TRAINING META-SURROGATE MODEL FOR TRANSFERABLE ADVERSARIAL ATTACK.

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider adversarial attacks to a black-box model when no queries are allowed. In this setting, many methods directly attack surrogate models and transfer the obtained adversarial examples to fool the target model. Plenty of previous works investigated what kind of attacks to the surrogate model can generate more transferable adversarial examples, but their performances are still limited due to the mismatches between surrogate models and the target model. In this paper, we tackle this problem from a novel angle—instead of using the original surrogate models, can we obtain a **Meta-Surrogate Model** (MSM) such that attacks to this model can be easier transferred to other models? We show that this goal can be mathematically formulated as a well-posed (bi-level-like) optimization problem and design a differentiable attacker to make training feasible. Given one or a set of surrogate models, our method can thus obtain an MSM such that adversarial examples generated on MSM enjoy eximious transferability. Comprehensive experiments on Cifar-10 and ImageNet demonstrate that by attacking the MSM, we can obtain stronger transferable adversarial examples to fool black-box models including adversarially trained ones, with much higher success rates than existing methods.

1 INTRODUCTION

The developments of Convolutional Neural Network (CNN) (LeCun et al., 1995; Krizhevsky et al., 2012) have greatly promoted the advancements in Computer Vision (Ren et al., 2016). However, previous works (Goodfellow et al., 2014; Carlini & Wagner, 2017; Croce & Hein, 2020a; Ganeshan et al., 2019) have shown a critical robustness issue that CNN models are vulnerable to human-imperceptible perturbations of input images, also known as adversarial examples (AEs). The design of AEs is useful for revealing the security threats on machine learning systems (Croce & Hein, 2020b) and for understanding the representations learned by CNN models (Ilyas et al., 2019).

In this paper, we consider the problem of black-box attack, where the target victim model is entirely hidden from the attacker. In this setting, standard white-box attacks (Moosavi-Dezfooli et al., 2016; Carlini & Wagner, 2017) or even query-based black-box attacks (Ilyas et al., 2018; Cheng et al., 2018; 2020; 2019) cannot be used, and the prevailing way to attack the victim is through transfer attack (Papernot et al., 2017; Wu et al., 2018). In transfer attack (Demontis et al., 2019; Dong et al., 2018), the attackers commonly generate AEs by attacking one or an ensemble of **surrogate models** and hope the obtained AEs can also successfully fool the victim black-box model.

Although great efforts have been made to improve the transferability of adversarial attacks (Tramèr et al., 2017; Xie et al., 2019; Wu et al., 2020a), the transfer attack-based methods still encounter poor success rates, especially when attacking adversarially trained target models. This is caused by a fundamental limitation of current approaches—they all leverage the surrogate models trained by standard learning tasks (e.g., classification, object detection), while it is not always the case that attacks fooling such models can be easily transferred. We thus pose the following important question on transfer attack that has not been well studied in the literature: Instead of using standard (naturally trained) models as surrogate, can we artificially construct another **Meta-Surrogate Model** (MSM) such that attacks to this model can be easier transferred to other models?

We answer this question in the affirmative by developing a novel black-box attack pipeline called **Meta-Transfer Attack** (MTA). Assume a set of source models (standard surrogate models) are given, instead of directly attacking these source models, our algorithm aims to obtain a "meta-surrogate

model (MSM)", which is designed in the way that attacks to this model can be easier transferred to fool other models, and conduct attacks on the MSM to obtain transferable AEs. We show that this goal can be mathematically formulated as a well-posed (bi-level-like) training objective by unrolling the attacks on the MSM and defining a loss to measure the transferability of the resulting AEs. To avoid discrete operations in the white-box attack, we propose a Customized PGD attacker that enables back-propagation through the whole procedure. With this bi-level-like optimization (Finn et al., 2017; Qin et al., 2020), the source models supervise the MSM to improve the transferability of the AEs created on it. Through extensive experiments on various models and datasets, we show that the proposed MTA method leads to significantly improved transfer attacks, demonstrating the effectiveness of the MSM.

We summarize the main contributions of our work as follows. 1) We propose a novel MTA framework to train an MSM to improve the transferability of AEs. To the best of our knowledge, our work is the first attempt to explore a better surrogate model for producing stronger transferable AEs. 2) We compare MTA with state-of-the-art transfer attack methods (*e.g.*, MI (Dong et al., 2018), DI (Xie et al., 2019), TI (Dong et al., 2019), SGM (Wu et al., 2020a), AEG (Bose et al., 2020), IR (Wang et al., 2021a), SI-NI (Lin et al., 2020)) on Cifar-10 (Krizhevsky et al., 2009) and Imagenet (Deng et al., 2009). The comparisons demonstrate the effectiveness of the proposed MTA—the AEs generated by attacking MSM significantly outperform previous methods, in attacking both naturally trained and adversarially trained black-box target models.

2 BACKGROUND

Adversarial attacks. Szegedy et al. (2014) reveals the interesting phenomenon that CNN models are vunerable to adversarial attacks. After that, many attacks have been developed (Gao et al., 2020; Zhou et al., 2018; Wu et al., 2020b; Li et al., 2020b; Kaidi et al., 2019; Sriramanan et al., 2020). Adversarial attacks can be mainly classified into white-box and black-box attacks (Maksym et al., 2020) according to how much information about the target model is exposed to the attacker. White-box attacks (Kurakin et al., 2016) are often more effective than than black-box attacks (Brendel et al., 2017; Cheng et al., 2018; 2020) as they can leverage full knowledge of the target model including the model weights and architecture. For example, Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) uses 1-step gradient ascent to produce adversarial examples that enlarge the model's loss. Projected gradient descent (PGD) attack can be viewed as a multi-step FGSM attack (Madry et al., 2018). Many other white-box attacks have also been developed by leveraging full information of the target model (Moosavi-Dezfooli et al., 2016; Croce & Hein, 2020a). In the black-box setting, query-based black-box attacks (Huang & Zhang, 2020; Du et al., 2020) assume model information is hidden but attackers can query the model and observe the corresponding hard-label or soft-label predictions. Among them, (Chen et al., 2017; Ilyas et al., 2018) considered soft-label probability predictions and (Chen et al., 2020; Huang & Zhang, 2020; Cheng et al., 2018) considered hard-label decision-based predictions. Considering that using a large number of queries to attack an image is impractical, several works try to further reduce the query counts (Li et al., 2020a; Wang et al., 2020).

Transferability of adversarial examples. In this paper, we consider the black-box attack scenario when the attacker cannot make any query to the target model (Lin et al., 2020; Huang et al., 2019; Wang et al., 2021b). In this case, the common attack method is based on transfer attack—the attacker generates AEs by attacking one or few surrogate models and hopes the AEs can also fool the target model (Papernot et al., 2016; Liu et al., 2017; Yuan et al., 2021; Zhou et al., 2018). Compared with query-based attacks, crafting AEs from the surrogate model consumes less computational resources and is more realistic in practice. Along this direction, subsequent works have made attempts to improve the transferability of AEs (Guo et al., 2020; Wu et al., 2020c; Naseer et al., 2019; Li et al., 2020c; Wang & He, 2021). For instance, Dong et al. (2018) boosted the transferability by integrating the momentum term into the iterative process. Other techniques like data augmentations (Xie et al., 2019), exploiting gradients of skip-connection (Wu et al., 2020a), and negative interaction between pixels (Wang et al., 2021a) also contribute to stronger transferable attacks. In addition to using the original surrogate models, AEG (Bose et al., 2020) adversarially trains a robust classifier together with an encoder-decoder-based transferable perturbation generator. After the training, AEG uses the generator to generate transferable AEs to attack a set of classifiers. Compared to all the existing works, our method is the first that meta-trains a new meta-surrogate model (MSM) such that attacks on MSM can be easier transferred to other models. This not only differs from all the previous methods that attack standard surrogate models but also differs from the encoder-decoder based method such as AEG (Bose et al., 2020).

3 METHODOLOGY

We consider the black-box attack setting where the target model is hidden to the attacker and queries are not allowed. This setting is also known as the transfer attack setting (Dong et al., 2018; 2019; Xie et al., 2019; Wang et al., 2021a) and the attacker 1) cannot access the weight, the architecture, and the gradient of the target model; and 2) cannot querying the target model. The attacker can access 1) the dataset used by the target model; and 2) a single or a set of **surrogate models** (also known as **source models**) that may share the dataset with the target model. For example, it is common to assume that the attacker can access one or multiple well-performed (pretrained) image classification models. Existing transferable adversarial attack methods conduct various attacks to these models and hope to get transferable AEs that can fool an unknown target model. Instead of proposing another attack method on surrogate models, we propose a novel framework MTA to train a **Meta-Surrogate Model (MSM)** with the goal that attacking the MSM can generate stronger transferable AEs than directly attacking the original surrogate models. When evaluating, the transferable AEs are generated by attacking the MSM with standard white-box attack methods (e.g., PGD attack). In the following, we will first review exiting attacks and then show how to form a bi-level optimization objective to train the MSM model.

3.1 REVIEWS OF FGSM AND PGD

We follow the settings of existing works (Dong et al., 2018; Xie et al., 2019; Wu et al., 2020a; Wang et al., 2021a) to focus on untargeted attack, where the attack is considered successful as long as the perturbed image is wrongly predicted.

FGSM (Goodfellow et al., 2014) conducts one-step gradient ascent to generate AEs to enlarge the prediction loss. The formulation can be written as

$$x_{adv} = \text{Clip}\big(x + \epsilon \cdot \text{sign}(\nabla_x L(f(x), y))\big), \tag{1}$$

where x is a clean image and y is the corresponding label; ϵ is the attack step size that determines the maximum L_∞ perturbation of each pixel; f is the victim model that is transparent to the FGSM attacker; Clip is the function that clipping the values of x_{adv} to the legal range (*e.g.*, clipping the RGB AEs to the range of $[0, 255]$); L is usually the cross-entropy loss.

PGD (Kurakin et al., 2016), also known as I-FGSM attack, is a multi-step extension of FGSM. The formulation of PGD is

$$x^k_{adv} = \text{Clip}\big(x^{k-1}_{adv} + \frac{\epsilon}{T} \cdot \text{sign}(\nabla_{x^{k-1}_{adv}} L(f(x^{k-1}_{adv}), y))\big). \tag{2}$$

x^k_{adv} is the AEs generated in the k-th gradient ascent step. Note that x^0_{adv} is the clean image equals to x. Eq 2 will be run for T iterations to obtain x^T_{adv} with perturbation size ϵ.

