Simulating Unknown Target Models for Query-Efficient Black-box Attacks

Chen Ma, Li Chen*, and Jun-Hai Yong
School of Software, BNRist, Tsinghua University, Beijing, China
mac16@mails.tsinghua.edu.cn, {chenlee,yongjh}@tsinghua.edu.cn

Abstract

Many adversarial attacks have been proposed to investigate the security issues of deep neural networks. In the black-box setting, current model stealing attacks train a substitute model to counterfeit the functionality of the target model. However, the training requires querying the target model. Consequently, the query complexity remains high, and such attacks can be defended easily. This study aims to train a generalized substitute model called "Simulator", which can mimic the functionality of any unknown target model. To this end, we build the training data with the form of multiple tasks by collecting query sequences generated during the attacks of various existing networks. The learning process uses a mean square error-based knowledge-distillation loss in the meta-learning to minimize the difference between the Simulator and the sampled networks. The meta-gradients of this loss are then computed and accumulated from multiple tasks to update the Simulator and subsequently improve generalization. When attacking a target model that is unseen in training, the trained Simulator can accurately simulate its functionality using its limited feedback. As a result, a large fraction of queries can be transferred to the Simulator, thereby reducing query complexity. Results of the comprehensive experiments conducted using the CIFAR-10, CIFAR-100, and TinyImageNet datasets demonstrate that the proposed approach reduces query complexity by several orders of magnitude compared to the baseline method. The implementation source code is released online[1].

1. Introduction

Deep neural networks (DNNs) are vulnerable to adversarial attacks [3, 13, 39], which add human-imperceptible perturbations to benign images for the misclassification of the *target model*. The study of adversarial attacks is crucial in the implementation of robust DNNs [29]. Adversarial attacks can be categorized into two types, namely, white-box and black-box attacks. In the white-box attack setting, the target model is fully exposed to the adversary. Thus, the perturbation can be crafted easily by using gradients [4, 13]. In the black-box attack setting, the adversary only has partial information of the target model, and adversarial examples are crafted without any gradient information. Hence, black-box attacks (*i.e.,* query- and transfer-based attacks) are more practical in real-world scenarios.

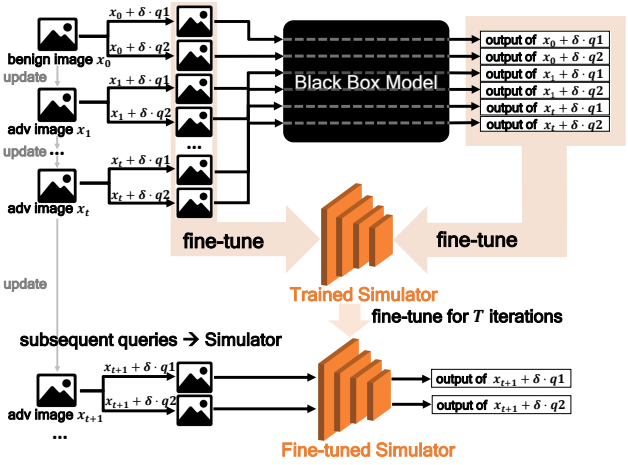


Figure 1: The procedure of the Simulator Attack, where $q1$ and $q2$ are the corresponding perturbations for generating query pairs in the attack (Algorithm 2). The queries of the first t iterations are fed into the target model to estimate the gradients. These queries and the corresponding outputs are collected to fine-tune the Simulator, which is trained without using the target model. The fine-tuned Simulator can accurately simulate the unknown target model, thereby transferring the queries and improving overall query efficiency.

Query-based attacks focus on estimating gradients through queries [6, 41, 19, 20]. These attacks are considered highly effective because of their satisfactory attack success rate. However, despite their practical merits, high query complexity inevitably arises when estimating the approximate gradient with high precision, resulting in costly procedures. In addition, the queries are typically underutilized,

*Corresponding author.
[1] https://github.com/machanic/SimulatorAttack

i.e., the implicit but profound messages returned from the target model are overlooked, because they are abandoned after estimating the gradients. **Thus, how to make full use of the feedback of the target model to enhance the query efficiency of attacks should be thoroughly investigated.**

Transfer-based attacks generate adversarial examples by using a white-box attack method on a source model to fool the target model [25, 33, 10, 18]. Transfer-based attacks have two disadvantages: (1) they cannot achieve a high success rate, and (2) they are weak in a targeted attack. To improve transferability, model stealing attacks train a local substitute model to mimic the black-box model using a synthetic dataset, in which the labels are given by the target model through queries [40, 36, 34]. In this way, the difference between the substitute and the target model is minimized, resulting in an increased attack success rate. However, such a training requires querying the target model. Consequently, the query complexity increases and such attacks can be defended easily by deploying a defense mechanism (*e.g.,* [35, 24]). Furthermore, the inevitable re-training to substitute a new target model is an expensive process. **Hence, how to train a substitute model without the target model requirement is worthy of further exploration.**

To eliminate the target model requirement in training, we propose a novel meta-learning-based framework to learn a generalized substitute model (*i.e.,* "Simulator") over many different networks, thereby exploiting their characteristics to achieve fast adaptation. Once trained and fine-tuned, the Simulator can mimic the output of any target model that is unseen in training, enabling it to eventually replace the target model (Fig. 1). Specifically, the intermediate queries of the real black-box attack are moved to the training stage, thus allowing the Simulator to learn how to distinguish the subtle differences among queries. All the training data are reorganized into a format consisting of multiple tasks. Each task is a small data subset consisting of a query sequence of one network. In this system, a large number of tasks allow the Simulator to experience the attacks of various networks.

We propose three components to optimize the generalization. First, a query-sequence level partition strategy is adopted to divide each task into meta-train and meta-test sets (Fig. 2) that match the iterations of fine-tuning and simulation in the attack, respectively (Fig. 1). Second, the mean square error (MSE)-based knowledge-distillation loss carries out the inner and outer loops of meta-learning. Finally, the meta-gradients of a batch of tasks are computed and then aggregated to update the Simulator and improve generalization. These strategies well address the problem of the target model requirement during training. In the attack (named "Simulator Attack"), the trained Simulator is fine-tuned using the limited feedback of the unknown target model to accurately simulate its output, thereby transferring its query stress (Fig. 1). Therefore, the feedback of the target model is fully utilized to improve query efficiency. In the proposed approach, the elimination of target models in training poses a new security threat, *i.e.,* the adversary with the minimal information about the target model can also counterfeit this model for a successful attack.

In this study, we evaluate the proposed method using the CIFAR-10 [23], CIFAR-100 [23], and TinyImageNet [38] datasets and compare it with natural evolution strategies (NES) [19], Bandits [20], Meta Attack [12], random gradient-free (RGF) [32], and prior-guided RGF (P-RGF) [8]. Experimental results show that the Simulator Attack can significantly reduce query complexity compared with the baseline method.

The main contributions of this work are summarized as follows:

(1) We propose a novel black-box attack by training a generalized substitute model named "Simulator". The training uses a knowledge-distillation loss to carry out the meta-learning between the Simulator and the sampled networks. After training, the Simulator only requires a few queries to accurately mimic any target model that is unseen in training.

(2) We identify a new type of security threat upon eliminating the target models in training: the adversary with the minimal information about the target model can also counterfeit this model for achieving the query-efficient attack.

(3) By conducting extensive experiments using the CIFAR-10, CIFAR-100, and TinyImageNet datasets, we demonstrate that the proposed approach achieves similar success rates as those of state-of-the-art attacks but with an unprecedented low number of queries.

2. Related Works

Query-based Attacks. Black-box attacks can be divided into query- and transfer-based attacks. Query-based attacks can be further divided into score- and decision-based attacks based on how much returned information from the target model can be used by the adversary. In score-based attacks, the adversary uses the output scores of the target model to generate adversarial examples. Most score-based attacks estimate the approximate gradient through zeroth-order optimizations [6, 2]. Then, the adversary can optimize the adversarial example with the estimated gradient. Although this type of approach can deliver a successful attack, it requires a large number of queries as each pixel needs two queries. Several improved methods have been introduced in the literature to reduce query complexity by using the principal components of the data [2], a latent space with reduced dimension [41], prior gradient information [20, 27], random search [14, 1], and active learning [37]. Decision-based attacks [5, 7] only use the output label of the target model. In this study, we focus on the score-based attacks.

Transfer-based Attacks. Transfer-based attacks generate adversarial examples on a source model and then transfer

them to the target model [25, 10, 18]. However, this type of attack cannot achieve a high success rate due to the large difference between the source model and the target model. Many efforts, including the use of model stealing attacks, have been made to improve the attack success rate. The original goal of model stealing attacks is to replicate the functionality of public service [42, 40, 30, 34]. Papernot *et al*. [36] expand the scope of use of model stealing attacks. They train a substitute model using a synthetic dataset labeled by the target model. Then, this substitute is used to craft adversarial examples. In this study, we focus on training a substitute model without using the target model.

Meta-learning. Meta-learning is useful in few-shot classification. It trains a meta-learner that can adapt rapidly to new environments with only a few samples. Ma *et al*. [28] propose MetaAdvDet to detect new types of adversarial attacks with high accuracy in order to utilize meta-learning in the adversarial attack field. The Meta Attack [12] trains an auto-encoder to predict the gradients of a target model to reduce the query complexity. However, its auto-encoder is only trained on natural image and gradient pairs and not on data from real attacks. Hence its prediction accuracy is not satisfied in the attack. The prediction of the large gradient map is also difficult for its lightweight auto-encoder. Thus, the Meta Attack only extracts the gradients with the top-128 values to update examples, resulting in poor performance. In comparison, the proposed Simulator in the current study is trained with knowledge-distillation loss for logits prediction; hence, the performance is not affected by the resolution of images. In addition, the training data are query sequences of black-box attacks, which are divided into meta-train set and meta-test set. The former corresponds to the fine-tuning iterations and the latter corresponds to simulation iterations in the attack. These strategies connect the training and the attack seamlessly to maximize the performance.

3. Method

3.1. Task Generation

During an attack, the trained Simulator must accurately simulate the outputs of any unknown target model when the feeding queries are only slightly different from one another. To this end, the Simulator should learn from the real attack, *i.e.,* the intermediate data (query sequences and outputs) generated in the attacks of various networks. For this purpose, several classification networks $\mathbb{N}_1, \ldots, \mathbb{N}_n$ are collected to construct the training tasks, creating a huge simulation environment to improve the general simulation capability (Fig. 2). Each task contains V query pairs $Q_1, \ldots, Q_V$ $\left(Q_i \in \mathbb{R}^D, i \in \{1, \cdots, V\}\right)$, where D is the image dimensionality. These pairs are generated by using Bandits to attack a randomly selected network. The data sources used by Bandits can be any image downloaded from the Internet. In this study, we use the training sets of the standard datasets with different data distributions from the tested images. Each task is divided into two subsets, namely, the meta-train set $\mathcal{D}_{mtr}$, which consists of the first t query pairs $Q_1, \ldots, Q_t$, and the meta-test set $\mathcal{D}_{mte}$ with the following query pairs $Q_{t+1}, \ldots, Q_V$. The former is used in the inner-update step of the training corresponding to the fine-tuning step in the attack stage. The latter corresponds to the attack iterations of using the Simulator as the substitute (Fig. 1). This partition connects the training and attack stages seamlessly. The logits outputs of $\mathbb{N}_1, \ldots, \mathbb{N}_n$ are termed as "pseudo labels". All query sequences and pseudo labels are cached in the hard drive to accelerate training.

