Adaptive Perturbation for Adversarial Attack

Zheng Yuan*, Student Member, IEEE, Jie Zhang*, Member, IEEE, Zhaoyan Jiang*, Liangliang Li*, and Shiguang Shan*, Fellow, IEEE

Abstract—In recent years, the security of deep learning models achieves more and more attentions with the rapid development of neural networks, which are vulnerable to adversarial examples. Almost all existing gradient-based attack methods use the sign function in the generation to meet the requirement of perturbation budget on $L_\infty$ norm. However, we find that the sign function may be improper for generating adversarial examples since it modifies the exact gradient direction. Instead of using the sign function, we propose to directly utilize the exact gradient direction with a scaling factor for generating adversarial perturbations, which improves the attack success rates of adversarial examples even with fewer perturbations. At the same time, we also theoretically prove that this method can achieve better black-box transferability. Moreover, considering that the best scaling factor varies across different images, we propose an adaptive scaling factor generator to seek an appropriate scaling factor for each image, which avoids the computational cost for manually searching the scaling factor. Our method can be integrated with almost all existing gradient-based attack methods to further improve their attack success rates. Extensive experiments on the CIFAR10 and ImageNet datasets show that our method exhibits higher transferability and outperforms the state-of-the-art methods.

Index Terms—Adversarial attack, transfer-based attack, adversarial example, adaptive perturbation.

I. INTRODUCTION

With the rapid progress and significant success in the deep learning in recent years, its security issue has attracted more and more attention. One of the most concerned security problems is its vulnerability to small, human-imperceptible adversarial noise [1], which implies severe risk of being attacked intentionally especially for technologies like face recognition and automatic driving. While it is important to study how to strengthen deep learning models to defend adversarial attack, it is equally important to explore how to attack these models.

Existing attack methods generate an adversarial example by adding to input some elaborately designed adversarial perturbations, which are usually generated either by a generative network [2], [3], [4], [5], [6], [7] or by the gradient-based optimization [8], [9], [10], [11], [12], [13], [14]. The latter, i.e., gradient-based methods, are the mainstream. Their key idea is generating the perturbation by exploiting the gradient computed via maximizing the loss function of the target task.

In these methods, since the gradient varies in different pixels, the sign function is usually used to normalize the gradient, which is convenient to set the step size of each step during the attack. Under the most commonly used $L_\infty$ norm setting in the adversarial attack, i.e., constraining the maximal $L_\infty$ norm of the generated adversarial perturbations, the use of the sign function can also leverage the largest perturbation budget to enhance the aggressiveness of adversarial examples. Due to the influence of the sign function, all values of the gradient are normalized into $\{0, +1, -1\}$. Although this approach can scale the gradient value, so as to make full use of the perturbation budget in the $L_\infty$ attack, the resultant update directions in adversarial attack are limited, there are only eight possible update directions in the case of a two-dimensional space. The inaccurate update direction may cause the generated adversarial examples to be sub-optimal (as shown in Fig. 1).

To solve the above problem of existing methods, we propose a method called Adaptive Perturbation for Adversarial Attack (APAA), which directly multiplies a scaling factor to the gradient of loss function instead of using the sign function to normalize. The scaling factor can either be elaborately selected manually, or adaptively generated according to different image characteristics. Specifically, to further take the characteristics of different images into consideration, we propose an adaptive scaling factor generator to automatically generate the suitable scaling factor in each attack step during the generation of adversarial examples. From Fig. 1, we can clearly see that, since our method can adaptively adjust the step size in each attack step, a large step size can be used in the first few steps of iterative attacks to make full use of the perturbation budget, and when close to the global optimal point, the step size can be adaptively reduced. With more accurate update direction, our APAA may reach the global optimum with fewer update steps and perturbations, which means the generated adversarial examples are more aggressive and thus can improve the corresponding attack success rate.

Adversarial examples have an intriguing property of transferability, where adversarial examples generated by one model...
the development of each other in recent years, which are briefly reviewed in this section, respectively.

A. Attack Methods

The attack methods mainly consist of the generative-based [2], [3], [4], [5], [6], [7] and gradient-based methods [8], [9], [10], [11], [12], [13], [14], and the latter ones are the mainstream. Our work mainly focuses on the gradient-based attack methods under the setting of $L_\infty$ norm. Before introducing the methods in detail, we first introduce some notations that will be used later. Let $x$ and $y$ be the original image and its corresponding class label, respectively. Let $J(x, y)$ be the loss function of cross-entropy. Let $x_{adv}$ be the generated adversarial example. Let $\epsilon$ and $\alpha$ be the total perturbation budget and the budget in each step of the iterative methods. $\Pi_{x, \epsilon}$ means to clip the generated adversarial examples within the $\epsilon$-neighborhood of the original image on $L_\infty$ norm.

Fast Gradient Sign Method. FGSM [8] is a one-step method for white-box attack, which directly utilizes the gradient of loss function to generate the adversarial example:

$$x_{adv} = x + \epsilon \cdot \text{sign}(\nabla_x J(f(x, y))).$$

Basic Iterative Method. BIM [9] is an extension of FGSM, which uses the iterative method to improve the attack success rate of adversarial examples:

$$x_{adv}^{t+1} = \Pi_{x, \epsilon}(x_{adv}^t + \alpha \cdot \text{sign}(\nabla_x J(f(x_{adv}^t, y))),$$

where $x_{adv}^0 = x$ and the subscript $t$ is the index of iteration.

Momentum Iterative Fast Gradient Sign Method. MIFGS [11] proposes a momentum term to accumulate the gradient in previous steps to achieve more stable update directions, which greatly improves the transferability of generated adversarial examples:

$$g_{t+1} = \mu \cdot g_t + \frac{\nabla_x J(f(x_{adv}^t, y))}{\| \nabla_x J(f(x_{adv}^t, y)) \|},$$

$$x_{adv}^{t+1} = \Pi_{x, \epsilon}(x_{adv}^t + \alpha \cdot \text{sign}(g_{t+1})), \quad (4)$$

where $g_t$ denotes the momentum item of gradient in the $t$-th iteration and $\mu$ is a decay factor.

Since then, various methods have been further proposed to improve the transferability of adversarial examples. A randomization operation of random resizing and zero-padding to the original image is proposed in DIM [12]. TILM [13] proposes a translation-invariant attack method by convolving the gradient with a Gaussian kernel to further improve the transferability of adversarial examples. Inspired by Nesterov accelerated gradient [16], SIM [14] amends the accumulation of the gradients to effectively look ahead and improve the transferability of adversarial examples. In addition, SIM also proposes to use several copies of the original image with different scales to generate the adversarial example. SGM [17] finds that using more gradients from the skip connections rather than the residual modules can craft adversarial examples with higher transferability. VT [18] considers the gradient variance of the previous iteration to tune the current gradient so as to stabilize the update direction and escape from poor local optima. EMI [19] accumulates the

II. RELATED WORK

The phenomenon of adversarial examples is proposed by Szegedy et al. [1]. The attack and defense methods promote
gradients of data points sampled in the gradient direction of previous iteration to find more stable direction of the gradient. IR [15] discovers the negative correlation between the adversarial transferability and the interaction inside adversarial perturbations and proposes to directly penalize interactions during the attacking process, which significantly improves the adversarial transferability. AIFGTM [20] also considers the limitations of the basic sign structure and proposes an ADAM iterative fast gradient tanh method to generate indistinguishable adversarial examples with high transferability.