3.2 META-TRANSFER ATTACK

How to train the MSM where attacks to this model can be easier transferred to other models? We show this can be formulated as a bi-level training objective. Let $\mathcal{A}$ denote an attack algorithm (e.g., FGSM or PGD) and $\mathcal{M}_\theta$ denote the **MSM** parameterized by θ. For a given image x, the AE generated by attacking $\mathcal{M}_\theta$ can be denoted as $\mathcal{A}(\mathcal{M}_\theta, x, y)$. For example, if $\mathcal{A}$ is FGSM, then $\mathcal{A}(\mathcal{M}_\theta, x, y) = x_{adv} = \text{Clip}\big(x + \epsilon \cdot \text{sign}(\nabla_x L(\mathcal{M}_\theta(x), y))\big)$. Since in the attack time we only have access to a set of source models $\mathcal{F}_1, \ldots, \mathcal{F}_N$, we can evaluate the transferability of the adversarial example $\mathcal{A}(\mathcal{M}_\theta, x, y)$ on the source models and optimize the MSM via maximizing the adversarial losses of those N source models, leading to the following training objective:

$$\arg\max_\theta \mathbb{E}_{(x,y)\sim D}\big[\sum_{i=1}^N L(\underbrace{\mathcal{F}_i(\overbrace{\mathcal{A}(\mathcal{M}_\theta, x, y)}^{\text{AE}})}_{\mathcal{F}_i's \text{ prediction for AE}}, y)\big], \tag{3}$$

where D is the distribution of training data. The structure of this objective and the training procedure can be illustrated in Figure 1, where we can view it as a meta-learning or bi-level optimization method. At the lower level, the AE is generated by a white-box attack (usually gradient ascent) on MSM, while at the higher level, we feed the AE to the source models to compute the robust loss. Solving Eq 3 will find an MSM where attacking it leads to stronger transferable AEs. The optimization steps of Eq 3 are detailed below.

First, $\mathcal{A}$ should be some strong white-box attacks, such as FGSM or PGD. However, directly using those attacks will make the gradient of meta training objective Eq 3 ill-defined since the sign function in both FGSM and PGD introduce a discrete operation. This results in that the gradient back-propagating through sign be zero and further prohibits the training of the MSM.

To overcome this challenge, we design $\mathcal{A}$ as an approximation of PGD and denote it as Customized PGD. Section 3.3 will show more explanation about how the sign function in PGD prohibits back-propagation and how Customized PGD enables the back-propagation. The crucial difference between PGD and the Customized PGD is the operation to the gradient $\nabla_{x_{adv}^{k-1}} L(\mathcal{M}_\theta(x_{adv}^{k-1}), y)$, where L is the cross entropy loss. For simplicity, we denote the vanilla gradient $\nabla_{x_{adv}^k} L(\mathcal{M}_\theta(x_{adv}^k), y)$ at the k-th step as g^k, and generate another map g_{ens}^k via Eq 4:

$$\begin{cases} g_1^k = \frac{g^k}{\text{sum}(\text{abs}(g^k))} \\ g_t^k = \frac{2}{\pi} \cdot \arctan(\frac{g^k}{\text{mean}(\text{abs}(g^k))}) \\ g_s^k = \text{sign}(g^k) \\ g_{ens}^k = g_1^k + \gamma_1 \cdot g_t^k + \gamma_2 \cdot g_s^k \end{cases} \tag{4}$$

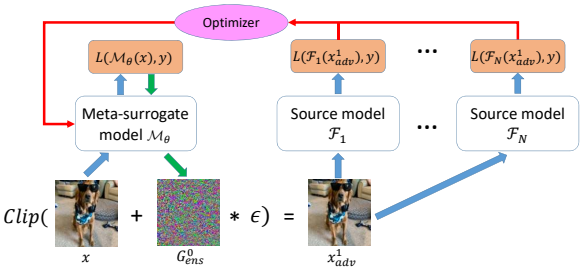

Figure 1: The framework of the proposed MTA when $T = 1$ and $\mathcal{A}(\mathcal{M}_\theta(x)) = x_{adv}^1$. The clean image x is first feed into the MSM $\mathcal{M}_\theta$ and obtain the loss $L(\mathcal{M}_\theta(x), y)$. Next we back-propagate the loss and use Eq 4 to obtain the noise g_{ens}^0. Then, via Eq 5, we obtain the adversarial example x_{adv}^1 which will be feed into the source models $\mathcal{F}_1$, $\mathcal{F}_2$, ..., and $\mathcal{F}_N$. Finally, by maximizing the source models' loss, we can optimize the MSM to learn a particular weight so that the adversarial example x_{adv}^1 attacking it can fool source models.

Note that we set $\gamma_1 = \gamma_2 = 0.01$ as default for all the experiments. Both g_1^k and g_t^k ensure the objective in Eq 3 be differentiable with respect to the MSM's weight θ; $\arctan(\cdot)$ is a smooth approximation of sign and $\frac{1}{\text{mean}(\text{abs}(g^k))}$ prevents arctan from falling into the saturation or linear region. The item $\gamma_2 \cdot g_s^k$ provides the lower-bound for each pixel's perturbation in g_{ens}^k. The experiments in Section 4.3 will demonstrate the importances of g_t^k and g_s^k for Customized PGD. With Eq 4, the Customized PGD conducts the following update to generate AE:

$$x_{adv}^k = \text{Clip}(x_{adv}^{k-1} + \frac{\epsilon_c}{T} \cdot g_{ens}^{k-1}). \tag{5}$$

Note that ϵ_c differs from the perturbation ϵ in FGSM and PGD because g_{ens}^{k-1} in our update is not a sign vector and its size will depend on the magnitude of the original gradient. Finally, we get x_{adv}^T after T iterations of Eq 5.

Second, we feed x_{adv}^T into N source models and calculate the corresponding adversarial losses $L(\mathcal{F}_i(x_{adv}^T), y)$ for all $i = 1, \ldots, N$. Larger losses of the N source models indicate a higher likelihood that x_{adv}^T fooling the MSM can transfer to other models.

Third, we optimize the MSM by maximizing the objective function defined in Eq 3. The update rule can be written as

$$\theta' = \theta + \alpha \cdot \textstyle\sum_{i=1}^N \nabla_\theta L(\mathcal{F}_i(x_{adv}^T), y), \tag{6}$$

where x_{adv}^T can be written as a function of θ by unrolling the attack update rule Eq 5 T times. We will show how to explicitly compute the gradient in Section 3.3. With this training procedure, the MSM is trained to learn a particular weight with which the white-box AEs fooling it can also fool other models. We summarize the training and testing of MTA in Algorithm 1 and Section A.1, respectively. Each capitalized notation represents a batch of the variable denoted with lower case. For example, X denotes a batch of x. Note that Customized PGD is just a continuous approximation of PGD used to train the MSM. In the inference phase, we use standard attacks such as PGD to craft AEs on the MSM.

3.3 Gradient Calculation

In the calculation we set both N and T in Eq 6 to 1, so the gradient in Eq 6 is $\nabla_\theta L(\mathcal{F}_1(x_{adv}^1), y)$. According to Eq 5, we can replace x_{adv}^1 in Eq 6 with $\text{Clip}(x_{adv}^0 + \epsilon_c \cdot g_{ens}^0)$, where x_{adv}^0 equals to x. For simplicity, we ignore the clip function in the analysis and simplify the derivation as $\nabla_\theta L(\mathcal{F}_1(x + \epsilon_c \cdot g_{ens}^0), y)$. By chain rule and since x is independent to θ, we can further rewrite this

as

$$\frac{\partial L(\mathcal{F}_1(x+\epsilon_c \cdot g^0_{ens}), y)}{\partial g^0_{ens}} \cdot \frac{\partial g^0_{ens}}{\partial \theta}. \tag{7}$$

By replacing g^0_{ens} with Eq 4, the second term of Eq 7 can be expanded as

$$\nabla_\theta g^0_{ens} = \nabla_\theta g^0_1 + \gamma_1 \cdot \nabla_\theta g^0_t + \gamma_2 \cdot \nabla_\theta g^0_s. \tag{8}$$

Note that g^0_s equals to $\text{sign}(g^0)$ and the sign function introduces discrete operation so that the gradient of g^0_s with respect to θ becomes 0 (unless $g^0 = 0$). Therefore, $\nabla_\theta g^0_{ens}$ can be further written as

$$\begin{aligned} \nabla_\theta g^0_{ens} &= \nabla_\theta g^0_1 + \gamma_1 \cdot \nabla_\theta g^0_t \\ &= \nabla_\theta\left(\frac{\nabla_x L(\mathcal{M}_\theta(x), y)}{\text{sum}(\text{abs}(\nabla_x L(\mathcal{M}_\theta(x), Y)))}\right) + \gamma_1 \cdot \nabla_\theta\left(\arctan\left(\frac{\nabla_x L(\mathcal{M}_\theta(x), y)}{\text{mean}(\text{abs}(\nabla_x L(\mathcal{M}_\theta(x), y)))}\right)\right). \end{aligned} \tag{9}$$

In this formulation, $\nabla_x L(\mathcal{M}_\theta(x), y)$ depends on θ and the second-order derivative of $\nabla_x L(\mathcal{M}_\theta(x), y)$ *w.r.t* θ can be obtained with lots of deep learning libraries (Abadi et al., 2016; Paszke et al., 2017). In summary, by integrating Eqs.6-9, the MSM can be optimized by an SGD-based optimizer.

4 EXPERIMENT

We conduct experiments to show that the proposed method, under the same set of source models, can generate stronger transferable AEs than existing transfer attack methods.