3.2. Simulator Learning

Initialization. Algorithm 1 and Fig. 2 present the training procedure. In the training, we sample K tasks randomly to form a mini-batch. At the beginning of learning each task, the Simulator $\mathbb{M}$ reinitializes its weights using the weights θ learned by the last mini-batch. These weights are kept for computing meta-gradients in the outer-update step.

Meta-train. $\mathbb{M}$ performs the gradient descent on the meta-

Algorithm 1 Training procedure of the Simulator

Input: Training dataset D, Bandits attack algorithm $\mathcal{A}$, pre-trained classification networks $\mathbb{N}_1, \ldots, \mathbb{N}_n$, the Simulator network $\mathbb{M}$ and its parameters θ, feed-forward function f of $\mathbb{M}$, loss function $\mathcal{L}(\cdot, \cdot)$ defined in Eq. (1).

Parameters: Training iterations N, query sequence size V, meta-train set size t, batch size K, inner-update learning rate λ_1, outer-update learning rate λ_2, inner-update iterations T.

Output: The learned Simulator $\mathbb{M}$.

for $iter \leftarrow 1$ to N **do**
  sample K benign images $x_1, \ldots, x_K$ from D
  for $k \leftarrow 1$ to K **do** ▷ iterate over K tasks
    a network $\mathbb{N}_i \leftarrow$ sample from $\mathbb{N}_1, \ldots, \mathbb{N}_n$
    $Q_1, \ldots, Q_V \leftarrow \mathcal{A}(x_k, \mathbb{N}_i)$ ▷ query sequence
    $\mathcal{D}_{mtr} \leftarrow Q_1, \ldots, Q_t$
    $\mathcal{D}_{mte} \leftarrow Q_{t+1}, \ldots, Q_V$
    $\mathbf{p}_{\texttt{train}} \leftarrow \mathbb{N}_i(\mathcal{D}_{mtr})$
    $\mathbf{p}_{\texttt{test}} \leftarrow \mathbb{N}_i(\mathcal{D}_{mte})$ ▷ pseudo labels
    $\theta' \leftarrow \theta$ ▷ reinitialize $\mathbb{M}$'s weights
    for $j \leftarrow 1$ to T **do**
      $\theta' \leftarrow \theta' - \lambda_1 \cdot \nabla_{\theta'} \mathcal{L}\left(f_{\theta'}\left(\mathcal{D}_{mtr}\right), \mathbf{p}_{\texttt{train}}\right)$
    end for
    $L_i \leftarrow \mathcal{L}\left(f_{\theta'}\left(\mathcal{D}_{mte}\right), \mathbf{p}_{\texttt{test}}\right)$
  end for
  $\theta \leftarrow \theta - \lambda_2 \cdot \frac{1}{K} \sum_{i=1}^{K} \nabla_\theta L_i$ ▷ the outer update
end for
return $\mathbb{M}$

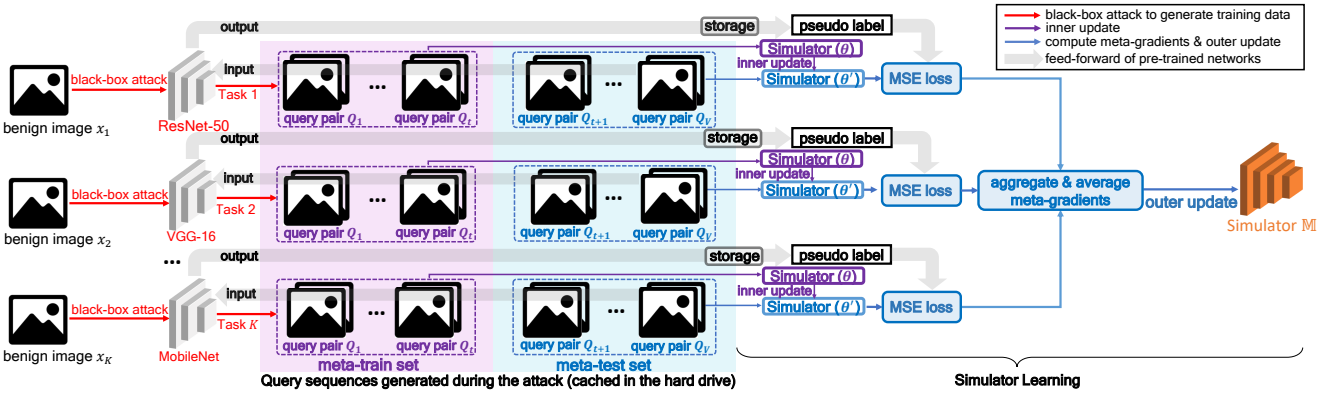


Figure 2: The procedure of training the Simulator in one mini-batch. Here, the sequences of query pairs generated during the attacks are collected as training data and then reorganized into multiple tasks. Each task contains the data generated from attacking one network and is further divided into meta-train set and meta-test set. Next, the Simulator network $\mathbb{M}$ reinitializes its weights to θ at the beginning of learning each task, after which it subsequently trains on the meta-train set. After several iterations (inner update), $\mathbb{M}$ converges and its weights are updated to θ'. The meta-gradients of $\mathbb{M}$ are computed based on the meta-test sets of K tasks and are then accumulated to update $\mathbb{M}$ (the outer update). The updated $\mathbb{M}$ is prepared for the next mini-batch learning. Finally, the learned Simulator can simulate any unknown black-box model using limited queries in the attack stage.

train set $\mathcal{D}_{mtr}$ for several iterations (the inner update). This step is similar to training a student model in a knowledge distillation, which matches the fine-tuning step of the attack.

Meta-test. After several iterations, $\mathbb{M}$'s weights are updated to θ'. Then, the loss L_i is computed based on meta-test set of the i-th task with θ'. Afterwards, the meta-gradient $\nabla_\theta L_i$ is calculated as a higher-order gradient. Then, $\nabla_\theta L_1, \ldots, \nabla_\theta L_K$ of K tasks are averaged as $\frac{1}{K}\sum_{i=1}^{K}\nabla_\theta L_i$ for updating $\mathbb{M}$ (the outer update), thus enabling $\mathbb{M}$ to learn the general simulation capability.

Loss Function. In the training, we adopt a knowledge-distillation-fashioned loss to induce the Simulator to output a similar prediction with the sampled network $\mathbb{N}_i$, which we use in both the inner and outer steps. Given the two queries $Q_{i,1}$ and $Q_{i,2}$ of the i-th query pair Q_i generated by Bandits[2], where $i \in \{1, \ldots, n\}$ and n represents the number of query pairs in the meta-train or meta-test set. The logits outputs of the Simulator and $\mathbb{N}_i$ are denoted as $\hat{\mathbf{p}}$ and $\mathbf{p}$, respectively. The MSE loss function defined in Eq. (1) pushes the predictions of the Simulator and the pseudo label closer.

$$\mathcal{L}(\hat{\mathbf{p}}, \mathbf{p}) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{\mathbf{p}}_{Q_{i,1}} - \mathbf{p}_{Q_{i,1}}\right)^2 + \frac{1}{n}\sum_{i=1}^{n}\left(\hat{\mathbf{p}}_{Q_{i,2}} - \mathbf{p}_{Q_{i,2}}\right)^2 \tag{1}$$

3.3. Simulator Attack

Algorithm 2 shows the Simulator Attack under the ℓ_p norm constraint. The query pairs of the first t iterations are fed to the target model (the warm-up phase). These queries and corresponding outputs are collected into a double-ended queue $\mathbb{D}$. Then, $\mathbb{D}$ drops the oldest item once it is full, which is beneficial in terms of focusing on new queries when fine-tuning $\mathbb{M}$ using $\mathbb{D}$. After warm-up, subsequent queries are fed into the target model every m iterations, and the fine-tuned $\mathbb{M}$ takes the rest. To be consistent with training, the gradient estimation steps follow that of Bandits. The attack objective loss function shown in Eq. (2) is maximized during the attack:

$$\mathcal{L}(\hat{y}, t) = \begin{cases} \max_{j \neq t} \hat{y}_j - \hat{y}_t, & \text{if untargeted attack;} \\ \hat{y}_t - \max_{j \neq t} \hat{y}_j, & \text{if targeted attack;} \end{cases} \tag{2}$$

where $\hat{y}$ represents the logits output of the Simulator or the target model, t is the target class in the targeted attack or the true class in the untargeted attack, and j indexes the other classes.

3.4. Discussion

During an attack, the Simulator must accurately simulate the outputs when feeding queries of the real attack. Thus, the Simulator is trained on the intermediate data of the real attack in a knowledge-distillation manner. None of existing meta-learning methods learn a simulator in this way, as they all focus on the few-shot classification or reinforcement learning problems. In addition, Algorithm 2 alternately feeds queries to $\mathbb{M}$ and the target model to learn the latest queries. The periodic fine-tuning is crucial in achieving a high success rate when faced with a difficult attack (*e.g.,* the result of the targeted attack in Fig. 3b).

[2]Bandits attack requires two queries in the finite difference for estimating a gradient. Thus, a query pair is generated in each iteration.

Algorithm 2 Simulator Attack under the ℓ_p norm constraint

Input: Input image $x \in \mathbb{R}^D$ where D is the image dimensionality, true label y of x, feed-forward function f of target model, Simulator $\mathbb{M}$, attack objective loss $\mathcal{L}(\cdot, \cdot)$.
Parameters: Warm-up iterations t, simulator-predict interval m, Bandits exploration τ, finite difference probe δ, OCO learning rate η_g, image learning rate η.
Output: x_{adv} that satisfies $\|x_{\text{adv}} - x\|_p \leq \epsilon$.

