 10.38% better than SU on average across different models. For generator-based attacks, our GAKer also achieves a similar attack success rate to the ESMA method, outperforming the HGN method. Notably, our method improves performance on several models by an average of 10.5% and 5.47% over HGN and ESMA, respectively.

4.3 Results on Other Models

To further evaluate the generalization ability of our method, we also test our method on defense models and Large Vision-Language Models (LVLM).

Results on Defense Models. In addition to attacking normally trained models, we evaluate the attack performance of various methods on adversarially trained models when using ResNet50 as the white-box model. As shown in Tab. 3, our method surpasses the HGN method in attacking targets in unknown and known classes. For example, with the defense model as Inc-v3$_{adv}$, we outperform HGN by **6.46%** in terms of target success rate on the known classes. On the unknown classes, HGN cannot achieve an attack at all, similar to the performance of the clean samples, whereas our GAKer still exhibits an attack success rate of **5.95%**. This result indicates that our generator has discovered more common model vulnerabilities, regardless of whether the model has been specifically trained for defense.

Table 3: Targeted Transfer Success Rates on Defense Models. Targeted success rates (%) for each attack method using Res-50 as the substitute model (known classes / unknown classes).

Attacks	Inc-v3$_{adv}$	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$
Clean	0.03/0.04	0.04/0.03	0.02/0.01	0.03/0.03
HGN	29.22/0.06	26.24/0.06	**24.88**/0.07	21.69/0.04
GAKer (Ours)	**35.68**/**5.95**	**29.60**/**4.10**	23.66/**3.38**	**25.25**/**3.21**

Table 4: Attack Success Rates for Unknown Classes on LVLM. We report attack success rates (%) of each method and the topmost row denotes the substitute model. (untargeted attack success rates / targeted attack success rates)

Substitute Models	Res-50		Dense-121		VGG-19	
Black-box	Qwen-VL	LLaVA	Qwen-VL	LLaVA	Qwen-VL	LLaVA
Clean	5.77/8.15	9.03/7.02	5.77/8.15	9.03/7.02	5.77/8.15	9.03/7.02
SU	20.16/35.36	18.77/42.73	19.59/28.76	17.83/32.73	20.40/33.35	18.13/**46.80**
HGN	24.98/12.50	20.76/13.85	22.12/12.27	20.76/13.85	23.65/12.79	20.23/14.34
GAKer (Ours)	**33.82**/**52.60**	**23.41**/**56.45**	**36.71**/**54.43**	**27.54**/**60.41**	**39.86**/**36.31**	**24.50**/41.61

Results on Large Vision-Language Models. The growing importance of Large Vision-Language Models (LVLM), including LLaVA [18, 19] and Qwen-VL [1], has been noted in recent times. Our research examines how well our method can be adapted for use with these LVLMs. Specifically, we test multiple templates on LVLMs to reduce the impact of prompt bias:

- Is there any (origin class / target class) in this image? Please begin answer with 'Yes,' or 'No,'
- Does this image contain any (origin class / target class)? Please begin answer with 'Yes,' or 'No,'
- In this image, is there a (origin class / target class) present? Please begin answer with 'Yes,' or 'No,'

We then assessed the effectiveness of our approach by calculating both the untargeted and targeted attack success rates against LVLMs. We define the untargeted attack success rate as the percentage of adversarial examples that mislead the model into failing to recognize the original class. The targeted attack success rate is defined as the percentage of adversarial examples that are misclassified into the target class. As shown in Tab. 4 and Tab. 5, our method achieves higher targeted attack success rates on LVLM than HGN. For example, when the substitute model is Res-50 and the target is unknown classes, our method achieves **52.60%** and **56.45%** targeted attack success rates on LLaVA and Qwen-VL, respectively, while HGN only achieves 12.50% and 13.85%. This result demonstrates that even when using a "small" substitute model like Res-50, our method can successfully attack "large" models such as Qwen-VL and LLaVA. We also show some cases on GPT-4V, which can be found in Appendix E.

