Enhancing Generalization of Universal Adversarial Perturbation through Gradient Aggregation

Xuannan Liu, Yaoyao Zhong, Yuhang Zhang, Lixiong Qin, Weihong Deng*
Beijing University of Posts and Telecommunications
{liuxuannan, zhongyaoyao, zyhzyh, lxqin, whdeng}@bupt.edu.cn

Abstract

Deep neural networks are vulnerable to universal adversarial perturbation (UAP), an instance-agnostic perturbation capable of fooling the target model for most samples. Compared to instance-specific adversarial examples, UAP is more challenging as it needs to generalize across various samples and models. In this paper, we examine the serious dilemma of UAP generation methods from a generalization perspective – the gradient vanishing problem using small-batch stochastic gradient optimization and the local optima problem using large-batch optimization. To address these problems, we propose a simple and effective method called Stochastic Gradient Aggregation (SGA), which alleviates the gradient vanishing and escapes from poor local optima at the same time. Specifically, SGA employs the small-batch training to perform multiple iterations of inner pre-search. Then, all the inner gradients are aggregated as a one-step gradient estimation to enhance the gradient stability and reduce quantization errors. Extensive experiments on the standard ImageNet dataset demonstrate that our method significantly enhances the generalization ability of UAP and outperforms other state-of-the-art methods. The code is available at https://github.com/liuxuannan/Stochastic-Gradient-Aggregation.

1. Introduction

Deep neural networks (DNNs) have achieved significant success in computer vision [9, 35, 34, 8, 11, 12], but are widely known to be vulnerable to adversarial examples [36, 23, 26, 7, 15, 19, 39]. A more critical property of adversarial examples is that they have shown good transferability between different models [18, 25, 5, 6, 17, 40, 50, 38, 29]. Unlike the instance-specific adversarial examples, a recent work [22] reveals the existence of universal adversarial perturbation (UAP) which can deceive the majority of samples.

Compared to the instance-specific adversarial examples, the generation of UAP is more challenging. Since most UAPs are generated with limited training samples and are expected to be applied to various unknown samples [14, 28, 33, 16] and even a variety of tasks [41, 4, 30, 1, 27, 51], the diversity of samples and models has created dual demands for higher generalization ability of UAPs. Therefore, we aim to investigate the limitation of the current UAPs and improve the generalization ability.

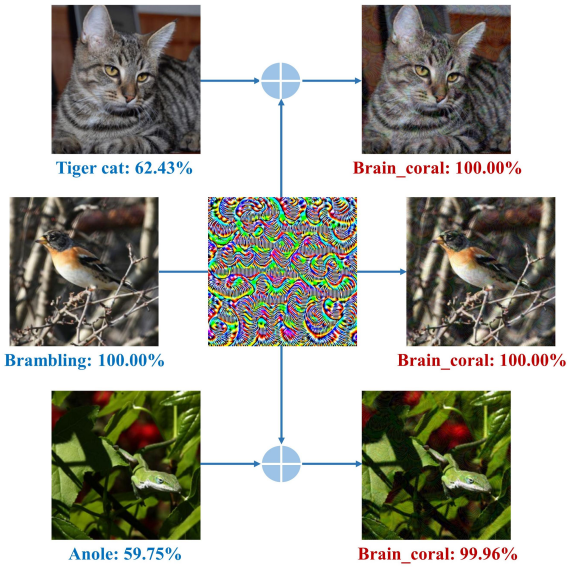


Figure 1. A universal adversarial perturbation is applied to images belonging to different categories to get visually similar adversarial examples with high attack success rates in the white-box scenario. **Left images:** the original natural images. **Central image:** the UAP by applying SGA on the VGG16 model (rescaled to [0,255]). **Right images:** the adversarial images.

Despite many works on UAP, there are two main issues in the generation of UAP. (1) Gradient instability. Due to the inherent difference of samples and model parameters, the optimization paths for adversarial perturbations vary widely, as shown in Fig 2 (a). Such instability may hinder the optimization of adversarial attacks in the correct direction [38, 42, 37]. (2) Quantization error. The adversar-

*Corresponding author

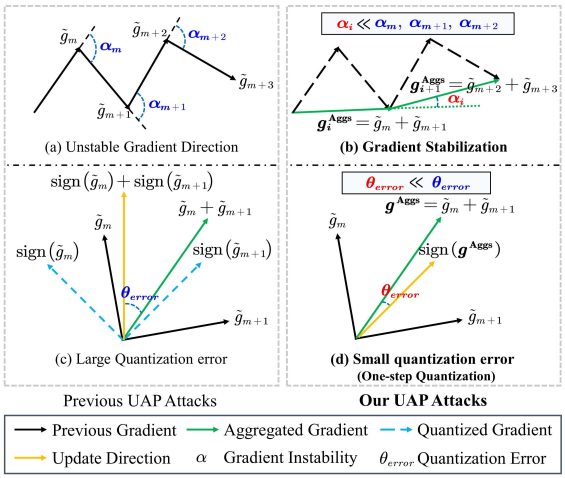


Figure 2. Illustration of the two issues, *i.e.*, gradient instability and quantization error, under previous UAP attacks and our attacks, respectively. Where $\tilde{g}_m$ and g^{Aggs} denote the pre-search gradient and the aggregated gradient respectively. In (b), we use the subscript i to distinguish between different g_i^{Aggs} in the outer iteration. (a) and (c): Previous works sequentially quantize the unstable gradients to accumulate the adversarial perturbations, resulting in severe gradient deviation. (b) and (d): Our work accumulates the pre-search gradients as a one-step gradient update with low variance before quantization to reduce the gradient deviation.

ial attack methods generally exploit the sign operations for quickly crafting adversarial perturbations according to the linearity of models. Frequent use of sign will accumulate a large amount of quantization errors [2, 48] in Fig 2 (c). Furthermore, when quantizing gradients with high fluctuations using the sign operations, the large values of the gradients in the forward optimization process can be easily eliminated by the negative small gradient values in the backward process, leading to the gradient vanishing phenomenon.

