Learning to Learn Transferable Attack

Shuman Fang,[1*] **Jie Li,**[1*] **Xianming Lin,**[1†] **Rongrong Ji**[1,2]

[1]MAC Lab, Department of Artificial Intelligence, School of Informatics, Xiamen University
[2]Peng Cheng Laboratory, Shenzhen, China
fangshuman@stu.xmu.edu.cn, lijie.32@outlook.com, linxm@xmu.edu.cn, rrji@xmu.edu.cn

Abstract

Transfer adversarial attack is a non-trivial black-box adversarial attack that aims to craft adversarial perturbations on the surrogate model and then apply such perturbations to the victim model. However, the transferability of perturbations from existing methods is still limited, since the adversarial perturbations are easily overfitting with a single surrogate model and specific data pattern. In this paper, we propose a *Learning to Learn Transferable Attack (LLTA)* method, which makes the adversarial perturbations more generalized via learning from both data and model augmentation. For data augmentation, we adopt simple random resizing and padding. For model augmentation, we randomly alter the back propagation instead of the forward propagation to eliminate the effect on the model prediction. By treating the attack of both specific data and a modified model as a task, we expect the adversarial perturbations to adopt enough tasks for generalization. To this end, the meta-learning algorithm is further introduced during the iteration of perturbation generation. Empirical results on the widely-used dataset demonstrate the effectiveness of our attack method with a 12.85% higher success rate of transfer attack comparing with the state-of-the-art methods. We also evaluate our method on the real-world online system, *i.e.*, Google Cloud Vision API, to further show the practical potentials of our method.

Introduction

Though showing dominant advance in various tasks, deep neural networks (DNNs) are proven vulnerable to adversarial examples (Goodfellow, Shlens, and Szegedy 2015), *i.e.*, adding well-designed human-imperceptible perturbations into natural images can mislead DNNs. Such a drawback has raised high-security concerns when deploying DNNs in security-sensitive scenarios (Xiao et al. 2021; Fang et al. 2021), which has caused researchers to attach attention to model security (Wu, Xia, and Wang 2020; Naseer et al. 2020). According to the knowledge of victim model, the adversarial attack can be categorized into white-box and black-box attack. The black-box attack is more practical but challenging, which can be further divided into query-based and transfer-based attack. The query-based attack polishes the

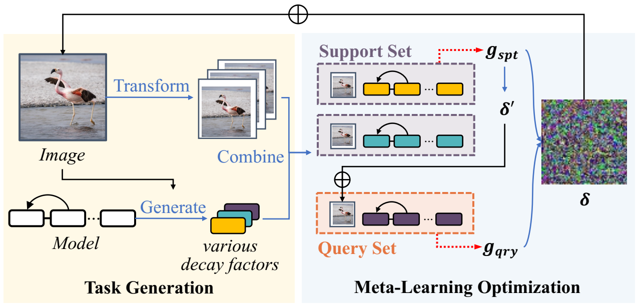

Figure 1: Overview of LLTA, which includes task generation and meta-learning optimization. In *task generation*, treating attacking of an augmented data and model as a task, we generate multiple tasks by adopting random transformation (data augmentation) and introducing a group of learnable decay factors on backpropagation (model augmentation). In *meta-learning optimization*, we split these tasks into support set and query set, craft a temporary perturbation from support set and finetune it on the query set. Finally we update the actual perturbation with the gradient from two sets.

adversarial perturbations by querying the victim model iteratively, which achieves a high attack success rate but brings more resource cost (Ilyas, Engstrom, and Madry 2019; Li et al. 2020a, 2021). Instead, the transfer-based attack crafts the adversarial perturbations on a white-box surrogate model and then transfer to the victim model directly, which is more efficient and avoids suspicion. Essentially, the adversarial transferability is highly expected since the surrogate model should be ideally similar to the victim model (Orekondy, Schiele, and Fritz 2019; Wang et al. 2021). However, as in reality the perturbations usually overfit to the surrogate model, the transfer success rate remains unsatisfactory.

To this end, many efforts have been put to learn more generalized perturbations. Dong et al. (2018) and Lin et al. (2020) integrated optimization techniques. These techniques are effective but with limited improvement space for adversarial attack due to the small number of optimization steps here. Huang et al. (2019) and Li, Guo, and Chen (2020) only perturbed features from the intermediate layer of DNNs, but the layer should be picked manually and carefully. There are also some augmentation-based methods (Xie et al. 2019; Lin

[*]These authors contributed equally.
[†]Corresponding author.

et al. 2020; Zhong and Deng 2020). These methods narrow the gap between the surrogate model and the victim one, but still suffer from insufficient augmentation or extra costs, *e.g.*, multiple surrogate models for model ensemble.

In this paper, we propose a novel transfer adversarial attack framework, termed *Learning to Learn Transferable Attack* (LLTA), to enhance the transferability of adversarial perturbations. As depicted in Fig. 1, our method contains a task generation module based on data and model augmentation, and a meta-learning optimization module to finally update the perturbations. For data augmentation, we follow Xie et al. (2019) to use simple random transformations so that different data patterns can be dug to avoid overfitting. For model augmentation, we aim to obtain various models by modifying a single surrogate model and only focus on backpropagation to ensure the model prediction keep the same. Inspired by SGM (Wu et al. 2020), we introduce a set of learnable decay factors where one factor per layer to suppress the gradient from the residual branch. Particularly, we first optimize the decay factors according to the hypothesis that the "linear nature" of DNNs causes the adversarial vulnerability (Goodfellow, Shlens, and Szegedy 2015; Guo, Li, and Chen 2020), and then randomly change them to generate diverse models. After augmentation, we define attacking a combination of specific input and a modified model as a task, and polish the perturbations iteratively on abundant tasks for better generalization. We finally introduce the meta-learning optimization to reduce the number of required tasks. Unlike previous works (Yuan et al. 2021; Yin et al. 2021) performing meta learning on multiple surrogate models, we perform meta learning on augmented tasks. To verify the effectiveness of our method, we conduct extensive experiments on a widely used dataset and achieve 12.85% better performance compared with the state-of-the-art methods. To further demonstrate the practical potentials, we evaluate it with baselines on the real-world online system, *i.e.*, Google Cloud Vision API, where our method achieves 42% attack success rate which is 5% higher than the second-best one.