B. Defense Methods

Adversarial defense aims to improve the robustness of the target model in the case of adversarial examples being the inputs. The defense methods can mainly be categorized into adversarial training, input transformation, model ensemble, and certified defenses. The adversarial training methods [10], [21], [22], [23], [24] use the adversarial examples as the extra training data to improve the robustness of the model. The input transformation methods [25], [26], [27], [28], [29] tend to denoise the adversarial examples before feeding them into the classifier. The model ensemble methods [30], [31], [32] use multiple models simultaneously to reduce the influence of adversarial examples on the single model and achieve more robust results. Certified defense methods [33], [34], [35], [36] guarantee that the target model can correctly classify the adversarial examples within the given distance from the original images.

III. METHOD

In this section, we first analyze the defect of the sign function used in existing gradient-based attack methods in Section III-A. Then we propose our method of Adaptive Perturbation for Adversarial Attack (APAA) in Section III-B, i.e., utilizing a scaling factor to multiply the gradient instead of the sign function normalization. The scaling factor can either be elaborately selected manually (in Section III-B1), or adaptively generated according to different image characteristics by a generator (in Section III-B2). Finally, we theoretically demonstrate that our proposed method can improve the black-box transferability of adversarial examples in Section III-C.

A. Rethinking the Sign Function

The task of adversarial attack is to do a minor modification on the original images with human-imperceptible noises to fool the target model, i.e., misclassifying the adversarial examples. The gradient-based methods generate the adversarial examples by maximizing the cross-entropy loss function, which can be formulated as follows:

$$\arg \max_{x^{adv}} J(f(x^{adv}), y), \quad \text{s.t. } \|x^{adv} - x\|_p \leq \epsilon,$$

where $p$ could be 0,1,2 and $\infty$.

Since the gradient varies in different pixels, the sign function is usually used to normalize the gradient, which is convenient to set the step size of each step during the attack (e.g., (1)). Under the most commonly used $L_{\infty}$ norm setting in the adversarial attack, i.e., constraining the maximal $L_{\infty}$ norm of the generated adversarial perturbations, the use of the sign function can also leverage the largest perturbation budget to enhance the aggressiveness of adversarial examples.

The sign function normalizes all values of the gradient into $\{0, +1, -1\}$. Although this method can scale the gradient value to make full use of the perturbation budget in the $L_{\infty}$ attack, the resultant update directions in adversarial attack are limited, e.g., there are only eight possible update directions in the case of a two-dimensional space ($(0, 1)$, $(0, -1)$, $(1, 1)$, $(1, -1)$, $(1, 0)$, $(-1, 1)$, $(-1, 0)$, $(-1, 1)$).

We use a two-dimensional toy example (as shown in Fig. 1) to demonstrate the limitation of the existing attack methods with the sign function. The loss function in Fig. 1 is a Gaussian mixture model of the following expression:

$$f(x, y) = \exp\{-[(x+2.8)^2 + (y-2)^2 + 0.5(x+2.8)(y-2)]\} + 0.7 \exp\{-[(x-1)^2 + (y-1)^2 - 0.8(x-1)(y-1)]\} - \exp\{-(x+3)^2 + (y+2)^2 - 0.5(x+3)(y+2)\}.$$ (6)

From the figure, we can clearly see that the attack directions in sign-based methods (e.g., BIM [9]) are distracted and not accurate anymore, which conversely needs more update steps and perturbations budgets to implement a successful attack. The inaccurate update direction may cause the generated adversarial examples to be sub-optimal.

B. Adaptive Perturbation for Adversarial Attack

To solve the problem mentioned above, we propose a method of Adaptive Perturbation for Adversarial Attack (APAA). Specifically, we propose to directly multiply the gradient by a scaling factor instead of normalizing it with the sign function. The scaling factor can be determined either elaborately selected manually, or adaptively achieved from a generator according to different image characteristics. We will introduce each of them in the following, respectively.

1) Fixed Scaling Factor: First, we propose to directly multiply the gradient by a fixed scaling factor, which not only maintains the accurate gradient directions but also can flexibly utilize the perturbation budget through the adjustment of the scaling factor:

$$x_{t+1}^{adv} = \Pi_{x, \epsilon}(x_t^{adv} + \gamma \cdot \nabla_x J(f(x_t^{adv}), y)),$$ (7)

where $\gamma$ is the scaling factor, $\Pi_{x, \epsilon}$ means to clip the generated adversarial examples within the $\epsilon$-neighborhood of the original image on $L_{\infty}$ norm.

Our method is easy to implement and can be combined with all existing gradient-based attack methods (e.g., MIFGSM [11], DIM [12], TIM [13], SIM [14]). We take the MIFGSM method as an example. When integrating with MIFGSM, the full update formulation is as follows:

$$x_0^{adv} = x,$$

$$g_{t+1} = \mu \cdot g_t + \frac{\nabla_x J(f(x_t^{adv}), y)}{\|\nabla_x J(f(x_t^{adv}), y)\|_1},$$ (9)

$$x_{t+1}^{adv} = \Pi_{x, \epsilon}(x_t^{adv} + \gamma \cdot g_{t+1}),$$ (10)
where $\mu$ is the decay factor in MIFGSM, $\gamma$ is the scaling factor.

2) Adaptive Scaling Factor: Considering that suitable scaling factors vary across different images, we further propose a generator to directly generate the adaptive scaling factor in each step of adversarial examples generation, which also gets rid of manually hyperparameter searching.

The overview of our proposed adaptive scaling factor generator is shown in Fig. 2. We employ an iterative method to generate the adversarial example with $T$ steps. In each step $t$, the gradient of the adversarial examples generated in the previous step is calculated through a randomly selected white-box model $C_x$. The corresponding scaling factor $\gamma_t$ is also generated through the generator $G_t$ by feeding the adversarial example together with the gradient information as the input. Then the gradient information and scaling factor are used to generate the new adversarial perturbation. We used the BIM-style update method $(x_{t+1}^{adv} = x_t^{adv} + \gamma_t \cdot \nabla_x J(C_x(x_t^{adv}), y))$ as an example in the figure for convenience, but in fact any existing gradient-based attack method can be utilized. The generated adversarial examples in all steps are used to calculate the cross-entropy loss by another classifier $C_y$. Then the parameters in the generator $G_t$ are updated by maximizing the cross-entropy loss through backpropagation, aiming to generate the adversarial examples, which can mislead the classifier. The generators used in each step share the same architecture but do not share the parameters. The procedure of training the generator is summarized in Algorithm 1.

From Fig. 1, we can observe the difference between our proposed APAA method and existing sign-based methods (e.g., BIM [9]) in generating adversarial examples. To show the generating process, we set the initial point at $(-2.900, -1.850)$ for both our APAA and BIM, and the step size for BIM to be 0.5. For our proposed APAA, a single-layer neural network is trained to dynamically generate the adaptive scaling factor, considering the simplicity of this toy example. In the figure, the orange and blue lines depict the process of generating adversarial examples using BIM and APAA, respectively. As can be seen from the figure, BIM finally reaches the point $(1.003, 0.752)$ after 12 steps, approximating a local optimum $(1.000, 1.000)$, while only 9 steps are needed for APAA to reach the point $(-2.730, 2.033)$, very close to the global optimum $(-2.800, 2.000)$. The reason behind is that our APAA enables a more accurate update direction and allows adaptive step size, while the update direction in BIM is con-strained and lacks precision. In our APAA, a large step size can be used in the first few steps of iterative attacks to make full use of the perturbation budget, and when close to the global optimal point, the step size can be adaptively reduced. Without loss of generality, we believe our

![Algorithm 1: Training of Adaptive Scaling Factor Generator](image-url)
APAA has a higher probability of reaching the global optimum in fewer steps under real scenarios.