The generation of UAP can be described as a non-convex optimization problem and adopt stochastic gradient descent (SGD) with mini-batch training. Therefore, an intuitive idea is to use large-batch methods to stabilize gradient update directions, thereby suppressing gradient vanishing. However, when optimizing via SGD, small-batch methods have been shown to converge more easily to flat local minima than large-batch methods, thus effectively reducing the generalization gap [13]. Inevitably, the generalization of UAP faces the grim dilemma of choosing small-batch training with the gradient vanishing problem or choosing large-batch training with the local optima problem.

In this paper, we propose a novel method called stochastic gradient aggregation (SGA), to address the gradient vanishing and the local optima problems to enhance the generalization of UAP. Specifically, our method consists of the inner-outer iterations. At each outer iteration for the gradient estimating, we randomly select multiple small-batch samples to perform the inner iterations for pre-search. Then we aggregate all inner gradients as a one-step iterative gradient for updating UAP, as illustrated in Fig 2 (b) and (d). The key idea is to cope with gradient vanishing by enhancing the gradient stability and decreasing the use of quantization operations while introducing the noisy gradients to escape from sharp local optima. To the best of our knowledge, this is the first work to investigate the limitation of existing universal attacks through the perspective of generalization. The main contributions of our paper are as follows:

- We investigate two issues behind the low generalization ability of existing UAP works, *i.e.*, gradient instability and quantization error, and further identify the gradient vanishing phenomenon when the iterative gradients have high fluctuations.
- We propose stochasitc gradient aggregation (SGA) that stabilizes the update directions and reduces quantization errors to alleviate the gradient vanishing in the small-batch optimization.
- Our method can be easily integrated with transferability attack methods. Extensive experiments demonstrate the superior generalization of UAP generated by the proposed SGA compared to the state-of-the-art methods under various attack settings.

2. Related work

Instance-specific attack methods. Szegedy *et al*. [36] first found the adversarial example and proposed an attack method based on box constraints, named as L-BFGS. By exploiting the decision boundary, Moosavi-Dezfooli *et al*. [23] proposed to generate the minimal perturbation at each step, called DeepFool. Moreover, Papernot *et al*. [26] constructed adversarial saliency maps to search for critical input regions for generating adversarial examples. To improve efficiency, Goodfellow *et al*. [7] performed a one-step attack updating along the sign direction of the gradient, widely known as the fast gradient sign method (FGSM). Kurakin *et al*. [15] further extended it to an iterative version of the FGSM, called I-FGSM. In addition, projected gradient descent (PGD) [19] is another gradient-based iterative attack and has demonstrated excellent attack performance. To ensure the naturalness of adversarial examples, Xiao *et al*. [39] proposed to exploit Generative Adversarial Networks to directly synthesize tiny perturbations.

Universal attack methods. Previous works on UAP mainly focus on the cross-sample university. Moosavi-Dezfooli *et al*. [22] first revealed the existence of UAPs and generated them by iteratively aggregating perturbation vectors obtained by the DeepFool method. SV-UAP [14] exploited a few feature maps to calculate the singular vectors of the Jacobian matrices for crafting UAP. Poursaeed *et*

al. [28] and Mopuri *et al*. [24] both proposed to use generative models to synthesize UAP, named GAP and NAG respectively. To make UAP only misclassify the target class, Zhang *et al*. [43] proposed a class discriminative UAP (CD-UAP). Other works of Zhang [44, 45] utilized dominant features and the cosine similarity loss to generate UAP, denoted as DF-UAP and Cos-UAP respectively. Besides, Shafahi *et al*. [33], Matachana *et al*. [21] and Co *et al*. [3] all introduced stochastic gradient method for efficiently generating UAP by using PGD with mini-batch training. Recently, Li *et al*. [16] proposed to integrate instance-specific and universal attacks from a feature perspective to generate a more powerful UAP, called AT-UAP. However, unlike research [13, 10, 20, 47, 46] on the generalization of model training, there are few works on the generalization of UAP.

Gradient optimization. Various of gradient optimization methods are proposed to improve the model transferability of adversarial examples. Dong *et al*. [5] proposed to add the momentum item to stabilize update directions and escape from poor local maxima. Furthermore, Lin *et al*. [17] introduced Nesterov accelerated gradient and scale-invariant property. Wang [38] proposed to shrink the gradient variance at each step to converge to better local optima. To encourage adversarial examples to converge flat local minimums, Qin *et al*. [29] proposed a new attack method by additionally injecting worst-case perturbations at each step to avoid overfitting to the surrogate model.

3. Methodology

3.1. Preliminaries of UAP

Problem formulation. The objective of universal adversarial attack is to craft a single perturbation δ to fool the target model f for most sample $x_i \in X$. In this way, we can consider the following optimization problem,

$$\begin{aligned} \arg\max_{\delta} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x_i+\delta\right), y_i\right), \\ s.t. \left\|\delta\right\|_{\infty} \leqslant \epsilon, \end{aligned} \tag{1}$$

where $f\left(x\right)$ represents the output of the model f with the input x, y_i is the corresponding label of the input x_i calculated by $y_i = \arg\max f\left(x_i\right)$. $\mathcal{L}\left(\cdot\right)$ denotes the adversarial loss used for generating adversarial perturbations. ϵ limits the maximum deviation of UAP.

SPGD. Among existing adversarial attack methods, PGD [19, 32] is considered to be a strong and efficient method. To accelerate the generation of UAP, Shafihi *et al*. [33] first combined the stochastic gradient method with the PGD attack method to solve the above optimization problem, termed SPGD by:

$$\tilde{g}_k = \frac{1}{|B_k|} \sum_{x_i \in B_k} \nabla_{\delta} \mathcal{L}\left(x_i+\delta_k\right), \tag{2}$$

$$\delta_{k+1} = \delta_k + \alpha \cdot \operatorname{sign}\left(\tilde{g}_k\right), \tag{3}$$

where B_k is a batch of samples in the data set, α is a step size and $\operatorname{sign}\left(\cdot\right)$ denotes the sign function.

Loss function. Two widely used adversarial loss functions are the cross-entropy (CE) loss [40] and the logit loss [49], which will be both implemented in later experiments. To prevent the infinite cross-entropy loss value of a single sample from dominating the optimization of the entire objective function, we use the clipped version [33] of cross-entropy loss $\tilde{\mathcal{L}}_{\mathrm{CE}} = \min\left\{\mathcal{L}_{\mathrm{CE}}, \beta\right\}$, where β represents the threshold for cross-entropy loss.