Table 5: Attack Success Rates for Known Classes on LVLM. We report attack success rates (%) of each method and the topmost row denotes the substitute model. (untargeted attack success rates / targeted attack success rates)

Substitute Models	Res-50		Dense-121		VGG-19	
Black-box	Qwen-VL	LLaVA	Qwen-VL	LLaVA	Qwen-VL	LLaVA
Clean	5.15/10.33	8.43/7.42	5.15/10.33	8.43/7.42	5.15/10.33	8.43/7.42
SU	20.10/42.28	18.78/46.02	19.70/35.67	17.13/37.28	20.53/38.77	18.38/**48.88**
HGN	22.95/36.57	18.67/34.35	21.70/31.13	18.22/29.78	22.80/28.87	16.15/30.95
ESMA	**50.07**/70.53	**33.67**/66.05	40.60/41.73	**28.97**/31.12	38.68/**54.02**	23.48/45.75
GAKer (Ours)	35.20/**73.62**	24.32/**73.37**	**47.60**/**63.10**	27.30/**72.00**	**45.00**/42.50	**26.95**/44.90

4.4 Ablation Study

This section discusses how different parameter selections in training dataset construction impact the generator's attack capability.

Numbers of Known Classes. Figure 4 demonstrates the impact of the number N of known classes on the generator's performance. Specifically, to mitigate the impact of adding new classes as the number of known classes increases, we evaluate targeted attack success rate on a common set of 10 known classes and 500 common unknown classes. When N is 10 or 50, the attack success rate on unknown classes is low due to limited training data. With N increased to 500, the white-box success rate reaches 49.45% on 500 unknown classes. Despite higher training costs, performance does not significantly improve with more known classes. Therefore, N is set to 200 to balance performance and training cost.

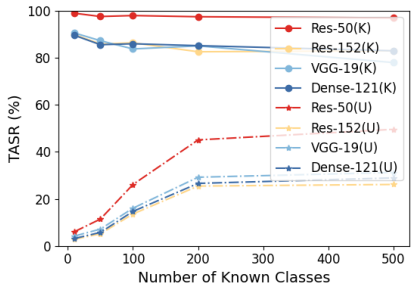

Fig. 4: Comparison of targeted attack success rates across a range of known classes. The Res-50 serves as the substitute model, while the performance of black-box models, including Res-152, VGG-19, and Dense-121, is evaluated for both known (K) and unknown (U) classes.

When N is 10 or 50, the attack success rate on unknown classes is low due to limited training data. With N increased to 500, the white-box success rate reaches 49.45% on 500 unknown classes. Despite higher training costs, performance does not significantly improve with more known classes. Therefore, N is set to 200 to balance performance and cost.

Strategies for Choosing Known Classes. Previous work [44] on multi-target generators has highlighted the importance of not only determining the number of target classes but also selecting which target classes to attack. When there

are significant differences between the selected target classes, the generator can achieve a better targeted attack success rate.

We introduce a similarity greedy algorithm to select a set of classes with the largest feature differences. To simplify, we represent each target class using the average of its image features and measure the similarity using cosine similarity. Specifically, the algorithm starts by randomly selecting an initial feature vector as the first class and adds it to the selected group. It then iteratively selects a vector from the remaining pool that has the lowest average cosine similarity to the vectors in the selected group. We add this selected class to the selected group and repeat the process until achieve the specified number of target classes.

We conduct multiple experiments to eliminate the randomness introduced by the initial selection. To validate our selection method's effectiveness, we compare it with randomly selecting strategy. Figure 5 illustrates that our method significantly outperforms random selection on known classes. Compared with the random strategy, we achieve a success rate increase of **16.52%** on black-box model Res-152 when the substitute model is Res-50. This demonstrates the necessity of selecting known classes and the effectiveness of our selecting method.

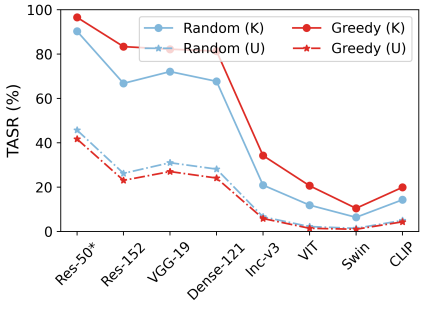

Fig. 5: Comparison of targeted attack transfer success rates under different known classes selection strategies.

