Generating Adversarial Examples with Better Transferability via Masking Unimportant Parameters of Surrogate Model

Dingcheng Yang[1,2], Wenjian Yu[1,*], Zihao Xiao[2], Jiaqi Luo[1]
[1]Dept. Computer Science & Tech., BNRist, Tsinghua University, Beijing, China.
[2]RealAI.
ydc19@mails.tsinghua.edu.cn, yu-wj@tsinghua.edu.cn, zihao.xiao@realai.ai, luojq19@mails.tsinghua.edu.cn

***Abstract*—Deep neural networks (DNNs) have been shown to be vulnerable to adversarial examples. Moreover, the transferability of the adversarial examples has received broad attention in recent years, which means that adversarial examples crafted by a surrogate model can also attack unknown models. This phenomenon gave birth to the transfer-based adversarial attacks, which aim to improve the transferability of the generated adversarial examples. In this paper, we propose to improve the transferability of adversarial examples in the transfer-based attack via masking unimportant parameters (MUP). The key idea in MUP is to refine the pretrained surrogate models to boost the transfer-based attack. Based on this idea, a Taylor expansion-based metric is used to evaluate the parameter importance score and the unimportant parameters are masked during the generation of adversarial examples. This process is simple, yet can be naturally combined with various existing gradient-based optimizers for generating adversarial examples, thus further improving the transferability of the generated adversarial examples. Extensive experiments are conducted to validate the effectiveness of the proposed MUP-based methods.**

***Index Terms*—Adversarial attack, Transfer-based attack, Network pruning.**

I. Introduction

Deep neural networks (DNNs) have achieved remarkable success in many areas, such as computer vision and natural language processing. However, they are vulnerable to adversarial examples, which are generated by adding carefully designed imperceptible perturbations on clean data. An example is illustrated in Fig. 1. The left image is an image of a dung beetle, which is also successfully classified as a dung beetle by an Inception-v3 network [1]. However, a carefully constructed adversarial example (the right image) fools Inception-v3 to misclassify it as a dragonfly, even though it does not look any different from the left image to a human. The generation of adversarial samples is known as the adversarial attack, which raises serious concerns about the security of DNN for deployments in real-world scenarios, such as face recognition and autonomous driving.

Even worse, researchers have found that adversarial examples have transferability. For example, although the adversarial example in Fig. 1 was generated from an Inception-v3, it can also successfully fool an Inception-v4 model. Such a phenomenon allows attacking unknown victim models by attacking a given surrogate model, which can be obtained by training locally or by using a publicly available pretrained model. Generating adversarial examples from a given model is called the *white-box attack*. In contrast, the attack that generates adversarial examples from a surrogate model and leverages its transferability to fool unknown victim models is called the *transfer-based attack*. The transfer-based attack is more practical than the white-box attack due to the inaccessibility of real-world commercial models.

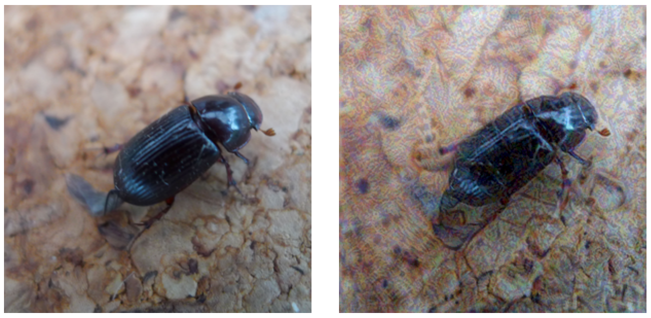
(a) Original Image. (b) Advdersarial Example.

Fig. 1. A visualization of adversarial examples. The left image is a clean image of a dung beetle, and the right image is an adversarial example generated from an Inception-v3 model using the method proposed in this paper, and which is misclassified as a dragonfly.

To better verify the safety of a DNN, investigating the method of transfer-based attack has received widespread attention. Dong et al. were the first to treat the problem of generating adversarial examples as a model training problem [2]. Specifically, the surrogate models were treated as the training data and the victim models were treated as the testing data, so the transferability of adversarial examples was naturally regarded as the generalizability of DNNs. Inspired by this analogy, many ideas to alleviate the overfitting phenomenon were borrowed to propose new methods for the transfer-based attack. Dong et al. [2] proposed a momentum iterative gradient-based method to escape from pool local maxima and experimentally improved the transferability. Xie et al. [3] proposed to use data augmentations to escape local optima. Then, there are a series of works that improve the transferability by using different data augmentation methods, including

[*]corresponding author.

translation [4], Mixup [5], and scaling [6]. More recently, Huang et al. [7] proposed an algorithm named Transferable Attack based on Integrated Gradients on Random piece-wise linear path (TAIG-R) to avoid overfitting the surface of the surrogate model.