Algorithm 1 Training of Meta-Transfer Attack

input: N source models $\mathcal{F}_1, \ldots, \mathcal{F}_N$, Training set $\mathbb{D}$, batch size b, initialized MSM $\mathcal{M}_\theta$.
output: Optimized weight θ.

while not done **do**
  sample data (X=$[x_1, \ldots, x_b]$, Y=$[y_1, \ldots, y_b]$) $\in \mathbb{D}$
  $X^0_{adv} = X$
  for k in [1, 2, ..., T]:
    $G^k = \nabla_{X^{k-1}_{adv}} L(\mathcal{M}_\theta(X^{k-1}_{adv}), Y)$
    obtain G^k_{ens} via Eq 4
    obtain X^k_{adv} via Eq 5
  end for
  for each source model $\mathcal{F}_i \in [\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_N]$, **do**
    evaluate X^T_{adv} on $\mathcal{F}_i$ and obtain $L(\mathcal{F}_i(X^T_{adv}), Y)$
  end for
  $\theta = \theta + \alpha \cdot \nabla_\theta \sum_i^N L(\mathcal{F}_i(X^T_{adv}), Y)$
return θ

We first present our general experimental settings. **1)** We conduct experiments on both Cifar-10 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009). **2)** We compare the proposed MTA with seven state-of-the-art transferable adversarial attack methods, including MI (Dong et al., 2018), DI (Xie et al., 2019), TI (Dong et al., 2019), SGM (Wu et al., 2020a), SI-NI (Lin et al., 2020), AEG (Bose et al., 2020), IR (Wang et al., 2021a), and FIA (Wang et al., 2021b). Note that since SGM is based on enlarging the gradient of skip connections, we only include this method on ImageNet experiments when the source models have sufficient skip connections. AEG is compared only on Cifar-10 because the official AEG is evaluated only on small scale datasets (Mnist and Cifar-10), and it is computational costly to train the perturbation generator on large-scale datasets. FIA is implemented only on ImageNet using the same intermediate feature layers introduces in (Wang et al., 2021b). **3)** Since the number of attack iterations T is different between training and testing, we denote it as T_t in training and T_v in testing respectively to avoid confusion. **4)** When training the MSM, we use the Customized PGD with γ_1=γ_2=0.01 to attack the MSM. When evaluating, we use PGD with T_v=10 and ϵ=15 to attack the MSM. **5)** When using the baseline methods to generate AEs on multiple source models, we follow Dong et al. (2018) to ensemble the logits of the source models before loss calculation. **6)** We use source and target models to train and to evaluate the MSM, respectively. **7)** For fair comparisons between MTA and baselines, we implement baselines with the number of iterations T=10 and ϵ=15, and other hyper-parameters are tuned for their best possible performances (implementations are detailed in Section A.8). **8)** More experiments (*e.g.*, targeted transfer attack, attacks with smaller ϵ, more comparisons between MTA and baselines) will be shown in Section A.3.

4.1 EXPERIMENTS ON CIFAR-10

4.1.1 EXPERIMENTAL CONFIGURATIONS

On Cifar-10, we use 8 source models including ResNet-10, -18, -34 (He et al., 2016), SeResNet-14, -26, -50 (Hu et al., 2018), MobileNet-V1 (Howard et al., 2017), and -V2 (Sandler et al., 2018) to train the MSM. To ensure mismatches between the source and target models and to avoid saturated

Table 1: Transfer attack success rates on eight target networks on Cifar-10. The MSM is trained with eight source models. From left to right, the eight target models are MobileNet-V3 (MN-V3), ShuffleNet-V1 (SN-V1), -V2 (SN-V2), SqueezeNet-A (SN-A), -B (SN-B), and adversarially trained ResNet-18 (Res-18$_{adv}$), ResNet-34 (Res-34$_{adv}$), and SeResNet-50 (SeRes-50$_{adv}$).

Method	MN-V3	SN-V1	SN-V2	SN-A	SN-B	Res-18$_{adv}$	Res-34$_{adv}$	SE-50$_{adv}$
PGD	51.8%	64.1%	49.4%	57.2%	56.3%	67.7%	63.9%	63.4%
DI	57.8%	72.5%	56.4%	65.7%	64.6%	80.7%	73.1%	71.0%
MI	70.2%	85.6%	72.6%	83.7%	83.0%	92.9%	90.9%	89.1%
A-PGD	74.1%	88.9%	75.8%	84.2%	83.6%	90.7%	89.3%	89.1%
TI	54.5%	59.9%	54.2%	71.8%	71.4%	57.6%	46.3%	46.6%
AEG	90.8%	92.5%	85.8%	91.3%	91.0%	96.1%	93.6%	93.1%
IR	59.3%	77.9%	62.5%	71.6%	69.1%	79.8%	73.7%	72.1%
MTA	**91.8%**	**98.4%**	**90.9%**	**94.9%**	**93.8%**	**98.4%**	**96.5%**	**97.1%**
MTA$_{\gamma_1=0}$	70.0%	80.9%	68.5%	58.5%	59.4%	67.7%	59.2%	68.9%
MTA$_{\gamma_2=0}$	90.0%	98.2%	90.5%	93.9%	93.1%	97.6%	96.0%	96.3%
MTA$_{dense}$	86.9%	96.2%	87.1%	89.0%	87.6%	96.2%	91.3%	93.6%

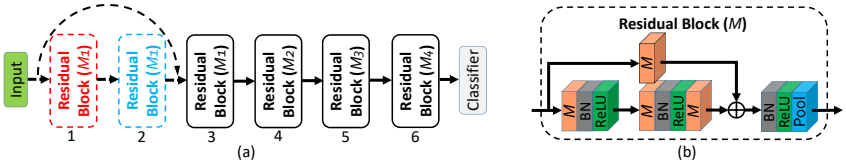

Figure 2: (a) The structures of ResNet-13 and -19. ResNet-13 contains the top four blocks in the solid-line box and the classifier. ResNet-19 contains all the six blocks and the classifier. The parameter $M*$ of each block denotes the number of filters of its convolution layers. (b) The detailed structure of residual block. The orange cube is the convolution layer and the number on it denotes its number of filters. Pool in the sixth block is global-average pooling while all the other pool is max-pooling with both stride and kernel size of 2×2. The convolution layer in the shortcut path uses 1×1 kernel size while all the other convolution layers use 3×3.

transfer attack performances (*i.e.*, attack success rates close to 100%), we select the 8 target models including MobileNet-V3 (Howard et al., 2019), ShuffleNet-V1, -V2 (Zhang et al., 2018), SqueezeNet-A, -B (Iandola et al., 2016), and adversarially trained ResNet-18, -34, and SeResNet-50. The network architectures of all 16 models are defined on public GitHub repositories[1,2,3]. We train all the source and target models and describe the training details of these models in Section A.2. The trained models and the code will be released to the community for reproducibility.

Training the MSM. The default network architecture of the MSM is ResNet-13 shown in Figure 2, with $M1$, $M2$, $M3$, and $M4$ set to 64, 128, 256, and 512, respectively. We use the 8 source models to train the MSM for 60 epochs with the number of attack steps T_t of 7. ϵ_c of the Customized PGD is initialized to 1,600 and is exponentially decayed by 0.9× for every 4,000 iterations. The learning rate α and the batch size are set to 0.001 and 64, respectively.

Evaluating the MSM. On each target model, we only attack the correctly classified test images because attacking wrongly classified clean images is less meaningful.

4.1.2 Experimental results

Table 1 shows the experimental results. The recently proposed white-box attack method A-PGD (Croce & Hein, 2020b) is also treated as a compared method here. Apparently, MTA performs much better than all the previous methods with significantly increased transfer attack success rates. For example, compared with IR (Wang et al., 2021a), MTA improves the success rates by 54.8%, 26.3%, 45.4%, 32.5%, 35.7%, 23.3%, 30.9%, and 34.7% on the eight target models. The results of MTA$_{\gamma_1=0}$, MTA$_{\gamma_2=0}$, and MTA$_{dense}$ will be discussed in ablation study (Section 4.3).

[1]https://github.com/yxlijun/cifar-tensorflow
[2]https://github.com/TropComplique/ShuffleNet-tensorflow
[3]https://github.com/TropComplique/shufflenet-v2-tensorflow

Table 2: Transfer attack results on seven black-box networks when using one source model.

Source	Method	Inc-V3	Inc-V4	IncRes-V2	Res-152	Inc-V3$_{ens3}$	Inc-V3$_{ens4}$	IncRes-V2$_{ens}$
Inc-V3	DI	/	35.2%	28.2%	22.3%	5.1%	4.3%	2.5%
	MI	/	38.1%	35.8%	29.6%	9.1%	8.8%	4.5%
	MI-DI	/	61.7%	57.3%	48.0%	13.6%	12.0%	6.5%
	SI-NI	/	63.8%	62.0%	51.7%	25.5%	25.2%	12.4%
	IR	/	33.6%	28.1%	15.9%	5.1%	5.5%	3.0%
	FIA	/	69.0%	66.8%	52.5%	29.3%	27.7%	14.9%
	MTA	99.9%	90.9%	87.3%	74.1%	67.7%	39.3%	26.1%
	MTA-IR	/	**95.5%**	**93.2%**	**85.0%**	**83.5%**	**56.9%**	**40.7%**
Inc-V4	DI	44.9%	/	30.5%	26.7%	5.9%	5.5%	3.3%
	MI	52.7%	/	41.8%	37.3%	12.4%	11.0%	5.8%
	MI-DI	69.1%	/	58.7%	49.3%	16.6%	14.1%	8.2%
	SI-NI	74.6%	/	67.3%	61.6%	39.2%	35.9%	22.0%
	IR	46.5%	/	33.2%	18.9%	8.1%	8.8%	4.9%
	FIA	63.6%	/	55.2%	45.9%	28.5%	26.1%	16.8%
	MTA	87.3%	99.9%	84.7%	73.1%	61.7%	38.2%	29.0%
	MTA-IR	**93.3%**	/	**90.5%**	**82.0%**	**77.2%**	**57.7%**	**44.9%**
IncRes-V2	DI	46.9%	42.0%	/	29.5%	8.6%	6.5%	5.5%
	MI	53.2%	45.2%	/	38.8%	16.2%	13.3%	9.7%
	MI-DI	64.7%	61.7%	/	50.6%	23.7%	18.6%	13.6%
	SI-NI	**78.2%**	**70.7%**	/	63.8%	45.2%	38.8%	**32.9%**
	IR	49.7%	44.9%	/	25.2%	13.6%	11.2%	10.9%
	FIA	63.2%	57.8%	/	51.3%	35.1%	30.3%	25.0%
	MTA	44.7%	41.7%	98.0%	57.9%	23.5%	19.4%	17.5%
	MTA$_{Inc}$	64.3%	51.7%	/	76.0%	46.2%	39.3%	27.5%
	MTA-IR$_{Inc}$	66.2%	52.3%	/	**78.3%**	**49.0%**	**42.2%**	31.7%
Res-152	DI	51.8%	48.1%	40.6%	/	9.7%	8.3%	6.2%
	MI	50.2%	44.9%	39.4%	/	13.9%	12.0%	7.8%
	MI-DI	**76.2%**	73.3%	**69.5%**	/	24.6%	21.1%	12.7%
	SI-NI	59.6%	50.1%	51.3%	/	37.9%	34.0%	20.7%
	IR	42.3%	33.8%	34.1%	/	22.0%	20.6%	16.2%
	FIA	73.8%	67.2%	67.9%	/	48.0%	43.7%	30.4%
	MTA	70.7%	77.5%	62.8%	99.1%	53.0%	59.2%	56.3%
	MTA-IR	72.8%	**78.0%**	64.3%	/	**54.9%**	**63.0%**	**59.3%**
	SGM$^{*}_{\epsilon=16}$	57.2%	48.6%	45.4%	/	31.6%	27.8%	20.0%
	IR$^{*}_{\epsilon=16}$	53.6%	50.6%	46.0%	/	/	/	/
	MTA$_{\epsilon=16}$	**76.0%**	**80.5%**	**67.6%**	/	**60.5%**	**68.4%**	**62.6%**