Concretely, we summarize our contributions as follows:

- We propose a novel transfer adversarial attack framework termed *Learning to Learn Transferable Attack* (LLTA) based on meta learning to generate generalized adversarial perturbations with data and model augmentation on a single surrogate model.
- We propose an efficient strategy to enhance the transferability of adversarial perturbations by modifying model backpropagation, which can be extended as a manner of model augmentation.
- Compared with the state-of-the-art methods, LLTA achieves a higher transfer attack success rate on ImageNet pre-trained models and real-world systems, demonstrating its effectiveness.

Background and Related Work

Given an input $\boldsymbol{x}$ with its label y and a model $f(\cdot)$, the objective of adversarial attack can be formulated as:

$$\max_{\boldsymbol{x}_{adv}} \mathcal{L}\big(f(\boldsymbol{x}_{adv}), y\big), \quad \text{s.t. } \|\boldsymbol{x}_{adv} - \boldsymbol{x}\|_p \leq \epsilon, \tag{1}$$

where $\mathcal{L}(\cdot,\cdot)$ denotes the prediction loss (*e.g.*, the cross-entropy loss), and $\boldsymbol{x}_{adv}$ denotes the adversary example. The ℓ_p norm of adversarial perturbation $(\boldsymbol{x}_{adv} - \boldsymbol{x})$ is constrained by the predefined value ϵ, and ℓ_p is set as ℓ_∞ in this paper following the most common setting (Dong et al. 2018).

For transfer adversarial attack, with the full accessible surrogate model, white-box attack methods based-on gradient are used. The classic method, Fast Gradient Sign Method (FGSM) (Goodfellow, Shlens, and Szegedy 2015), crafts the adversarial example with one-step update as:

$$\boldsymbol{x}_{adv} = \boldsymbol{x} + \epsilon \cdot \text{sign}\Big(\nabla_{\boldsymbol{x}} \mathcal{L}\big(f(\boldsymbol{x}), y\big)\Big) \tag{2}$$

Kurakin et al. (2016) extended FGSM to an iterative version (I-FGSM) with a small step size α as:

$$\boldsymbol{x}_{adv}^{t+1} = \Pi_\epsilon^{\boldsymbol{x}}\left(\boldsymbol{x}_{adv}^t + \alpha \cdot \text{sign}\Big(\nabla_{\boldsymbol{x}} \mathcal{L}\big(f(\boldsymbol{x}_{adv}^t), y\big)\Big)\right), \tag{3}$$

where $\Pi_\epsilon^{\boldsymbol{x}}(\cdot)$ ensures $\boldsymbol{x}_{adv}$ within the ϵ-ball of $\boldsymbol{x}$. I-FGSM improves the power of the white-box attack while impeding the transferability. To improve transferability, recent works can be divided into three categories: *i.e.*, gradient optimization, input transformation, and model modification.

Gradient Optimization: Some optimization techniques used to improve generalization have been introduced to improve adversarial transferability. Momentum Iterative Method (MI) (Dong et al. 2018) integrates momentum (Polyak 1964) into the gradient of each iteration, which allows previous gradients to guide the current updating direction to alleviate overfitting. Instead of using the current gradient for momentum, Wang and He (2021) tuned it with the gradient variance of the previous iteration. Lin et al. (2020) integrated Nesterov Accelerated Gradient (Nesterov 1983) which is also a popular optimization method.

Input Transformation: As an effective method for generalization, data augmentation has been utilized to make the perturbation underfit to specific data patterns. Diverse Input Method (DI) (Xie et al. 2019) applies different transformations (*e.g.*, random resizing and padding) to the input image each iteration. Translation Invariant Method (TI) (Dong et al. 2019) aims to calculate gradients of a set of transformed images, and applies a predefined kernel to the gradient according to translation invariance. Scale Invariant Method (SI) (Lin et al. 2020) optimizes perturbations over the scale copies of the input image.

Model Modification: To avoid overfitting the specific surrogate model, some works try to modify or enhance the surrogate model to a more general one. Model ensemble (Liu et al. 2017) is the most straightforward method via using multiple surrogate models, and meta learning (Yuan et al. 2021; Yin et al. 2021) is utilized to further improve the performance. However, multiple surrogate models are hard to obtain and extra computation is consumed. More works tend to focus on only one surrogate model. Ghost Networks (Li et al. 2020b) applies random dropout and skip-connection erasion to simulate plenty of models. Intermediate Level Attack (ILA) (Huang et al. 2019) crafts transferable adversarial examples by removing the later layers of the surrogate model as low-level features extracted by DNNs may be shared. Instead of altering the forward propagation process which influences predictions, some methods only modify the back propagation process. Wu et al. (2020) experimentally concluded the skip connection benefits adversarial transferability, and introduces a decay factor to reduce gradients from the residual modules. Inspired by the linear nature hypothesis (Goodfellow, Shlens, and Szegedy 2015), Linear Backpropagation (LinBP) (Guo, Li, and Chen 2020) skips some non-linear activation layers on back propagation.

Methodology

To improve the transferability of adversarial examples, we propose a novel framework termed *Learning to Learn Transfer Attack* (LLTA), which generates generalized adversarial perturbations via meta learning with both data and model augmentation as depicted in Fig. 1. By treating attacking a combination of a specific data and model as a task, we expect the crafted adversarial perturbations work on various tasks for transferability. To this end, we first propose a task generation module with both data and model augmentation, and tasks are generated during optimization iteration. Then we introduce the meta-learning strategy to make full use of generated tasks, where tasks are divided into two sets and adversarial perturbations are crafted on one set and finetuned on another set.