This work is also inspired by [2], but instead of proposing an optimizer to avoid overfitting a given surrogate model, we want to refine the given surrogate model thus boosting the transfer-based attack. This is similar to the data cleaning process of model training, where the model performance is improved by removing noisy data. Our motivation comes from the phenomenon observed in the field of network pruning, where a DNN often has a large number of unimportant parameters. We propose to mask these unimportant parameters when generating adversarial examples to avoid overfitting them. We first borrow the idea from the work on model pruning [8], using a Taylor expansion-based metric to evaluate the parameter importance scores of the surrogate model. Then, we propose a technique for boosting transfer-based attacks which improves the transferability of adversarial examples via masking unimportant parameters (MUP), where the unimportant parameters are identified by the Taylor expansion-based metric. The experimental results on ImageNet-compatible dataset show that the proposed method can consistently improve the transferability of adversarial examples.

Note that we are not the only work to mask model parameters during the generation of adversarial examples. The ghost network (GN) [9] leveraged dropout layers [10] to prevent overfitting, which is equivalent to randomly masking some neurons. The key difference in our method is the way to identify the parameters to be masked, i.e., we use a Taylor expansion-based metric to identify the unimportant parameters rather than randomly dropout. Experimental comparison with the method [9] will be presented in Section IV.B.

The major contributions of this work are as follows:

- We propose the idea of improving the transferability of adversarial examples via masking unimportant parameters (MUP) in the surrogate model.
- A technique for boosting transfer-based attacks is presented based on the proposed idea, which utilizes a Taylor expansion-based metric to evaluate the parameter importance score of surrogate models, and then masks unimportant parameters to improve existing optimizers for generating adversarial examples.
- Extensive experiments on ImageNet-compatible dataset demonstrate the effectiveness of the proposed method. With the TAIG-R optimizer [7], the proposed method improves the attack success rates by 4.9%, 4.2%, and 2.9% on average respectively when Inc-v3, Inc-v4, and DN121 are used as surrogate models. Moreover, the experimental results also show that the proposed method is better than the comparative work GN [9].

II. Related Work

A. Adversarial Attack

Although DNNs have demonstrated to be successful on many tasks, they are vulnerable to adversarial examples [11]. Let $x \in \mathbb{R}^d$ denote an image with dimension d, y denote its ground truth label. Given a pretrained DNN θ, the adversarial attack is implemented by solving the following optimization problem:

$$\begin{aligned} x^* &= \text{argmax}_{x'} \mathcal{L}(x', y, \theta) \ , \\ \text{s.t.} \quad & \|x' - x\|_\infty \leq \epsilon \ , \end{aligned} \tag{1}$$

where the $\mathcal{L}(x, y, \theta)$ denotes the loss function that guides the DNN θ to predict the class of image x as ground-truth y, ϵ is the maximum allowed magnitude of perturbation, and x^* is the generated adversarial example. Some gradient-based optimizers were proposed to solve (1). Szegedy et al. were the first to use a box-constrained L-BFGS optimizer [11]. Then, a one-step optimizer, i.e., fast gradient sign method (FGSM) was proposed to efficiently generate adversarial examples [12]. It was later extended to the iterative version of FGSM (I-FGSM) [13], which generates adversarial examples that fool the model θ with a higher success rate.

The generated adversarial example is shown to have the ability to attack an unknown model, and this ability is called the transferability of the adversarial example. In this scenario, the model θ used to generate the adversarial example is called the *surrogate model*, and the unknown model to be attacked is called the *victim model*. This type of adversarial attack is called the transfer-based attack. Dong et al. systematically investigated the transferability of adversarial examples and found an interesting phenomenon that is although the iterative method (I-FGSM) is a stronger optimizer than the one-step method (FGSM), the transferability of the adversarial examples generated by I-FGSM is worse than FGSM [2]. Specifically, their experimental results on the white-box attacks show that an Inception-v3 has only a 72.3% probability of being fooled by FGSM, but a 100% probability of being fooled by I-FGSM. However, when the adversarial examples generated from Inception-v3 were then used to attack an Inception-v4 model, the attack success rates of the adversarial examples generated by FGSM and I-FGSM were 28.2% and 22.8%, respectively. It shows that the adversarial examples generated by FGSM are more transferable.

To explain this phenomenon, Dong et al. analogized the surrogate model to the training data and the victim model to the testing data, so the transferability of the generated adversarial examples can be analogous to the generalizability of the trained models. Then, inspired by the widespread use of momentum-based optimizers to escape local maxima in training DNNs, Dong et al. propose a Momentum Iterative fast gradient sign Method (MIM) to generate more transferable adversarial examples [2]. Suppose the number of iterations is

N, the MIM performs the following steps in an iteration:

$$\delta_{t+1} = \nabla_x \mathcal{L}(x_t, y, \theta) \ , \tag{2}$$

$$g_{t+1} = \mu \cdot g_t + \delta_{t+1}/\|\delta_{t+1}\|_1 \ , \tag{3}$$

$$x_{t+1} = \mathrm{Clip}_x^{\epsilon}\{x_t + \beta \cdot \mathrm{sign}(g_{t+1})\} \ , \quad t = 0, 1, \cdots \tag{4}$$

with $x_0 = x$ and $g_0 = \mathbf{0}$, and the generated adversarial example is x_N. Here, the $\mathrm{Clip}_x^{\epsilon}$ function is used to guarantee that the adversarial example is in the ϵ-ball of x under the L_∞ norm, μ denotes a momentum factor and β is a step size.