3.2. Stochastic gradient aggregation

3.2.1 Gradient vanishing

In UAP attacks, gradient instability is a normal issue due to the sample diversity and sign is a regular operation for quickly crafting adversarial examples. However, we discover that the sign operations will lead to huge optimization errors when the iterative gradients are strongly unstable. To illustrate this phenomenon, consider a simple toy example. Let $\tilde{g}_m$ and $\tilde{g}_{m+1}$ be gradients at iteration m and $m+1$:

$$\tilde{g}_m = [\cdots, \underline{-0.01}, 0.10, 0.05, \underline{0.70}, \cdots]^{\mathrm{T}}, \tag{4}$$

$$\tilde{g}_{m+1} = [\cdots, \underline{1.00}, 0.02, 0.30, \underline{-0.01}, \cdots]^{T}. \tag{5}$$

When using sign on the gradients for accumulating perturbation δ, large value for the right region of $\tilde{g}_m$ is eliminated by the small negative value of $\tilde{g}_{m+1}$. Similarly in the left region, the sign operations lead to a significant deviation in the current optimization direction:

$$\operatorname{sign}\left(\tilde{g}_m\right) = [\cdots, \underline{-1}, 1, 1, \underline{1}, \cdots]^{T}, \tag{6}$$

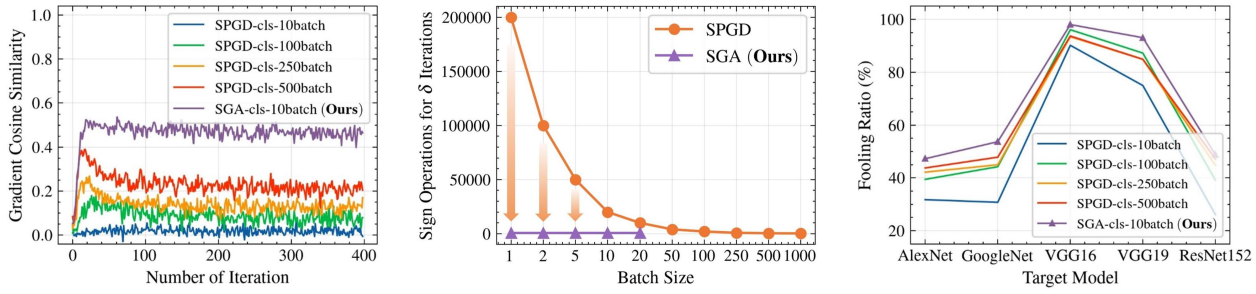
$$\operatorname{sign}\left(\tilde{g}_{m+1}\right) = [\cdots, \underline{1}, 1, 1, \underline{-1}, \cdots]^{T}, \tag{7}$$

$$\begin{aligned} \delta &= \alpha \cdot \operatorname{sign}\left(\tilde{g}_m\right) + \alpha \cdot \operatorname{sign}\left(\tilde{g}_{m+1}\right) \\ &= \alpha \cdot [\cdots, \underline{0}, 2, 2, \underline{0}, \cdots]^{T}. \end{aligned} \tag{8}$$

Here we refer to this phenomenon as gradient vanishing, which is caused by the combination of gradient instability and sign operations.

3.2.2 Large-batch or small-batch

Since the optimization problem for the generation of UAP can be solved by stochastic gradient algorithm with mini-batch training, it is easy to find using large-batch methods can effectively cope with gradient vanishing. However, in the deep learning tasks, recent works [13, 10, 20] verify that the large-batch methods tend to be attracted to sharp minima leading to the large generalization gap. Similarly, when UAP locates at a sharp local extremum in large-batch optimization, the difference in samples and models will result in



(a) (b) (c)

Figure 3. Analysis of two issues, *i.e.*, gradient instability and quantization error under different batch sizes in stochastic gradient universal attacks. (a) represents the gradient cosine similarity between iterations; (b) represents the number of times the sign operator is used for the updates of UAP; (c) represents the average fooling ratio of five models. The gradient cosine similarity is used to describe the gradient stability and is obtained by calculating the cosine value between the forward gradient and the backward gradient, with a larger cosine similarity indicating a more stable gradient update direction.

a significant change in attack loss, making the UAP unable to generalize to unknown samples and models. The small-batch optimization has the advantage of escaping from poor local optima by exploiting the inherent noise in the gradient estimation. However, as shown in Fig 2 (a) and (c), we empirically find that, it will further deteriorate two issues, *i.e.*, gradient instability and quantization error, which will cause severe gradient vanishing, as illustrated in Section 3.2.1.

To further illustrate the impact of gradient instability and quantization error, we explore the relationship between two issues and attack performance under different batch sizes. First, in Fig 3 (a), the stability between the gradients obtained by small-batch training in the regular UAP attacks is poor. The large-batch methods can effectively stable the gradient update directions. Second, Fig 3 (b) illustrates that the sign operators are much more frequently used for the UAP iteration in small-batch methods than in large-batch methods, indicating that it is easier to accumulate larger quantization errors. Furthermore, Fig 3 (c) suggests that by enhancing gradient stability and reducing the usage of sign, large-batch methods can mitigate the gradient vanishing to improve the attack performance, but optimization with over large batch size also inhibits the generalization of UAP. In this way, both the gradient vanishing in small-batch training and the optimization problem in large-batch training will make the generalization of UAP in trouble.

3.2.3 Attack algorithms

As discussed above, our target is to enhance gradient stability and reduce quantization errors when optimizing with small-batch methods.