Numbers of Sample in Each Known Class. We observe that noisy images, such as those with occlusion or blurring, can decrease performance. To validate this, we employ three selection methods to form groups, each comprising 325 images: low quality (largest classification loss), high quality (lowest loss), and random. Figure 6(a) demonstrates that image quality significantly impacts the attack success rate. Consequently, we prioritize training images by their classification loss, favoring those with the lowest losses. Specifically, we experiment with different numbers of images ($M \in \{1, 130, 325, 650, 1300\}$) for each known class, as illustrated in Figure 6(b). When the number of images (M) is less than 325, the performance is lower, likely because the generator could not capture the full breadth of class characteristics. Conversely, when M exceeds 325, the attack success rate slightly decreased, potentially due to the inclusion of more noisy images. Thus, we ultimately select 325 as the optimal number of samples for each known class. This choice yields a maximum attack success rate of 41.69% for the unknown class in the ResNet-50 model. This finding underscores the equal importance of both the quantity and quality of training samples.

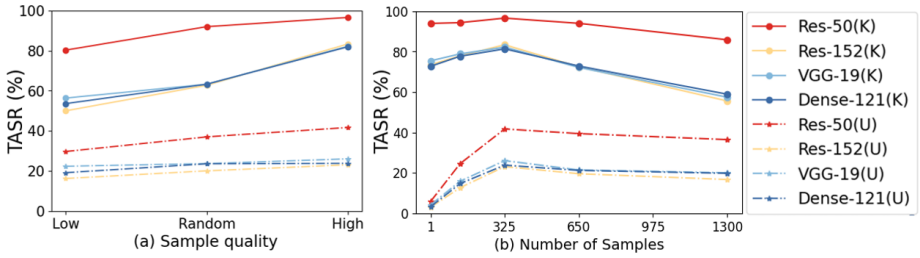

Fig. 6: Comparisons of targeted attack success rates at sample quality and different numbers of targeted images per known class.

5 Conclusion

Generator-based targeted attacks are able to mislead DNNs to any target they have been trained on, showing their dangers. For the first time, we find that the attack also extends to untrained unknown classes, and the extent of its potential harm is revealed. We propose the Generalized Adversarial attacker, which injects target feature into adversarial examples to attack unknown classes. Through comprehensive experiments across standard, defense, and large vision-language models, we demonstrate that our method can effectively attack unknown and known classes across models. We hope our work will draw attention to the potential dangers of generator-based targeted attacks and inspire future research in this area.

6 Societal Impacts & Limitation.

Societal Impacts. Previous research on generator-based target attacks requires the attacker to know the target class. Our proposed algorithm allows successful targeted attacks without this information, highlighting the risk of relying solely on dataset and model confidentiality for security. Moreover, our method's success on unknown classes reveals inherent model vulnerabilities, offering new insights for advancing security.

Limitation. While our method validates the possibility of attacking unknown classes using generator-based methods, the gap in target attack success rates between known and unknown classes still exists. In the future, we will focus on analyzing the reasons for this difference.

Acknowledgements

This study is supported by grants from the National Natural Science Foundation of China (Grant No. 62122018, No. 62020106008, No. U22A2097, No. U23A20315), Kuaishou, and SongShan Laboratory YYJC012022019. It is also supported by the Postdoctoral Fellowship Program of CPSF under Grant Number GZB20240114.