Furthermore, many optimizers have been proposed to improve MIM by introducing data augmentation to the objective function $\mathcal{L}(x', y, \theta)$ in (1) to avoid overfitting. For example, Lin et al. proposed a Scale-Invariant attack Method (SIM) [6], which optimizes an adversarial example over a set of scaled images. Specifically, the loss function for generating adversarial examples with SIM is $\sum_{i=0}^{m-1} \mathcal{L}(x/2^i, y, \theta)$, where m denotes the number of scaled images. More recently, Huang et al. proposed an algorithm named Transferable Attack based on Integrated Gradient (TAIG) [7], which computes the gradient over a set of sampling images. Two versions of TAIG were proposed, the stronger version is called TAIG-R, which is equivalent to optimizing the following loss function: $\sum_{i=1}^{S} \mathcal{L}(\frac{i}{S}x + U(-\epsilon, \epsilon), y, \theta)$, where U denotes a uniform distribution, and S denotes the number of sampling images.

Unlike the above methods, instead of proposing a new optimizer to generate adversarial examples (by solving (1)), we propose the idea of masking unimportant parameters in the surrogate model θ when using them. Therefore, our approach can assist the above methods, as will be shown in Section III.B. To the best of our knowledge, the most relevant work to us is the ghost network (GN) [9], which manually injects some dropout layers into the surrogate model to randomly drop some neurons. Our experimental results in Section IV.B will show that our approach works better than GN.

B. Network Pruning

Network pruning is a popular technique in the field of model compression, which aims to remove unimportant parameters to compress and accelerate DNNs. To guarantee the accuracy of the pruned model, many heuristic strategies have been proposed to evaluate whether a parameter is important or not. Optimal Brain Damage [14] pruned the parameters based on the Hessian matrix of the loss function. Deep Compression [15] pruned the parameters based on their absolute values. Li et al. [16] proposed an L_1-norm-based metric to evaluate the importance of channels. Molchanov et al. [8] proposed a Taylor expansion-based metric to evaluate the importance of channels. In this paper, we borrow the idea from the work [8] to evaluate the parameter importance scores based on the first-order gradient information of the parameters.

III. Method

In this section, we first present a Taylor expansion-based metric for calculating the parameter importance score. Based on this metric, we present a technique for boosting transfer-based attacks that masks the unimportant parameters at each iteration.

A. Parameter Importance Score

Existing transfer-based attack methods generate adversarial examples by maximizing the loss function (1) and improving the transferability of the adversarial examples by using advanced optimizers that help to escape the local maxima. The parameter θ in (1) is considered as a constant in these works. Inspired by the analogy of surrogate models to training data in [2], we propose that the quality of θ is also a key factor for the transferability, just as noisy data can be detrimental to generalizability.

However, work on network pruning suggests that DNNs are often over-parameterized. For example, Li et al. [16] showed that even after removing 50% of the convolutional kernels from the first convolutional layer of a VGG16 model [17] based on the L_1-norm, the accuracy degradation of the VGG16 model on CIFAR-10 dataset [18] was still negligible. Since there are redundant parameters for prediction, it is natural to be concerned about the impact of these redundant parameters on the generation of adversarial examples. While previous work on preventing overfitting can alleviate this effect, explicitly removing these parameters is also a worthwhile consideration, and the two types of approaches can be naturally combined to achieve better performance. To this end, we first introduce a method to calculate the importance scores of model parameters in this subsection. Based on the calculated scores, we can explicitly identify and mask unimportant parameters during the generation of adversarial examples, which will be presented in the next subsection.

Given an image x and its true label y, assume that V_i represents the importance score of the i-th parameter θ_i for predicting the category of image x as label y. It can be defined intuitively as $V_i = |\mathcal{L}(x, y, \theta) - \mathcal{L}(x, y, \theta - \theta_i e_i)|$, where e_i denotes a one-hot vector consisting of zeros in all elements except for a single one in i-th element. It means how much the loss function change after setting θ_i to zero. However, calculating the importance scores for all parameters in this way requires a number of forward propagations proportional to the model size, which is intractable for large DNNs. To reduce the computational overhead, It was approximated in [8] by using the first-order gradient information based on the Taylor expansion as follows:

$$V_i = |\mathcal{L}(x, y, \theta) - \mathcal{L}(x, y, \theta - \theta_i e_i)| \approx |\frac{\partial \mathcal{L}(x, y, \theta)}{\partial \theta_i} \theta_i|. \tag{5}$$

According to (5), the importance scores of all parameters can be calculated simultaneously in just one step of backpropagation.