It is worth noting that two methods proposed above (i.e., APAA with fixed scaling factor and APAA with adaptive scaling factor) are complementary, each with its own distinct advantages. So the attacker can consider to choose which one to use depending on the specific situation. Specifically, as shown in the experiment part, APAA with adaptive scaling factor performs better and can automatically obtain adaptive scaling factors for each step in the attack but requires training a scaling factor generator. However, due to the lightweight structure of the generator (as illustrated in Table II), the extra inference time brought by the generator during the generation of adversarial examples is negligible. On the other hand, APAA with fixed scaling factor is simple, does not require training a generator network, and outperforms existing methods significantly, making it an excellent choice as well. Therefore, attackers can select one of them based on the specific scenario when conducting the attack.

C. Theoretical Analysis

Adversarial examples have an intriguing property of transferability, where adversarial examples generated by one model can also fool other unknown models. In addition to the intuitive idea to illustrate that our method can provide more accurate attack directions, we also provide a theoretical proof to show that our proposed method can meanwhile improve the black-box transferability of adversarial examples. Wang et al. [15] utilizes the Shapley interaction index proposed in game theory [37], [38] to analyze the interactions inside adversarial perturbations. Through extensive experiments, they discover the negative correlation between the transferability and interactions, i.e., the adversarial examples with smaller interactions have the better black-box transferability. In the following, we take MIFGSM [11] as an example to prove that the adversarial examples generated by our APAA method have smaller interaction values, which also confirms that our method has better black-box transferability.

To simplify the proof, we do not consider some tricks in the adversarial attack, such as gradient normalization and the clip operation. The MIFGSM method combined with APAA can be formulated as:

\[ g_t = \mu \cdot g_{t-1} + g(x + \delta_{t-1}), \]

\[ \delta_t = \sum_{i=1}^{t} \gamma \cdot g_t, \]

where \( g(x) = \frac{\partial L(x)}{\partial x} \).

Proposition 1 (The Perturbations Generated by MIFGSM With APAA). The adversarial perturbation generated by MIFGSM with APAA at \( m \)-th step is given as:

\[ g_m = a_m \cdot g + b_m \cdot \gamma H g, \]

\[ \delta_m = c_m \cdot \gamma g + d_m \cdot \gamma^2 H g, \]

where \( g \) and \( H \) are the first and second order gradients of \( L(x) \) with respect to \( x \), respectively,

\[ a_m = \sum_{i=1}^{m} \mu_i^{i-1}, \]

\[ b_m = \sum_{i=1}^{m} (m - i + 1)(i - 1) \mu_i^{i-2}, \]

\[ c_m = \sum_{i=1}^{m} (m - i + 1) \mu_i^{i-1}, \]

\[ d_m = \sum_{i=1}^{m} \frac{(m - i + 2)(m - i + 1)(i - 1)}{2} \mu_i^{i-2}. \]

Detailed proofs are provided in the appendix (Section A), available online.

Further, we can calculate the interaction inside perturbations generated by MIFGSM with APAA.

Proposition 2 (The Interaction Inside Perturbations Generated by MIFGSM With APAA). The interaction inside adversarial perturbations generated by MIFGSM with APAA at \( m \)-th step is given as:

\[ E_{\alpha,b}(I_{ab}; \gamma) = A_\gamma^2 + 2B\gamma^3, \]
We use both normally trained models and that of MIFGSM [11]. Smaller interaction values correspond to better black-box transferability. The experiment is conducted on 1000 images of CIFAR10 testset.

where

\[
A = \mathbb{E}_{a,b}\left(c_{a}^2g_{a}g_{b}H_{ab}\right), \quad B = \mathbb{E}_{a,b}\left(c_{m}d_{m}g_{a}H_{ab}g_{a}^{T}H_{ab}\right) \geq 0,
\]

\(g\) and \(H\) are the first and second order gradients of \(L(x)\) with respect to \(x\), respectively. \(g_{a}\) and \(g_{b}\) are the \(a\)-th and \(b\)-th elements in \(g\), \(H_{ab}\) represents the \(b\)-th column of Hessian matrix \(H\).

Detailed proofs are provided in the appendix (Section B), available online.

Through experiments, we find that the magnitude of gradients obtained by derivation in the MIFGSM method is predominately smaller than \(10^{-2}\). Given that the gradient values are usually small, normalizing the gradient with the sign function can be approximated as multiplying the gradient by a rather large coefficient. When we convert these gradients to \([-1, 1]\) with the sign function, it is equivalent to multiplying by a \(\gamma_{\text{MIFGSM}}\) on the order of greater than \(10^{2}\). On the other hand, as evident from the experiments that follows, our \(\gamma_{\text{APAA}}\) takes values below 10 (0.4 or 0.8 on ImageNet and 8 on CIFAR10). Hence, we can conclude that \(0 < \gamma_{\text{APAA}} \ll \gamma_{\text{MIFGSM}}\). We treat (22) as a cubic function of \(\gamma\), taking into account that coefficient \(B\) is greater than 0, so we can achieve \(\mathbb{E}_{a,b}(I_{ab};\gamma_{\text{APAA}}) < \mathbb{E}_{a,b}(I_{ab};\gamma_{\text{MIFGSM}})\), which means our APAA method have better black-box transferability. Moreover, we also verify our proof by experiments. As shown in Fig. 3, we can clearly see that the interaction inside perturbations generated by our APAA is significantly smaller than that of MIFGSM. Combined with the negative correlation between the adversarial transferability and the interaction inside adversarial perturbations, our proposed APAA method is theoretically proven to have better black-box transferability.

IV. EXPERIMENTS

We first introduce the setting of experiments in Section IV-A. Then we demonstrate the effectiveness of our proposed APAA with fixed scaling factor in Section IV-B. Finally, we conduct several experiments to show the superiority of the adaptive scaling factor generator in Section IV-C. We denote the APAA with the fixed scaling factor as APAA\(_{f}\) and the APAA with the adaptive scaling factor generator as APAA\(_{a}\).

A. Experiment Setting

Datasets. We use ImageNet [39] and CIFAR10 [40] datasets to conduct the experiments. CIFAR10 has 50000 training images and 10000 test images in different classes. For ImageNet, we use two sets of subsets\(^1\),\(^2\) in the ImageNet dataset [39] to conduct experiments. Each set contains 1000 images, covering almost all categories in ImageNet, which has been widely used in previous works. The maximization perturbation budgets on CIFAR10 and ImageNet are 8 and 16 with \(L_{\infty}\) norm under the scale of 0-255, respectively.

Evaluation Models. We use both normally trained models and defense models to evaluate all attack methods. For CIFAR10, we use totally 8 normally trained models and 7 defense models for comprehensive evaluations. For ImageNet, we use totally 9 normally trained models and 12 defense models to evaluate. Specifically, the models used for CIFAR10 includes RegNet [41], Res-18 [42], SENet-18 [43], Dense-121 [44], WideRes\(_{28\times10}\) [45], DPN [46], Pyramid [47], ShakeShake [48], Dense-121\(_{adv}\) [10], GoogLeNet\(_{adv}\) [10], Res-18\(_{adv}\) [10], k-WTA [49], Odds [50], Generative [51] and Ensemble [31]. The models used for ImageNet include IncRes-v2 [52], Inc-v3 [53], Inc-v4 [52], Res-101 [54], Res-152 [54], Mob\(_{1,0}\) [55], Mob\(_{1,4}\) [55], PNAS [56], NAS [57], Inc-v3\(_{adv}\) [21], Inc-v3\(_{ens}\) [21], Inc-v3\(_{ens4}\) [21], IncRes-v2\(_{ens}\) [21], HGD [26], R&P [58], NIPS-r3\(^3\) Bit-Red [59], JPEG [60], FD [28], ComDefend [29] and RS [36].

Metrics. We use the attack success rates on both white-box and black-box models to evaluate the effectiveness of different methods. Since all methods constrain the same \(L_{\infty}\) perturbation budget, we additionally use mean absolute distance (MAD) and root mean square distance (RMSD) to compare the magnitude of perturbations generated by different methods.