Task Generation

The poor transferability is caused by the gap between the surrogate model and the victim model, which can be further considered as the different data patterns extracted by two models and different behaviors of two models on forward and backward propagation. To narrow the gap, we need to optimize the adversarial perturbation via both data and model augmentation in case of overfitting to the surrogate model. Reviewing Eq. 3 in I-FGSM, the adversarial perturbations is updated mainly on the gradient, *i.e.*, $\nabla_{\boldsymbol{x}_{adv}^t}\mathcal{L}\big(f(\boldsymbol{x}_{adv}^t), y\big)$. We rewrite the gradient as $G(\boldsymbol{x}_{adv}^t, f)$ and define a task as attacking an augmented data and model. Then, we can optimize the adversarial perturbation with tasks generated on the fly as:

$$\boldsymbol{x}_{adv}^{t+1} = \Pi_{\epsilon}^{\boldsymbol{x}}\bigg(\boldsymbol{x}_{adv}^t + \alpha \cdot \mathrm{sign}\big(\boldsymbol{g}^t\big)\bigg), \quad \boldsymbol{g}^t = \bigg(\frac{1}{n}\cdot\sum_n G\big(T_d(\boldsymbol{x}_{adv}^t), T_m(f)\big)\bigg), \tag{4}$$

where $T_d(\cdot)$ and $T_m(\cdot)$ denote random data transformation and model augmentation respectively.

Data Augmentation. Data augmentation is an effective method to enhance generalization and alleviate overfitting, which has been verified on various vision tasks. As complex augmentation strategies are commonly non-differentiable which prevents us from optimizing perturbations by utilizing gradients, we tend to seek simple and differentiable image transformations. Following previous works (Xie et al. 2018, 2019), we adopt the random resizing and padding operations. Particularly, given an input image with the width and height of W, we first resize it to a random size $W' \in [W, W/0.875]$ following the common practice. We then add random zeros pixels to the left and top of the resized image, and pad it on the right and bottom side for the size of $W/0.875$. To align with the size of the initial image, we finally resize the padded image to the size of W. Same as DI, we also transform the image with a probability of 0.5 to make sure the adversarial perturbation works for the original input. Based on the above discussion, the data augmentation function can be represented as:

$$T_d(\boldsymbol{x}) = \begin{cases}\mathrm{Resize}(\mathrm{RandPad}(\mathrm{RandResize}(\boldsymbol{x}))), & p \le 0.5\\ \boldsymbol{x}, & p > 0.5\end{cases},$$

where $p \sim \mathcal{U}(0,1)$ controls to use a transformed image or the original one.

Model Augmentation. Unlike model ensemble that needs multiple surrogate models, we aim to modify only one surrogate model for model augmentation. Moreover, to avoid perturbing the prediction of the pre-trained model, we focus on adjusting the back propagation process. Wu et al. (2020) has proposed the SGM method that introduces a decay factor in back propagation to reduce the gradient from the residual modules, as the skip connections are widely used and the gradient from it is more helpful for transferability. It inspires us to adopt different decay factors for model augmentation.

In SGM, only one single decay factor is used, which is chosen according to a validation set and is fixed during optimization for all input images. The inflexible factor setting harms the efficiency of SGM, and hinders model augmentation. As different layers play different roles and the decay factors should also vary for different inputs, we introduce a set of changeable decay factors instead. Denoting the set of decay factors as $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, ..., \gamma_L]^{\mathrm{T}} \in [0,1]^L$ and the factor for i-th residual layer as γ_i, we re-write the forward and back propagation of the residual layer as:

$$\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + \gamma_i \cdot f_{i+1}(\boldsymbol{z}_i) + C, \quad G = \nabla_{\boldsymbol{x}}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_L}\prod_{i=0}^{L-1}(\gamma_{i+1}\frac{\partial f_{i+1}}{\partial \boldsymbol{z}_i} + 1)\frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{x}}, \tag{5}$$

where $\boldsymbol{z}_i$ denotes the input of $(i+1)$-th residual layer, f_i denotes the i-th residual module, and C is a constant whose value is equal to $(1-\gamma_i)\cdot f_{i+1}(\boldsymbol{z}_i)$. Such modification is easy to implement and brings nearly no extra computation cost. Now, with the changeable decay factors, we can naturally augment the surrogate model by altering the value of factors.

Note that SGM achieves better results with a carefully picked decay factor, which inspired us that well-tuned decay factors will bring further performance improvement. To this end, we tend to sample the decay factors $\boldsymbol{\gamma}$ from an optimized distribution instead of a random one. Here we optimized the $\boldsymbol{\gamma}$ to reduce the gradient w.r.t. inputs, *i.e.*, $\|G\|_2$. The design behind the objective is two-fold. On the one hand, the $\boldsymbol{\gamma}$ should make the model towards locally linear for better transferability according to the "linear nature" hypothesis (Goodfellow, Shlens, and Szegedy 2015; Guo, Li, and Chen 2020). The linearity for a function $\ell(\boldsymbol{x})$ can be measured by the difference between it and its first-order Taylor expansion:

$$\begin{aligned}&|\ell(\boldsymbol{x}+\boldsymbol{\delta})-\ell(\boldsymbol{x})-\boldsymbol{\delta}^{\mathrm{T}}\nabla_{\boldsymbol{x}}\ell(\boldsymbol{x})|\\ \leq&|\ell(\boldsymbol{x}+\boldsymbol{\delta})-\ell(\boldsymbol{x})|+|\boldsymbol{\delta}^{\mathrm{T}}\nabla_{\boldsymbol{x}}\ell(\boldsymbol{x})|.\end{aligned} \tag{6}$$

Assuming the function is K-Lipschitz continuous, we can further bound the first term on the right side in Eq. 6 as:

$$|\ell(\boldsymbol{x}+\boldsymbol{\delta})-\ell(\boldsymbol{x})| \leq K|\boldsymbol{\delta}|. \tag{7}$$

Combining Eq. 6 and Eq. 7, we can conclude that reducing the gradient contributes linearity and transferability.

On the other hand, minimizing the gradient is also a classic method for robust model (Jakubovitz and Giryes 2018) which hinders adversarial attack. According to some regularization methods (Wu, Xia, and Wang 2020; Zheng, Zhang, and Mao 2021), optimizing in an adversarial situation brings a flatter loss landscape which actually introduces more generalization, which also supports our design.