Initialize the adversarial example $x_{\text{adv}} \leftarrow x$
Initialize the gradient to be estimated $\mathbf{g} \leftarrow \mathbf{0}$
Initialize $\mathbb{D} \leftarrow deque(maxlen = t)$ ▷ a bounded double-ended queue with maximum length of t, adding a full $\mathbb{D}$ leads it to drop its oldest item automatically.
for $i \leftarrow 1$ to N **do**
  $\mathbf{u} \leftarrow \mathcal{N}(\mathbf{0}, \frac{1}{D}\mathbf{I})$ ▷ the same dimension with x
  $q1 \leftarrow \mathbf{g} + \tau\mathbf{u}, \quad q2 \leftarrow \mathbf{g} - \tau\mathbf{u}$
  $q1 \leftarrow q1/\|q1\|_2, \quad q2 \leftarrow q2/\|q2\|_2$
  if $i \leq t$ or $(i-t) \bmod m = 0$ **then**
    $\hat{y}_1 \leftarrow f(x_{\text{adv}} + \delta \cdot q1)$
    $\hat{y}_2 \leftarrow f(x_{\text{adv}} + \delta \cdot q2)$
    $\{x_{\text{adv}} + \delta \cdot q1, \hat{y}_1, x_{\text{adv}} + \delta \cdot q2, \hat{y}_2\}$ append $\mathbb{D}$
    if $i \geq t$ **then**
      Fine-tune $\mathbb{M}$ using $\mathbb{D}$ ▷ fine-tune $\mathbb{M}$ every m iterations after the warm-up phase.
    end if
  else
    $\hat{y}_1 \leftarrow \mathbb{M}(x_{\text{adv}} + \delta \cdot q1), \quad \hat{y}_2 \leftarrow \mathbb{M}(x_{\text{adv}} + \delta \cdot q2)$
  end if
  $\Delta_g \leftarrow \frac{\mathcal{L}(\hat{y}_1, y) - \mathcal{L}(\hat{y}_2, y)}{\tau\delta}\mathbf{u}$
  if $p = 2$ **then**
    $\mathbf{g} \leftarrow \mathbf{g} + \eta_g \cdot \Delta_g$
    $x_{\text{adv}} \leftarrow \prod_{\mathcal{B}_2(x,\epsilon)}(x_{\text{adv}} + \eta \cdot \frac{\mathbf{g}}{\|\mathbf{g}\|_2})$ ▷ $\prod_{\mathcal{B}_p(x,\epsilon)}$ denotes the ℓ_p norm projection under ℓ_p norm bound.
  else if $p = \infty$ **then** ▷ using the exponentiated gradient update [20] in the ℓ_∞ norm attack as follows.
    $\hat{\mathbf{g}} \leftarrow \frac{\mathbf{g}+\mathbf{1}}{2}$
    $\mathbf{g} \leftarrow \frac{\hat{\mathbf{g}} \cdot \exp(\eta_g \cdot \Delta_g) - (\mathbf{1}-\hat{\mathbf{g}}) \cdot \exp(-\eta_g \cdot \Delta_g)}{\hat{\mathbf{g}} \cdot \exp(\eta_g \cdot \Delta_g) + (\mathbf{1}-\hat{\mathbf{g}}) \cdot \exp(-\eta_g \cdot \Delta_g)}$
    $x_{\text{adv}} \leftarrow \prod_{\mathcal{B}_\infty(x,\epsilon)}(x_{\text{adv}} + \eta \cdot \text{sign}(\mathbf{g}))$
  end if
  $x_{\text{adv}} \leftarrow \text{Clip}(x_{\text{adv}}, 0, 1)$
end for
return x_{adv}

4. Experiment

4.1. Experiment Setting

Dataset and Target Models. We conduct the experiments using the CIFAR-10 [23], CIFAR-100 [23], and TinyImageNet [38] datasets. Following previous studies [45], 1,000 tested images are randomly selected from their validation sets for evaluation. In the CIFAR-10 and CIFAR-100 datasets, we follow Yan *et al*. [45] to select the target models: (1) a 272-layer PyramidNet+Shakedrop network (PyramidNet-272) [15, 44] trained using AutoAugment [9]; (2) a model obtained via neural architecture search called GDAS [11]; (3) a WRN-28 [46] with 28 layers and 10 times width expansion; and (4) a WRN-40 with 40 layers. In the TinyImageNet dataset, we select ResNeXt-101 (32x4d) [43], ResNeXt-101 (64x4d), and DenseNet-121 [17] with a growth rate of 32.

Method Setting. In the training, we generate the query sequence data $Q_1, \ldots, Q_{100}$ in each task. The meta-train set $\mathcal{D}_{mtr}$ contains $Q_1, \ldots, Q_{50}$, and the meta-test set $\mathcal{D}_{mte}$ consists of $Q_{51}, \ldots, Q_{100}$. We select ResNet-34 [16] as the backbone of the Simulator, which we trained for three epochs over 30,000 tasks. Here, 30 sampled tasks constitute a mini-batch. Training each Simulator with an NVIDIA Tesla V100 GPU lasted for 72 hours. The fine-tune iteration number is set to 10 in the first fine-tuning and then reduced to a random number from 3 to 5 for subsequent ones. In the targeted attacks, we set the target class to $y_{adv} = (y + 1) \bmod C$ for all attacks, where y_{adv} is the target class, y is the true class, and C is the class number. Following previous studies [8, 45], we use the attack success rate as well as the average and median values of queries as the evaluation metrics. Table 1 presents the default parameters.

Pre-trained Networks. In order to evaluate the capability of simulating unknown target models, we ensure that the selection of $\mathbb{N}_1, \ldots, \mathbb{N}_n$ in Algorithm 1 is different from the target models. A total of 14 networks are selected in the CIFAR-10 and CIFAR-100 datasets, and 16 networks are selected for the TinyImageNet dataset. The details can be found in the supplementary material. In experiments involving attacks of defensive models, we re-train the Simulator by removing the data of ResNet networks. This is because the defensive models adopt a backbone of ResNet-50.

Compared Methods. The compared methods include NES [19], Bandits [20], Meta Attack [12], RGF [32], and P-RGF [8]. Bandits is selected as the baseline. To ensure a fair comparison, the training data (*i.e.,* images and gradients) of the Meta Attack are generated by directly using the pre-trained classification networks of the present study. We translate the codes of NES, RGF, and P-RGF from the official implementations of TensorFlow into the PyTorch version for the experiments. P-RGF improves RGF query efficiency by utilizing a surrogate model, which adopts ResNet-110 [16] in the CIFAR-10 and CIFAR-100 datasets and ResNet-101 [16] in the TinyImageNet dataset. We exclude the experiments of RGF and P-RGF in the targeted attack experiments, because their official implements only support untargeted attacks. All methods are limited to the maximum of 10,000 queries in both untargeted and targeted attacks. We set the same ϵ values for all attacks, which are 4.6 and $8/255$ in the ℓ_2 norm attack and ℓ_∞ norm attack, respec-

name	default	description
λ_1 of the inner update	0.01	learning rate in the inner update.
λ_2 of the outer update	0.001	learning rate in the outer update.
maximum query times	10,000	the limitation of queries of each sample.
ϵ of ℓ_2 norm attack	4.6	the maximum distortion in ℓ_2 norm attack.
ϵ of ℓ_∞ norm attack	8/255	the maximum distortion in ℓ_∞ norm attack.
η of ℓ_2 norm attack	0.1	the image learning rate for updating image.
η of ℓ_∞ norm attack	1/255	the image learning rate for updating image.
η_g of ℓ_2 norm attack	0.1	OCO learning rate for updating $\mathbf{g}$.
η_g of ℓ_∞ norm attack	1.0	OCO learning rate for updating $\mathbf{g}$.
inner-update iterations	12	update iterations of learning meta-train set.
simulator-predict interval	5	the prediction iteration's interval of $\mathbb{M}$.
warm-up iterations t	10	the first t iterations of the attack.
deque $\mathbb{D}$'s length	10	the maximum length of $\mathbb{D}$.

Table 1: The default parameters setting of Simulator Attack.

Target Model	Method	Avg. Query	Med. Query	Max Query	Success Rate
PyramidNet-272	Rnd_init Simulator	105	52	1470	100%
	Vanilla Simulator	102	52	1374	100%
	Simulator Attack	**92**	**52**	**834**	100%

Table 2: Comparison of different simulators by performing ℓ_2 norm attack on the CIFAR-10 dataset. The Rnd_init Simulator uses an untrained ResNet-34 as the simulator; the Vanilla Simulator uses a ResNet-34 that is trained without using meta-learning as the simulator.

tively. The detailed configurations of all compared methods are provided in the supplementary material.

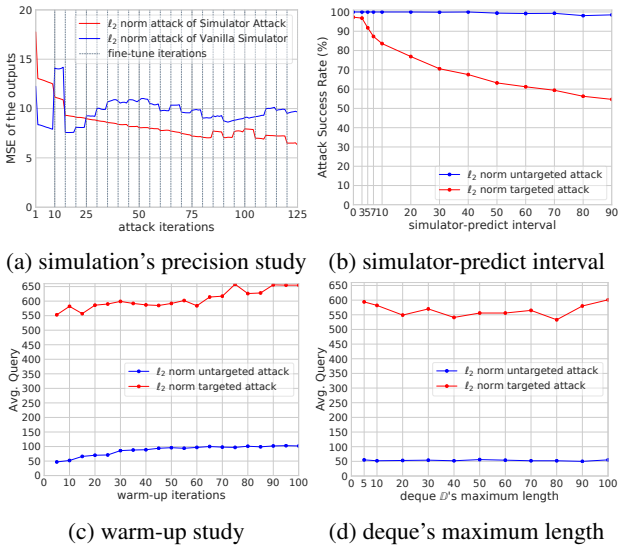
4.2. Ablation Study

The ablation study is conducted to validate the benefit of meta training and determine the effects of key parameters.

Meta Training. We validate the benefits of meta training by equipping with different simulators in the proposed algorithm. Simulator $\mathbb{M}$ is replaced with two networks for comparison, *i.e.,* Rnd_init Simulator: a randomly initialized ResNet-34 network without training, and Vanilla Simulator: a ResNet-34 network trained on the data of the present study but without using meta-learning. Table 2 shows the experimental results, which indicate that the Simulator Attack is able to achieve the minimum number of queries, thereby confirming the benefit of meta training. To inspect the simulation capacity in detail, we calculate the average MSE between outputs of simulators and the target model at different attack iterations (Fig. 3a). As indicated by the results, the Simulator Attack achieves the lowest MSE at most iterations, thus exhibiting its satisfactory simulation capability.

In control experiments, we check the effects of the key parameters of the Simulator Attack by adjusting one parameter while keeping others fixed, as listed in Table 1. The corresponding results are shown in Figs. 3b, 3c, and 3d.

Simulator-Predict Interval m. This parameter is the iteration interval that uses Simulator $\mathbb{M}$ to make predictions. A larger m results in fewer opportunities to fine-tune $\mathbb{M}$. When this happens, the Simulator cannot accurately simulate the target model in case of a difficult attack (*e.g.,* the



(a) simulation's precision study (b) simulator-predict interval

(c) warm-up study (d) deque's maximum length

Figure 3: We conduct ablation studies of the simulation's precision, simulator-predict interval, warm-up iterations, and deque $\mathbb{D}$'s maximum length by attacking a WRN-28 model in the CIFAR-10 dataset. The results indicate the following: (1) the meta training is beneficial for achieving an accurate simulation (Fig. 3a), (2) a difficult attack (*e.g.,* targeted attack) requires a small simulator-predict interval (Fig. 3b), and (3) more warm-up iterations cause higher average queries (Fig. 3c).

targeted attack in Fig. 3b), resulting in a low success rate.

Warm-up. As shown in Fig. 3c, more warm-up iterations lead to a higher average query, because more queries are fed into the target model in the warm-up phase.