B. The MUP-Based Attack Algorithm

In this subsection, we propose a technique for boosting existing optimizers by masking unimportant parameters (MUP). We take assisting MIM as an example, call it MUP-MIM, and

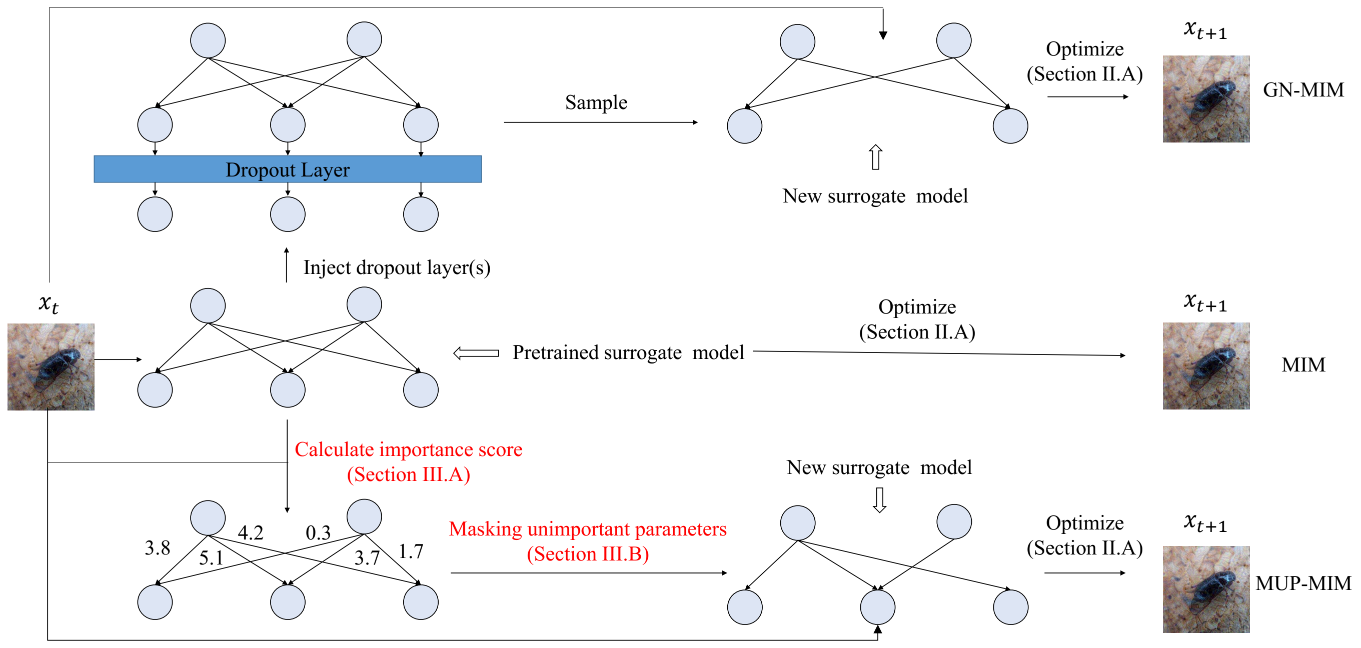

Fig. 2. An illustration of the MUP technique and the approach of ghost network (GN) during a round of iteration for generating adversarial examples (taking the optimizer MIM as an example).

illustrate it in Fig. 2. Only two layers of the surrogate models are drawn in Fig. 2 for simplicity. The standard MIM and the MIM assisted by the comparative work ghost network (GN) [9] are also illustrated in Fig. 2, and the latter is called GN-MIM. As can be seen in Fig. 2, the pretrained surrogate model is fixed in MIM, while it is changing dynamically when using GN and MUP.

Given a hyperparameter masking ratio r, MUP will mask unimportant parameters at each iteration. To do this, the important scores V are calculated as described in Section III.A. Then, the $r \times |V|$ elements in V with the smallest score are considered as unimportant parameters. This process is simple and can be naturally integrated into any gradient-based optimizer. We take the basic MIM as an example and summarize the algorithm MUP-MIM for assisting MIM with MUP in Algorithm 1. At each iteration, the importance scores are first calculated (Step 3). Then, a binary mask of the same shape as θ is obtained based on the given masking ratio r (Step 4-5). The modification to the standard MIM is that the gradient vector δ_{t+1} is computed with the model $\theta \odot M$, whose unimportant parameters are masked (Step 6). Here the $\odot$ denotes an element-wise product. The subsequent steps are the same as for MIM (Step 7-8). Finally, the produced x_N is the generated adversarial example. Since the unimportant parameters are masked at each iteration, the generated adversarial example avoids overfitting them and has higher transferability, as will be demonstrated in Section IV.B.