In this paper, instead of using the average gradient of a batch sample in conventional UAP attacks, a simple strategy is to aggregate multi-step noisy forward gradients for a one-step quantization update at each iteration. Specifically, we first randomly select multiple small-batch samples $\boldsymbol{x}_m^{\mathrm{SB}}$

Algorithm 1: The SGA attack algorithm

input : A surrogate model f, loss function $\mathcal{L}$
input : The training image set $\boldsymbol{X}$, large-batch $\boldsymbol{x}^{\mathrm{LB}}$, small-batch $\boldsymbol{x}^{\mathrm{SB}}$
input : Maximum perturbation magnitude ϵ, number of epochs T, step size α
output: A universal adversarial perturbation δ

Initialize $\delta = 0$;
for $t = 0$ *to* $T-1$ **do**
  for $\boldsymbol{x}^{\mathrm{LB}} \in \boldsymbol{X}$ **do**
    $\delta_0^{\mathrm{inner}} = \delta$;
    $g^{\mathrm{Aggs}} = 0$;
    for $m = 0$ *to* $M-1$ **do**
      Random select $\boldsymbol{x}_m^{\mathrm{SB}} \in \boldsymbol{x}^{\mathrm{LB}}$;
      $\tilde{g}_m = \frac{1}{|\boldsymbol{x}^{\mathrm{SB}}|} \nabla_\delta \mathcal{L}\left(\boldsymbol{x}_m^{\mathrm{SB}} + \delta_m^{\mathrm{inner}}\right)$;
      $\delta_{m+1}^{\mathrm{inner}} = \mathrm{Clip}_\delta^\epsilon\left(\delta_m^{\mathrm{inner}} + \alpha \cdot \mathrm{sign}\left(\tilde{g}_m\right)\right)$;
      $g^{\mathrm{Aggs}} \leftarrow g^{\mathrm{Aggs}} + \tilde{g}_m$;
    end
    $\delta \leftarrow \mathrm{Clip}_\delta^\epsilon\left(\delta + \alpha \cdot \mathrm{sign}\left(g^{\mathrm{Aggs}}\right)\right)$;
  end
end
return δ.

from a large-batch set $\boldsymbol{x}^{\mathrm{LB}}$ to perform pre-search by updating the inner adversarial perturbation $\delta_m^{\mathrm{inner}}$:

$$\tilde{g}_m = \frac{1}{|\boldsymbol{x}^{\mathrm{SB}}|} \nabla_\delta \mathcal{L}\left(\boldsymbol{x}_m^{\mathrm{SB}} + \delta_m^{\mathrm{inner}}\right), \tag{9}$$

$$\delta_{m+1}^{\mathrm{inner}} = \mathrm{Clip}_\delta^\epsilon\left(\delta_m^{\mathrm{inner}} + \alpha \cdot \mathrm{sign}\left(\tilde{g}_m\right)\right). \tag{10}$$

Then, we accumulate the stochastic gradients $\tilde{g}_m$ for updating the aggregated gradient g^{Aggs}:

$$g^{\mathrm{Aggs}} \leftarrow g^{\mathrm{Aggs}} + \tilde{g}_m. \tag{11}$$

After aggregating all the inner gradients into a one-step gradient of the outer iteration, we update the outer adversarial perturbation δ using the aggregated gradient:

$$\delta \leftarrow \mathrm{Clip}_{\delta}^{\epsilon}\left(\delta + \alpha \cdot \mathrm{sign}\left(g^{\mathrm{Aggs}}\right)\right), \tag{12}$$

where α is a step size and $\mathrm{Clip}(\cdot)$ operation constrains the perturbation amplitude under the l_∞ norm.

By means of accumulating more important gradient information, the iterative gradient estimate items are more accurate with a small variance while greatly decreasing quantization operations, thus suppressing gradient vanishing. Moreover, the existence of noise gradients can help escape from poor local optima. The algorithm of stochastic gradient aggregation is summarized in the Algorithm 1.

In short, our method differs from SPGD in that SGA has inner iterations, where SGA obtains multiple noisy gradients through M updates. Model transferability methods in instance-specific adversarial attacks (*e.g.* Momentum [5], Nesterov accelerated gradient [17], *etc.*) can be easily integrated with SGA in the inner iteration.

Method	AlexNet	GoogleNet	VGG16	VGG19	ResNet152	Average
UAP [22]	93.30	78.90	78.30	77.80	84.00	82.46
SV-UAP [14]	-	-	52.00	60.00	-	56.00
NAG [24]	96.44	90.37	77.57	83.78	87.24	87.08
GAP [28]	-	82.70	83.70	80.10	-	82.17
DF-UAP [44]	96.17	88.94	94.30	94.98	90.08	92.89
Cos-UAP [45]	96.50	90.50	97.40	96.40	90.20	94.20
AT-UAP [16]	97.01	90.82	97.51	97.56	91.52	94.88
Ours	**97.43**	**92.12**	**98.36**	**97.69**	**94.04**	**95.93**

Table 1. The fooling ratio (%) in the white-box setting by various UAP attack methods. The UAPs are crafted on five normally trained models, *i.e.*, AlexNet, GoogleNet, VGG16, VGG19, and ResNet152.

4. Experiment

4.1. Experimental setup

Dataset. Following [22], we randomly choose 10 images from each category in the ImageNet training set [31], a total of 10,000 images, for the generation of UAP. Then we evaluate our method on the ImageNet validation set which contains 50,000 images.

Models. We use five normally trained models including AlexNet [9], Googlenet [35], VGG-16 [34], VGG-19 [34], and ResNet152 [8].

Evaluation metrics. To effectively evaluate the attack performance of our method, we report the fooling ratio metric which is most widely implemented in the UAP tasks [22, 28, 24, 44, 45, 16]. The fooling ratio is obtained by calculating the proportion of samples with labels changes when applying UAP.

Baselines. The proposed method is compared with the following UAP methods in the white-box attack scenario: UAP [22], SV-UAP [14], NAG [24], GAP [28], DF-UAP [44], Cos-UAP [45] and AT-UAP [16]. In the black-box attack setting, we regard SPGD [33] as our baseline, and also consider other regular methods, *i.e.*, UAP and GAP. To evaluate our method combined with the momentum method [5], we further compare with the state-of-the-art method, AT-UAP which adopts the Adam optimizer.