4.2 Experiments on ImageNet

4.2.1 Experimental configurations

We directly use the public trained ImageNet models[4,5,6] including ResNet-50, -101, -152 (He et al., 2016), DenseNet-121, -161 (Huang et al., 2017), Inception-V3 (Szegedy et al., 2016), -V4 (Szegedy et al., 2017), Inception-ResNet-V2, Inception-V3$_{ens3}$, Inception-V3$_{ens4}$, and Inception-ResNet-V2$_{ens}$. The former eight models are normally trained models while the latter three are secure models trained by ensemble adversarial training (Tramèr et al., 2017). We shorten these models as Res-50, Res-101, Res-152, DN-121, DN-161, Inc-V3, Inc-V4, IncRes-V2, Inc-V3$_{ens3}$, Inc-V3$_{ens4}$, and IncRes-V3$_{ens}$.

Training the MSM. The default network architecture of the MSM is ResNet-19 shown in Figure 2, with $M1$, $M2$, $M3$, and $M4$ set to 32, 80, 200, and 500, respectively. We follow previous works (Dong et al., 2018; Wu et al., 2020a) to evaluate the transferability of AEs in two settings: using a single source model and using multiple source models. We set the input resolution of the MSM to 224×224. Note that, when the resolution of the source model differs from that of the MSM, we resize the AE x^T_{adv} to the resolution of the source model before feeding it into the source model. More details about training the MSM will be shown in Section A.4.

Evaluating the MSM. Following the official testing data settings in the papers of DI (Xie et al., 2019) and SGM (Wu et al., 2020a), we also randomly choose 5,000 validation images from ImageNet that are correctly classified by all models for evaluation. Note that, when the resolutions of the MSM and the target model are different, we resize the AE x^T_{adv} to the resolution of the target model. For instance, when attacking Inc-V3 whose resolution is 299×299, we first resize x^T_{adv} from 224×224 to 299×299 and then use the resized x^T_{adv} to attack Inc-V3.

[4]https://github.com/pudae/tensorflow-densenet

[5]https://github.com/tensorflow/models/tree/r1.12.0/research/slim

[6]https://github.com/tensorflow/models/tree/r1.12.0/research/adv_imagenet_models

Table 3: Transfer attack results on seven black-box models when using multiple source models.

Source	Method	Inc-V3	Inc-V4	IncRes-V2	Res-101	Inc-V3$_{ens3}$	Inc-V3$_{ens4}$	IncRes-V2$_{ens}$
Res-50 + Res-152 + DN-161	DI	86.9%	84.3%	81.8%	96.7%	59.7%	55.1%	41.9%
	MI	82.0%	76.1%	76.0%	98.0%	63.6%	60.3%	49.6%
	TI-DI	60.6%	59.2%	50.2%	86.8%	54.9%	56.2%	46.9%
	SGM	81.8%	74.7%	73.9%	**98.7%**	54.9%	50.1%	38.7%
	SGM-DI	86.2%	83.9%	81.6%	98.3%	69.8%	64.9%	54.4%
	SGM-MI	86.5%	84.3%	82.7%	98.2%	71.1%	67.4%	60.8%
	IR	75.2%	70.3%	67.9%	90.6%	51.7%	49.1%	37.5%
	MTA	90.4%	94.3%	87.6%	97.5%	75.5%	79.7%	79.0%
	MTA-IR	**93.1%**	**95.8%**	**90.5%**	98.3%	**83.6%**	**87.2%**	**85.0%**
Res-50 + Inc-V1 + DN-121	DI	84.1%	82.3%	79.4%	93.9%	56.3%	50.1%	35.2%
	MI	79.9%	73.6%	72.3%	93.7%	59.3%	56.0%	42.7%
	TI-DI	61.9%	58.5%	49.0%	79.7%	53.1%	54.1%	41.9%
	SGM	62.7%	53.5%	50.9%	89.1%	33.8%	30.4%	19.3%
	SGM-DI	87.2%	83.6%	**79.5%**	95.1%	59.6%	54.9%	37.9%
	SGM-MI	82.8%	76.0%	74.3%	**95.9%**	62.2%	59.7%	45.3%
	IR	76.5%	70.9%	64.0%	92.1%	51.3%	44.9%	31.5%
	MTA	91.7%	86.4%	76.0%	93.6%	81.7%	**79.6%**	**61.6%**
	MTA-IR	**92.8%**	**87.9%**	77.2%	93.8%	**82.6%**	79.3%	61.5%
Res-50 + Inc-V1	DI	76.1%	69.3%	66.3%	90.0%	43.5%	39.2%	25.5%
	MI	69.5%	60.1%	59.5%	91.5%	47.1%	44.7%	32.5%
	TI-DI	51.6%	46.9%	38.4%	73.4%	43.4%	44.2%	32.8%
	SGM	46.1%	35.6%	33.3%	82.0%	22.1%	19.5%	12.3%
	SGM-DI	79.2%	70.6%	68.7%	91.9%	47.9%	42.0%	28.1%
	SGM-MI	71.9%	62.0%	61.3%	94.3%	49.6%	47.2%	33.8%
	IR	60.2%	49.0%	46.2%	93.0%	36.5%	30.6%	21.0%
	MTA	84.1%	88.8%	78.4%	93.9%	60.6%	61.1%	55.1%
	MTA-IR	**87.6%**	**91.8%**	**83.9%**	**95.2%**	**71.5%**	**72.6%**	**63.7%**

4.2.2 USING ONE SOURCE MODEL

Table 2 reports the experimental results of using one source model. Note that, in this work, we only focus on the transfer attack testing scene and neglect the white-box attack testing scene. So we left the results of the testing scenes where the target model is the source model itself to $/$. MI-DI is a combination of MI and DI. IR is our re-implementation with ϵ=15 and the implementation details will be shown in Section A.8. Obviously, MTA outperforms the baselines on almost all testing scenes with great margins, especially when attacking adversarially trained models. For example, compared with FIA, MTA improves the transfer attack success rates by about 31.7%, 30.7%, 41.1%, 131.1%, 41.9%, and 75.2% when using the Inc-V3 source model and attacking the target models (Inc-V4, IncRes-152, Res-152, Inc-V3$_{ens3}$, Inc-V3$_{ens4}$, IncRes-V2$_{ens}$). MTA-IR combines MTA with IR. Instead of attacking the MSM using PGD, MTA-IR generates AEs by attacking the MSM using IR. Compared with MTA, MTA-IR improves the attack success rates by about 5.1%, 6.8%, 14.7%, 23.3%, 44.8%, and 55.9% when using the Inc-V3 source model and attacking the target models, indicating that existing transferable attack methods can further improve MTA.

Recall that SGM only works for source models with lots of skip connections (e.g., ResNet). And the original paper sets ϵ to 16, which differs from most of the other methods. The official IR also sets ϵ to 16. Therefore, we copy their results with $\epsilon = 16$ from their official paper to Table 2 and denote them as SGM$^*_{\epsilon=16}$ and IR$^*_{\epsilon=16}$, respectively. To compare MTA with them, we further set ϵ to 16 for MTA and denote the new result as MTA$_{\epsilon=16}$. The comparisons show that MTA$_{\epsilon=16}$ outperforms SGM$^*_{\epsilon=16}$ and IR$^*_{\epsilon=16}$ significantly.

When using IncRes-V2 source model, MTA sometimes performs slightly worse than MI-DI, possibly because the MSM with ResNet-19 backbone is not suitable to be trained to attack IncRes-V2. We then replace the backbone from ResNet-19 with another simplified Inception network (the architecture will be shown in Section A.6) and retrain the MSM. The newly trained MSM is denoted as MTA$_{Inc}$. Compared with ResNet-19, the simplified Inception backbone is more similar to IncRes-V2 so that MTA$_{Inc}$ turns to be easier to generate adversarial attacks to fool IncRes-V2 than MTA, leading to easier convergence of MTA$_{Inc}$. The experimental results show that MTA$_{Inc}$ outperforms not only MTA but also the compared methods in most testing scenes, indicating 1) the advantage of the proposed MTA framework and 2) MTA can be further improved by using more suitable backbones.

4.2.3 USING MULTIPLE SOURCE MODELS

The experimental results of using multiple source models are reported in Table 3. We use three source model groups (Res-50+Res-152+DN161, Res-50+Inc-V1+DN-121, Res-50+Inc-V1) to train the MSM, respectively, and use seven target models (Inc-V3, Inc-V4, InvRes-V2, Res-101, Inc-V3$_{ens3}$,

Inc-V3$_{ens4}$, IncRes-V2$_{ens}$) to evaluate the transferability of the attacks to the MSM. SGM-X is the combination of SGM and X (X=DI or MI). TI-DI is the combination of TI and DI, which is also known as TI-DIM (Dong et al., 2019). The results show that MTA outperforms the baselines in almost all testing scenes, especially when attacking defensive models. For instance, compared with SGM-DI, MTA improves the transfer attack success rates by 6.2%, 25.8%, 14.1%, 2.2%, 26.5%, 45.5%, and 96.1% on the seven target models when using Res-50 and Inc-V1 source models. Besides, MTA-IR outperforms MTA.