It is straightforward to integrate MUP into other optimizers. Two improved versions of MIM are considered in this paper, namely SIM [6] and TAIG-R [7], which are introduced in Section II.A. Assisting SIM with MUP (called MUP-SIM), only the gradient δ_{t+1} in Step 6 needs to be computed as follows:

$$\delta_{t+1} = \nabla_x \sum_{i=0}^{m-1} \mathcal{L}(x_t/2^i, y, \theta \odot M). \tag{6}$$

Similarly, using MUP to assist TAIG-R (called MUP-TAIG-R) only requires to compute δ_{t+1} in Step 6 as follows:

$$\delta_{t+1} = \nabla_x \sum_{i=1}^{S} \mathcal{L}(\frac{i}{S}x_t + U(-\epsilon,\epsilon), y, \theta \odot M). \tag{7}$$

The comparative work ghost network (GN) [9] is also shown in Fig. 2, which manually injects some dropout layers into the pretrained surrogate model to assist MIM. The authors

Algorithm 1 MUP-MIM:Masking unimportant parameters for assisting MIM.

Input: Input image x and its ground-truth label y, pretrained surrogate model θ, number of iteration N, momentum factor μ, step size β, the maximum allowed magnitude of perturbation ϵ, masking ratio r.
Output: The generated adversarial example x_N with $\|x_N - x\|_\infty \le \epsilon$

```
1: $x_0 \leftarrow x, g_0 \leftarrow \mathbf{0}$.
2: for $t \leftarrow 0$ to $N-1$ do
3:     $V \leftarrow |\frac{\partial \mathcal{L}(x_t,y,\theta)}{\partial \theta} \odot \theta|$.                ▷ (5)
4:     Let $\tau$ be the $r \times |V|$-th smallest element in $V$.
5:     $M \leftarrow V > \tau$.                ▷ A binary mask.
6:     $\delta_{t+1} \leftarrow \nabla_x \mathcal{L}(x_t, y, \theta \odot M)$.                ▷ (2)
7:     $g_{t+1} = \mu \cdot g_t + \delta_{t+1}/\|\delta_{t+1}\|_1$.                ▷ (3)
8:     $x_{t+1} = \text{Clip}_x^\epsilon\{x_t + \beta \cdot \text{sign}(g_{t+1})\}$.                ▷ (4)
9: end for
```

of GN claim to do this in order to sample a sub-network at each iteration. And in our opinion, this is equivalent to randomly masking some *neurons*. In contrast, MUP is selectively masking some parameters, which are called *synapses* in computational neuroscience. Therefore, MUP has two advantages over GN in terms of the way to identify the parameters to be masked. Firstly, the proposed MUP is input-dependent and is able to model the effect of input data on the importance scores, and is therefore more theoretical. Secondly, MUP has a finer granularity in identifying unimportant parameters, since masking a neuron in GN is equivalent to masking a group of synapses. Besides the difference in the way to identify the parameters to be masked, GN needs to manually specify some locations to inject dropout layers, while not needed in MUP, so MUP is more convenient to be applied on a new architecture without any customization.

IV. Experimental Results

In this section, we first provide our experimental setup in Section IV.A. Then, we demonstrate the effectiveness of our MUP on ImageNet-compatible dataset[1] in Section IV.B. Finally, we perform some ablation studies in Section IV.C to investigate the impact of different hyperparameters.

A. Setup

Victim Models. We consider nine publicly available models as victim models, which have been widely used in previous work [2], [6]. The first three of them are normally trained models, including Inception-v3 (Inc-v3) [1], Inception-v4 (Inc-v4), and Inception-ResNet-v2 (IncRes-v2) [19]. The rest are robust models: Inc-v3$_{ens3}$, Inc-v3$_{ens4}$, and IncRes-v2$_{ens}$ [20], high-level representation guided denoiser (HGD) [21], input transformation through resizing and padding (R&P) [22], and the rank-3 submission in NIPS2017 adversarial competition (NIPS-r3)[2].

Surrogate Models. Following the setting of the comparative work ghost network (GN), we first choose Inc-v3 and Inc-v4 as the architectures of the surrogate model, since they were experimented in [9]. To demonstrate the versatility of MUP over more architectures, we additionally conduct experiments on the DenseNet121 (DN121) surrogate model.

Optimizers. We use MUP to assist three optimizers for generating adversarial examples to evaluate the compatibility of the proposed MUP with prior works, including MIM [2], SIM [6], and the state-of-the-art TAIG-R [7].

Hyperparameters. If not specified otherwise, the masking ratio r is set to 15%, 30% and 25% for the surrogate model Inc-v3, Inc-v4, and DN121, respectively. Notice that it is independent with respect to the victim model since the victim model is unknown in the assumptions of the transfer-based attack. For the hyperparameters related to different optimizers, we follow the recommended values of their corresponding papers to set them. Specifically, we set the step size β to 2, the number of iterations to 10, the momentum factor μ for MIM to 1.0, the number of scaled images m for SIM to 5, and the number of sampling images S for TAIG-R to 20. All the experiments are conducted with the maximum allowed magnitude of perturbation $\epsilon = 16$.