To optimize $\boldsymbol{\gamma}$, we adopt the MLE-guided parameter search method (MGS) (Welleck and Cho 2021). MGS is a general optimization method regardless of whether the objective function is derivable or not, and it introduces more uncertainty which suits the model augmentation compared with the simple gradient descent method, In each iteration, the MGS samples some update directions from a predefined distribution (*e.g.*, a Gaussian distribution $\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})$), mixes them using the corresponding improvement in the objective function as weight, and finally update the parameter to be optimized. This can be formulated as:

$$\begin{aligned}&\boldsymbol{\Delta}_{\boldsymbol{\gamma}}^{n}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})\\ &p(\boldsymbol{\Delta}_{\boldsymbol{\gamma}}^{n}|\boldsymbol{\gamma}^t)=\frac{\exp\left(\|G(\boldsymbol{\gamma}^t)\|_2-\|G(\boldsymbol{\gamma}^t+\boldsymbol{\Delta}_{\boldsymbol{\gamma}}^{n})\|_2\right)}{\mathcal{N}(\boldsymbol{\Delta}_{\boldsymbol{\gamma}}^{n};\mathbf{0},\sigma^2\mathbf{I})}\\ &\boldsymbol{\gamma}=\boldsymbol{\gamma}+\sum\frac{p(\boldsymbol{\Delta}_{\boldsymbol{\gamma}}^{n}|\boldsymbol{\gamma}^t)}{\sum p(\boldsymbol{\Delta}_{\boldsymbol{\gamma}}^{n}|\boldsymbol{\gamma}^t)}\cdot\boldsymbol{\Delta}_{\boldsymbol{\gamma}}^{n}.\end{aligned} \tag{8}$$

We then obtain an optimized $\boldsymbol{\gamma}^\star$ after several iterations. Instead of performing MGS multiple times for multiple $\boldsymbol{\gamma}^\star$ leading high computational cost, we just optimize once, and then alter it randomly multiple times, *i.e.*, add noises sampled from the uniform distribution on $[-0.5, 0.5]^L$. This achieves similar attack performance but saves a great deal of computation. With multiple well-tuned decay factors, we finally complete model augmentation.

In summary, the model augmentation part can be stated as optimizing the decay factors $\boldsymbol{\gamma}^\star$ first, randomly changing them multiple times for multiple $\boldsymbol{\gamma}$, and applying $\boldsymbol{\gamma}$ on the surrogate model separately. As the $T_m(\cdot)$ is heavily about the $\boldsymbol{\gamma}$, we rewrite $G(\boldsymbol{x}, f)$ as $G(\boldsymbol{x}, \boldsymbol{\gamma})$.

Meta-Learning Optimization

Instead of simply generating the adversarial perturbation directly on these tasks, we use the philosophy of meta-learning to optimize so that the perturbation can better fit various scenarios. Unlike previous works (Yuan et al. 2021; Yin et al. 2021) which perform meta learning directly on multiple surrogate models, we only have one surrogate model and perform meta learning on the tasks with augmentation. Note that, the tasks we constructed are based on the diversity of transferable attacks, so making the perturbation adapt these tasks better will not lead to overfitting. The meta learning method is performed on each iteration of the perturbation update. In each iteration, with the tasks split into a support set and a query set, we perform meta training (training on support set) and meta testing (finetuning on query set) multiple times, and finally update the adversarial perturbations.

Algorithm 1: Learning to Learn Transferable Attack

```
Input: Source image $\boldsymbol{x}$, number of perturbation
       iterations $T$ and meta iteration $M$, size of
       support set $\#\mathcal{S}$ and query set $\#\mathcal{Q}$.
Output: Adversarial example $\boldsymbol{x}_{adv}$.
1  Initialize adversarial example $\boldsymbol{x}_{adv}^0=\boldsymbol{x}$.
2  for $t=0$ to $T-1$ do
3      // Pre-process decay factors $\boldsymbol{\gamma}$
4      Initialize $\boldsymbol{\gamma}$ with $\mathbf{0.5}$.
5      Optimize $\boldsymbol{\gamma}^\star$ by Eq. 8.
6      // Create task sets
7      $\mathcal{S}=\{\}, \mathcal{Q}=\{\}$
8      for $\mathcal{D}\in\{\mathcal{S},\mathcal{Q}\}$ do
9          for $i=0$ to $\#\mathcal{D}-1$ do
10             $\mathcal{D}=\mathcal{D}\cup\{(T_d(\boldsymbol{x}_{adv}^t),\boldsymbol{\gamma}^\star+\boldsymbol{\Delta})\}$,
               $\boldsymbol{\Delta}\sim\mathcal{U}([-0.5,0.5]^L)$.
11     // Meta-learning optimization
12     for $m=0$ to $M-1$ do
13         Sample a mini-batch $\mathcal{S}_i$ from $\mathcal{S}$.
14         // Meta-Training
15         $\boldsymbol{g}_{spt}=\frac{1}{|\mathcal{S}_i|}\sum_{(\boldsymbol{x}_s,\boldsymbol{\gamma}_s)\in\mathcal{S}_i}G(\boldsymbol{x}_s,\boldsymbol{\gamma}_s)$
16         $\boldsymbol{\delta'}=\epsilon\cdot\mathrm{sign}(\boldsymbol{g}_{spt})$
17         // Meta-Testing
18         $\boldsymbol{g}_{qry}=\frac{1}{|\mathcal{Q}|}\sum_{(\boldsymbol{x}_q,\boldsymbol{\gamma}_q)\in\mathcal{Q}}G(\boldsymbol{x}_q+\boldsymbol{\delta'},\boldsymbol{\gamma}_q)$
19     $\boldsymbol{x}_{adv}^{t+1}=\Pi_\epsilon^{\boldsymbol{x}}(\boldsymbol{x}_{adv}^t+\alpha\cdot\mathrm{sign}(\overline{\boldsymbol{g}}_{spt}+\overline{\boldsymbol{g}}_{qry}))$
20 return $\boldsymbol{x}_{adv}^T$
```

Meta-Training. With the tasks generation method we can generate sufficient tasks and then split them into the support set $\mathcal{S}$ and the query set $\mathcal{Q}$. In the meta-training stage, we use the support set $\mathcal{S}$ to craft a temporary adversarial perturbation. In each meta learning iteration, we sample a subset $\mathcal{S}_i\subset\mathcal{S}$ and follow Eq. 4 to calculate the average gradient w.r.t. input as:

$$\boldsymbol{g}_{spt}=\frac{1}{|\mathcal{S}_i|}\sum_{(\boldsymbol{x}_s,\boldsymbol{\gamma}_s)\in\mathcal{S}_i}G(\boldsymbol{x}_s,\boldsymbol{\gamma}_s). \tag{9}$$