Hyper-parameters. For fair comparison with previous works [22, 28, 24, 44, 45, 16], we set the maximum perturbation $\epsilon = 10/255$. For SPGD, we follow the setting in [33] with the maximum cross-entropy loss $\beta = 9$, and the step size $\alpha = 1/255$. Moreover, the number of epochs for the SPGD and SGA is set to 20. To ensure the same number of perturbation iterations, the batch size of SPGD and the outer large batch size $|\boldsymbol{x}^{\mathrm{LB}}|$ in SGA are both set to 250. For SGA, the inner small batch size of $|\boldsymbol{x}^{\mathrm{SB}}|$ for AlexNet, GoogleNet, and the other three models is set to 1, 2, and 10 respectively, and the inner iteration number M is set to twice the number of $|\boldsymbol{x}^{\mathrm{LB}}|/|\boldsymbol{x}^{\mathrm{SB}}|$ for AlexNet and GoogleNet, and four times in the remaining three models. In the momentum methods, the decay factor is set to 0.9.

4.2. Generalization performance of UAP

We first perform universal adversarial attacks under the white-box and black-box settings respectively and evaluate the overall performance of our proposed SGA with baselines on the ImageNet validation set.

White-Box Setting. The results on five models of various UAP generation methods in the white-box scenario are depicted in Table 1. In addition to our method, the results of UAP, SV-UAP, NAG, GAP, DF-UAP, Cos-UAP, and AT-UAP are reported as in the original papers. We can observe that our method achieves the highest attack performance across all the models. For stronger models with deeper networks, the advantage of our proposed method is even clearer, *e.g.* improvement of over 2% in the ResNet152 model. The results demonstrate that the UAPs generated by our method can generalize better to unknown samples.

Black-Box Setting. We also evaluate the proposed method with regular comparison methods, *i.e.*, UAP, GAP, and SPGD in the black-box scenario. Both SPGD and SGA methods implement two different loss functions, *i.e.*, logit loss and cross-entropy loss, namely SPGD-logit, SPGD-cls, SGA-logit, and SGA-cls respectively. We craft UAPs on normally trained models and evaluate them on all the five networks. As shown in Table 2, the SGA can significantly improve the fooling ratio across all the models on both loss functions. For the UAPs crafted on the AlexNet model, the average fooling ratio increases from 54.96% and 56.38 % to 60.21% and 61.71%, respectively. Furthermore, the average fooling ratio achieved by our methods outperforms comparison UAP attack methods by 3.60% $\sim$ 19.28% on average, which verifies SGA method can effectively en-

Model	Method	AlexNet	GoogleNet	VGG16	VGG19	ResNet152	Average
AlexNet	UAP	86.53*	27.82	37.67	35.47	20.99	41.70
	GAP	89.06*	33.05	52.02	48.60	28.70	50.29
	SPGD-logit	95.23*	37.68	57.62	53.86	30.39	54.96
	SGA-logit (**Ours**)	**96.60***	**46.18**	**63.82**	**59.52**	**34.95**	**60.21**
	SPGD-cls	95.97*	42.78	58.59	54.03	30.52	56.38
	SGA-cls (**Ours**)	**97.23***	**48.97**	**66.46**	**60.60**	**35.29**	**61.71**
GoogleNet	UAP	44.63	74.51*	52.92	52.63	32.74	51.48
	GAP	56.55	76.60*	69.65	67.59	46.29	63.34
	SPGD-logit	53.73	**88.88***	77.13	74.88	54.21	69.77
	SGA-logit (**Ours**)	**65.92**	86.54*	**80.85**	**78.18**	**56.53**	**73.60**
	SPGD-cls	50.10	**90.70***	76.47	73.88	51.65	68.56
	SGA-cls (**Ours**)	**61.34**	90.64*	**81.11**	**79.06**	**55.69**	**73.57**
VGG16	UAP	33.35	33.51	76.73*	64.14	29.39	47.42
	GAP	22.33	42.50	82.21*	76.30	29.46	50.56
	SPGD-logit	39.20	54.12	93.56*	86.78	48.78	64.49
	SGA-logit (**Ours**)	**42.74**	**57.58**	**95.01***	**89.99**	**52.25**	**67.51**
	SPGD-cls	42.17	44.97	93.45*	84.89	44.77	62.05
	SGA-cls (**Ours**)	**47.26**	**53.70**	**98.04***	**93.08**	**48.78**	**68.17**
VGG19	UAP	34.45	35.21	65.46	77.79*	28.49	48.28
	GAP	**52.71**	49.11	75.08	79.11*	35.21	58.24
	SPGD-logit	44.47	54.94	86.63	93.87*	48.15	65.61
	SGA-logit (**Ours**)	**47.20**	**57.53**	**91.33**	**96.07***	**50.13**	**68.45**
	SPGD-cls	43.15	47.51	84.62	92.94*	42.09	62.06
	SGA-cls (**Ours**)	**46.53**	**51.53**	**93.35**	**97.39***	**45.47**	**66.86**
ResNet152	UAP	40.59	44.89	64.00	60.50	79.39*	57.87
	GAP	47.74	52.43	64.41	63.41	67.76*	59.15
	SPGD-logit	43.02	54.13	76.47	74.40	90.83*	67.77
	SGA-logit (**Ours**)	**47.07**	**56.91**	**78.84**	**77.86**	**93.27***	**70.79**
	SPGD-cls	46.48	53.86	76.12	73.15	90.75*	68.07
	SGA-cls (**Ours**)	**50.12**	**61.61**	**80.98**	**78.29**	**93.16***	**72.83**

Table 2. The fooling ratio (%) on five models in the black-box setting by regular UAP attack methods. The UAPs are crafted on AlexNet, GoogleNet, VGG16, VGG19, and ResNet152 respectively. * indicates the white-box model.

Method	# samples	AlexNet	GoogleNet	VGG16	VGG19	ResNet152	Average
UAP	500	57.33	16.61	25.29	25.04	19.11	28.68
GAP	500	86.89	57.07	70.40	65.89	47.58	65.57
SPGD	500	92.35	41.68	81.70	75.74	23.44	62.98
SGA (**Ours**)	500	**94.03**	**68.33**	**89.83**	**88.70**	**52.12**	**78.60**

Table 3. The fooling ratio (%) on five models in the limit-sample setting by regular UAP attack methods. The UAPs are crafted on AlexNet, GoogleNet, VGG16, VGG19, and ResNet152 respectively.

hance the cross-model generalization ability of UAP. The visualization results are provided in the supplementary.