B. Results on ImageNet-compatible dataset

In this subsection, we will first show that using the proposed MUP can improve the success rate of existing optimizers. In addition, we compared the proposed MUP with the comparative work ghost network (GN), which was used to assist MIM in [9]. To follow the experimental setup of GN, we first consider assisting MIM, as well as Inc-v3 and Inc-v4 as the architecture of the surrogate models. Then, a DN121 is additionally considered as a surrogate model, which reflects the versatility of MUP for different architectures of the surrogate model. In contrast, GN cannot be applied directly to a new architecture, as it requires specifically designed locations for adding dropout layers, as we present in Section III.B. All of the experiments are conducted on the ImageNet-compatible dataset used in the NIPS 2017 adversarial competition.

We list the experimental results in Table I. The case where the surrogate model is the same as the victim model is not considered, as this is a white-box attack that we are not interested in. We also report the average attack success rates in Table I, which is averaged over all nine victim models for DN121 and over eight victim models other than itself for Inc-v3 and Inc-v4. From the results, we can see that the MUP-MIM significantly outperforms MIM in terms of attack success rates in all experiments. Specifically, when Inc-v3, Inc-v4, and DN121 are used as surrogate models, MUP-MIM improves the attack success rate by 5.1%, 5.8%, and 11.3%, respectively.

MUP and GN can also be compared based on the results in Table I. From it we can see that MUP-MIM outperforms GN-MIM in all experiments when Inc-v3 is used as the surrogate model. However, GN-MIM performs slightly better than MUP-MIM in attacking IncRes-v2$_{ens}$, R&D, and NIPS-r3 when Inc-v4 is used as the surrogate model. This indicates that the performance of MUP and GN are comparable when using MIM to generate adversarial examples and makes us curious about the comparison results between them when assisting an advanced optimizer.

Therefore, we further conduct experiments to compare the compatibility of GN and MUP with more optimizers. We considered SIM and TAIG-R and list the experimental results in Table II. Surprisingly, although GN performs well in assisting the MIM optimizer as claimed by [9], the results in Table II show that using GN is worse than not using it when assisting more advanced optimizers. For example, when using Inc-v3 as a surrogate model, GN decreases the attack success rates of SIM and TAIG-R optimizer by 1.1% and 4.7%, respectively, on average. This drop is even more significant when using Inc-v4 as a surrogate model. On the contrary, our MUP assists the SIM and TAIG-R well, bringing further improvements in terms of attack success rates in all experiments. As a result, compared to TAIG-R, using MUP improves the attack success

[1]https://www.kaggle.com/google-brain/nips-2017-adversarial-learning-development-set

[2]https://github.com/anlthms/nips-2017/tree/master/mmd

TABLE I
THE ATTACK SUCCESS RATES (%) OF DIFFERENT METHODS BASED ON MIM [2]. "-" INDICATES THE WHITE-BOX ATTACK.

Surrogate Model	Method	Average	Inc-v3	Inc-v4	IncRes-v2	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$	HGD	R&D	NIPS-r3
Inc-v3	MIM	23.4	-	54.1	48.1	21.8	21.3	10.7	6.5	10.0	14.4
	GN-MIM	26.2	-	61.6	58.9	23.1	22.4	11.6	6.6	10.3	15.2
	MUP-MIM	**28.5**	-	**66.2**	**64.4**	**25.1**	**23.2**	**12.8**	**6.9**	**12.6**	**16.4**
Inc-v4	MIM	23.8	64.2	-	49.9	18.3	18.6	10.3	6.0	9.9	12.9
	GN-MIM	29.0	75.2	-	63.3	23.7	22.0	**12.8**	5.4	**12.3**	**16.9**
	MUP-MIM	**29.6**	**76.8**	-	**65.4**	**24.0**	**23.0**	12.4	**7.3**	11.8	16.4
DN121	MIM	48.6	67.6	61.4	55.7	47.9	46.4	36.4	46.9	35.7	39.8
	MUP-MIM	**59.9**	**82.3**	**76.0**	**69.5**	**58.3**	**56.4**	**44.1**	**57.8**	**44.9**	**49.6**

TABLE II
THE ATTACK SUCCESS RATES (%) OF DIFFERENT METHODS BASED ON SIM [6] AND TAIG-R [7]. "-" INDICATES THE WHITE-BOX ATTACK.