4.3 Ablation Study

Network structure The comparison between MTA and MTA$_{Inc}$ shown in Table 2 has validated the effect of backbone on the MSM. Here we conduct another experiment on Cifar-10 to further verify the effect of backbone by replacing the backbone from ResNet-13 to DenseNet-22BC (the structure of DenseNet-22BC will be shown in Section A.6). We denote the MSM using DenseNet-22BC backbone as MTA$_{dense}$ and report its experimental results in Table 1. The comparisons among MTA, MTA$_{dense}$, and the other compared methods indicate that 1) the backbone affects the performance of MTA; 2) MTA outperforms the compared methods with various backbones. This also inspires us to design more suitable backbones to improve MTA as future work.

Number of attack iterations We perform several experiments on Cifar-10 to validate how the number of attack iterations T_t affects the performance. T_t is set to 7 by default on Cifar-10. Here we set T_t to 1, 3, 5, 9, and 11 and keep all the other settings be consistent with the default settings. Figure 3 shows the corresponding performances of MTA. It is observed that when $T_t < 7$, the performances of MTA will be improved with the increase of T_t while when $T_t > 7$, the performance tends to drop. We think this is due to the difficulty of unrolling too many attack steps when training the MSM. We also verify how T_v affects the performance by changing T_v. T_v is default set to 10 in all our experiments. Figure 3 shows the experimental results using different numbers of T_v. When $T_v = 1$, the performances can be denoted as MTA-FGSM (one-step PGD). With the increase of T_v, the transfer attack success rates are clearly increased.

The effects of γ_1 and γ_2 We perform two experiments on Cifar-10 to verify how the parameters γ_1 and γ_2 in Eq 5 affect the transfer attack performance. In the two experiments, we set γ_1 and γ_2 to zero respectively, and amplify ϵ_c appropriately to offset the decrease of the training perturbation size caused by zeroing γ_1 or γ_2. We denote the two newly performed MTA as MTA$_{\gamma_1=0}$ and MTA$_{\gamma_2=0}$. Table 1 shows the experimental results. The results show that by setting γ_1 to zero, the performances of MTA are greatly damaged on all target models, indicating the indispensability of the arctan component in the Customized PGD. Setting γ_2 to zero also decreases MTA's performances, but the effect is much smaller than that of γ_1. Overall, the two experiments demonstrate the indispensability of Customized PGD for the proposed MTA framework. Further, both the arctan and sign components in Customized PGD are important to train the MSM, especially arctan.

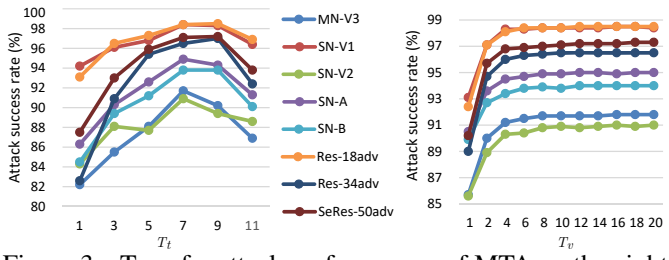

Figure 3: Transfer attack performances of MTA on the eight target models of Cifar-10. Left: Attack success rates with different T_t. Right: Attack success rates with different T_v. y-axis denotes the attack success rate.

5 Conclusion

Existing query free black-box adversarial attack methods directly use image classification models as surrogate models to generate transferable adversarial attacks to attack black-box models neglecting the study of surrogate models. In this paper, we propose a novel framework called meta-transfer attack (MTA) to improve the transferability of adversarial attacks via training an MSM using these surrogate models. The MSM is a particular model trained to learn how to make the adversarial attacks to it can fool the surrogate models. To enable and improve the training of the MSM, a novel Customized PGD is also developed. Through extensive experiments, we validate that by attacking the trained MSM, we can get transferable adversarial attacks that are generalizable to attack black-box target models with much higher success rates than existing methods, demonstrating the effectiveness of the proposed MTA framework.

6 ETHICS STATEMENT

Our work is promising to evaluate and improve the security of deep models, and has no potential negative societal impacts.

7 REPRODUCIBILITY STATEMENT

We provide our code in supplemental material and describe all the experimental settings in Sections 4.1.1, 4.2.1, and Appendix. The hyperparameter settings and the network structure are clear. The training details of source and target models used on Cifar-10 are described in Section A.2, and the network architecture descriptions of these models can be found in Section 4.1.1 and our code. The source and target models used on ImageNet can be found in the repositories described in Section 4.2.1. We include a very simple code example of our method at the end of Appendix, which also helps readers to understand and to reproduce our results. Overall, our work is easy to reproduce and follow.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

Avishek Joey Bose, Gauthier Gidel, Hugo Berrard, Andre Cianflone, Pascal Vincent, Simon Lacoste-Julien, and William L Hamilton. Adversarial example games. *Advances in neural information processing systems*, 2020.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017.

Jianbo Chen, Michael I Jordan, and Martin J Wainwright. HopSkipJumpAttack: a query-efficient decision-based adversarial attack. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 15–26, 2017.

Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

Minhao Cheng, Simranjit Singh, Patrick H. Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. Sign-opt: A query-efficient hard-label adversarial attack. In *international conference on learning representations*, 2020.

Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Improving black-box adversarial attacks with a transfer-based prior. pp. 10932–10942, 2019.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pp. 2196–2205. PMLR, 2020a.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pp. 2206–2216. PMLR, 2020b.

Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. *USENIX Security Symposium*, pp. 321–338, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4312–4321, 2019.

Jiawei Du, Hu Zhang, Tianyi Joey Zhou, Yi Yang, and Jiashi Feng. Query-efficient meta attack to deep neural networks. *International Conference on Learning Representations*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Aditya Ganeshan, Vivek BS, and R Venkatesh Babu. Fda: Feature disruptive attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8069–8079, 2019.

Lianli Gao, Qilong Zhang, Jingkuan Song, Xianglong Liu, and Heng Tao Shen. Patch-wise attack for fooling deep neural network. In *European Conference on Computer Vision*, pp. 307–322. Springer, 2020.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *international conference on learning representations*, 2014.

Yiwen Guo, Qizhang Li, and Hao Chen. Backpropagating linearly improves transferability of adversarial examples. In *Advances in neural information processing systems 33 (NIPS 2020)*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324, 2019.

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

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Qian Huang, Isay Katsman, Horace He, Zeqi Gu, Serge Belongie, and Ser-Nam Lim. Enhancing adversarial example transferability with an intermediate level attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4733–4742, 2019.

Zhichao Huang and Tong Zhang. Black-box adversarial attack with transferable model-based embedding. *International Conference on Learning Representations*, 2020.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pp. 2137–2146. PMLR, 2018.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019.

Xu Kaidi, Liu Sijia, Zhao Pu, Chen Pin-Yu, Zhang Huan, Fan Quanfu, Erdogmus Deniz, Wang Yanzhi, and Lin Xue. Structured adversarial attack: Towards general implementation and better interpretability. *International Conference on Learning Representations*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Huichen Li, Xiaojun Xu, Xiaolu Zhang, Shuang Yang, and Bo Li. Qeba: Query-efficient boundary-based blackbox attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1221–1230, 2020a.

Qizhang Li, Yiwen Guo, and Hao Chen. Practical no-box adversarial attacks against dnns. *Advances In Neural Information Processing Systems 2020*, 2020b.

Yingwei Li, Song Bai, Yuyin Zhou, Cihang Xie, Zhishuai Zhang, and Alan Yuille. Learning transferable adversarial examples via ghost networks. *AAAI*, pp. 11458–11465, 2020c.

Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. *International Conference on Learning Representations*, 2020.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *international conference on learning representations*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *international conference on learning representations*, 2018.

Andriushchenko Maksym, Croce Francesco, Flammarion Nicolas, and Hein Matthias. Square attack: a query-efficient black-box adversarial attack via random search. *european conference on computer vision*, pp. 484–501, 2020.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Muhammad Muzammal Naseer, Salman H Khan, Muhammad Haris Khan, Fahad Shahbaz Khan, and Fatih Porikli. Cross-domain transferability of adversarial perturbations. *Advances in Neural Information Processing Systems*, 32:12905–12915, 2019.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

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

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Yunxiao Qin, Weiguo Zhang, Zezheng Wang, Chenxu Zhao, and Jingping Shi. Layer-wise adaptive updating for few-shot image classification. *IEEE Signal Processing Letters*, 27:2044–2048, 2020.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, and Venkatesh R. Babu. Guided adversarial attack for evaluating and enhancing adversarial defenses. *Advances In Neural Information Processing Systems*, 2020.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, J. Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *international conference on learning representations*, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Lu Wang, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Yuan Jiang. Spanning attack: reinforce black-box attacks with unlabeled data. *Machine Learning*, 109(12):2349–2368, 2020.

Xiaosen Wang and Kun He. Enhancing the transferability of adversarial attacks through variance tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1924–1933, 2021.

Xin Wang, Jie Ren, Shuyun Lin, Xiangming Zhu, Yisen Wang, and Quanshi Zhang. A unified approach to interpreting and boosting adversarial transferability. *International Conference on Learning Representations*, 2021a.

Zhibo Wang, Hengchang Guo, Zhifei Zhang, Wenxin Liu, Zhan Qin, and Kui Ren. Feature importance-aware transferable adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021b.

Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. *international conference on learning representations*, 2020a.

Kaiwen Wu, Allen Wang, and Yaoliang Yu. Stronger and faster wasserstein adversarial attacks. *International Conference on Machine Learning*, pp. 10377–10387, 2020b.

Lei Wu, Zhanxing Zhu, Cheng Tai, et al. Understanding and enhancing the transferability of adversarial examples. *arXiv preprint arXiv:1802.09707*, 2018.

Weibin Wu, Yuxin Su, Xixian Chen, Shenglin Zhao, Irwin King, R. Michael Lyu, and Yu-Wing Tai. Boosting the transferability of adversarial samples via attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1158–1167, 2020c.

Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2730–2739, 2019.

Zheng Yuan, Jie Zhang, Yunpei Jia, Chuanqi Tan, Tao Xue, and Shiguang Shan. Meta gradient adversarial attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *computer vision and pattern recognition*, 2018.

Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. Transferable adversarial perturbations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 452–467, 2018.