4.3. UAPs via different number of training samples

We further investigate the generation of effective UAPs via different numbers of training samples. In the following experiments, without otherwise stated, the clipped cross-entropy loss is used in SPGD and SGA methods.

Limit-Sample Setting. Considering that the acquisition of a large number of training samples is difficult in some tasks, we randomly select 500 training samples from the ImageNet training set to simulate the limit-sample scenario. In Table 3, the fooling ratio across five models achieved by our method remarkably outperforms other comparison methods, showing that SGA can fully exploit the potential of limited samples. It is worth mentioning that the proposed method reaches about 90% fooling ratio on AlexNet, VGG16, and VGG19 models, which is close to the attack performance achieved on 10,000 training samples.

Diverse-Sample Setting. Furthermore, we explore the effect of the number of training samples on the attack performance. The results are shown in Fig 4. Compared with the baseline attack, our method can promote the improvement of the generalization ability of UAP under different numbers of training samples. However, when the number of samples reaches a certain number, the attack performance does not increase anymore correspondingly. The phenomenon reflects the limitations of excessively increas-

Model	Method	AlexNet	GoogleNet	VGG16	VGG19	ResNet152	Average
AlexNet	AT-UAP	97.01*	47.31	62.37	57.72	33.40	59.56
	M-SPGD	96.95*	43.29	62.26	56.61	31.91	58.20
	M-SGA (**Ours**)	**97.43***	**49.71**	**66.41**	**60.96**	**35.76**	**62.05**
	N-SPGD	96.96*	39.87	60.27	56.00	29.90	56.60
	N-SGA (**Ours**)	**97.35***	**47.93**	**65.74**	**60.17**	**35.00**	**61.24**
GoogleNet	AT-UAP	55.90	90.82*	78.71	76.01	54.49	71.19
	M-SPGD	53.06	**92.20***	78.43	76.05	54.62	70.87
	M-SGA (**Ours**)	**62.56**	92.12*	**83.62**	**82.11**	**59.09**	**75.90**
	N-SPGD	53.55	92.23*	77.93	76.03	53.60	70.67
	N-SGA (**Ours**)	**63.43**	**91.67***	**84.23**	**82.17**	**58.41**	**75.98**
VGG16	AT-UAP	45.58	53.63	97.51*	91.53	47.16	67.08
	M-SPGD	44.30	49.19	97.43*	91.06	43.51	65.10
	M-SGA (**Ours**)	**49.02**	**55.78**	**98.36***	**94.17**	**49.02**	**69.27**
	N-SPGD	44.49	49.16	97.39*	90.55	42.45	64.81
	N-SGA (**Ours**)	**48.74**	**55.61**	**98.36***	**93.98**	**50.17**	**69.37**
VGG19	AT-UAP	46.04	52.58	93.49	97.56*	43.53	66.64
	M-SPGD	44.89	45.26	91.25	96.36*	42.47	64.04
	M-SGA (**Ours**)	**50.67**	**56.87**	**95.52**	**97.69***	**51.08**	**70.37**
	N-SPGD	45.27	52.18	93.48	96.82*	46.46	66.84
	N-SGA (**Ours**)	**50.77**	**56.95**	**95.58**	**97.73***	**51.78**	**70.56**
ResNet152	AT-UAP	47.33	61.32	**81.93**	78.72	91.52*	72.16
	M-SPGD	48.84	58.84	79.46	76.37	92.83*	71.27
	M-SGA (**Ours**)	**51.59**	**64.05**	81.77	**79.01**	**94.04***	**74.09**
	N-SPGD	49.00	58.01	78.65	76.49	92.87*	71.00
	N-SGA (**Ours**)	**51.50**	**65.29**	**83.02**	**79.41**	**93.92***	**74.63**

Table 4. The fooling ratio (%) on five models in the single model setting by gradient-based UAP attack methods enhanced by Momentum and Nesterov accelerated gradient methods respectively. * indicates the white-box model.

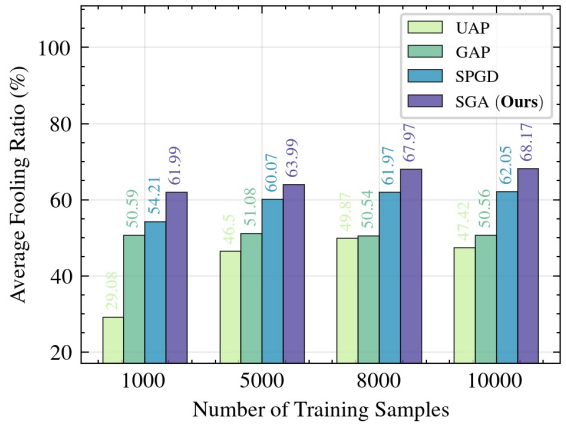


Figure 4. Average fooling ratio (%) on five models versus the number of training samples. The UAPs are crafted by SPGD and SGA methods on the VGG16 model.

ing the number of training samples.

4.4. Combination with transferability methods

In adversarial attacks, Momentum [5] and Nesterov accelerated gradient [17] are considered as the baselines to improve model transferability and can be well integrated with other methods. Therefore, we further incorporate SPGD and SGA with these two baseline methods, denoted as M-SPGD, N-SPGD, M-SGA, and N-SGA, under single-model and ensemble-model settings [18] respectively.

Single-Model Setting. In the integrated momentum methods, we report the results in the original paper of the current state-of-the-art attack method, AT-UAP. Experimental results on the ImageNet validation set are listed in Table 4. We can see that our method outperforms the baseline method by a large margin on all black-box models, while also improving the attack performance of white-box attacks. In general, the SGA-based method consistently outperforms the baseline method by 2.82% $\sim$ 6.33% and exceeds the current state-of-the-art method in attack performance.