4.3. Comparisons with State-of-the-Art Methods

Results of Attacks on Normal Models. In this study, the normal model is the classification model without the defensive mechanism. We conduct experiments on the target models described in Section 4.1. Tables 3 and 4 show the results of the CIFAR-10 and CIFAR-100 datasets, respectively, whereas Tables 6 and 7 present the results of the TinyImageNet dataset. The results reveal the following: (1) the Simulator Attack can gain up to 2× reduction in the average and median values of the queries compared with the baseline Bandits, and (2) the Simulator Attack can obtain significantly fewer queries and a higher attack success rate than the Meta Attack [12] (*e.g.,* the low success rates of Meta Attack in Tables 6 and 7). The poor performance of the Meta Attack can be attributed to its high-cost gradient estimation (specifically the use of ZOO [6]).

Experimental Figures. Tables 3, 4, 6, and 7 show the results obtained after setting the maximum number of queries to 10,000. To further inspect the attack success rates at different maximum queries, we perform ℓ_∞ norm attacks by

Dataset	Norm	Attack	Attack Success Rate				Avg. Query				Median Query			
			PyramidNet-272	GDAS	WRN-28	WRN-40	PyramidNet-272	GDAS	WRN-28	WRN-40	PyramidNet-272	GDAS	WRN-28	WRN-40
CIFAR-10	ℓ_2	NES [19]	99.5%	74.8%	99.9%	99.5%	200	123	159	154	150	100	100	100
		RGF [32]	100%	100%	100%	100%	216	168	153	150	204	152	102	152
		P-RGF [8]	100%	100%	100%	100%	**64**	40	76	73	62	**20**	64	64
		Meta Attack [12]	99.2%	99.4%	98.6%	99.6%	2359	1611	1853	1707	2211	1303	1432	1430
		Bandits [20]	100%	100%	100%	100%	151	66	107	98	110	54	80	78
		Simulator Attack	100%	100%	100%	100%	92	**34**	**48**	**51**	**52**	26	**34**	**34**
	ℓ_∞	NES [19]	86.8%	71.4%	74.2%	77.5%	1559	628	1235	1209	600	300	400	400
		RGF [32]	99%	93.8%	98.6%	98.8%	955	646	1178	928	668	460	663	612
		P-RGF [8]	97.3%	97.9%	97.7%	98%	**742**	337	703	564	**408**	128	236	217
		Meta Attack [12]	90.6%	98.8%	92.7%	94.2%	3456	2034	2198	1987	2991	1694	1564	1433
		Bandits [20]	99.6%	100%	99.4%	99.9%	1015	391	611	542	560	166	224	228
		Simulator Attack	96.5%	99.9%	98.1%	98.8%	779	**248**	**466**	**419**	469	**83**	**186**	**186**
CIFAR-100	ℓ_2	NES [19]	92.4%	90.2%	98.4%	99.6%	118	94	102	105	100	50	100	100
		RGF [32]	100%	100%	100%	100%	114	110	106	106	102	101	102	102
		P-RGF [8]	100%	100%	100%	100%	54	46	54	73	62	62	62	62
		Meta Attack [12]	99.7%	99.8%	99.4%	98.4%	1022	930	1193	1252	783	781	912	913
		Bandits [20]	100%	100%	100%	100%	58	54	64	65	42	42	52	53
		Simulator Attack	100%	100%	100%	100%	**29**	**29**	**33**	**34**	**24**	**24**	**26**	**26**
	ℓ_∞	NES [19]	91.3%	89.7%	92.4%	89.3%	439	271	673	596	204	153	255	255
		RGF [32]	99.7%	98.8%	98.9%	98.9%	385	420	544	619	256	255	357	357
		P-RGF [8]	99.3%	98.2%	98%	97.8%	308	220	371	480	147	116	136	181
		Meta Attack [12]	99.7%	99.8%	97.4%	97.3%	1102	1098	1294	1369	912	911	1042	1040
		Bandits [20]	100%	100%	99.8%	99.8%	266	209	262	260	68	57	107	92
		Simulator Attack	100%	100%	99.9%	99.9%	**129**	**124**	**196**	**209**	**34**	**28**	**58**	**54**

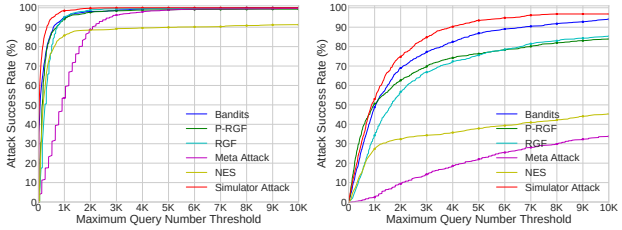
Table 3: Experimental results of untargeted attack in CIFAR-10 and CIFAR-100 datasets.

Dataset	Norm	Attack	Attack Success Rate				Avg. Query				Median Query			
			PyramidNet-272	GDAS	WRN-28	WRN-40	PyramidNet-272	GDAS	WRN-28	WRN-40	PyramidNet-272	GDAS	WRN-28	WRN-40
CIFAR-10	ℓ_2	NES [19]	93.7%	95.4%	98.5%	97.7%	1474	1515	1043	1088	1251	999	881	882
		Meta Attack [12]	92.2%	97.2%	74.1%	74.7%	4215	3137	3996	3797	3842	2817	3586	3329
		Bandits [20]	99.7%	100%	97.3%	98.4%	852	718	1082	997	458	538	338	399
		Simulator Attack (m=3)	99.1%	100%	98.5%	95.6%	896	718	990	980	373	**388**	217	249
		Simulator Attack (m=5)	97.6%	99.9%	96.4%	94%	**815**	**715**	**836**	**793**	**368**	400	**206**	**245**
	ℓ_∞	NES [19]	63.8%	80.8%	89.7%	88.8%	4355	3942	3046	3051	3717	3441	2535	2592
		Meta Attack [12]	75.6%	95.5%	59%	59.8%	4960	3461	3873	3899	4736	3073	3328	3586
		Bandits [20]	84.5%	98.3%	76.9%	79.8%	2830	1755	2037	2128	2081	1162	1178	1188
		Simulator Attack (m=3)	80.9%	97.8%	83.1%	82.2%	2655	1561	1855	1806	1943	918	1010	1018
		Simulator Attack (m=5)	78.7%	96.5%	80.8%	80.3%	**2474**	**1470**	**1676**	**1660**	**1910**	**917**	**957**	**956**
CIFAR-100	ℓ_2	NES [19]	87.6%	77%	89.3%	87.6%	1300	1405	1383	1424	1102	1172	1061	1049
		Meta Attack [12]	86.1%	88.7%	63.4%	43.3%	4000	3672	4879	4989	3457	3201	4482	4865
		Bandits [20]	99.6%	100%	98.9%	91.5%	1442	847	1645	2436	1058	679	1150	1584
		Simulator Attack (m=3)	99.3%	100%	98.6%	92.6%	921	724	1150	1552	666	519	779	1126
		Simulator Attack (m=5)	97.8%	99.6%	95.7%	83.9%	**829**	**679**	**1000**	**1211**	**644**	**508**	**706**	**906**
	ℓ_∞	NES [19]	72.1%	66.8%	68.4%	69.9%	4673	5174	4763	4770	4376	4832	4357	4508
		Meta Attack [12]	80.4%	81.2%	57.6%	40.1%	4136	3951	4893	4967	3714	3585	4609	4737
		Bandits [20]	81.2%	92.5%	72.4%	56%	3222	2798	3353	3465	2633	2132	2766	2774
		Simulator Attack (m=3)	89.4%	94.2%	79%	64.3%	2732	2281	3078	3238	1854	1589	2185	2548
		Simulator Attack (m=5)	83.7%	91.4%	74.2%	60%	**2410**	**2134**	**2619**	**2823**	**1754**	**1572**	**2080**	**2270**

Table 4: Experimental results of targeted attack in CIFAR-10 and CIFAR-100 datasets, where m is simulator-predict interval.

limiting the different maximum queries of each adversarial example. The superiority of the proposed approach in terms of attack success rate is shown in Fig. 4. Meanwhile, Fig. 5 demonstrates the average number of queries that reaches different desired success rates. Fig. 5 reveals that the proposed approach is more query-efficient than other attacks and that the gap is amplified for higher success rates.

Results of Attacks on the Defensive Models. Table 5 shows the experimental results obtained after attacking the defensive models. ComDefend (CD) [21] and Feature Distillation (FD) [26] are equipped with a denoiser to transform the input images to their clean versions before feeding



(a) PyramidNet-272 in CIFAR-100 (b) R_{32} in TinyImageNet

Figure 4: Comparison of the attack success rate at different limited maximum queries in untargeted attack under ℓ_∞ norm, where R_{32} indicates ResNext-101 (32×4d).

to the target model. Prototype conformity loss (PCL) [31]

Dataset	Attack	Attack Success Rate				Avg. Query				Median Query			
		CD [21]	PCL [31]	FD [26]	Adv Train [29]	CD [21]	PCL [31]	FD [26]	Adv Train [29]	CD [21]	PCL [31]	FD [26]	Adv Train [29]
CIFAR-10	NES [19]	60.4%	65%	54.5%	16.8%	1130	728	1474	**858**	400	150	450	**200**
	RGF [32]	48.7%	82.6%	44.4%	22.4%	2035	1107	1717	973	1071	306	768	510
	P-RGF [8]	62.8%	80.4%	65.8%	22.4%	1977	1006	1979	1158	1038	230	703	602
	Meta Attack [12]	26.8%	77.7%	38.4%	18.4%	2468	1756	2662	1894	1302	1042	1824	1561
	Bandits [20]	44.7%	84%	55.2%	34.8%	786	776	832	1941	100	126	114	759
	Simulator Attack	54.9%	78.2%	60.8%	32.3%	**433**	**641**	**391**	1529	**46**	**116**	**50**	589
CIFAR-100	NES [19]	78.1%	87.9%	77.6%	23.1%	892	429	1071	**865**	300	150	250	**250**
	RGF [32]	50.2%	95.5%	62%	29.2%	1753	645	1208	1009	765	204	408	510
	P-RGF [8]	54.2%	96.1%	73.4%	28.8%	1842	679	1169	1034	815	182	262	540
	Meta Attack [12]	20.8%	93%	59%	27%	2084	1122	2165	1863	781	651	1043	1562
	Bandits [20]	54.1%	97%	72.5%	44.9%	786	321	584	1609	56	34	32	484
	Simulator Attack	72.9%	93.1%	80.7%	35.6%	**330**	**233**	**250**	1318	**30**	**22**	**24**	442
TinyImageNet	NES [19]	69.5%	73.1%	33.3%	23.7%	1775	863	2908	**945**	850	250	1600	**200**
	RGF [32]	31.3%	91.8%	9.1%	34.7%	2446	1022	1619	1325	1377	408	765	612
	P-RGF [8]	37.3%	91.8%	25.9%	34.4%	1946	1065	2231	1287	891	436	985	602
	Meta Attack [12]	4.5%	75.8%	3.7%	20.1%	1877	2585	4187	3413	912	1792	2602	2945
	Bandits [20]	39.6%	95.8%	12.5%	49%	893	909	1272	1855	85	206	193	810
	Simulator Attack	43%	84.2%	21.3%	42.5%	**377**	**586**	**746**	1631	**32**	**148**	**157**	632

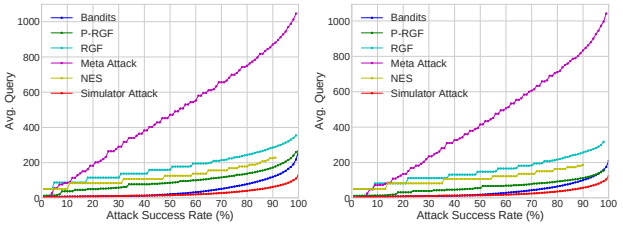
Table 5: Experimental results after performing the ℓ_∞ norm attacks on defensive models, where CD represents ComDefend [21], FD is Feature Distillation [26], and PCL is prototype conformity loss [31].