Baselines. We use MIFGSM [11], DIM [12], TIM [13], SIM [14], VT [18], EMI [19], AIFGTM [20], SGM [17], IR [15] and SVRE [61] as baselines to compare with our proposed method. The details of the hyperparameters in these methods are provided in Table I. The number of iteration \(T\) in the generation of adversarial examples for all methods, including ours, is 10 unless mentioned. We clip the adversarial examples to the range of the normal image (i.e., 0-255) and constrain the adversarial perturbations within the perturbation budget on \(L_{\infty}\) norm bound in each step of the iteration process.

Details: For the method of fixed scaling factor, we select 10000 training images of CIFAR10 and one subset of ImageNet\(^1\) as the validation set to search for the best scaling factor, and utilize the corresponding scaling factor to conduct the attack and evaluation in the testset of CIFAR10 and the other subset of ImageNet\(^2\), respectively. By searching on the validation set, we set the scaling factor \(\gamma\) as 8 for all methods on CIFAR10. For ImageNet, we use 0.4 in TIM, EMI, SGM, IR, and 0.8 in

\(^1\)[Online]. Available: https://github.com/cleverhans-lab/cleverhans/tree/master/cleverhans_v3.1.0/examples/nips17_adversarial_competition/dataset

\(^2\)[Online]. Available: https://drive.google.com/drive/folders/\%1Cf0bY6I8BFqfWPHL31FKFDipNgjWwAhS

\(^3\)[Online]. Available: https://github.com/anlthms/nips-2017/tree/master/master/cleverhans_v3.10/examples/nips17_adversarial_competition/dataset
methods combined with our proposed APAA, considering different values of $\gamma$ for normally trained and defense models. As $\gamma$ increases, the magnitude of the adversarial perturbation also increases, indicating a more aggressive perturbation. The attack success rate on normally trained models increases accordingly. However, for defense models, especially those obtained through adversarial training, the opposite conclusion is reached. Subtable (b) in Table III reveals that as $\gamma$ increases, the attack success rate of adversarial examples on adversarially trained models gradually decreases. This behavior is attributed to models typically being trained with a fixed perturbation size during adversarial training. The model obtained in this manner exhibits better defense capabilities against moderate perturbations but reduced robustness when facing smaller adversarial perturbations, which are rarely seen during the training time. To strike a balance between achieving high attack success rates on normally trained models and defense models, we conducted experiments on SGM [15] and IR [15] methods in Tables VI and VII, respectively. It demonstrates that our method can well integrate with almost all gradient-based attack methods to improve the attack success rates.

The Untargeted Attack: The experiments of adversarial examples generated on CIFAR10 and ImageNet under the untargeted attack setting are shown in Tables IV and V, respectively. From the results on both the single model and the ensemble model attacks, combined with our proposed method of the scaling factor, the attack success rates of generated adversarial examples in all the state-of-the-art methods are improved. In addition, under the same perturbation budget constraint on $L_\infty$ norm, the MAD and RMSD between the original images and generated adversarial examples on adversarially trained models gradually decrease. This behavior is attributed to models typically being trained with a fixed perturbation size during adversarial training. The model obtained in this manner exhibits better defense capabilities against moderate perturbations but reduced robustness when facing smaller adversarial perturbations, which are rarely seen during the training time. To strike a balance between achieving high attack success rates on normally trained models and defense models, we conducted experiments on SGM [15] and IR [15] methods in Tables VI and VII, respectively. It demonstrates that our method can well integrate with almost all gradient-based attack methods to improve the attack success rates.

The Targeted Attack: The experiments of adversarial examples generated on CIFAR10 under the targeted attack setting are shown in Table VIII. The target label of each image is randomly chosen among the 9 wrong labels. The average black-box attack success rates of the adversarial examples generated by APAA$_f$ against white-box and black-box models are about 10% higher than those of baselines under both settings of the single model attack and the ensemble model attack. It verifies that our proposed scaling factor method is also effective under the targeted attack setting.

The Influence of the Size of Perturbation. We conduct experiments to demonstrate the attack success rates versus perturbation budget curves on CIFAR10 in Fig. 4. We can clearly find that our proposed method of using the scaling factor improves the attack success rates on various perturbation budgets, which shows the excellent generalization of our method.

The Influence of the Number of Attack Steps. We conduct experiments to compare the results of different methods under various settings of attack iteration steps. From Table IX, it is obvious that as the number of attack steps increases, the success rates of all attack methods improve. However, when comparing
TABLE III

| Method | IncRes-v2 | IncRes-v3 | IncRes-v4 | RestNet | ResNet-152 | MobW_18 | MobW_44 | PNAS | NAS | Distance Metric |
|--------|-----------|-----------|-----------|---------|------------|---------|---------|------|----|----------------|
| TIM | 98.3 | 86.5 | 83.9 | 73.3 | 73.8 | 81.4 | 83.7 | 75.1 | 77.3 | 10.302 | 11.107 |
| TIM w/ APAAγ (γ=0.2) | 97.7 | 84.6 | 82.2 | 71.3 | 72.6 | 76.4 | 80.0 | 72.4 | 73.7 | 5.993 | 6.668 |
| TIM w/ APAAγ (γ=0.4) | 98.6 | 88.6 | 85.4 | 75.3 | 76.1 | 83.7 | 87.9 | 77.2 | 80.2 | 8.887 | 9.693 |
| TIM w/ APAAγ (γ=0.6) | 99.0 | 88.8 | 85.7 | 75.7 | 75.9 | 85.7 | 88.2 | 76.8 | 80.3 | 10.497 | 11.288 |
| TIM w/ APAAγ (γ=0.8) | 99.3 | 90.0 | 86.7 | 75.0 | 75.4 | 87.5 | 88.6 | 77.5 | 80.2 | 11.453 | 12.195 |
| TIM w/ APAAγ (γ=1.0) | 99.4 | 89.7 | 86.9 | 75.3 | 75.8 | 85.9 | 88.9 | 79.2 | 80.1 | 12.092 | 12.785 |
| EMI [19] | 99.5 | 93.1 | 93.0 | 88.0 | 87.3 | 90.8 | 93.8 | 89.8 | 92.0 | 10.279 | 11.282 |
| EMI w/ APAAγ (γ=0.2) | 99.6 | 96.0 | 94.3 | 88.7 | 88.4 | 91.4 | 93.8 | 89.7 | 91.2 | 6.335 | 7.074 |
| EMI w/ APAAγ (γ=0.4) | 99.6 | 97.0 | 95.9 | 90.2 | 90.5 | 93.6 | 95.2 | 91.8 | 92.7 | 9.390 | 10.234 |
| EMI w/ APAAγ (γ=0.6) | 99.7 | 97.0 | 95.7 | 90.3 | 90.0 | 94.3 | 96.0 | 91.1 | 93.5 | 10.940 | 11.737 |
| EMI w/ APAAγ (γ=0.8) | 99.9 | 96.9 | 95.8 | 89.5 | 89.6 | 93.8 | 97.3 | 91.8 | 94.3 | 11.843 | 12.576 |
| EMI w/ APAAγ (γ=1.0) | 99.8 | 97.1 | 95.0 | 89.5 | 89.1 | 93.4 | 96.6 | 92.0 | 93.0 | 12.416 | 13.088 |

(a) the evaluation on the normally trained models.

(b) the evaluation on the defense models.