Then, similar as FGSM, we obtain the temporary perturbation with a single step update:

$$\boldsymbol{\delta'}=\epsilon\cdot\mathrm{sign}(\boldsymbol{g}_{spt}). \tag{10}$$

Meta-Testing. To improve the generalization of temporary perturbation, we finetune it on the query set $\mathcal{Q}$ so that it can adapt more tasks. Particularly, we compute the query gradient $\boldsymbol{g}_{qry}$ here by adding the temporary perturbation:

$$\boldsymbol{g}_{qry} = \frac{1}{|\mathcal{Q}|} \sum_{(\boldsymbol{x}_q, \gamma_q) \in \mathcal{Q}} G(\boldsymbol{x}_q + \boldsymbol{\delta'}, \gamma_q). \tag{11}$$

Meta-Optimization. With, we finally update the actual adversarial perturbation with the gradient from both the support set and the query set for maximum utilization:

$$\boldsymbol{x}_{adv}^{t+1} = \Pi_\epsilon^{\boldsymbol{x}} \big(\boldsymbol{x}_{adv}^t + \alpha \cdot \text{sign}(\overline{\boldsymbol{g}}_{spt} + \overline{\boldsymbol{g}}_{qry}) \big), \tag{12}$$

where $\overline{\boldsymbol{g}}_{spt}$ and $\overline{\boldsymbol{g}}_{qry}$ denote the average gradient over meta learning iterations, respectively.

We refer to the method combining the task generation and meta learning optimization as Learning to Learn Transferable Attack (LLTA), and summarize it in Alg. 1.

Experiment

Experiment Setup

Datasets. Following most of the previous works, we report the results on the ImageNet-compatible dataset in the NIPS 2017 adversarial competition (Kurakin et al. 2018), which contain $1,000$ categories and one image per category. We tune hyper-parameters on another $1,000$ images randomly chosen from ImageNet validation set (Deng et al. 2009).

Surrogate and Victim Models. For surrogate models, following SGM (Wu et al. 2020), we choose eight different models from ResNet family (He et al. 2016) (ResNet-18/34/50/101/152) and DenseNet family (Huang et al. 2017) (DenseNet-121/169/201). We mainly report the results on the ResNet-50 and DenseNet-121 as they are more popular. For victim models, we consider two transferable attack scenarios, *i.e.*, naturally trained models and robust models. We pick popular models including above surrogate models along with VGG-16 (Simonyan and Zisserman 2015), Inception-v3 (IncV3) (Szegedy et al. 2016), Inception-v4 (IncV4), and Inception-ResNet-v2 (IncResV2) (Szegedy et al. 2017) for naturally trained models. We omit the white-box situation when the surrogate model and the victim model are the same. And for robust models, following the common practice (Dong et al. 2019; Xie et al. 2019; Lin et al. 2020), we choose four adversarially trained models (Tramèr et al. 2018), *i.e.*, IncV3$_{ens3}$, IncV3$_{ens4}$, IncResV2$_{ens3}$ and IncV3$_{adv}$[1].

Compared Baselines. To demonstrate the effectiveness of the our proposed LLTA, we compare it with existing competitive baselines, *i.e.* I-FGSM (Kurakin et al. 2016), DI (Xie et al. 2019), TI (Dong et al. 2019), MI (Dong et al. 2018), SGM (Wu et al. 2020) and LinBP (Guo, Li, and Chen 2020). We report the transfer attack success rate as the evaluation metric, which measures whether the victim model keeps the top-1 prediction with the adversarial perturbation.

Experiment Details. We follow the attack setting in most of previous works (Kurakin et al. 2016; Dong et al. 2018, 2019) with the maximum ℓ_p-norm $\epsilon = 16$, number of iteration $T = 10$, and step size $\alpha = \epsilon/T = 1.6$. We set other parameters following the original setting in baselines. LinBP applies the linearly backpropagate only on some specific layers, and we follow their setting to choose layers for the ResNet family. For DenseNet models which are not evaluated in their original paper, we test all layers and choose the layers for the best results on the validation set. For our LLTA, we set the size of the support set as 20, the size of the query set as 10, and the number of meta iterations as 5, respectively. For MGS used in LLTA, we set the number both of iteration and sampled update directions as 5.

[1]https://github.com/JHL-HUST/SI-NI-FGSM

Comparison of Transferability

Attacking Performance. We first evaluate the performance of our proposed LLTA and the baselines on naturally trained models. We report the performance with ResNet-50 and DenseNet-121 as the surrogate model in Tab. 1, and report the average attack success rates on black-box victim models as well. From Tab. 1, we can conclude that though with a little performance drop on the white-box scenario, the adversarial perturbations generated by our method beat other methods on all black-box scenarios. Averagely, our proposed LLTA achieves more than 86% attack success rates for both ResNet-50 and DenseNet-121 models, which are 6.67% higher than LinBP and 9.07% higher than SGM respectively. We owe this to both data augmentation and model augmentation used in our method, which makes the adversarial perturbations more generalized. We also find that the perturbations generated from ResNet-50 work better on ResNet models while worse on DenseNet models than perturbations from DenseNet-121. It is consistent with the intuition that the transfer attack works better when the surrogate model is similar to the victim one. Similarly, the performance of perturbations from both surrogate models degrades on Inception-based victim models as the models are more different. However, for shallow models like VGG-16, both perturbations work well.

To further validate the performance of our method, we attack four adversarially trained models that are more robust, and present the results in Tab. 2. The average attack success rates degrade 30%-40% for all methods compared with naturally trained victim models. Under this situation, our method achieves the best results with 46.18% and 57.28% success rates for two surrogate models, which are more than 9.84% higher than the second-best method MI.

We report the average attack success on black-box scenarios for all eight surrogate models in Fig. 2. Our method consistently outperforms baselines by a large margin for nearly all surrogate models. Unlike methods like LinBP where the performance varies rapidly for different models, our method keeps similar performance on different surrogate models except for ResNet-18, which indicates that our method does not heavily depend on the surrogate model and the adversarial perturbations are generalized. For ResNet-18 model, as it is shallow and few decay factors are used (4 factors compared with 12 for ResNet-50), the model augmentation is not fully utilized which leads to unsatisfactory performance.