Attack	Attack Success Rate			Avg. Query			Median Query		
	D_{121}	R_{32}	R_{64}	D_{121}	R_{32}	R_{64}	D_{121}	R_{32}	R_{64}
NES [19]	74.3%	45.3%	45.5%	1306	2104	2078	510	765	816
RGF [32]	96.4%	85.3%	87.4%	1146	2088	2087	667	1280	1305
P-RGF [8]	94.5%	83.9%	85.9%	883	1583	1581	448	**657**	**690**
Meta Attack [12]	71.1%	33.8%	36%	3789	4101	4012	3202	3712	3649
Bandits [20]	99.2%	94.1%	95.3%	964	1737	1662	520	954	1014
Simulator Attack	99.4%	96.8%	97.9%	**811**	**1380**	**1445**	**431**	850	878

Table 6: Experimental results of untargeted attack under ℓ_∞ norm in TinyImageNet dataset. D_{121}: DenseNet-121, R_{32}: ResNeXt-101 (32×4d), R_{64}: ResNeXt-101 (64×4d).

Attack	Attack Success Rate			Avg. Query			Median Query		
	D_{121}	R_{32}	R_{64}	D_{121}	R_{32}	R_{64}	D_{121}	R_{32}	R_{64}
NES [19]	88.5%	88%	88.2%	4625	4959	4758	4337	4703	4440
Meta Attack [12]	24.2%	21%	18.2%	5420	5440	5661	5506	5249	5250
Bandits [20]	85.1%	72.2%	72.4%	2724	3550	3542	1860	2700	2854
Simulator Attack	89.8%	84.9%	83.9%	**1959**	**2558**	**2488**	**1399**	**1966**	**1982**

Table 7: Experimental results of targeted attack under ℓ_2 norm in TinyImageNet dataset. D_{121}: DenseNet-121, R_{32}: ResNeXt-101 (32×4d), R_{64}: ResNeXt-101 (64×4d).



(a) PyramidNet-272 in CIFAR-100 (b) GDAS in CIFAR-100

Figure 5: Comparisons of the average query at different success rates under the untargeted ℓ_∞ norm attack. More results are presented in the supplementary material.

introduces a new loss function to maximally separate the intermediate features of each class. Here, the PCL defensive model is obtained without using adversarial training in our experiments. Adv Train [29] is a powerful defense method based on adversarial training. Following the results shown in Table 5, we derive the following conclusions:

(1) Among all methods, the Simulator Attack exhibits the best performance in breaking CD, particularly outperforming the baseline method Bandits significantly.

(2) The Meta Attack demonstrates poor performance in CD and FD based on its unsatisfactory success rate. In comparison, the Simulator Attack can break this type of defensive model with a high success rate.

(3) In experiments in which the Adv Train is attacked, the Simulator Attack consumes fewer queries to achieve a comparable success rate with Bandits.

5. Conclusion

In this study, we present a novel black-box attack named Simulator Attack. It focuses on training a generalized substitute model ("Simulator") to accurately mimic any unknown target model with the aim of reducing the query complexity of the attack. To this end, the query sequences generated while attacking many different networks are used as the training data. The proposed approach uses an MSE-based knowledge-distillation loss in the inner and outer updates of meta-learning to learn the Simulator. After training, a high number of queries can be transferred to the Simulator, thereby reducing the query complexity of the attack by several orders of magnitude compared with the baseline.

Acknowledgments

This research is supported by the National Key R&D Program of China (2019YFB1405703) and TC190A4DA/3, the National Natural Science Foundation of China (Grant Nos. 61972221, 61572274).

References

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 484–501, Cham, 2020. Springer International Publishing. 2

[2] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *European Conference on Computer Vision*, pages 158–174. Springer, 2018. 2

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013. 1

[4] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 39–57, May 2017. 1

[5] 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. 2

[6] 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*, pages 15–26. ACM, 2017. 1, 2, 6

[7] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, JinFeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 2

[8] Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Improving black-box adversarial attacks with a transfer-based prior. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 2, 5, 7, 8, 11

[9] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 5

[10] 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. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 321–338, Santa Clara, CA, Aug. 2019. USENIX Association. 2, 3

[11] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019. 5

[12] Jiawei Du, Hu Zhang, Joey Tianyi Zhou, Yi Yang, and Jiashi Feng. Query-efficient meta attack to deep neural networks. In *International Conference on Learning Representations*, 2020. 2, 3, 5, 6, 7, 8, 11

[13] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 1

[14] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2484–2493. PMLR, 09–15 Jun 2019. 2

[15] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5927–5935, 2017. 5

[16] 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*, pages 770–778, 2016. 5, 13

[17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 5

[18] 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 International Conference on Computer Vision*, pages 4733–4742, 2019. 2, 3

[19] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2137–2146. PMLR, 10–15 Jul 2018. 1, 2, 5, 7, 8, 11, 12

[20] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *International Conference on Learning Representations*, 2019. 1, 2, 5, 7, 8, 11

[21] Xiaojun Jia, Xingxing Wei, Xiaochun Cao, and Hassan Foroosh. Comdefend: An efficient image compression model to defend adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6084–6092, 2019. 7, 8, 12

[22] P. Diederik Kingma and Lei Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 12

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 5

[24] T. Lee, B. Edwards, I. Molloy, and D. Su. Defending against neural network model stealing attacks using deceptive perturbations. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 43–49, May 2019. 2

[25] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *Proceedings of 5th International Conference on Learning Representations*, 2017. 2, 3

[26] Zihao Liu, Qi Liu, Tao Liu, Nuo Xu, Xue Lin, Yanzhi Wang, and Wujie Wen. Feature distillation: Dnn-oriented jpeg compression against adversarial examples. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 860–868. IEEE, 2019. 7, 8, 13

[27] Chen Ma, Shuyu Cheng, Li Chen, and Junhai Yong. Switching transferable gradient directions for query-efficient black-box adversarial attacks. *arXiv preprint arXiv:2009.07191*, 2020. 2

[28] Chen Ma, Chenxu Zhao, Hailin Shi, Li Chen, Junhai Yong, and Dan Zeng. Metaadvdet: Towards robust detection of evolving adversarial attacks. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, page 692–701, New York, NY, USA, 2019. Association for Computing Machinery. 3

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. 1, 8, 13

[30] Smitha Milli, Ludwig Schmidt, Anca D Dragan, and Moritz Hardt. Model reconstruction from model explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 1–9, 2019. 3

[31] Aamir Mustafa, Salman Khan, Munawar Hayat, Roland Goecke, Jianbing Shen, and Ling Shao. Adversarial defense by restricting the hidden space of deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3385–3394, 2019. 7, 8, 13

[32] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017. 2, 5, 7, 8, 11

[33] Seong Joon Oh, Bernt Schiele, and Mario Fritz. Towards reverse-engineering black-box neural networks. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 121–144. Springer, 2019. 2

[34] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019. 2, 3

[35] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks. In *International Conference on Learning Representations*, 2020. 2

[36] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017. 2, 3

[37] Li Pengcheng, Jinfeng Yi, and Lijun Zhang. Query-efficient black-box attack by active learning. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1200–1205. IEEE, 2018. 2

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 2, 5

[39] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. 1

[40] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016. 2, 3

[41] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 742–749, 2019. 1, 2

[42] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 36–52. IEEE, 2018. 3

[43] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 5

[44] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise. Shakedrop regularization for deep residual learning. *IEEE Access*, 7:186126–186136, 2019. 5

[45] Ziang Yan, Yiwen Guo, and Changshui Zhang. Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks. In *Advances in Neural Information Processing Systems*, pages 3820–3829, 2019. 5

[46] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 5

Supplementary Material

A. Experiment Settings

A.1. Compared Methods

Bandits. Table 8 shows the default hyperparameters of Bandits [20], which is a subset of hyperparameters of the Simulator Attack. Specifically, the OCO learning rate is used to update the prior, which is an alias of the gradient $\mathbf{g}$ for updating the input image.

RGF and P-RGF. Table 9 shows the default hyperparameters of random gradient-free (RGF) [32] and prior-guided RGF (P-RGF) [8]. P-RGF improves RGF by using surrogate models (see the last row block of Table 9). The experiments of RGF and P-RGF are conducted by using the implementation of PyTorch version that is translated from the official TensorFlow version.

NES. The default hyperparameters for natural evolution strategies (NES) [19] are listed in Table 12. In the targeted attack, NES uses an initial image of the target class and reduce its distortion iteratively while keeping the image residing in the adversarial region of the target class. Finally, the samples whose ℓ_p norm distance to the original benign image is less than a preset ϵ are considered as successful samples. Thus, the hyperparameters of NES are carefully tuned in the untargeted and targeted attack separately, so as to achieve the highest attack success rate. The experiments of NES are conducted by using the implementation of PyTorch version, which is translated from the official TensorFlow implementation.

Meta Attack. The default hyperparameters of the Meta Attack [12] are listed in Table 13. Specifically, the meta interval m is set to 3 in two cases, namely, the targeted attack and all the experiments of TinyImageNet dataset. In other cases, the meta interval m is set to 5. The gradients of training data are generated by using the classification networks listed in Table 14. The Meta Attack uses the official PyTorch implementation to conduct ℓ_2 norm attack experiments, and we add the additional code in the official implementation to enable it to support the ℓ_∞ norm attack.

Simulator Attack. The default hyperparameters of the proposed method are listed in Table 10. Those hyperparameters that are also used in Bandits are set to the same values as Bandits.

Norm	Hyperparameter	Value
ℓ_2	δ, finite difference probe	0.01
	η, image learning rate	0.1
	η_g, OCO learning rate	0.1
	τ, Bandits exploration	0.3
	ϵ, radius of ℓ_2 norm ball	4.6
	maximum query times	10,000
ℓ_∞	δ, finite difference probe	0.1
	η, image learning rate	1/255
	η_g, OCO learning rate	1.0
	τ, Bandits exploration	0.3
	ϵ, radius of ℓ_∞ norm ball	8/255
	maximum query times	10,000

Table 8: The hyperparameters of Bandits [20].

Norm	Hyperparameter	Value
ℓ_2	h, image learning rate	2.0
	σ, sampling variance	1e-4
	ϵ, radius of ℓ_2 norm ball	4.6
ℓ_∞	h, image learning rate	0.005
	σ, sampling variance	1e-4
	ϵ, radius of ℓ_∞ norm ball	8/255
ℓ_2, ℓ_∞	surrogate model used in CIFAR-10/100	ResNet-110
ℓ_2, ℓ_∞	surrogate model used in TinyImageNet	ResNet-101

Table 9: The hyperparameters of RGF [32] and P-RGF [8], and the networks shown in the last row block are used as the surrogate models of P-RGF.