---

**Comparison With Other Alternative Methods to the Sign Function.** To provide a more comprehensive demonstration of the superiority of our approach, which employs a scaling factor to replace the sign function method, we integrate the baseline methods with both the Adam optimizer [62] and the arctanh function, subsequently comparing them with our proposed APAAγ. The results presented in Table X demonstrate that the method utilizing the arctanh function yields unsatisfactory performance, which is even worse than the baselines. On the other hand, the utilization of the Adam optimizer, while showing some improvements over baseline methods, still falls short of matching the performance of our APAAγ. Notably, when combined with SIM, our APAA achieves average improvements attack methods with the same number of steps, our APAAγ consistently achieves higher attack success rates than baselines. Furthermore, when comparing the results of our 10-step APAAγ with the 100-step baseline methods, it can be observed that our APAAγ achieves comparable or even higher attack success rates with fewer attack steps. It confirms our opinion that accurate gradient directions can conduct a successful attack with fewer attack steps. This observation well validates the idea presented in Fig. 1.
TABLE IV
ATTACK SUCCESS RATES OF THE ADVERSARIAL EXAMPLES ON CIFAR10 UNDER UNTARGETED ATTACK SETTING

| Source Model | Method          | RegNet | Res-18 | SENet-18 | Dense-121 | WideResNet-12 | DPN | Pyramid | ShakeShake | Distance Metric |
|--------------|-----------------|--------|--------|----------|-----------|----------------|-----|---------|------------|-----------------|
| **RegNet**   | MIFGSM [11]     | 99.1   | 88.8   | 85.8     | 90.2      | 84.6           | 86.1| 79.3    | 83.1       | 5.470           |
|              | MIFGSM w/ APAAP | 99.8   | 92.7   | 92.3     | 93.8      | 85.8           | 89.9| 82.4    | 87.1       | 5.377           |
|              | DIM [12]        | 98.0   | 91.0   | 90.7     | 91.8      | 88.2           | 88.2| 85.1    | 86.6       | 5.544           |
|              | DIM w/ APAAP    | 99.4   | 94.9   | 94.6     | 95.5      | 92.5           | 92.3| 92.2    | 91.3       | 5.309           |
|              | SIM [14]        | 98.1   | 93.2   | 93.2     | 94.0      | 91.5           | 90.6| 88.2    | 89.6       | 5.606           |
|              | SIM w/ APAAP    | 99.4   | 96.5   | 96.4     | 97.1      | 95.6           | 94.4| 92.7    | 93.5       | 5.343           |
| **RegNet**   | MIFGSM [11]     | 98.0   | 98.5   | 99.5     | 99.0      | 96.1           | 94.5| 94.2    | 95.4       | 5.636           |
|              | MIFGSM w/ APAAP | 99.4   | 99.6   | 100.0    | 99.9      | 96.7           | 97.8| 97.5    | 98.2       | 5.496           |
|              | DIM [12]        | 97.6   | 98.4   | 99.3     | 98.7      | 96.6           | 94.7| 94.8    | 95.9       | 5.668           |
|              | DIM w/ APAAP    | 99.4   | 99.7   | 99.9     | 99.7      | 98.7           | 97.6| 97.9    | 98.5       | 5.427           |
|              | SIM [14]        | 97.8   | 98.6   | 99.4     | 98.9      | 97.3           | 95.6| 95.7    | 96.6       | 5.747           |
|              | SIM w/ APAAP    | 99.3   | 99.7   | 99.9     | 99.8      | 99.3           | 98.1| 98.3    | 98.8       | 5.462           |

(a) The evaluation on the normally trained models.

(b) The evaluation on the defense models.

Fig. 4. Attack success rates versus perturbation budget curve on CIFAR10. The curves with dotted lines are the results of baseline methods, and those with solid lines are the results of our method. Three subfigures are the average attack success rates of different methods on the white-box model, the black-box normally trained models and the defense models, respectively. The experiment chooses RegNet as the white-box model.

5%-10% in attack success rate, which demonstrates the superiority of APAAP. We suppose that the Adam optimizer, originally designed for model parameter optimization, may not be also suitable for generating adversarial examples, which could potentially result in suboptimal results. Similarly, the arctanh function may not be well-suited for the task of generating adversarial examples.

**Comparison With Generative Attack Methods.** We further compare our proposed APAAP with some typical generative attack methods [2], [3]. Table XI demonstrates that in comparison to AdvGAN [2], which shares consistent experimental settings with our study, our APAAP method consistently achieves significantly higher attack success rates with smaller adversarial perturbations. In contrast to NaturalAdversary [3], which conducting attack without perturbation constraints, thus introducing much larger perturbations compared to APAAP, we still attain higher attack success rates across nearly all models (except for Dense-121_adv). Therefore, our APAAP method also exhibits clear advantages over generative attack methods. We suppose that the poor performance of generative attack methods may be attributed to the fact that generative adversarial attack methods typically operate in a black-box fashion. Attackers may struggle to precisely control the generated adversarial examples, resulting in the poor transferability on black-box models.


### Table V

| Source Model | Method | Target Model | Distance Metric |
|--------------|--------|--------------|-----------------|
| TIM [13]    |        |              |                 |
| TIM w/ APAAf | IncRes-v2 | 98.3 | 86.5 | 83.9 | 73.3 | 73.8 | 81.4 | 83.7 | 75.1 | 77.3 | 10.302 | 11.108 |
| VT [18]     |        |              |                 |
| VT w/ APAAf | IncRes-v4 | 98.6 | 88.6 | 85.8 | 75.3 | 76.1 | 83.7 | 87.9 | 77.2 | 80.2 | 8.887 | 9.693 |
| EMI [19]    |        |              |                 |
| EMI w/ APAAf | IncRes-v4 | 99.5 | 92.3 | 90.9 | 82.6 | 82.9 | 87.5 | 90.6 | 83.8 | 85.6 | 9.513 | 10.299 |
| AIFGTM [20] |        |              |                 |
| AIFGTM w/ APAAf | IncRes-v2 | 97.4 | 82.2 | 78.0 | 72.5 | 72.4 | 77.1 | 80.4 | 74.0 | 75.1 | 10.415 | 11.269 |
| TIM [13]    |        |              |                 |
| TIM w/ APAAf | IncRes-v2 | 98.5 | 99.4 | 99.0 | 97.0 | 93.0 | 92.8 | 94.3 | 91.8 | 93.5 | 10.289 | 11.155 |
| VT [18]     |        |              |                 |
| VT w/ APAAf | IncRes-v4 | 99.7 | 99.9 | 99.7 | 99.0 | 97.6 | 95.9 | 98.2 | 95.5 | 96.5 | 9.019 | 9.898 |
| EMI [19]    |        |              |                 |
| EMI w/ APAAf | IncRes-v4 | 98.3 | 99.7 | 98.8 | 85.5 | 94.9 | 95.1 | 96.4 | 94.6 | 95.5 | 9.417 | 10.257 |
| AIFGTM [20] |        |              |                 |
| AIFGTM w/ APAAf | IncRes-v2 | 98.3 | 94.4 | 99.6 | 98.2 | 97.1 | 97.0 | 98.3 | 97.1 | 97.7 | 10.357 | 11.391 |