Surrogate Model	Method	Average	Inc-v3	Inc-v4	IncRes-v2	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$	HGD	R&D	NIPS-r3
Inc-v3	SIM	37.7	-	72.1	71.4	37.3	36.1	20.2	16.0	19.7	28.9
	GN-SIM	33.6	-	74.4	72.2	30.6	29.0	15.7	9.5	15.4	22.3
	MUP-SIM	**42.8**	-	**83.6**	**83.0**	**41.8**	**40.5**	**21.8**	**17.3**	**22.1**	**32.4**
	TAIG-R	55.6	-	86.0	84.1	59.0	57.9	37.8	31.7	37.8	50.1
	GN-TAIG-R	50.9	-	89.5	87.6	51.9	51.5	31.0	23.1	30.7	41.7
	MUP-TAIG-R	**60.5**	-	**92.1**	**90.4**	**63.7**	**63.9**	**40.7**	**34.1**	**42.8**	**56.7**
Inc-v4	SIM	45.6	83.9	-	75.4	47.4	42.5	27.2	22.6	28.2	37.8
	GN-SIM	34.5	81.9	-	74.1	30.3	28.6	15.7	8.0	15.3	22.4
	MUP-SIM	**53.2**	**92.9**	-	**90.0**	**56.6**	**52.7**	**31.8**	**25.7**	**32.3**	**43.8**
	TAIG-R	61.4	89.3	-	86.2	64.8	61.9	47.4	37.1	48.0	56.3
	GN-TAIG-R	49.5	93.8	-	88.0	50.1	45.7	29.1	17.2	29.9	42.3
	MUP-TAIG-R	**66.6**	**94.9**	-	**94.0**	**72.1**	**68.7**	**50.2**	**38.4**	**51.6**	**62.9**
DN121	SIM	67.8	83.1	78.5	73.8	68.3	67.5	55.1	66.6	57.2	60.4
	MUP-SIM	**74.0**	**90.3**	**88.7**	**84.2**	**74.0**	**71.4**	**57.5**	**73.7**	**60.5**	**65.5**
	TAIG-R	85.7	91.9	92.1	88.8	87.1	85.6	77.2	86.7	79.2	82.5
	MUP-TAIG-R	**88.6**	**95.3**	**94.9**	**92.7**	**89.1**	**88.4**	**80.1**	**89.9**	**81.5**	**85.4**

rates by 4.9%, 4.2%, and 2.9% on average respectively when Inc-v3, Inc-v4, and DN121 are used as surrogate models. In addition, MUP improves the attack success rates of DN121 on nine victim models to at least **80.1%** (IncRes-v2$_{ens}$). It should be pointed out that a high attack success rate is what is of interest in the field of transfer-based attack, since the goal of a malicious adversary is to fool a DNN system with as high a probability as possible. Thus, our MUP is more practical in real-world attack scenarios, although it is not always better than the GN when using a relatively weak optimizer (MIM).

C. Ablation Studies

In this subsection, we first conduct experiments to investigate the effect of the masking ratio r. Then, we have considered another metric for calculating the parameter importance scores, to highlight the effectiveness of the Taylor expansion-based metric now in use.

The masking ratio r plays an important role in MUP. When r is equal to zero, it means that no parameters are masked. On the other hand, if r is too large, most of the parameters of the surrogate model will be set to zero, which will impair the accuracy of the model prediction and lead to poor transferability. A natural question to ask is how should the hyperparameter r be set. Will the optimal masking ratio r vary greatly for different victim models? To answer this question, we enumerate different r from $\{0\%, 5\%, 10\%, \cdots, 45\%, 50\%\}$ and conduct experiments on Inc-v3, Inc-v4 and DN121, respectively. We fix the optimizer for generating adversarial examples as TAIG-R since we are more interested in the performance of MUP to assist a stronger optimizer. The experimental results of using different masking ratios to assist TAIG-R are shown in Fig. 3, which shows that for the same surrogate model, the curves of attack success rates versus masking ratio r are similar for all different victim models. Specifically, all experiments show that the attack success rate will initially increase, then reach the saturation region, and begin to decline rapidly. Moreover, *the saturation region does not depend on the victim model*, because all curves of the same surrogate model have highly overlapping saturation regions. Therefore, determining the hyperparameter r of MUP on a new surrogate model requires only choosing any public model as the victim model to search for an optimal r. This is critical for the transfer-based attack because the adversary has no knowledge of the victim model.

Finally, we additionally considered a metric for calculating the parameter importance scores, which is simply using the absolute value of the parameters as the importance scores, namely $V_i = |\theta_i|$. This metric was previously applied in