Hyperparameter	Default Value
backbone	ResNet-34
λ_1, the learning rate of the inner update	0.01
λ_2, the learning rate of of the outer update	0.001
ϵ, the maximum distortion of ℓ_2 norm attack	4.6
ϵ, the maximum distortion of ℓ_∞ norm attack	8/255
δ, finite difference probe of ℓ_2 norm attack	0.01
δ, finite difference probe of ℓ_∞ norm attack	0.1
η, the image learning rate of ℓ_2 norm attack	0.1
η, the image learning rate of ℓ_∞ norm attack	1/255
η_g, OCO learning rate of ℓ_2 norm attack	0.1
η_g, OCO learning rate of ℓ_∞ norm attack	1.0
τ, Bandits exploration	0.3
inner-update iterations	12
meta-predict interval m	5
warm-up iterations t	10
deque $\mathbb{D}$'s maximum length	10

Table 10: The hyperparameters of the Simulator Attack.

Dataset	Network	Model Details		
		Params(M)	MACs(G)	Layers
CIFAR-10	PyramidNet-272	26.21	4.55	272
	GDAS	3.02	0.41	20
	WRN-28	36.48	5.25	28
	WRN-40	55.84	8.08	40
CIFAR-100	PyramidNet-272	26.29	4.55	272
	GDAS	3.14	0.41	20
	WRN-28	36.54	5.25	28
	WRN-40	55.90	8.08	40
TinyImageNet	DenseNet-121	7.16	0.23	121
	ResNeXt-101 (32×4d)	42.54	0.65	101
	ResNeXt-101 (64×4d)	81.82	1.27	101

Table 11: The details of black-box target models which are used for evaluating attack methods, where MAC is the multiply–accumulate operation count.

Dataset	Attack	Norm	Hyperparameter	Value
CIFAR-10	Untargeted	ℓ_2	ϵ, radius of ℓ_2 norm ball	4.6
			h, image learning rate	2.0
		ℓ_∞	ϵ, radius of ℓ_∞ norm ball	8/255
			h, image learning rate	1e-2
	Targeted	ℓ_2	ϵ_0, initial distance from the source image	20.0
			ϵ, final radius of ℓ_2 norm ball	4.6
			δ_{ϵ_0}, initial rate of decaying ϵ	1.0
			$\delta_{\epsilon_{\text{min}}}$, the minimum rate of decaying ϵ	0.1
			h_{max}, the maximum image learning rate	2.0
			h_{min}, the minimum image learning rate	5e-5
		ℓ_∞	ϵ_0, initial distance from the source image	1.0
			ϵ, final radius of ℓ_∞ norm ball	8/255
			δ_{ϵ_0}, initial rate of decaying ϵ	0.1
			$\delta_{\epsilon_{\text{min}}}$, the minimum rate of decaying ϵ	0.01
			h_{max}, the maximum image learning rate	0.1
			h_{min}, the minimum image learning rate	0.01
CIFAR-100	Untargeted	ℓ_2	ϵ, radius of ℓ_2 norm ball	4.6
			h, image learning rate	2.0
		ℓ_∞	ϵ, radius of ℓ_∞ norm ball	8/255
			h, image learning rate	1e-2
	Targeted	ℓ_2	ϵ_0, initial distance from the source image	20.0
			ϵ, final radius of ℓ_2 norm ball	4.6
			δ_{ϵ_0}, initial rate of decaying ϵ	1.0
			$\delta_{\epsilon_{\text{min}}}$, the minimum rate of decaying ϵ	0.3
			h_{max}, the maximum image learning rate	1.0
			h_{min}, the minimum image learning rate	5e-5
		ℓ_∞	ϵ_0, initial distance from the source image	1.0
			ϵ, final radius of ℓ_∞ norm ball	8/255
			δ_{ϵ_0}, initial rate of decaying ϵ	0.1
			$\delta_{\epsilon_{\text{min}}}$, the minimum rate of decaying ϵ	0.01
			h_{max}, the maximum image learning rate	0.1
			h_{min}, the minimum image learning rate	0.01
TinyImageNet	Untargeted	ℓ_2	ϵ, radius of ℓ_2 norm ball	4.6
			h, image learning rate	2.0
		ℓ_∞	ϵ, radius of ℓ_∞ norm ball	8/255
			h, image learning rate	1e-2
	Targeted	ℓ_2	ϵ_0, initial distance from the source image	40.0
			ϵ, final radius of ℓ_2 norm ball	4.6
			δ_{ϵ_0}, initial rate of decaying ϵ	1.0
			$\delta_{\epsilon_{\text{min}}}$, the minimum rate of decaying ϵ	0.1
			h_{max}, the maximum image learning rate	2.0
			h_{min}, the minimum image learning rate	0.5
		ℓ_∞	ϵ_0, initial distance from the source image	1.0
			ϵ, final radius of ℓ_∞ norm ball	8/255
			δ_{ϵ_0}, initial rate of decaying ϵ	0.1
			$\delta_{\epsilon_{\text{min}}}$, the minimum rate of decaying ϵ	1e-3
			h_{max}, the maximum image learning rate	0.1
			h_{min}, the minimum image learning rate	0.01

Table 12: The hyperparameters of NES [19], where the sampling variance σ for gradient estimation is set to 1e-3, and the number of samples per draw is set to 50.

Dataset	Attack	Norm	Hyperparameter	Value
CIFAR-10/100	Untargeted	ℓ_2	h, image learning rate	1e-2
			top-q coordinates for estimating gradient	125
			m, meta interval	5
			use_tanh, change-of-variables method	true
			ϵ, radius of ℓ_2 norm ball	4.6
		ℓ_∞	h, image learning rate	1e-2
			top-q coordinates for estimating gradient	125
			m, meta interval	5
			use_tanh, change-of-variables method	false
			ϵ, radius of ℓ_∞ norm ball	8/255
	Targeted	ℓ_2	h, image learning rate	1e-2
			top-q coordinates for estimating gradient	125
			m, meta interval	3
			use_tanh, change-of-variables method	true
			ϵ, radius of ℓ_2 norm ball	4.6
		ℓ_∞	h, image learning rate	1e-2
			top-q coordinates for estimating gradient	125
			m, meta interval	3
			use_tanh, change-of-variables method	false
			ϵ, radius of ℓ_∞ norm ball	8/255
TinyImageNet	Untargeted	ℓ_2	h, image learning rate	1e-2
			top-q coordinates for estimating gradient	125
			m, meta interval	3
			use_tanh, change-of-variables method	true
			ϵ, radius of ℓ_2 norm ball	4.6
		ℓ_∞	h, image learning rate	1e-2
			top-q coordinates for estimating gradient	125
			m, meta interval	3
			use_tanh, change-of-variables method	false
			ϵ, radius of ℓ_∞ norm ball	8/255
	Targeted	ℓ_2	h, image learning rate	1e-2
			top-q coordinates for estimating gradient	125
			m, meta interval	3
			use_tanh, change-of-variables method	true
			ϵ, radius of ℓ_2 norm ball	4.6
		ℓ_∞	h, image learning rate	1e-2
			top-q coordinates for estimating gradient	125
			m, meta interval	3
			use_tanh, change-of-variables method	false
			ϵ, radius of ℓ_∞ norm ball	8/255

Table 13: The hyperparameters of the Meta Attack, where the binary step is set to 1, and the solver of gradient estimation adopts the Adam optimizer [22].

A.2. Pre-trained Networks and Target Models

Pre-trained Networks. In the training of the Simulator and the auto-encoder of the Meta Attack, we collect various types of classification networks to generate the training data. In our experiments, we select 14 networks for generating training data of CIFAR-10 and CIFAR-100 datasets, and select 16 networks for generating training data of TinyImageNet datasets. The names of these networks and their training configurations are shown in Table 14.

Target Models. To evaluate the performance of attacking unknown target models, we specify the target models to equip with completely different architectures from the pre-trained networks. The target models and their complexity are listed in Table 11.

B. Experimental Results

B.1. Detailed Experimental Figures

Attack Success Rates at Different Maximum Queries. We conduct experiments by limiting different maximum queries of attacks and compare their attack success rates. Figs. 6, 7, 8, and 9 show the results which are obtained by attacking normal models and defensive models with different maximum number of queries. Four defensive models are adopted, namely, ComDefend [21], Feature Distil-

Dataset	Network	Training Configuration					Hyperparameters	
		epochs	lr	lr decay epochs	lr decay rate	weight decay	layer depth	other hyperparameters
CIFAR-10/100	AlexNet	164	0.1	81, 122	0.1	5e-4	9	-
	DenseNet-100	300	0.1	150, 225	0.1	1e-4	100	growth rate:12, compression rate:2
	DenseNet-190	300	0.1	150, 225	0.1	1e-4	190	growth rate:40, compression rate:2
	PreResNet-110	164	0.1	81, 122	0.1	1e-4	110	block name: BasicBlock
	ResNeXt-29 ($8 \times 64d$)	300	0.1	150, 225	0.1	5e-4	29	widen factor:4, cardinality:8
	ResNeXt-29 ($16 \times 64d$)	300	0.1	150, 225	0.1	5e-4	29	widen factor:4, cardinality:16
	VGG-19 (BN)	164	0.1	81, 122	0.1	5e-4	19	-
	ResNet-20	164	0.1	81, 122	0.1	1e-4	20	block name: BasicBlock
	ResNet-32	164	0.1	81, 122	0.1	1e-4	32	block name: BasicBlock
	ResNet-44	164	0.1	81, 122	0.1	1e-4	44	block name: BasicBlock
	ResNet-50	164	0.1	81, 122	0.1	1e-4	50	block name: BasicBlock
	ResNet-56	164	0.1	81, 122	0.1	1e-4	56	block name: BasicBlock
	ResNet-110	164	0.1	81, 122	0.1	1e-4	110	block name: BasicBlock
	ResNet-1202	164	0.1	81, 122	0.1	1e-4	1202	block name: BasicBlock
TinyImageNet	VGG-11	300	1e-3	100, 200	0.1	1e-4	11	-
	VGG-11 (BN)	300	1e-3	100, 200	0.1	1e-4	11	-
	VGG-13	300	1e-3	100, 200	0.1	1e-4	13	-
	VGG-13 (BN)	300	1e-3	100, 200	0.1	1e-4	13	-
	VGG-16	300	1e-3	100, 200	0.1	1e-4	16	-
	VGG-16 (BN)	300	1e-3	100, 200	0.1	1e-4	16	-
	VGG-19	300	1e-3	100, 200	0.1	1e-4	19	-
	VGG-19 (BN)	300	1e-3	100, 200	0.1	1e-4	19	-
	ResNet-18	300	1e-3	100, 200	0.1	1e-4	18	block name: BasicBlock
	ResNet-34	300	1e-3	100, 200	0.1	1e-4	34	block name: BasicBlock
	ResNet-50	300	1e-3	100, 200	0.1	1e-4	50	block name: Bottleneck
	ResNet-101	300	1e-3	100, 200	0.1	1e-4	101	block name: Bottleneck
	ResNet-152	300	1e-3	100, 200	0.1	1e-4	152	block name: Bottleneck
	DenseNet-161	300	1e-3	100, 200	0.1	1e-4	161	growth rate: 32
	DenseNet-169	300	1e-3	100, 200	0.1	1e-4	169	growth rate: 32
	DenseNet-201	300	1e-3	100, 200	0.1	1e-4	201	growth rate: 32

Table 14: The details of pre-trained classification networks, which are $\mathbb{N}_1, \cdots, \mathbb{N}_n$ used for the generation of training data in both the Simulator Attack and the Meta Attack. All the data of ResNet networks are excluded in the experiments of attacking defensive models.