### Table VI

| Source Model | Method | Target Model | Distance Metric |
|--------------|--------|--------------|-----------------|
| TIM [13]    |        |              |                 |
| TIM w/ APAAf | Inc-v3 | 68.2 | 64.2 | 60.3 | 54.0 | 62.2 | 54.9 | 61.0 | 43.9 | 76.6 | 79.7 | 73.3 | 10.431 |
| VT [18]     |        |              |                 |
| VT w/ APAAf | Inc-v4 | 72.2 | 70.5 | 65.9 | 64.2 | 67.4 | 62.6 | 65.6 | 64.1 | 75.8 | 77.7 | 73.5 | 10.373 |
| EMI [19]    |        |              |                 |
| EMI w/ APAAf | Inc-v4 | 68.7 | 68.1 | 61.1 | 63.1 | 64.9 | 62.2 | 64.2 | 44.3 | 71.8 | 67.4 | 67.6 | 8.379 |
| AIFGTM [20] |        |              |                 |
| AIFGTM w/ APAAf | Inc-v3 | 85.2 | 86.7 | 83.3 | 78.3 | 86.6 | 79.9 | 84.3 | 62.9 | 91.4 | 83.6 | 88.4 | 9.563 |
| TIM [13]    |        |              |                 |
| TIM w/ APAAf | Inc-v3 | 85.1 | 85.4 | 83.8 | 79.1 | 83.6 | 79.3 | 82.3 | 66.7 | 88.8 | 84.1 | 86.6 | 80.8 |
| VT [18]     |        |              |                 |
| VT w/ APAAf | Inc-v4 | 93.6 | 93.1 | 91.5 | 86.0 | 91.6 | 87.2 | 89.6 | 73.4 | 95.0 | 88.4 | 92.7 | 90.0 |
| EMI [19]    |        |              |                 |
| EMI w/ APAAf | Inc-v4 | 96.6 | 95.6 | 93.9 | 89.4 | 95.1 | 91.0 | 92.7 | 78.9 | 97.2 | 93.6 | 95.5 | 73.0 |
| AIFGTM [20] |        |              |                 |
| AIFGTM w/ APAAf | Inc-v3 | 88.7 | 88.0 | 86.1 | 81.4 | 86.8 | 81.9 | 84.5 | 63.4 | 88.7 | 82.0 | 85.5 | 57.3 |
| TIM [13]    |        |              |                 |
| TIM w/ APAAf | Inc-v3 | 85.2 | 86.7 | 83.3 | 78.3 | 86.6 | 79.9 | 84.3 | 62.9 | 91.4 | 83.6 | 88.4 | 9.563 |
| VT [18]     |        |              |                 |
| VT w/ APAAf | Inc-v4 | 93.6 | 93.1 | 91.5 | 86.0 | 91.6 | 87.2 | 89.6 | 73.4 | 95.0 | 88.4 | 92.7 | 90.0 |
| EMI [19]    |        |              |                 |
| EMI w/ APAAf | Inc-v4 | 96.6 | 95.6 | 93.9 | 89.4 | 95.1 | 91.0 | 92.7 | 78.9 | 97.2 | 93.6 | 95.5 | 73.0 |

### Table VII

| Source Model | Method | Target Model | Distance Metric |
|--------------|--------|--------------|-----------------|
| Dense-201   | SGM [17] | 100.0 | 94.9 | 94.6 | 90.1 | 90.2 | 86.5 | 84.6 | 80.4 | 79.1 | 9.947 | 10.805 |
| Dense-152   | SGM w/ APAAf | 100.0 | 95.7 | 95.9 | 92.1 | 90.8 | 88.5 | 85.4 | 82.7 | 81.5 | 9.656 | 10.528 |
| Res-34      | SGM [17] | 97.8 | 99.9 | 93.5 | 91.9 | 89.4 | 78.7 | 81.8 | 75.5 | 75.8 | 9.487 | 10.357 |
| Res-121     | SGM w/ APAAf | 57.8 | 57.8 | 55.5 | 46.1 | 76.4 | 67.2 | 52.0 | 43.4 | 62.8 | 75.7 | 47.6 | 42.0 |

### Table VIII

| Source Model | Method | Target Model | Distance Metric |
|--------------|--------|--------------|-----------------|
| Dense-201   | IR [15] | 97.7 | 92.0 | 90.6 | 88.5 | 90.0 | 72.2 | 68.5 | 63.3 | 59.8 |
| Dense-152   | IR w/ APAAf | 93.9 | 89.0 | 83.2 | 93.4 | 74.2 | 69.6 | 64.7 | 58.2 |
| Res-34      | IR [15] | 89.8 | 98.1 | 89.7 | 87.7 | 96.2 | 78.9 | 73.3 | 70.0 | 66.5 |
| Res-121     | IR w/ APAAf | 89.8 | 98.1 | 89.7 | 87.7 | 96.2 | 78.9 | 73.3 | 70.0 | 66.5 |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
| Source Model | Method | Target Model | Distance Metric |
|-------------|--------|-------------|----------------|
| RegNet      | MIFGSM [11] | Reg-Net, Res-18, SENet-18, Dense-121 | 5.363, 5.734 |
| RegNet      | MIFGSM w/ APAAF | Res-18, SENet-18, Dense-121 | 5.206, 5.585 |
| RegNet      | DIM [12] | Dense-121, Dense-ResNet-10 | 5.436, 5.790 |
| RegNet      | SIM [14] | Dense-121, Dense-ResNet-10 | 5.147, 5.523 |
| SIM w/ APAAF | 91.0 | Dense-121, Dense-ResNet-10 | 5.488, 5.837 |
| MIFGSM      | 78.8 | Dense-121, Dense-ResNet-10 | 5.168, 5.537 |
| MIFGSM w/ APAAF | 89.0 | Dense-121, Dense-ResNet-10 | 5.365, 5.729 |
| SIM w/ APAAF | 84.4 | Dense-121, Dense-ResNet-10 | 5.351, 5.865 |
| SIM w/ APAAF | 75.3 | Dense-121, Dense-ResNet-10 | 5.592, 5.917 |
| SIM w/ APAAF | 87.3 | Dense-121, Dense-ResNet-10 | 5.279, 5.632 |

TABLE IX
ATTACK SUCCESS RATES OF THE ADVERSARIAL EXAMPLES ON CIFAR10 UNDER UNTARGETED ATTACK SETTING WITH DIFFERENT NUMBER OF ITERATIONS

| Source Model | Method | Step Number | Target Model | Distance Metric |
|-------------|--------|-------------|-------------|----------------|
| RegNet      | MIFGSM [11] | Reg-Net, Res-18, SENet-18, Dense-121 | 87.8, 92.7 | 87.9, 90.2 |
| RegNet      | MIFGSM w/ APAAF | Res-18, SENet-18, Dense-121 | 87.8, 92.7 | 87.9, 90.2 |
| RegNet      | DIM [12] | Dense-121, Dense-ResNet-10 | 87.8, 92.7 | 87.9, 90.2 |
| RegNet      | SIM [14] | Dense-121, Dense-ResNet-10 | 87.8, 92.7 | 87.9, 90.2 |
| SIM w/ APAAF | 87.8 | Dense-121, Dense-ResNet-10 | 87.8, 92.7 | 87.9, 90.2 |
| SIM w/ APAAF | 90.0 | Dense-121, Dense-ResNet-10 | 87.8, 92.7 | 87.9, 90.2 |

TABLE X
COMPARE WITH OTHER ALTERNATIVES TO THE SIGN FUNCTION (i.e., ARCTANH AND ADAM)

| Source Model | Method | Target Model | Distance Metric |
|-------------|--------|-------------|----------------|
| RegNet      | MIFGSM [11] | Reg-Net, Res-18, SENet-18, Dense-121 | 87.8, 92.7 | 87.9, 90.2 |
| RegNet      | MIFGSM w/ ArcTanh | Res-18, SENet-18, Dense-121 | 87.8, 92.7 | 87.9, 90.2 |
| RegNet      | MIFGSM w/ Adam | Res-18, SENet-18, Dense-121 | 87.8, 92.7 | 87.9, 90.2 |
| RegNet      | DIM [12] | Dense-121, Dense-ResNet-10 | 87.8, 92.7 | 87.9, 90.2 |
| RegNet      | SIM [14] | Dense-121, Dense-ResNet-10 | 87.8, 92.7 | 87.9, 90.2 |
| SIM w/ ArcTanh | 90.0 | Dense-121, Dense-ResNet-10 | 87.8, 92.7 | 87.9, 90.2 |
| SIM w/ Adam | 90.0 | Dense-121, Dense-ResNet-10 | 87.8, 92.7 | 87.9, 90.2 |