A APPENDIX

A.1 TESTING PSEUDO CODE OF MTA

We summarize the testing pseudo code of MTA in Algorithm.2, where $\hat{\mathcal{F}}$ is the target model and $\tilde{y}$ is the target model's prediction for the adversarial example x_{adv}^T. Note that all the clean examples in $\hat{\mathbb{D}}$ are correctly classified by the target model. Len($\hat{\mathbb{D}}$) denotes the number of examples in $\hat{\mathbb{D}}$.

A.2 TRAINING THE SOURCE AND TARGET MODELS ON CIFAR-10

Algorithm 2 Testing of Meta-Transfer Attack

input: Black-box target model $\hat{\mathcal{F}}$, Testing examples $\hat{\mathbb{D}}$ that are correctly classified by the target model, Optimized meta-surrogate model $\mathcal{M}_\theta$.
output: Transfer attack success rate.

$P = 0$
for $(x, y) \in \hat{\mathbb{D}}$ **do**
  $x_{adv}^0 = x$
  for k in [1, 2, ..., T] **do**
    $g^k = \nabla_{x_{adv}^{k-1}} L(\mathcal{M}_\theta(x_{adv}^{k-1}), y)$
    $x_{adv}^k = \text{Clip}\big(x_{adv}^{k-1} + \frac{\epsilon}{T} \cdot \text{sign}(g^k)\big)$
  end for
  evaluate x_{adv}^T on $\hat{\mathcal{F}}$ and obtain $\tilde{y} = \hat{\mathcal{F}}(x_{adv}^T)$
  if $y \neq \tilde{y}$ **do**
    $P+ = 1$
  end if
return $\frac{P}{\text{Len}(\hat{\mathbb{D}})}$

On Cifar-10, we use 16 source and target models to train and test the meta-surrogate model (MSM). The 8 source models are ResNet-10, -18, -34, SeResNet-14, -26, -50, MobileNet-V1, and -V2. The 8 target models are MobileNet-V3, ShuffleNet-V1, -V2, SqueezeNet-A, -B, and adversarially trained ResNet-18, -34 and SeResNet-50. It is not easy to collect the 16 trained Cifar-10 models on the internet. Therefore, before the experiments of MTA, we first use consistent hyper-parameters to train the 16 models on Cifar-10 for 200 epochs. The learning rate, L2 weight decay, and batch size are set to 0.01, 1e-5, and 128, respectively. For each adversarially trained model, we first use FGSM and the normally trained model to generate one adversarial example for each training image with $\epsilon = 3$, and then train the model on both clean and adversarial images. The 8 source models obtain 90.0%, 91.8%, 92.6%, 85.6%, 88.3%, 90.5%, 82.0%, and 81.8% accuracies on the test set, and the 8 target models obtain 80.0%, 82.5%, 76.4%, 86.4%, 86.9%, 88.9%, 90.5%, and 87.5% accuracies.

A.3 MORE EXPERIMENTS ON CIFAR-10

Here we show more experiments on Cifar-10.

A.3.1 TARGETED TRANSFER ATTACK

We conduct targeted transfer attack and show the experimental results in Table 4. MTA has a great advantage over the compared methods in the targeted transfer attack setting.

A.3.2 TRANSFER ATTACK WITH SMALLER ϵ

We set ϵ to 8 to evaluate how does MTA perform with smaller ϵ. The results shown in Table 5 indicate that MTA outperforms the compared methods no matter the value of ϵ.

A.3.3 COMPARISON BETWEEN MTA AND METAATTACK

MetaAttack[14] is developed for query-based black-box adversarial attack but not for transfer attack. We implement MetaAttack in the transfer attack scene on Cifar-10 and compare it with MTA in Table 6. The comparison indicates that MTA greatly outperforms MetaAttack in transfer attack.

A.3.4 MORE EXPERIMENTS ABOUT THE CUSTOMIZED PGD

As introduced in Section 3, the sign function in the vanilla PGD with L_∞ constraint introduces a discrete operation. This results in that the gradient back-propagating through sign be zero and further

Table 4: Targeted transfer attack results on Cifar-10.

Method	MN-V3	SN-V1	SN-V2	SN-A	SN-B
DI	16.3%	26.4%	17.2%	22.3%	21.6%
MI	29.6%	43.6%	29.8%	37.1%	35.4%
TI	17.6%	21.1%	16.5%	26.1%	25.8%
IR	10.8%	19.6%	9.5%	13.7%	12.5%
AEG	47.2%	53.8%	36.5%	42.6%	41.0%
MTA	**49.0%**	**70.3%**	**47.7%**	**60.3%**	**58.5%**

Table 5: Transfer attack results with $\epsilon = 8/255$ on Cifar-10.

Method	MN-V3	SN-V1	SN-V2	SN-A	SN-B
DI	31.5%	42.1%	30.0%	38.2%	36.9%
MI	44.2%	59.8%	43.2%	55.7%	54.9%
TI	29.5%	31.3%	29.6%	37.7%	36.8%
IR	29.2%	51.1%	35.3%	38.5%	37.4%
AEG	58.0%	66.5%	50.4%	61.9%	59.6%
MTA	**62.5%**	**79.6%**	**58.2%**	**70.5%**	**69.3%**

prohibits the training of the MSM. We propose the Customized PGD to enable the training of the MSM. Here we conduct other four experiments to validate the indispensability and the effect of the Customized PGD on the proposed MTA framework.

As PGD with L2 constraint contains no sign, in the first experiment, we use PGD with L2 constraint (PGD_{L2}) instead of the Customized PGD to attack the MSM in the training phase and denote the trained MSM as $\text{MTA}_{PGD_{L2}}$.

PGD with L1 constraint also contains no sign. In the second experiment, we use PGD with L1 constraint (PGD_{L1}) to attack the MSM in the training phase and denote the trained MSM as $\text{MTA}_{PGD_{L1}}$.

Both γ_1 and γ_2 of the Customized PGD are set to 0.01 by default. In the third experiment, we set γ_1 to 0.05. Note that we decrease ϵ_c appropriately to offset the increase of the training perturbation size caused by setting γ_1 to 0.05. All the other experimental settings are consistent with the default settings. We denote the MSM trained in this experiment as $\text{MTA}_{\gamma_1=0.05}$.

In the fourth experiment, we set γ_2 to 0.05 and denote the trained MSM as $\text{MTA}_{\gamma_2=0.05}$.

Table 6 reports all the four experimental results. We can get three conclusions. First, directly using PGD_{L1} or PGD_{L2} in MTA's training stage is also effective to train the MSM but leads to limited performance. Second, the proposed Customized PGD is important for the proposed MTA framework to achieve superior performance. Third, larger γ_1 or γ_2 damages the performances of MTA.

A.3.5 More experiments about source and target models

Here we change the setting of source and target models, and evaluate MTA under this new setting. In this setting, the source models are MobileNet-V2, ShuffleNet-V1, ShuffleNet-V2, SqueezeNet-A, and SqueezeNet-B, and the target models are ResNet-10, ResNet-18, ResNet-34, SeResNet-14, SeResNet-26, SeResNet-50, MobileNet-V1, and MobileNet-V2. All the other experimental settings are consistent with those introduced before. The experimental results are summarized in Table 7, where MTA still shows its advantage in the transfer attack problem.

A.3.6 The experiment where source models do not share training samples with target models.

In our previous experiment, the source and target models are all trained on the same training set. Here we conduct a new experiment, where we train the source and target models on different training samples, and use the new trained models to perform transfer attack. This experiment is performed on Cifar-10, which contains 10 categories and each category in the training set contains 5000 images.

Table 6: More transfer attack experimental results on Cifar-10.

Method	MN-V3	SN-V1	SN-V2	SN-A	SN-B
MetaAttack	39.2%	43.9%	32.1%	38.6%	37.8%
MTA$_{PGD_{L2}}$	80.8%	92.7%	83.5%	89.0%	86.8%
MTA$_{PGD_{L1}}$	81.5%	91.3%	82.4%	85.3%	83.7%
MTA$_{\gamma_1=0.05}$	90.5%	98.0%	90.2%	94.5%	93.1%
MTA$_{\gamma_2=0.05}$	86.7%	95.3%	85.8%	89.5%	88.4%
MTA	**91.8%**	**98.4%**	**90.9%**	**94.9%**	**93.8%**

Table 7: The MSM is trained with source models MobileNet-V3, ShuffleNet-V1, ShuffleNet-V2, SqueezeNet-A, and SqueezeNet-B. From left to right, the target models are ResNet-10 (Res-10), ResNet-18 (Res-18), ResNet-34 (Res-34), SeResNet-14 (SE-14), SeResNet-26 (SE-26), SeResNet-50 (Res-18), MobileNet-V1 (MB-V1), and MobileNet-V2 (MB-V2).

Method	Res-10	Res-18	Res-34	SE-14	SE-26	SE-50	MB-V1	MB-V2
PGD	46.9%	42.5%	50.1%	49.6%	50.2%	45.9%	47.9%	54.5%
DI	65.2%	56.9%	69.6%	70.2%	71.5%	65.7%	69.5%	71.2%
MI	89.5%	86.1%	90.7%	88.3%	91.0%	89.1%	86.6%	88.8%
TI	48.1%	39.2%	49.9%	53.8%	55.6%	47.8%	63.9%	60.8%
MTA	**96.7%**	**94.6%**	**98.3%**	**98.7%**	**98.8%**	**97.5%**	**96.7%**	**98.7%**

In this experiment, we split the training set into two sub-training sets and each of the sub-training set contains all the 10 categories. Every category in the first sub-training set contains 2500 images and every category in the second sub-training set contains the remaining 2500 images. Therefore, there is no overlapping samples between the two sub-training sets, and the two sub-training sets share only the label set. We use the first sub-training set to train the source models and the meta-surrogate model, and use the second sub-training set to train the target models. Thus the source models and the meta-surrogate model does not use the training images of the target models. The source models are ResNet-10, ResNet-18, ResNet-34, SeResNet-14, SeResNet-26, SeResNet-50, MobileNet-V1, MobileNet-V2 with testing accuracies of 86.8%, 86.7%, 87.2%, 84.2%, 85.4%, 87.7%, 80.7%, and 80.9%, respectively. The target models are MobileNet-V3, ShuffleNet-V1, ShuffleNet-V2, SqueezeNet-A, and SqueezeNet-B with testing accuracies of 73.9%, 81.1%, 72.6%, 82.3%, and 83.0%, respectively. Then we use the source models and the trained meta-surrogate model to attack the target models. The experimental results are reported in Table 8. It is clear that when we know the label set but do not know the training images of the target models, MTA still outperforms the baselines with clear margins.