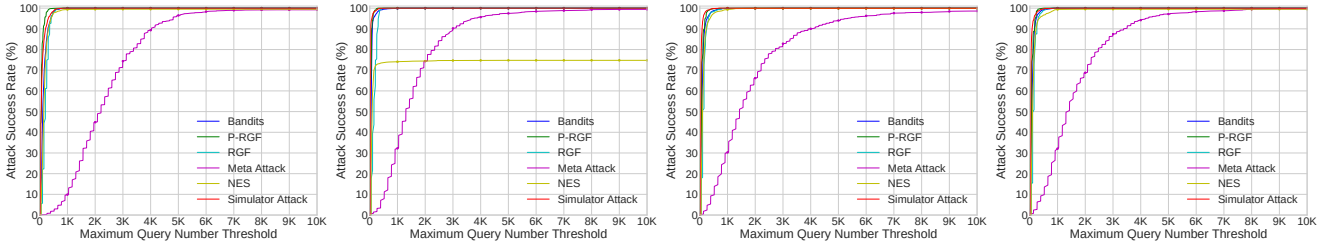
lation [26], prototype conformity loss (PCL) [31] and Adv Train [29]. The ResNet-50 [16] is selected as the backbone in these defensive models.

Average Queries at Different Success Rates. The second type of figure measures the average number of queries that reaches different desired success rates. It demonstrates the relation between the query number and attack success rate from a different angle. Specifically, given a desired success rate a and the query list Q of all successful attacked samples, the average query (Avg. Q_a) is defined as follows:

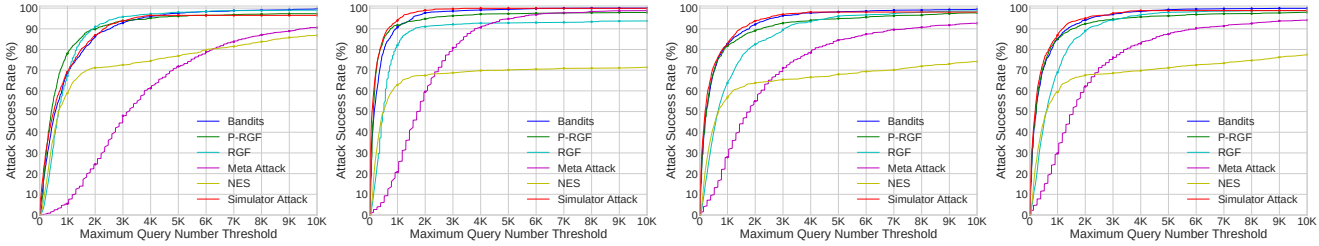
$$\text{Avg. } Q_a = \frac{\sum_{i=1}^{N} \hat{Q}_i}{N}, \qquad \text{where } \hat{Q} = Q[Q \leq P_a], \quad (3)$$

where P_a is the a-th percentile value of Q and N is the length of $\hat{Q}$. Figs. 10, 11, 12, and 13 show the results. All the experimental results demonstrate that the Simulator Attack requires the lowest queries and achieves the highest attack success rate, so the superior performance of the Simulator Attack is verified.

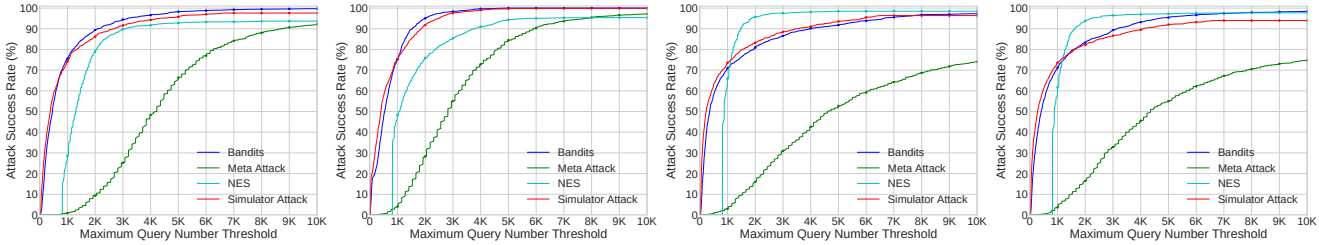
Histogram of Query Numbers. To observe the distribution of query numbers in detail, we collect the query number of each adversarial example to draw the histogram figures. Specifically, we divide the range of query number into 10 intervals, and then count the number of samples in each interval. These intervals are separated by the vertical lines of figures. Each bar indicates one attack, and its height indicates the number of samples with the queries belong to this query interval. Figs. 14, 15, 16, and 17 show the histograms of query numbers in the CIFAR-10, CIFAR-100, and TinyImageNet datasets, respectively. The results demonstrates that the highest red bars (the Simulator Attack) are located in the area with low number of queries, which confirms that most adversarial examples of the Simulator Attack have the minimum number of queries.



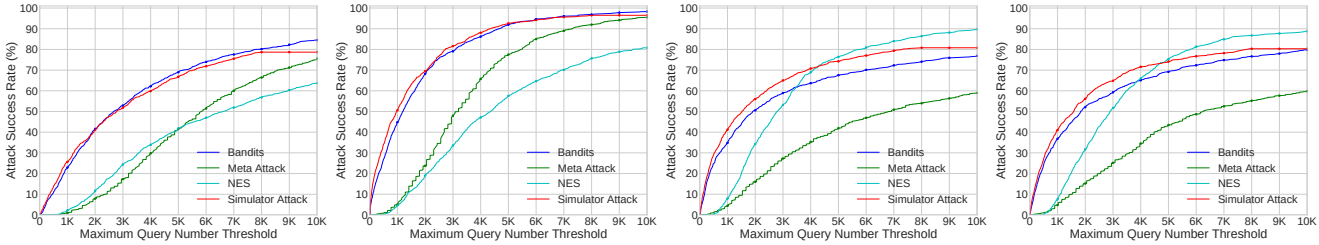
(a) untargeted ℓ_2 attack PyramidNet-272 (b) untargeted ℓ_2 attack GDAS (c) untargeted ℓ_2 attack WRN-28 (d) untargeted ℓ_2 attack WRN-40



(e) untargeted ℓ_∞ attack PyramidNet-272 (f) untargeted ℓ_∞ attack GDAS (g) untargeted ℓ_∞ attack WRN-28 (h) untargeted ℓ_∞ attack WRN-40

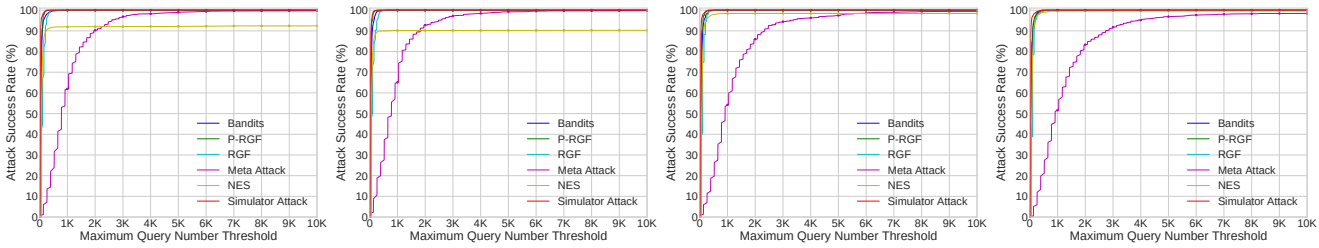


(i) targeted ℓ_2 attack PyramidNet-272 (j) targeted ℓ_2 attack GDAS (k) targeted ℓ_2 attack WRN-28 (l) targeted ℓ_2 attack WRN-40

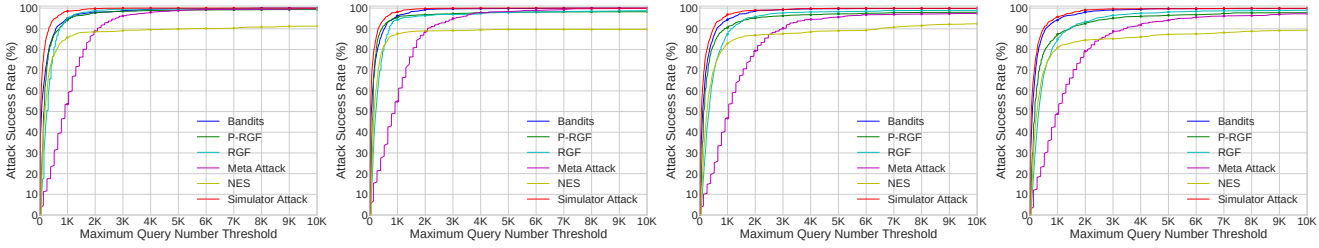


(m) targeted ℓ_∞ attack PyramidNet-272 (n) targeted ℓ_∞ attack GDAS (o) targeted ℓ_∞ attack WRN-28 (p) targeted ℓ_∞ attack WRN-40

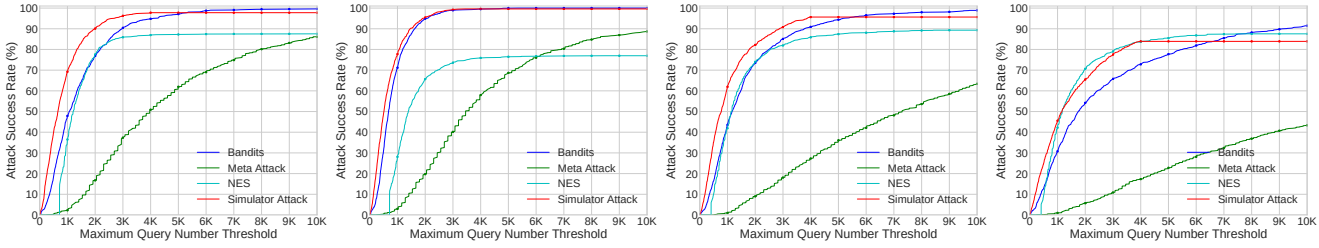
Figure 6: Comparisons of attack success rates at different limited maximum queries in CIFAR-10 dataset.