Combining with Existing Methods. Considering that our method actually utilizes DI which may bring unfair compar-

	Attack	VGG16	RN18	RN34	RN50	RN101	RN152	DN121	DN169	DN201	IncV3	IncV4	IncResV2	Avg.
RN50	I-FGSM	60.1	63.9	68.5	**100.0***	79.6	69.2	68.5	63.8	61.7	30.7	25.5	19.5	55.55
	TI	59.7	66.7	70.4	**100.0***	77.0	72.4	74.2	70.4	68.7	48.8	45.1	38.7	62.92
	DI	74.7	80.2	81.5	**100.0***	86.7	81.8	81.2	80.0	77.1	49.6	44.7	36.3	70.35
	MI	80.0	84.1	87.0	99.9*	90.5	86.8	87.2	85.0	83.4	62.2	52.4	50.8	77.22
	LinBP	88.6	88.7	89.1	**100.0***	93.6	89.1	90.1	87.7	85.5	55.6	56.5	48.5	79.36
	SGM	86.8	85.3	86.7	**100.0***	92.8	88.2	86.1	85.2	83.4	55.0	54.1	45.4	77.18
	LLTA	**93.2**	**92.1**	**94.3**	99.8*	**95.5**	**93.5**	**92.9**	**92.8**	**91.3**	**72.5**	**68.6**	**66.7**	**86.67**
DN121	I-FGSM	65.5	66.5	65.5	73.2	63.9	56.9	**100.0***	83.2	79.2	36.2	33.2	23.5	58.80
	TI	59.6	64.9	64.7	67.9	60.8	58.7	**100.0***	79.2	78.1	52.2	45.8	40.7	61.15
	DI	74.8	75.8	78.6	81.0	73.0	72.2	**100.0***	89.4	85.4	50.8	47.7	38.8	69.77
	MI	82.7	83.6	81.0	85.5	80.7	77.5	**100.0***	91.9	90.4	64.1	59.3	53.6	77.30
	LinBP	91.5	87.7	86.8	90.7	83.6	81.5	**100.0***	94.4	91.1	61.0	63.5	52.8	80.42
	SGM	84.5	84.5	85.9	88.3	83.5	81.9	**100.0***	92.3	91.5	57.9	57.3	48.1	77.79
	LLTA	**91.3**	**92.2**	**92.0**	**93.3**	**90.5**	**89.7**	99.9*	**94.8**	**94.2**	**73.4**	**72.2**	**67.3**	**86.45**

Table 1: The transfer attack success rate (%) on naturally trained models. (RN: ResNet, DN: DenseNet, Avg.: average success rate on all black-box victim models, *: white-box attack.)

	Attack	IncV3$_{ens3}$	IncV3$_{ens4}$	IncResV2$_{ens3}$	IncV3$_{adv}$	Avg
RN50	I-FGSM	17.3	18.5	11.2	18.5	16.38
	TI	39.0	37.5	25.3	42.4	36.05
	DI	29.7	29.5	17.9	31.2	27.08
	MI	41.9	43.4	28.7	43.4	39.35
	LinBP	34.5	32.5	20.9	39.8	31.93
	SGM	30.4	28.4	18.6	36.9	28.58
	LLTA	**50.6**	**47.3**	**33.6**	**53.2**	**46.18**
DN121	I-FGSM	21.8	21.5	13.1	23.5	19.98
	TI	40.4	41.2	28.2	47.6	39.35
	DI	30.6	32.0	20.5	35.2	29.58
	MI	46.4	48.6	33.8	48.9	44.43
	LinBP	39.3	38.3	22.6	44.5	36.18
	SGM	36.8	36.8	25.5	42.5	35.40
	LLTA	**59.1**	**60.5**	**46.8**	**62.7**	**57.28**

Table 2: Results on adversarially trained models.

Attack	ResNet-50	DenseNet-121
LinBP	79.36	80.42
LinBP+DI	61.45	59.66
LinBP+MI	88.51	88.93
LinBP+DI+MI	81.12	79.05
SGM	77.18	77.79
SGM+DI	79.46	79.57
SGM+MI	86.10	84.95
SGM+DI+MI	88.31	87.76
LLTA	86.67	86.45
LLTA+MI	**92.24**	**91.76**

Table 3: The results of combining with DI and MI.

ison and the performance of optimization methods like MI is competitive, we finally report the average performance of LinBP, SGM, and our method combining with DI and MI on naturally trained models in Tab. 3. We find that DI conflicts with LinBP that leads to large performance degradation, while promotes SGM with about 2% performance improvement. And the two methods are still not competitive with our methods with DI. Combining with MI will bring improvement for all methods, and our method is still the best with at least a 5.32% higher success rate than others.

Various Epsilon ϵ. We further evaluate the performance for attacking natural models under different epsilon ϵ, *i.e.*, $\epsilon = 1, 2, 4, 8, 16, 32$. The results are depicted in Fig. 3. The performance of all methods increases along with ϵ increases, and the curves are similar for ResNet-50 and DenseNet-121. When the ϵ is small enough (*e.g.*, $\epsilon = 1$), all methods perform similarly and the success rates are all low. Except this, the curves of our method are above others with different values of ϵ, which demonstrates the superiority of our method.

Ablation Study

To verify the contribution of each part in LLTA, we conduct the ablation study by removing each part. Our method relies on data augmentation and model augmentation along with the meta learning strategy. Moreover, for model augmentation, we first train the decay factors for a better initialization, and then randomly alter them to obtain multiple models. Based on the above discussion, we report the ablation results in Tab. 4. We first remove the data augmentation and model augmentation parts separately, which brings 2.09% and 23.60% performance degradation, respectively. It indicates that both data augmentation and model augmentation contribute the transferability, and model augmentation plays a more significant role. It is easy to understand as data patterns are extracted from model and model augmentation can bring diverse data patterns as well. We further check each part in the model augmentation, *i.e.*, optimizing decay factors and random alteration. Without optimizing decay factors, the decay factors are all initialized as 0.5 and then randomly altered. We observe a performance decline of more than 7.62%. It demonstrates well-tuned decay factors work better than random ones. Without random alteration, it means only one augmented model with the optimized decay factors is used. As it contributes to the model augmentation mainly, a poor attack result is obtained. Finally, we evaluate the performance after removing the meta learning strategy and conclude that it is as significant as the random alteration. In addition, we also verify whether using multiple decay factors instead of a single one as SGM is helpful. We obtain an average success rate of 83.39% with a single factor, and 84.83% with 4 factors (one factor per block) respectively, compared with our method of 86.56%. Since