References

1. Bai, J., Bai, S., Yang, S., Wang, S., Tan, S., Wang, P., Lin, J., Zhou, C., Zhou, J.: Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. CoRR (2023)
2. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
3. Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., Li, J.: Boosting adversarial attacks with momentum. In: CVPR (2018)
4. Dong, Y., Pang, T., Su, H., Zhu, J.: Evading defenses to transferable adversarial examples by translation-invariant attacks. In: CVPR (2019)
5. Gao, J., Qi, B., Li, Y., Guo, Z., Li, D., Xing, Y., Zhang, D.: Perturbation towards easy samples improves targeted adversarial transferability. In: NeurIPS (2023)
6. Gao, L., Zhang, Q., Song, J., Liu, X., Shen, H.T.: Patch-wise attack for fooling deep neural network. In: ECCV (2020)
7. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICLR (2015)
8. Gu, J., Jia, X., de Jorge, P., Yu, W., Liu, X., Ma, A., Xun, Y., Hu, A., Khakzar, A., Li, Z., Cao, X., Torr, P.H.S.: A survey on transferability of adversarial examples across deep neural networks. CoRR (2023)
9. Guo, C., Rana, M., Cissé, M., van der Maaten, L.: Countering adversarial images using input transformations. In: ICLR (2018)
10. Han, J., Dong, X., Zhang, R., Chen, D., Zhang, W., Yu, N., Luo, P., Wang, X.: Once a MAN: towards multi-target attack via learning multi-target adversarial network once. In: ICCV (2019)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
12. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: NeurIPS (2020)
13. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR (2017)
14. Jia, X., Wei, X., Cao, X., Foroosh, H.: Comdefend: An efficient image compression model to defend adversarial examples. In: CVPR (2019)
15. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. In: ICLR (2017)
16. Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., Zhu, J.: Defense against adversarial attacks using high-level representation guided denoiser. In: CVPR (2018)
17. Lin, J., Song, C., He, K., Wang, L., Hopcroft, J.E.: Nesterov accelerated gradient and scale invariance for adversarial attacks. In: ICLR (2020)
18. Liu, H., Li, C., Li, Y., Lee, Y.J.: Improved baselines with visual instruction tuning. CoRR (2023)
19. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: NeurIPS (2023)
20. Naseer, M., Khan, S.H., Hayat, M., Khan, F.S., Porikli, F.: A self-supervised approach for adversarial robustness. In: CVPR (2020)
21. Naseer, M., Khan, S.H., Hayat, M., Khan, F.S., Porikli, F.: On generating transferable targeted perturbations. In: ICCV (2021)
22. Naseer, M., Khan, S.H., Khan, M.H., Khan, F.S., Porikli, F.: Cross-domain transferability of adversarial perturbations. In: NeurIPS (2019)
23. Naseer, M., Ranasinghe, K., Khan, S., Khan, F., Porikli, F.: On improving adversarial transferability of vision transformers. In: ICLR (2022)

24. Naseer, M., Ranasinghe, K., Khan, S.H., Khan, F.S., Porikli, F.: On improving adversarial transferability of vision transformers. In: ICLR (2021)
25. Poursaeed, O., Katsman, I., Gao, B., Belongie, S.J.: Generative adversarial perturbations. In: CVPR (2018)
26. Qin, Z., Fan, Y., Liu, Y., Shen, L., Zhang, Y., Wang, J., Wu, B.: Boosting the transferability of adversarial attacks with reverse adversarial perturbation. In: NeurIPS (2022)
27. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: ICML (2021)
28. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
29. Su, S., Liu, J., Gao, L., Song, J.: F^3-pruning: A training-free and generalized pruning strategy towards faster and finer text-to-video synthesis. In: AAAI (2024)
30. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR (2016)
31. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: ICLR (2014)
32. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML (2021)
33. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML (2021)
34. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models. CoRR (2023)
35. Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I.J., Boneh, D., McDaniel, P.D.: Ensemble adversarial training: Attacks and defenses. In: ICLR (2018)
36. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
37. Wang, X., He, K.: Enhancing the transferability of adversarial attacks through variance tuning. In: CVPR (2021)
38. Wang, X., Zhang, Z., Zhang, J.: Structure invariant transformation for better adversarial transferability. In: ICCV (2023)
39. Wang, Z., Yang, H., Feng, Y., Sun, P., Guo, H., Zhang, Z., Ren, K.: Towards transferable targeted adversarial examples. In: CVPR (2023)
40. Wei, Z., Chen, J., Wu, Z., Jiang, Y.: Enhancing the self-universality for transferable targeted attacks. In: CVPR (2023)
41. Xie, C., Wang, J., Zhang, Z., Ren, Z., Yuille, A.L.: Mitigating adversarial effects through randomization. In: ICLR (2018)
42. Xie, C., Wu, Y., van der Maaten, L., Yuille, A.L., He, K.: Feature denoising for improving adversarial robustness. In: CVPR (2019)
43. Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z., Yuille, A.L.: Improving transferability of adversarial examples with input diversity. In: CVPR (2019)
44. Yang, X., Dong, Y., Pang, T., Su, H., Zhu, J.: Boosting transferability of targeted adversarial examples via hierarchical generative networks. In: ECCV (2022)
45. Yuan, S., Zhang, Q., Gao, L., Cheng, Y., Song, J.: Natural color fool: Towards boosting black-box unrestricted attacks. In: NeurIPS (2022)
46. Zhao, A., Chu, T., Liu, Y., Li, W., Li, J., Duan, L.: Minimizing maximum model discrepancy for transferable black-box targeted attacks. In: CVPR (2023)