A.4 The supplemental experimental settings of MTA on ImageNet.

In our experiment on ImageNet, we found that the MSM directly trained on the resolution of 224×224 often suffers from slow and unstable convergence due to the high dimensionality. Therefore, we develop a three-stage training strategy for gradually and stably training the MSM. The **first** training stage only trains the top 4 blocks and the classifier of the MSM. The input data x_{adv}^{k-1} is down-sampled by $4\times$ and is fed into the 3rd block skipping the 1st and 2nd blocks. The perturbation g_{ens}^{k-1} is first up-sampled by $4\times$ and is then added to x_{adv}^{k-1} to obtain x_{adv}^{k}. The **second** stage trains the top 5 blocks and the classifier. The input x_{adv}^{k-1} is down-sampled by $2\times$ and is fed into the 2nd block skipping the 1st block. The **third** stage trains all layers. Note that, except for the newly added block in the second or third stage and the layers directly connected with the newly added block, all the other layers inherit the weights trained in the previous stage. Due to memory limitation, we set T_t to a small number of 2.

The first, second, and third training stages take 100,000, 50,000, and 50,000 iterations, with the batch size of 50, 36, and 24, respectively. Both the second and the third stages train the newly added blocks and the layers directly connected with them in the first 20,000 iterations and fine-tune all the blocks in the later 30,000 iterations. The learning rate α and the number of iterations T_t are set to 0.001 and 2, respectively. In the first, second, and third training stages, ϵ_c is initialized to $3,000$, $1,200$, and $1,200$ respectively, and is exponentially decayed by $0.9\times$ for every $4,000$, $3,000$, and $3,000$ iterations, respectively.

Table 8: The transfer attack results on Cifar-10 when the source models do not share training images with target models. The source models are ResNet-10 (Res-10), ResNet-18 (Res-18), ResNet-34 (Res-34), SeResNet-14 (SE-14), SeResNet-26 (SE-26), SeResNet-50 (Res-18), MobileNet-V1 (MB-V1), and MobileNet-V2 (MB-V2). The target models are MobileNet-V2, ShuffleNet-V1, ShuffleNet-V2, SqueezeNet-A, and SqueezeNet-B.

Method	MN-V3	SN-V1	SN-V2	SN-A	SN-B
PGD	52.6%	69.2%	51.3%	57.8%	60.2%
DI	61.2%	75.0%	57.9%	63.2%	66.5%
MI	74.4%	87.1%	71.6%	80.3%	80.9%
TI	56.3%	59.1%	56.1%	63.3%	62.0%
MTA	**87.9%**	**95.1%**	**85.6%**	**82.7%**	**83.5%**

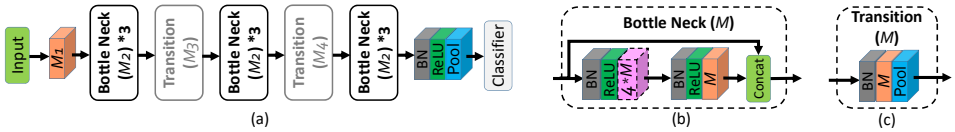

Figure 4: (a) DenseNet-22-BC. Orange cube is convolution layer with 3×3 kernel size. Pink cube is convolution layer with 1×1 kernel size. 'Bottle Neck (M_2) *3' denotes three cascaded 'Bottle Neck (M_2)'. The number (*e.g.*, M_1, $4 * M$, M) on each convolution layer denotes its number of filters. 'Pool' in the Transition block is Max Pooling with both stride and kernel size of 2×2, and the last 'Pool' before the classifier is Global Average Pooling. (b) The detailed structure of Bottle Neck. (c) The detailed structure of Transition.

We refer to the data pre-processing methods in the repository[7] on GitHub to pre-process the data used in our experiments on ImageNet. When the resolution of the source model is 224×224, we refer to 'vgg_preprocessing.py' while when the resolution is 299×299, we refer to 'inception_preprocessing.py'.

A.5 Attacking Transformer

We also conduct an experiment on ImageNet to evaluate how the proposed MTA performs in attacking Vision Transformer (ViT). In this experiment, the source model is Inception-V3, and the target model is Vit_base_patch16_224[8]. Experimental results are reported in Table 9. It is clear that MTA performs the best in attacking ViT.

A.6 The network architecture

Table 9: Transfer attack performances of MTA on ViT.

Method	PGD	TI	DI	MI	**MTA**
Success Rate	5.5%	8.6%	7.0%	15.6%	**21.3%**

DenseNet-22BC is shown in Figure 4. M_1, M_2, M_3, and M_4 are set to 80, 40, 100, and 110, respectively. We denote MTA with DenseNet-22BC backbone as MTA$_{dense}$ and show its performances in Table 1 of the main-body.

The simplified Inception network is a much shallower and thinner version of the official Inception-ResNet-V2. Figure 5 shows the structure of the simplified Inception. The official Inception-ResNet-

[7]https://github.com/tensorflow/models/tree/master/research/slim/preprocessing

[8]https://github.com/rwightman/pytorch-image-models

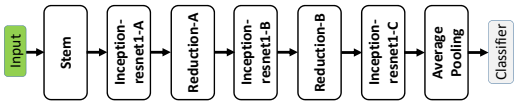

Figure 5: The simplified Inception network. All the blocks have the same inner structures with those of Inception-ResNet-V2.

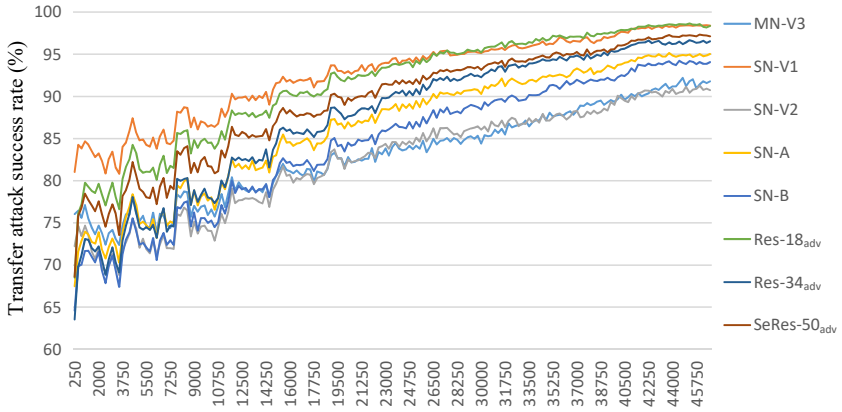

Figure 6: Transfer attack success rates of MTA on the eight black-box Cifar-10 models, across the training process.

V2 repeats each Indeption-resnet1-A, -B, or -C block for several times while the simplified Inception does not repeat them. We denote MTA with this backbone as MTA$_{Inc}$ and show its performances in Table 2 of the main-body.

A.7 Definition of attack success rate.

The formulation of attack success rate is $Rate = \frac{Card(\{x \| x \in D_t, M(x)=y \neq M(x_{adv})\})}{Card(\{x \| x \in D_t, M(x)=y\})}$, where D_t is the test set, x is a test image and x_{adv} is the adversarial image generated for x, y is the ground-truth label for x, M is the target model and $M(x)$ is the prediction of the target model for x. $\{x \| x \in D_t, M(x) = y\}$ is the set containing all clean images that are correctly classified by model M. $\{x \| x \in D_t, M(x) = y \neq M(x_{adv})\}$ is the set containing all clean images that not only are correctly classified by model M but also the corresponding adversarial images are misclassified by model M. $Card(\{x \| x \in D_t, M(x) = y\})$ denotes the number of elements in the set $\{x \| x \in D_t, M(x) = y\}$.

A.8 Implementations of the compared methods.

For fair comparisons between MTA and the compared methods, we tune the compared methods for their best possible performances in our re-implementation. ϵ is set to 15 by default for all methods and T_v is set to 10 for all PGD-based methods.

MI utilizes gradient momentum to make the generated adversarial examples more transferable. The most important hyper-parameter of MI is μ. In our implementation, we found that setting μ to 1 can achieve the best transfer attack performance.

DI. We follow the available public code[9] of DI to implement it in Tables 1, 2, and 3. As to the experiments on ImageNet, we set 'FLAGS.image_width' and 'FLAGS.image_resize' (two parameters of the input_diversity function in the official code[9]) to 224 and 256 respectively. On Cifar-10, we set 'FLAGS.image_width' and 'FLAGS.image_resize' to 32 and 36, respectively. For all experiments, we set p to 0.8.

TI. We directly utilize the public code[10] to implement TI and TI-DI in Tables 1, and 3, respectively.

SGM uses a parameter γ to reduce the gradient from all residual modules of ResNet or DenseNet. We utilize grid search to tune γ for each ResNet and DenseNet source model shown in Table 3. We denote γ for the source model of Res-50, Res-152, DN-161, and DN-121 as γ_{res50}, γ_{res152}, γ_{dn161}, and γ_{dn121}, respectively. The tuned best γ_{res50}, γ_{res152}, and γ_{dense} for the source model group Res-50+Res-152+DN-161 are 0.20, 0.45, and 0.70, respectively. The tuned best γ_{res50} and γ_{dn121} for the source model group Res-50+Inc-V1+DN-121 are 0.60 and 0.85, respectively. The tuned best γ_{res50} for the source model group Res-50+Inc-V1 is 0.65.

[9] https://github.com/cihangxie/DI-2-FGSM

[10] https://github.com/dongyp13/Translation-Invariant-Attacks

A-PGD. We directly utilize the public public code[11] of A-PGD to implement it in Table 1.

AEG. By referring to the AEG's paper and code[12], we re-implement AEG on Cifar-10 and train the generator and the critic for 500 epochs with the learning rate of 0.001. The architecture of the generator is the encoder-decoder defined in Tab.7 of AEG's paper. We do not implement AEG on ImageNet because training the generator and critic is expensive on ImageNet.

IR. We directly utilize the public code[13] of IR to implement it on ImageNet. When implementing IR on Cifar-10, we set the hyper-parameter 'args.grid_scale' to 1.