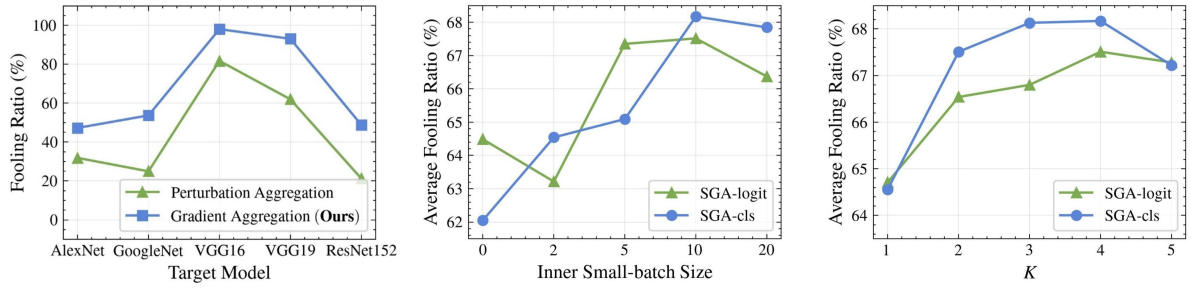
Ensemble-Model Setting. Here we implement the model ensemble method in [5] which averages the loss functions of two normally trained models, *i.e.*, AlexNet and VGG16. Except for the five widely used models, we also test the transferability on the ResNet50. The results are summarized in Table 5. The SGA-based methods can greatly improve the attack performance of the baselines across all models, achieving the average fooling ratio from 74.1%, 74.38%, 74.06% to 76.31%, 77.10% and 77.11% respectively. Such compelling results verify the effectiveness of our method in combination with gradient optimization and model ensemble methods for improving transferability.

4.5. Ablation study

In this subsection, we conduct a series of ablation experiments to explore the impact of the gradient aggregation and

Model	Method	AlexNet	VGG16	GoogleNet	VGG19	ResNet50	ResNet152	Average
Ensemble	SPGD	93.77*	94.28*	63.47	86.45	58.45	48.39	74.10
	SGA (**Ours**)	**94.54***	**95.50***	**66.16**	**88.54**	**62.06**	**51.04**	**76.31**
	M-SPGD	93.91*	95.21*	61.05	86.92	60.49	48.72	74.38
	M-SGA (**Ours**)	**94.65***	**96.03***	**66.83**	**90.00**	**62.72**	**52.39**	**77.10**
	N-SPGD	93.94*	95.31*	60.41	87.33	59.39	47.99	74.06
	N-SGA (**Ours**)	**94.20***	**96.19***	**64.49**	**89.21**	**65.23**	**53.36**	**77.11**

Table 5. The fooling ratio (%) on five models in the ensemble model setting by gradient-based UAP attack methods enhanced by Momentum and Nesterov accelerated gradient methods respectively. The UAPs are crafted on the ensemble models, *i.e.*, AlexNet and VGG16.



(a) (b) (c)

Figure 5. Ablation study on gradient aggregation, inner small-batch size and inner iteration number. (a) represents the average fooling ratio (%) of five models using different types of aggregation methods, *i.e.*, perturbation aggregation and gradient aggregation. (b) represents the average fooling ratio (%) of five models with different inner batch sizes. (c) represents the average fooling ratio (%) of five models with different inner iteration numbers. The UAPs are crafted by the VGG16 model under SGA.

the hyper-parameters for inner small-batch size and inner iteration number. All the UAPs are generated on VGG16 and evaluated on all the five widely used models. The results for the remaining models are provided in the supplementary.

On the gradient aggregation. To demonstrate that the gain of SGA is not by increasing the number of gradient calculations, we report the results of SGA with two types of aggregation methods, *i.e.*, perturbation aggregation and gradient aggregation. Perturbation aggregation is to update δ by using gradients without accumulation as in SPGD, which can be easily achieved by utilizing the inner iterative perturbation δ_M^{inner} to update UAP by $\delta \leftarrow \delta_M^{\text{inner}}$. The results are shown in Fig 5 (a). The gradient aggregation method achieves remarkably better attack performance than the perturbation aggregation method, indicating the effectiveness of gradient aggregation to mitigate the gradient vanishing.

On the inner small-batch size. We then investigate the impact of inner batch size $|\boldsymbol{x}^{\text{SB}}|$ on the attack performance of SGA. We try different inner batch sizes of 0, 2, 5, 10, and 20 respectively. When the inner batch size is set to 0, SGA degrades to SPGD. As shown in Fig 5 (b), the attack performance is excellent when $|\boldsymbol{x}^{\text{SB}}| = 10$. An appropriate batch can help enhance the generalization of UAP.

On the inner iteration number. After determining the inner batch size, we continue to explore the effect of the inner iteration number M. We set the inner iteration number as $M = K \times |\boldsymbol{x}^{\text{LB}}| \,/\, |\boldsymbol{x}^{\text{SB}}|$, where K represents the average number of times each image is traversed in one inner iteration. Fig 5 (c) shows that with the increase of the iteration number, the attack performance reaches the peak when $K = 4$. Similarly, an appropriate iteration number can help enhance the generalization of UAP. Besides, the discussion of the time consuming are provided in the supplementary.

5. Conclusion

In this paper, we propose a stochastic gradient aggregation (SGA) method to cope with the two main issues, *i.e.*, gradient instability and quantization error in the existing UAP attack methods with mini-batch training. Specifically, we first perform multiple rounds of inner pre-search by implementing small-batch optimization. Then SGA aggregates all the gradients of the inner iterations as a one-step gradient estimate and updates the outer adversarial perturbations by using the aggregated gradients. Extensive experiments demonstrate that the proposed method can significantly enhance the generalization ability of the baseline methods in various settings and reach an average fooling ratio of 95.95% exceeding the state-of-the-art methods.

6. Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant No.62276030 and 62236003, in part by BUPT Excellent Ph.D. Students Foundation No.CX2023111 and supported by Program for Youth Innovative Research Team of BUPT No.2023QNTD02.