	Attack	Natural	Robust	Total
RN50	**LLTA**	**86.67**	**46.18**	**75.87**
	-DataAugment	84.32 (-2.35)	42.28 (-3.90)	73.11 (-2.76)
	-ModelAugment	57.95 (-28.72)	28.48 (-17.70)	50.09 (-25.78)
	-OptimizeFactor	80.38 (-6.29)	34.90 (-11.28)	68.25 (-7.62)
	-RandomAlter	71.07 (-15.60)	38.35 (-7.83)	62.35 (-13.53)
	-MetaLearning	74.43 (-12.24)	28.58 (-17.60)	62.20 (-13.67)
DN121	**LLTA**	**86.45**	**57.28**	**79.11**
	-DataAugment	86.05 (-0.40)	53.10 (-4.17)	77.70 (-1.41)
	-ModelAugment	63.77 (-22.68)	34.55 (-23.73)	57.69 (-21.41)
	-OptimizeFactor	81.01 (-5.44)	42.38 (-14.90)	71.37 (-7.74)
	-RandomAlter	72.66 (-13.79)	44.55 (-12.73)	66.38 (-12.73)
	-MetaLearning	75.96 (-10.49)	37.88 (-19.40)	66.73 (-12.38)

Table 4: The result of albation experiments: w/ complete LLTA (line 1); w/o data augmentation (line 2); w/o model augmentation (line 3); w/ decay factors initialized as 0.5 and not optimized (line 4); w/ decay factors optimized w/o further random alteration (line 5); w/o meta learning (line 6).

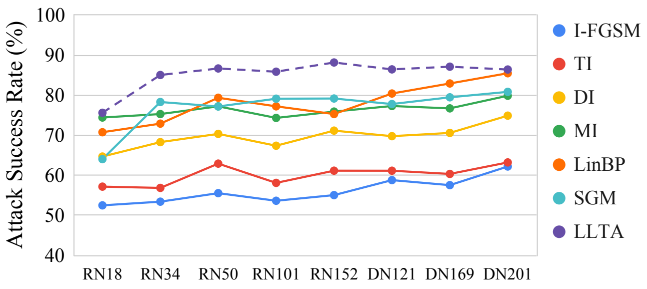

Figure 2: The average attack success rate on naturally trained models with various surrogate models.

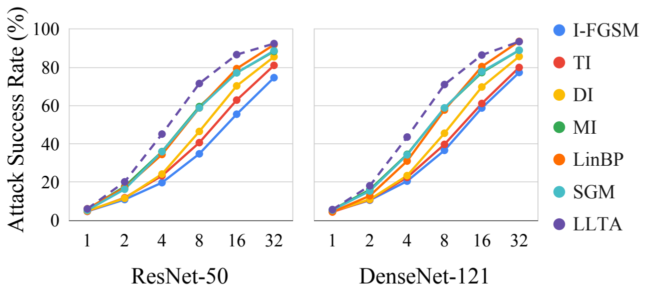

Figure 3: The average attack success rate on naturally trained models with various maximum perturbation ϵ.

both are lower than our method using one factor per residual layer and the result with 4 factors is higher than the one with 1 factor, we conclude that using multiple decay factors does help for transferability.

Results on Real-world System

To explore the practical potential of our method, we finally test the adversarial examples generated by LLTA along with SGM and LinBP on a real-world image recognition system, *i.e.*, Google Cloud Vision API[2]. Given an image, this API will return a list of probable labels. We randomly select 100 images from the dataset and generate adversarial examples

[2]https://cloud.google.com/vision/docs/drag-and-drop

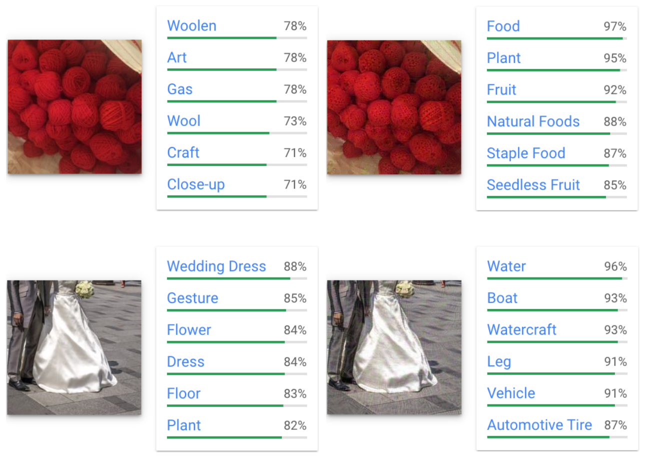

Figure 4: Examples for attacking Google Cloud Vision API. Images on the left are the original ones while images on the right are perturbed by LLTA on ResNet-50.

with ResNet-50 as the surrogate model for evaluation. We define a successful attack as the original top-1 prediction not appearing in the return list of the adversarial image, and get 22%, 37%, and 42% success rates for SGM, LinBP, and our method, respectively. We also give two visualization examples in Fig. 4, where woolen is recognized as fruit and wedding dress is recognized as boat above water.

Conclusion

In this paper, we track the problem of adversarial perturbations overfitting to the surrogate model in the transfer adversarial attack. we propose a novel framework termed Learning to Learn Transferable Attack (LLTA), which generates generalized adversarial perturbations by meta learning on tasks including both data and model augmentation. In particular, we first define a task as attacking a specific data and model and generate tasks via data and model augmentation. For data augmentation, simple random resizing and padding are utilized. For model augmentation, a set of changeable decay factors is introduced. Finally, we use the meta learning method to further optimize the adversarial perturbations over tasks. We evaluate our proposed method on the widely-used ImageNet-compatible benchmark with popular pre-trained neural networks, and show a higher transfer attack performance of LLTA compared with the state-of-the-art methods. We also check the practicality of our proposed method against the real-world image recognition service, *i.e.*, Google Cloud Vision API. We hope this work can serve as an inspiration for more generalized methods and robust models. We also believe that the decay factors learned by our method can help interpretability for deep models. Besides, LLTA consumes more time due to extra iterations for optimal γ and meta learning, which is unsuitable for the online scenario. We leave these for the future work.