A.9 TRAINING CURVES

In the training process of the MSM, we evaluate MTA's transfer attack performances on the target models for every 250 iterations. Figure 6 visualizes the performance curves on eight Cifar-10 target models. It is observed that with the training going on, the transfer attack success rates on the target models rise gradually. The periodic fluctuations of the performances are caused by the periodic decay of the hyper-parameter ϵ_c described in Section 4.1.1.

A.10 COMPUTATIONAL COST

We conduct all experiments on Tesla P40 GPU. The computational cost can be summarized into **training cost** and **inference cost** happened in the training and the inference phases, respectively. The training cost of the proposed MTA depends mainly on the backbone of MSM, the used source models, the dataset, the batch size, T_t, and *etc.*. On Cifar-10, the default backbone of the MSM is ResNet-13, the batch size is 64, $T_t = 7$, and we use 8 source models to train the MSM. The training costs one P40 GPU and approximately 2.5T FLOPs per iteration. On ImageNet, the default backbone of the MSM is ResNet-19, $T_t = 2$, the batch size is 24 in the third training stage. When using the Inc-V3 source model to train the MSM, the third training stage costs one P40 and approximately 3.2T FLOPs per iteration. When using the Res-152, Res-50, and DN-161 source models to train the MSM, the third training stage costs three P40 GPUs and approximately 6.5T FLOPs per iteration.

In the inference phase (generating adversarial examples and attack the target models), the cost of MTA depends mainly on the backbone of MSM and T_v. The inference cost of baselines depend on the source models and T_v. On ImageNet, when using the Res-152, Res-50, and DN-161 source models, the PGD-based baselines (DI, MI, TI, SGM) cost about 124.1 GFLOPs per gradient ascent step per image, and cost about 109.1M parameters. As a comparison, the inference cost of MTA is only 11.3 GFLOPs per gradient ascent step per image and the parameter the MTA needed is only 6.77M. Obviously, **both the inference cost and the parameter the MTA used is much smaller than those of the PGD-based baselines**, and this is another advantage of the proposed MTA over the PGD-based baselines.

A.11 VISUALIZATION OF ADVERSARIAL EXAMPLES

Figure 7 visualizes the adversarial examples and the noises generated for the corresponding clean images via MI, DI, TI, SGM, IR, and MTA. All the clean images are sampled from the testing set of ImageNet.

A.12 THE TENSORFLOW CODE

We show the simplified core code of MTA on the last two pages for a better understanding of our work. Note that the showed code is used for the experiments on Cifar-10 but not on ImageNet. The code used on ImageNet differs slightly from the showed code.

[11] https://github.com/fra31/auto-attack

[12] https://github.com/joeybose/Adversarial-Example-Games

[13] https://github.com/xherdan76/A-Unified-Approach-to-Interpreting-and-Boosting-Adversarial-Transferability

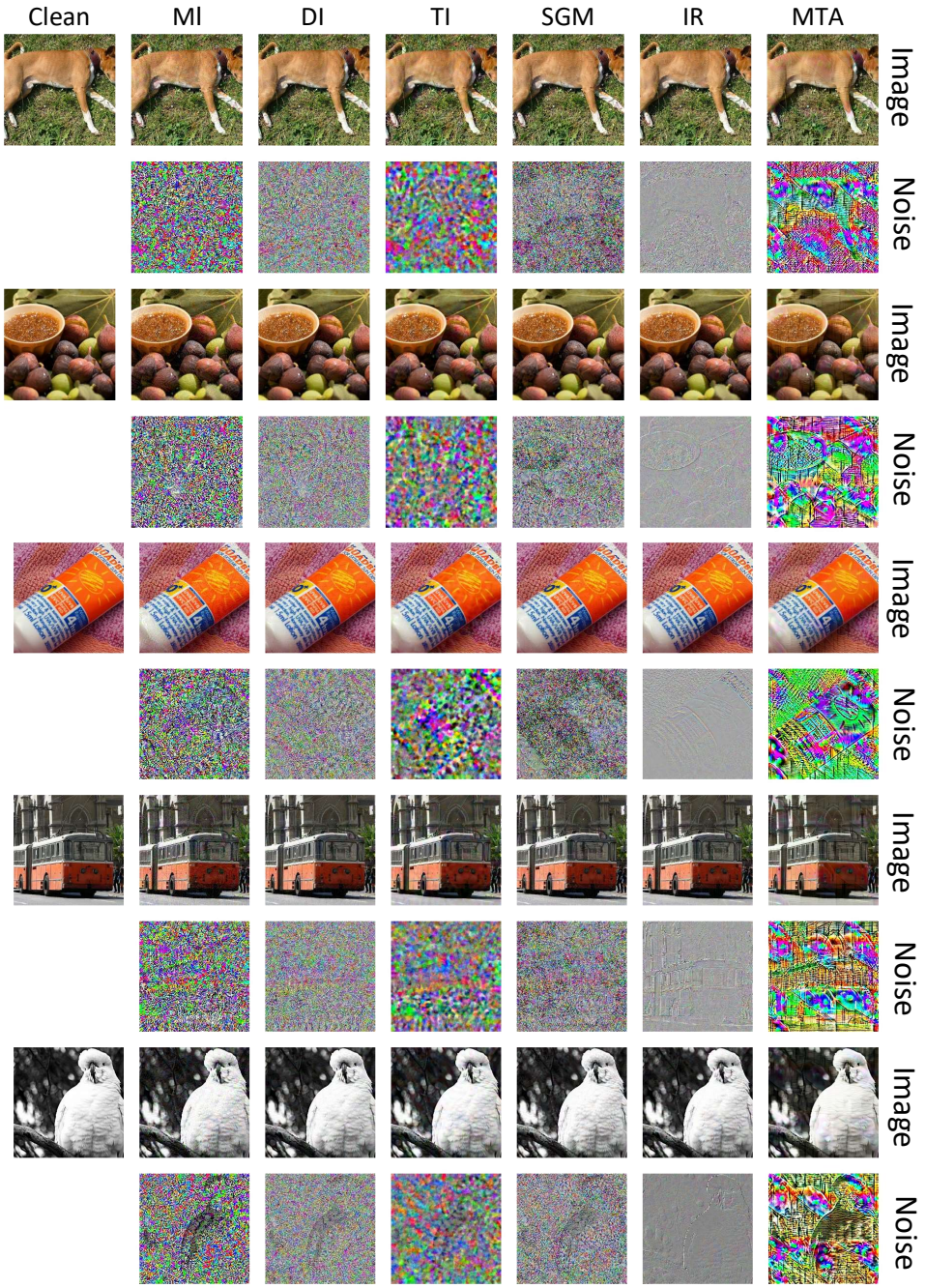

Figure 7: The adversarial examples and the noises generated via MI, DI, TI, SGM, IR, and MTA. The corresponding clean images are shown in the left most column. The source model is Res-152.

```python
import tensorflow as tf

class Meta_Transfer_Attack:
  def __init__(self):
    # Define some hyperparameters
    self.lr = tf.placeholder_with_default(0.001, ())
    self.epsilon_c = tf.placeholder_with_default(1, ())
    # Define the meta-surrogate model
    self.MSM = ResNet13()
    # Define the source models
    self.source_models = [ResNet(10), ..., MobileNet_V2()]
    # Define the input data and the label
    self.image = tf.placeholder(tf.float32, shape=[None, 32, 32, 3])
    self.label = tf.placeholder(tf.float32, shape=[None, 10])

  def build_training_graph(self, T):
    # The initial adversarial examples are the clean images
    attack = self.image
    with tf.variable_scope('surrogate', reuse=tf.AUTO_REUSE):
      for k in range(T):
        # Predict the adversarial examples
        surrogate_logits = self.MSM.predict(attack)

        # meta-surrogate models' loss on the adversarial examples.
        surrogate_loss = Cross_entropy(logits=surrogate_logits,
                                       labels=self.label)

        # calculate G^k
        grad = tf.gradients(surrogate_loss, attack)[0]

        # calculate G^k_1
        grad_1 = grad / tf.reduce_sum(tf.abs(grad), axis=[1,2,3],
                                      keep_dims=True)

        # calculate G^k_t
        mean_abs_grad = tf.reduce_mean(tf.abs(grad), axis=[1,2,3],
                                       keep_dims=True)
        norm_one_grad = grad / mean_abs_grad
        grad_atan = tf.atan(norm_one_grad) * (2 / 3.1415926)

        # calculate G^k_s
        grad_sign = tf.sign(grad)

        # calculate G^k_ens
        grad_ens = grad_1 + 0.01 * grad_sign + 0.01 * grad_atan

        # Obtain the adversarial examples X^k_adv
        attack_temp = attack + (self.epsilon_c / T) * grad_ens
        attack = tf.clip_by_value(attack, 0.0, 1.0)

      # Evaluate the adversarial examples X^T_adv on the source models
    with tf.variable_scope('Source', reuse=tf.AUTO_REUSE):
      for model in self.source_models:
        logits = model.predict(attack)
        loss = Cross_entropy(logits=logits, labels=self.label)
        self.source_loss += tf.reduce_mean(loss)/len(self.source_models)


  def build_optimizing_graph(self):
    opt = tf.train.AdamOptimizer(self.lr)
    # Optimize the MSM via maximizing the source models' loss.
    gvs = opt.compute_gradients(-self.source_loss, self.MSM.weight)
    gvs = [(tf.clip_by_value(grad, -15, 15), var) for grad, var in gvs]
    self.train_op = optimizer.apply_gradients(gvs)

```

```
def main():
  # Initialize the settings.
  Batch_size = 64
  Init_eps_c = 1600 / 255

  # Define the graph
  MTA = Meta_Transfer_Attack()
  MTA.build_training_graph(7)
  MTA.build_optimizing_graph()

  # Define the data loader
  Cifar10_dataloader = DataSet('Cifar10')

  sess = tf.InteractiveSession()
  tf.global_variables_initializer().run()

  # Restore the weights of all source models
  restore_source_weights(MTA.source_models, sess)

  for iter in range(47000):
    # Exponentially decay eps_c by 0.9× for every 4000 iterations.
    eps_c = Init_eps_c * ( 0.9 ** int(iter / 4000) )

    images, labels = Cifar10_dataloader.get_data(Batch_size)

    feed_dict = {}
    feed_dict[MTA.image] = images
    feed_dict[MTA.label] = labels
    feed_dict[MTA.lr] = 0.001
    feed_dict[MTA.epsilon_c] = eps_c

    # Train the MSM
    sess.run(MTA.train_op, feed_dict)
```