References

[1] Zachary Berger, Parth Agrawal, Tian Yu Liu, Stefano Soatto, and Alex Wong. Stereoscopic universal perturbations across different architectures and datasets. In *CVPR*, pages 15180–15190, 2022. 1

[2] Yaya Cheng, Jingkuan Song, Xiaosu Zhu, Qilong Zhang, Lianli Gao, and Heng Tao Shen. Fast gradient non-sign methods. *arXiv preprint arXiv:2110.12734*, 2021. 2

[3] Kenneth T Co, Luis Muñoz-González, Leslie Kanthan, Ben Glocker, and Emil C Lupu. Universal adversarial robustness of texture and shape-biased models. In *ICIP*, pages 799–803. IEEE, 2021. 3

[4] Wenjie Ding, Xing Wei, Rongrong Ji, Xiaopeng Hong, Qi Tian, and Yihong Gong. Beyond universal person re-identification attack. *IEEE transactions on information forensics and security*, 16:3442–3455, 2021. 1

[5] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *CVPR*, pages 9185–9193, 2018. 1, 3, 5, 7

[6] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *CVPR*, pages 4312–4321, 2019. 1

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015. 1, 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1, 5

[9] Geoffrey E Hinton, Alex Krizhevsky, and Ilya Sutskever. Imagenet classification with deep convolutional neural networks. *NIPS*, 25(1106-1114):1, 2012. 1, 5

[10] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *NIPS*, 30, 2017. 3

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1

[12] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018. 1

[13] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017. 2, 3

[14] Valentin Khrulkov and Ivan Oseledets. Art of singular vectors and universal adversarial perturbations. In *CVPR*, pages 8562–8570, 2018. 1, 2, 5

[15] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018. 1, 2

[16] Maosen Li, Yanhua Yang, Kun Wei, Xu Yang, and Heng Huang. Learning universal adversarial perturbation by adversarial example. In *AAAI*, 2022. 1, 3, 5

[17] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *ICLR*, 2020. 1, 3, 5, 7

[18] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017. 1, 7

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 1, 2, 3

[20] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018. 3

[21] Alberto G Matachana, Kenneth T Co, Luis Muñoz-González, David Martinez, and Emil C Lupu. Robustness and transferability of universal attacks on compressed models. *arXiv preprint arXiv:2012.06024*, 2020. 3

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *CVPR*, pages 1765–1773, 2017. 1, 2, 5

[23] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, pages 2574–2582, 2016. 1, 2

[24] Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R Venkatesh Babu. Nag: Network for adversary generation. In *CVPR*, pages 742–751, 2018. 3, 5

[25] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017. 1

[26] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016. 1, 2

[27] Zirui Peng, Shaofeng Li, Guoxing Chen, Cheng Zhang, Haojin Zhu, and Minhui Xue. Fingerprinting deep neural networks globally via universal adversarial perturbations. In *CVPR*, pages 13430–13439, 2022. 1

[28] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In *CVPR*, pages 4422–4431, 2018. 1, 3, 5

[29] Zeyu Qin, Yanbo Fan, Yi Liu, Li Shen, Yong Zhang, Jue Wang, and Baoyuan Wu. Boosting the transferability of adversarial attacks with reverse adversarial perturbation. *arXiv preprint arXiv:2210.05968*, 2022. 1, 3

[30] Arianna Rampini, Franco Pestarini, Luca Cosmo, Simone Melzi, and Emanuele Rodola. Universal spectral adversarial attacks for deformable shapes. In *CVPR*, pages 3216–3226, 2021. 1

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 5

[32] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *NIPS*, 32, 2019. 3

[33] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. Universal adversarial training. In *AAAI*, volume 34, pages 5636–5643, 2020. 1, 3, 5

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 5

[35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 1, 5

[36] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014. 1, 2

[37] Jiafeng Wang, Zhaoyu Chen, Kaixun Jiang, Dingkang Yang, Lingyi Hong, Yan Wang, and Wenqiang Zhang. Boosting the transferability of adversarial attacks with global momentum initialization. *arXiv preprint arXiv:2211.11236*, 2022. 1

[38] Xiaosen Wang and Kun He. Enhancing the transferability of adversarial attacks through variance tuning. In *CVPR*, pages 1924–1933, 2021. 1, 3

[39] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *IJCAI*, 2018. 1, 2

[40] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In *CVPR*, pages 2730–2739, 2019. 1, 3

[41] Yi Xie, Zhuohang Li, Cong Shi, Jian Liu, Yingying Chen, and Bo Yuan. Enabling fast and universal audio adversarial attack using generative model. In *AAAI*, volume 35, pages 14129–14137, 2021. 1

[42] Yifeng Xiong, Jiadong Lin, Min Zhang, John E Hopcroft, and Kun He. Stochastic variance reduced ensemble adversarial attack for boosting the adversarial transferability. In *CVPR*, pages 14983–14992, 2022. 1

[43] Chaoning Zhang, Philipp Benz, Tooba Imtiaz, and In-So Kweon. Cd-uap: Class discriminative universal adversarial perturbation. In *AAAI*, volume 34, pages 6754–6761, 2020. 3

[44] Chaoning Zhang, Philipp Benz, Tooba Imtiaz, and In So Kweon. Understanding adversarial examples from the mutual influence of images and perturbations. In *CVPR*, pages 14521–14530, 2020. 3, 5

[45] Chaoning Zhang, Philipp Benz, Adil Karjauv, and In So Kweon. Data-free universal adversarial perturbation and black-box attack. In *ICCV*, pages 7868–7877, 2021. 3, 5

[46] Yuhang Zhang, Weihong Deng, and Liang Zheng. Unsupervised evaluation of out-of-distribution detection: A data-centric perspective. *arXiv preprint arXiv:2302.08287*, 2023. 3

[47] Yuhang Zhang, Chengrui Wang, and Weihong Deng. Relative uncertainty learning for facial expression recognition. *NIPS*, 34:17616–17627, 2021. 3

[48] Yihua Zhang, Guanhua Zhang, Prashant Khanduri, Mingyi Hong, Shiyu Chang, and Sijia Liu. Revisiting and advancing fast adversarial training through the lens of bi-level optimization. In *ICML*, pages 26693–26712. PMLR, 2022. 2

[49] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. On success and simplicity: A second look at transferable targeted attacks. *NIPS*, 34:6115–6128, 2021. 3

[50] Yaoyao Zhong and Weihong Deng. Towards transferable adversarial attack against deep face recognition. *IEEE Transactions on Information Forensics and Security*, 16:1452–1466, 2020. 1

[51] Yaoyao Zhong and Weihong Deng. Opom: Customized invisible cloak towards face privacy protection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1