Acknowledgements

This work is supported by the National Science Fund for Distinguished Young Scholars (No.62025603), the National Natural Science Foundation of China (No.U1705262, No. 62072386, No. 62072387, No. 62072389, No. 62002305, No.61772443, No.61802324 and No.61702136), Guangdong Basic and Applied Basic Research Foundation (No.2019B1515120049), the Natural Science Foundation of Fujian Province of China (No.2021J01002), and the Fundamental Research Funds for the central universities (No. 20720200077, No. 20720200090 and No. 20720200091).

References

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting adversarial attacks with momentum. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Dong, Y.; Pang, T.; Su, H.; and Zhu, J. 2019. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Fang, J.; Yang, R.; Chen, Q. A.; Liu, M.; Li, B.; et al. 2021. Invisible for both camera and LiDAR: security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *IEEE Symposium on Security and Privacy*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Guo, Y.; Li, Q.; and Chen, H. 2020. Backpropagating linearly improves transferability of adversarial examples. In *Advances in Neural Information Processing Systems*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Huang, Q.; Katsman, I.; He, H.; Gu, Z.; Belongie, S.; and Lim, S.-N. 2019. Enhancing adversarial example transferability with an intermediate level attack. In *IEEE International Conference on Computer Vision*.

Ilyas, A.; Engstrom, L.; and Madry, A. 2019. Prior convictions: Black-box adversarial attacks with bandits and priors. In *International Conference on Learning Representations*.

Jakubovitz, D.; and Giryes, R. 2018. Improving DNN robustness to adversarial attacks using jacobian regularization. In *European Conference on Computer Vision*.

Kurakin, A.; Goodfellow, I.; Bengio, S.; Dong, Y.; Liao, F.; Liang, M.; Pang, T.; Zhu, J.; Hu, X.; Xie, C.; et al. 2018. Adversarial attacks and defences competition. In *The NIPS'17 Competition: Building Intelligent Systems*. Springer.

Kurakin, A.; Goodfellow, I.; Bengio, S.; et al. 2016. Adversarial examples in the physical world. In *International Conference on Learning Representations*.

Li, J.; Ji, R.; Chen, P.; Zhang, B.; Hong, X.; Zhang, R.; Li, S.; Li, J.; Huang, F.; and Wu, Y. 2021. Aha! Adaptive History-Driven Attack for Decision-Based Black-Box Models. In *IEEE International Conference on Computer Vision*.

Li, J.; Ji, R.; Liu, H.; Liu, J.; Zhong, B.; Deng, C.; and Tian, Q. 2020a. Projection & probability-driven black-box attack. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Li, Q.; Guo, Y.; and Chen, H. 2020. Yet another intermediate-level attack. In *European Conference on Computer Vision*.

Li, Y.; Bai, S.; Zhou, Y.; Xie, C.; Zhang, Z.; and Yuille, A. 2020b. Learning transferable adversarial examples via ghost networks. In *AAAI Conference on Artificial Intelligence*.

Lin, J.; Song, C.; He, K.; Wang, L.; and Hopcroft, J. E. 2020. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *International Conference on Learning Representations*.

Liu, Y.; Chen, X.; Liu, C.; and Song, D. 2017. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations*.

Naseer, M.; Khan, S.; Hayat, M.; Khan, F. S.; and Porikli, F. 2020. A self-supervised approach for adversarial robustness. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Nesterov, Y. 1983. A method for unconstrained convex minimization problem with the rate of convergence O (1/k$\hat{2}$). In *Doklady an ussr*.

Orekondy, T.; Schiele, B.; and Fritz, M. 2019. Knockoff nets: stealing functionality of black-Box models. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Polyak, B. T. 1964. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*.

Simonyan, K.; and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. A. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI Conference on Artificial Intelligence*.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2018. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*.

Wang, X.; and He, K. 2021. Enhancing the transferability of adversarial attacks through variance tuning. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, Y.; Li, J.; Liu, H.; Wang, Y.; Wu, Y.; Huang, F.; and Ji, R. 2021. Black-Box Dissector: Towards Erasing-based Hard-Label Model Stealing Attack. *arXiv preprint arXiv:2105.00623*.

Welleck, S.; and Cho, K. 2021. MLE-guided parameter search for task loss minimization in neural sequence modeling. In *AAAI Conference on Artificial Intelligence*.

Wu, D.; Wang, Y.; Xia, S.-T.; Bailey, J.; and Ma, X. 2020. Skip connections matter: On the transferability of adversarial examples generated with resnets. In *International Conference on Learning Representations*.

Wu, D.; Xia, S.-T.; and Wang, Y. 2020. Adversarial weight perturbation helps robust generalization. In *Advances in Neural Information Processing Systems*.

Xiao, Z.; Gao, X.; Fu, C.; Dong, Y.; Gao, W.; Zhang, X.; Zhou, J.; and Zhu, J. 2021. Improving transferability of adversarial patches on face recognition with generative models. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Xie, C.; Zhang, Z.; Yuille, A. L.; Wang, J.; and Ren, Z. 2018. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*.

Xie, C.; Zhang, Z.; Zhou, Y.; Bai, S.; Wang, J.; Ren, Z.; and Yuille, A. L. 2019. Improving transferability of adversarial examples with input diversity. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Yin, B.; Wang, W.; Yao, T.; Guo, J.; Kong, Z.; Ding, S.; Li, J.; and Liu, C. 2021. Adv-Makeup: A new imperceptible and transferable attack on face recognition. In *International Joint Conference on Artificial Intelligence*.

Yuan, Z.; Zhang, J.; Jia, Y.; Tan, C.; Xue, T.; and Shan, S. 2021. Meta gradient adversarial attack. In *IEEE International Conference on Computer Vision*.

Zheng, Y.; Zhang, R.; and Mao, Y. 2021. Regularizing neural networks via adversarial model perturbation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhong, Y.; and Deng, W. 2020. Towards transferable adversarial attack against deep face recognition. *IEEE Transactions on Information Forensics and Security*.