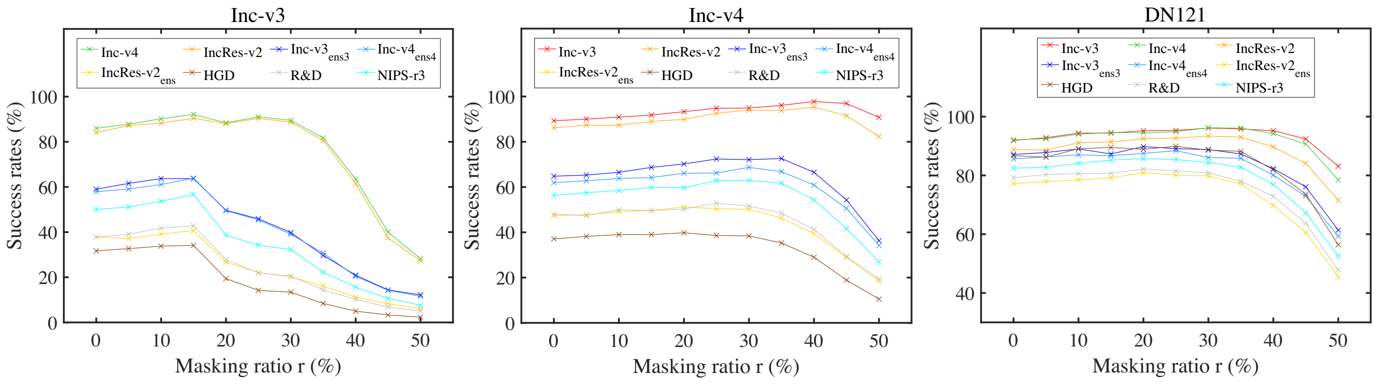

Fig. 3. The attack success rates (%) versus hyperparameter r curves. The surrogate models are Inc-v3, Inc-v4 and DN121 from left to right.

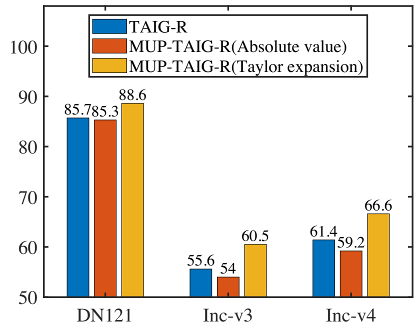

Fig. 4. The average attack success rate (%) for different methods using surrogate models Inc-v3, Inc-v4 and DN121, respectively.

model pruning by Han et al. [15]. By iteratively removing the parameters with the smallest absolute value, they compressed a VGG16 [17] model by 13X with no loss of accuracy. By comparing it with the Taylor expansion-based metric (see (1)), the only difference is that the latter has a gradient term $\frac{\partial \mathcal{L}(x,y,\theta)}{\partial \theta_i}$. We use the absolute value-based metric to perform the MUP algorithm with the same hyperparameter r as the MUP with the Taylor expansion-based metric, i.e., set to 15%, 30%, and 25% Inc-v3, Inc-v4, and DN121, respectively. The experimental results of using different metrics to assist TAIG-R are shown in Fig. 4, which shows that MUP with the absolute value-based metric does not boost the transfer-based attack. We believe this is because the absolute value-based metric is input-independent and ignores the features of inputs. In contrast, the Taylor expansion-based metric compensates for this shortcoming by multiplying an input-dependent gradient term. This indicates that the metric for calculating the importance scores is crucial for the proposed MUP-based methods.

V. Conclusions

In this paper, we propose the idea of improving the transferability of the adversarial examples by masking unimportant parameters (MUP). Based on this idea, we propose a technique for boosting transfer-based attacks, which utilizes a Taylor expansion-based metric to evaluate the parameter importance scores, and improves existing optimizers for the transfer-based attack by masking unimportant parameters. Extensive experiments verify the effectiveness of the proposed method.

VI. Acknowledgment

This work was supported by the National Key Research and Development Plan of China (2020AAA0103502), and National Key Research and Development Project of China (No. 2021ZD0110502).

References

[1] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
[2] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9185–9193.
[3] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2730–2739.
[4] Y. Dong, T. Pang, H. Su, and J. Zhu, "Evading defenses to transferable adversarial examples by translation-invariant attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4312–4321.
[5] X. Wang, X. He, J. Wang, and K. He, "Admix: Enhancing the transferability of adversarial attacks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 158–16 167.
[6] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft, "Nesterov accelerated gradient and scale invariance for adversarial attacks," in *International Conference on Learning Representations*, 2019.
[7] Y. Huang and A. W.-K. Kong, "Transferable adversarial attack based on integrated gradients," in *International Conference on Learning Representations*, 2022.
[8] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *International Conference on Learning Representations*, 2017.
[9] Y. Li, S. Bai, Y. Zhou, C. Xie, Z. Zhang, and A. Yuille, "Learning transferable adversarial examples via ghost networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 458–11 465.
[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[11] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2013.
[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[13] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” in *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.
[14] Y. LeCun, J. Denker, and S. Solla, “Optimal brain damage,” *Advances in neural information processing systems*, vol. 2, 1989.
[15] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” in *International Conference on Learning Representations*, 2016.
[16] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” in *International Conference on Learning Representations*, 2017.
[17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
[18] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.
[19] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Thirty-first AAAI conference on artificial intelligence*, 2017.
[20] F. Tramèr, D. Boneh, A. Kurakin, I. Goodfellow, N. Papernot, and P. McDaniel, “Ensemble adversarial training: Attacks and defenses,” in *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, 2018.
[21] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, “Defense against adversarial attacks using high-level representation guided denoiser,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1778–1787.
[22] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, “Mitigating adversarial effects through randomization,” in *International Conference on Learning Representations (ICLR)*, 2018.