TABLE XI
COMPARE WITH SEVERAL GENERATIVE METHODS

| Source Model | Method | Target Model | Distance Metric |
|-------------|--------|-------------|----------------|
| RegNet      | AdvGAN [2] | Reg-Net, Res-18, SENet-18, Dense-121 | 87.8, 92.7 | 87.9, 90.2 |
| RegNet      | SIM [14] | Dense-121, Dense-ResNet-10 | 87.8, 92.7 | 87.9, 90.2 |
| SIM w/ APAAF | 87.8 | Dense-121, Dense-ResNet-10 | 87.8, 92.7 | 87.9, 90.2 |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE XII
THE COMPARISONS BETWEEN THE ATTACK SUCCESS RATES OF THE ADVERSARIAL EXAMPLES BY BASELINES, OUR PROPOSED METHOD WITH FIXED SCALING FACTOR (APAA$_f$) AND OUR PROPOSED METHOD WITH ADAPTIVE SCALING FACTOR (APAA$_a$) AGAINST BOTH NORMALLY TRAINED MODELS AND DEFENSE MODELS ON CIFAR10 UNDER UNTARGETED ATTACK SETTING

| Source Model | Method | Target Model | Distance Metric | MAU | RXS
|--------------|--------|--------------|----------------|-----|-----
| RegNet | | | | | |
| Res-18 | MIFGSM | 85.6 | 76.0 | 86.0 | 84.8 | 88.6 | 94.7 | 63.1 | 23.0 | 79.2 | 82.2 | 4.976 | 5.390 |
| | w/ APAA$_f$ | 85.6 | 76.0 | 86.0 | 84.8 | 88.6 | 94.7 | 63.1 | 23.0 | 79.2 | 82.2 | 4.976 | 5.390 |
| | w/ APAA$_a$ | 95.5 | 79.3 | 93.8 | 82.4 | 87.8 | 96.8 | 64.8 | 27.4 | 80.5 | 83.3 | 6.488 | 5.504 |
| SEnet-18 | MIFGSM | 84.3 | 74.0 | 85.0 | 84.0 | 86.2 | 92.1 | 70.5 | 22.5 | 79.6 | 82.6 | 4.892 | 5.292 |
| | w/ APAA$_f$ | 84.3 | 74.0 | 85.0 | 84.0 | 86.2 | 92.1 | 70.5 | 22.5 | 79.6 | 82.6 | 4.892 | 5.292 |
| | w/ APAA$_a$ | 94.6 | 79.3 | 93.8 | 82.4 | 87.8 | 96.8 | 64.8 | 27.4 | 80.5 | 83.3 | 6.488 | 5.504 |
| | SIM | 89.7 | 79.3 | 93.8 | 87.3 | 90.8 | 93.2 | 68.2 | 25.3 | 80.3 | 83.5 | 5.555 | 5.953 |
| | w/ APAA$_f$ | 89.7 | 79.3 | 93.8 | 87.3 | 90.8 | 93.2 | 68.2 | 25.3 | 80.3 | 83.5 | 5.555 | 5.953 |
| | w/ APAA$_a$ | 94.6 | 79.3 | 93.8 | 82.4 | 87.8 | 96.8 | 64.8 | 27.4 | 80.5 | 83.3 | 6.488 | 5.504 |
| | SVRE | 93.6 | 89.3 | 91.0 | 91.2 | 94.8 | 85.5 | 64.5 | 19.7 | 87.5 | 90.2 | 5.554 | 5.597 |
| | w/ APAA$_f$ | 93.6 | 89.3 | 91.0 | 91.2 | 94.8 | 85.5 | 64.5 | 19.7 | 87.5 | 90.2 | 5.554 | 5.597 |
| | w/ APAA$_a$ | 94.6 | 79.3 | 93.8 | 82.4 | 87.8 | 96.8 | 64.8 | 27.4 | 80.5 | 83.3 | 6.488 | 5.504 |

For fair comparisons, we consider all the models used during the training of the generator ($C_1, \ldots, C_n$) as white-box models, and conduct both attacks of SIM and our method on the ensemble of all white-box models.

To conduct a comprehensive comparison among our proposed APAA$_f$, APAA$_a$ methods, and the baseline methods, Table XII presents a comparison of our APAA$_f$ and APAA$_a$ methods when combined with MIFGSM, DIM, SIM and SVRE, respectively, against baseline methods under six different source model settings. From Table XII, it becomes apparent that in most cases, APAA$_a$ achieves higher attack success rates than APAA$_f$. However, in some instances, APAA$_f$ uses smaller perturbations than APAA$_a$. Nevertheless, both APAA$_a$ and APAA$_f$ consistently outperform baseline methods in terms of attack success rates across various settings. It is noteworthy that although SVRE is a stronger baseline compared to MIFGSM, DIM and SIM, after integration with our APAA, SVRE achieves higher attack success rates with fewer perturbations. This further demonstrates the effectiveness of our APAA method again. Due to the distinct advantages of these two methods, attackers can freely select one of them based on the specific scenario when conducting the attack. For experiments on more models, please refer to Table XIII in the appendix, available online.

C. Attack With Adaptive Scaling Factor

We further demonstrate the effectiveness of our proposed adaptive scaling factor generator. The process of generating adversarial examples with APAA$_a$ is summarized in Algorithm 2.
Algorithm 2: Generating Adversarial Examples With Adaptive Scaling Factor Generator.

Input: the original image \( x \) and corresponding label \( y \)
Input: the number of attack iteration \( T \)
Input: the scaling factor generators \( G_1, G_2, \ldots, G_T \)
Input: the ensembled model \( C \), which contains all white-box classifier models \( C_1, \ldots, C_n \)
Output: the adversarial example \( x_{adv}^T \)

1: \( x_{adv}^0 = x \)
2: for \( t \in \{1, \ldots, T\} \) do
3: \( \text{grad}_t = \nabla_x f(C(x_{adv}^{t-1}, y)) \)
4: \( \gamma_t = G_t(x_{adv}^{t-1}, \text{grad}_t) \)
5: \( x_{adv}^t = \Pi_{x}(\gamma_t \cdot \text{grad}_t + x_{adv}^{t-1}) \) \( \triangleright \) using the update in BIM as an example
6: end for
7: return \( x_{adv}^T \)

V. CONCLUSION

In this work, we propose to use the scaling factor instead of the sign function to normalize the gradient of the input example when conducting the adversarial attack, which can achieve a more accurate gradient direction and thus improve the attack success rate. The scaling factor can either be elaborately selected manually, or adaptively achieved by a generator according to different image characteristics. We also theoretically demonstrate that our proposed method can improve the black-box transferability of adversarial examples. Extensive experiments on CIFAR10 and ImageNet show the superiority of our proposed methods, which can improve the black-box attack success rates on both normally trained models and defense models with fewer update steps and perturbation budgets.

REFERENCES

[1] C. Szegedy et al., “Intriguing properties of neural networks,” in Proc. Int. Conf. Learn. Representations, 2014.
[2] C. Xiao, B. Li, J. Zhu, W. He, M. Liu, and D. Song, “Generating adversarial examples with adversarial networks,” in Proc. Int. Joint Conf. Artif. Intell., 2018, pp. 3905–3911.
[3] Z. Zhao, D. Dua, and S. Singh, “Generating natural adversarial attacks,” in Proc. Int. Conf. Learn. Representations, 2018.
[4] Y. Song, R. Shu, N. Kushman, and S. Ermon, “Constructing unrestricted adversarial examples with generative models,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2018, pp. 8322–8333.
[5] A. Joshi, A. Mukherjee, S. Sarkar, and C. Hegde, “Semantic adversarial attacks: Parametric transformations that fool deep classifiers,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 4772–4782.
[6] H. Qiu, C. Xiao, L. Yang, X. Yan, H. Lee, and B. Li, “SemanticAdv: Generating adversarial examples via attribute-conditioned image editing,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 19–37.
[7] Z. Xiao et al., “Improving transferability of adversarial patches on face recognition with generative models,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 11 845–11 854.
[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explainng and harnessing adversarial examples,” in Proc. Int. Conf. Learn. Representations, 2015.
[9] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” in Proc. Int. Conf. Learn. Representations, 2017.
[10] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in Proc. Int. Conf. Learn. Representations, 2018.
[11] Y. Dong et al., “Boosting adversarial attacks with momentum,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 9185–9193.
[12] C. Xie et al., “Improving transferability of adversarial examples with input diversity,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 2760–2769.
[13] Y. Dong, T. Pang, H. Su, and J. Zhu, “Evading defenses to transferable adversarial examples by translation-invariant attacks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 4312–4321.
[14] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft, “Nesterov accelerated gradient and scale invariance for adversarial attacks,” in Proc. Int. Conf. Learn. Representations, 2020.
[15] X. Wang, J. Ren, S. Lin, X. Zhu, Y. Wang, and Q. Zhang, “A unified approach to interpreting and boosting adversarial transferability,” in Proc. Int. Conf. Learn. Representations, 2021.
[16] Y. Nesterov, “A method for unconstrained convex minimization problem with the rate of convergence \( o(1/k^2) \),” in Proc. USSR Acad. Sci., 1983, vol. 269, pp. 543–547.
[17] D. Wu, Y. Wang, S. Xia, J. Bailey, and X. Ma, “Skip connections matter: On the transferability of adversarial examples generated with ResNets,” in Proc. Int. Conf. Learn. Representations, 2020.
[18] X. Wang and K. He, “Enhancing the transferability of adversarial attacks through variance tuning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 1924–1933.
[19] X. Wang, J. Lin, H. Hu, J. Wang, and K. He, “Boosting adversarial transferability through enhanced momentum,” 2021, arXiv:2103.10609.
[20] J. Zou, Z. Pan, J. Qiu, Y. Duan, X. Liu, and Y. Pan, “Making adversarial examples more transferable and indistinguishable,” 2020, arXiv:2007.03838.
[21] F. Tramèr, A. Kurakin, N. Papernot, J. I. Goodfellow, D. Boneh, and P. D. McDaniel, “Ensemble adversarial training: Attacks and defenses,” in Proc. Int. Conf. Learn. Representations, 2018.
[22] C. Song, K. He, J. Lin, L. Wang, and J. E. Hopcroft, “Robust local features for improving the generalization of adversarial training,” in Proc. Int. Conf. Learn. Representations, 2020.
[23] T. Pang, X. Yang, Y. Dong, T. Xu, J. Zhu, and H. Su, “Boosting adversarial training with hypersphere embedding,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2020, Art. no. 652.
[24] E. Wong, L. Rice, and J. Z. Kolter, “Fast is better than free: Revisiting adversarial training,” in Proc. Int. Conf. Learn. Representations, 2020.
[25] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, “A study of the effect of JPEG compression on adversarial images,” 2016, arXiv:1608.08583.
[26] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, “Defense against adversarial attacks using high-level representation guided denoiser,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 1778–1787.
[27] P. Samangouei, M. Kabkab, and R. Chellappa, “Defense-GAN: Protecting classifiers against adversarial attacks using generative models,” in Proc. Int. Conf. Learn. Representations, 2018.
[28] Z. Liu et al., “Feature distillation: DNN-oriented JPEG compression against adversarial examples,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 860–868.
[29] X. Jia, X. Wei, X. Cao, and H. Foroosh, “ComDefend: An efficient image compression model to defend adversarial examples,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 6084–6092.
[30] X. Liu, M. Cheng, H. Zhang, and C. Hsieh, “Towards robust neural networks via random self-ensemble,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 369–385.
[31] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, “Improving adversarial robustness via promoting ensemble diversity,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 4970–4979.
[32] H. Yang et al., “DIVERGE: Diversifying vulnerabilities for enhanced robustness through variance tuning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 1924–1933.
[33] A. Raghunathan, J. Steinhardt, and P. Liang, “Certified defenses against adversarial examples,” 2020, Art. no. 462.
[34] A. Raghunathan, J. Steinhardt, and P. Liang, “Certified defenses against adversarial examples,” in Proc. Int. Conf. Learn. Representations, 2018.
[35] E. Wong, F. R. Schmidt, J. H. Metzen, and J. Z. Kolter, “Scaling provable adversarial defenses,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2018, pp. 8400–8409.
[36] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, “Certified adversarial robustness via randomized smoothing,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 1310–1320.
[37] J. Jia, X. Cao, B. Wang, and N. Z. Gong, “Certified robustness for top-k predictions against adversarial perturbations via randomized smoothing,” in Proc. Int. Conf. Learn. Representations, 2020.
M. Grabisch and M. Roubens, “An axiomatic approach to the concept of interaction among players in cooperative games,” Int. J. Game Theory, vol. 28, no. 4, pp. 547–565, 1999.

L. S. Shapley, “A value for n-person games,” in Contributions to the Theory of Games II, vol. 2. Princeton, NJ, USA: Princeton Univ. Press, 1953, pp. 307–318.

O. Russakovsky et al., “ImageNet large scale visual recognition challenge,” Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, 2015.

A. Krizhevsky, “Learning multiple layers of features from tiny images,” Univ. Toronto, Tech. Rep. 2009.

I. Radosavovic, R. P. Koraraj, R. B. Girshick, K. He, and P. Dollár, “Designing network design spaces,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 10 428–10 436.

K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 7132–7141.

G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 4700–4708.

S. Zagoruyko and N. Komodakis, “Wide residual networks,” 2016. arXiv:1605.07146.

Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, “Dual path networks,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2017, pp. 4467–4475.

D. Han, J. Kim, and J. Kim, “Deep pyramidal residual networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 5927–5935.

X. Gastaldi, “Shake-shake regularization,” 2017. arXiv:1705.07485.

C. Xiao, P. Zhong, and C. Zheng, “Enhancing adversarial defense by k-winners-take-all,” in Proc. Int. Conf. Learn. Representations, 2020.

K. Roth, Y. Kilcher, and T. Hofmann, “The odds are odd: A statistical test for detecting adversarial examples,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 5498–5507.

Y. Li, J. Bradshaw, and Y. Sharma, “Are generative classifiers more robust to adversarial attacks?,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 3804–3814.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-ResNet and the impact of residual connections on learning,” in Proc. AAAI Conf. Artif. Intell., 2017, pp. 4278–4284.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 2818–2826.

K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 630–645.

M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4510–4520.

C. Liu et al., “Progressive neural architecture search,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 19–34.

B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 8697–8710.

C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. L. Yuille, “Mitigating adversarial effects through randomization,” in Proc. Int. Conf. Learn. Representations, 2018.

W. Xu, D. Evans, and Y. Qi, “Feature squeezing: Detecting adversarial examples in deep neural networks,” in Proc. Netw. Distrib. Syst. Secur. Symp., 2018.

C. Guo, M. Rana, M. Cissé, and L. van der Maaten, “Countering adversarial images using input transformations,” in Proc. Int. Conf. Learn. Representations, 2018.

Y. Xiong, J. Lin, M. Zhang, J. E. Hopcroft, and K. He, “Stochastic variance reduced ensemble adversarial attack for boosting the adversarial transferability,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2022, pp. 14 963–14 972.

D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Representations, 2015.