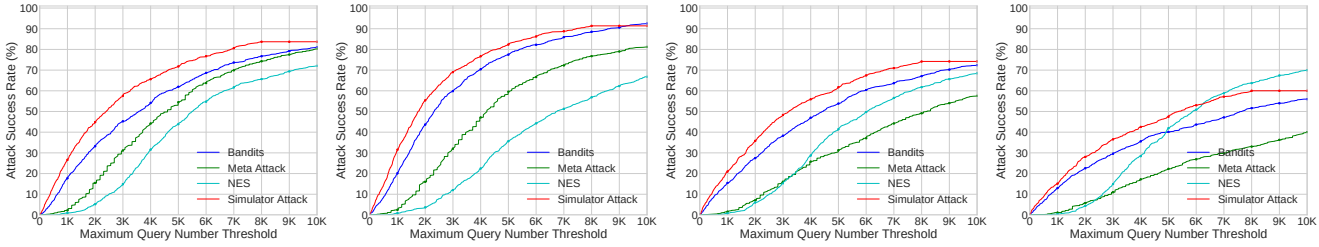
(a) untargeted ℓ_2 attack PyramidNet-272 (b) untargeted ℓ_2 attack GDAS (c) untargeted ℓ_2 attack WRN-28 (d) untargeted ℓ_2 attack WRN-40



(e) untargeted ℓ_∞ attack PyramidNet-272 (f) untargeted ℓ_∞ attack GDAS (g) untargeted ℓ_∞ attack WRN-28 (h) untargeted ℓ_∞ attack WRN-40

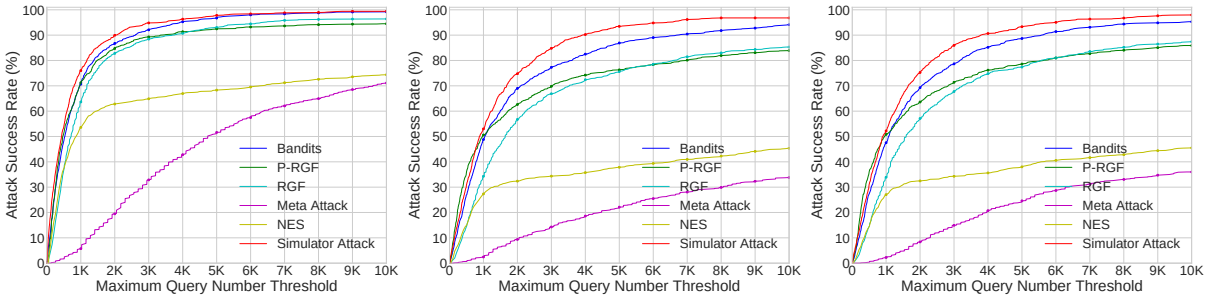


(i) targeted ℓ_2 attack PyramidNet-272 (j) targeted ℓ_2 attack GDAS (k) targeted ℓ_2 attack WRN-28 (l) targeted ℓ_2 attack WRN-40

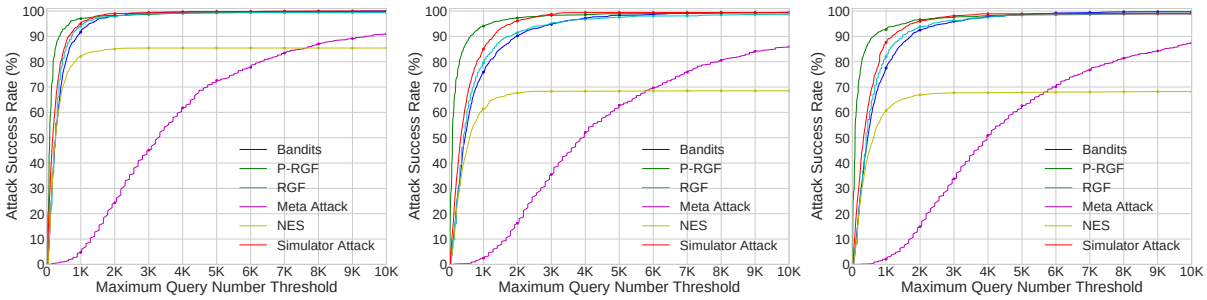


(m) targeted ℓ_∞ attack PyramidNet-272 (n) targeted ℓ_∞ attack GDAS (o) targeted ℓ_∞ attack WRN-28 (p) targeted ℓ_∞ attack WRN-40

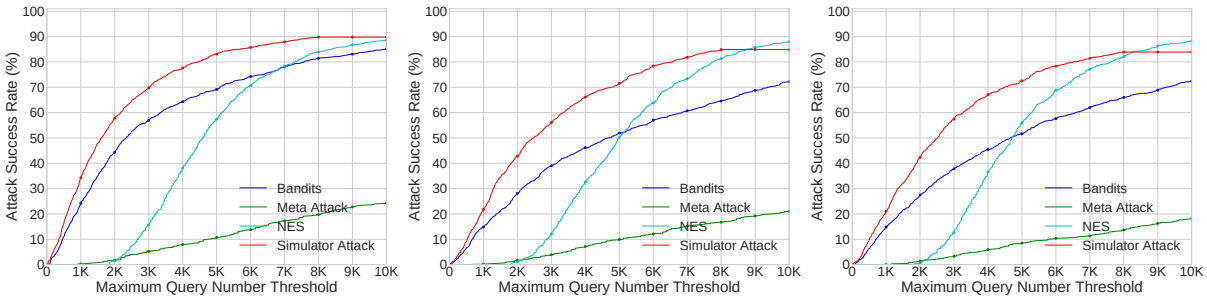
Figure 7: Comparisons of attack success rates at different limited maximum queries in CIFAR-100 dataset.



(a) untargeted ℓ_∞ attack DenseNet-121 (b) untargeted ℓ_∞ attack ResNeXt-101(32×4d) (c) untargeted ℓ_∞ attack ResNeXt-101(64×4d)

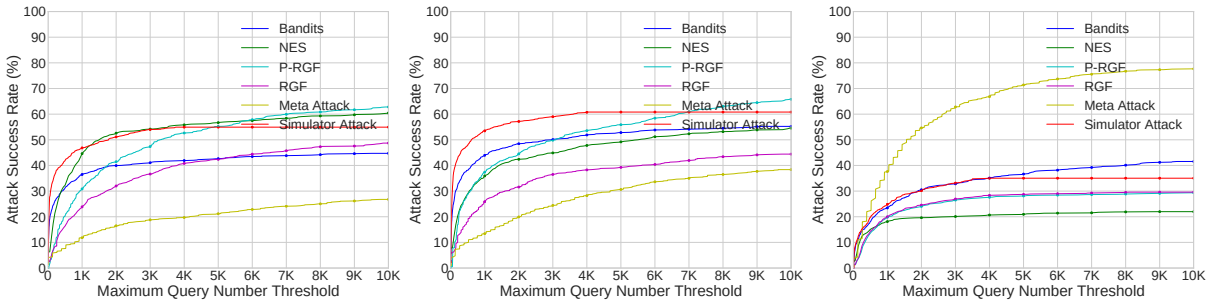


(d) untargeted ℓ_2 attack DenseNet-121 (e) untargeted ℓ_2 attack ResNeXt-101(32×4d) (f) untargeted ℓ_2 attack ResNeXt-101(64×4d)

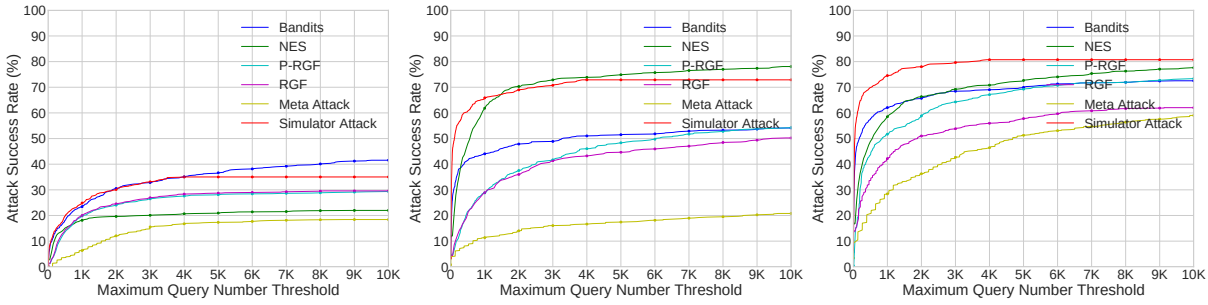


(g) targeted ℓ_2 attack DenseNet-121 (h) targeted ℓ_2 attack ResNeXt-101(32×4d) (i) targeted ℓ_2 attack ResNeXt-101(64×4d)

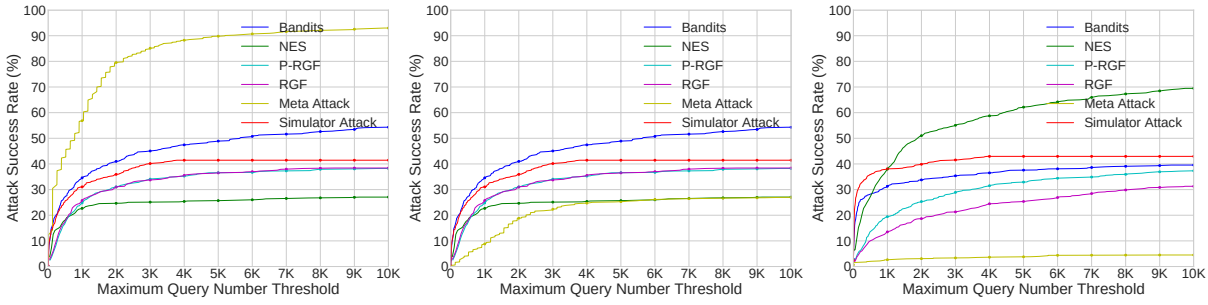
Figure 8: Comparisons of attack success rates at different limited maximum queries in TinyImageNet dataset.



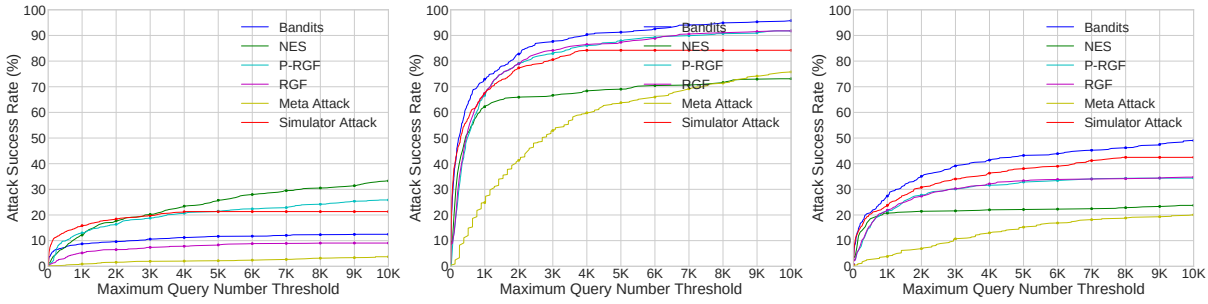
(a) attack ComDefend in CIFAR-10 (b) attack Feature Distillation in CIFAR-10 (c) attack PCL in CIFAR-10



(d) attack Adv Train in CIFAR-10 (e) attack ComDefend in CIFAR-100 (f) attack Feature Distillation in CIFAR-100

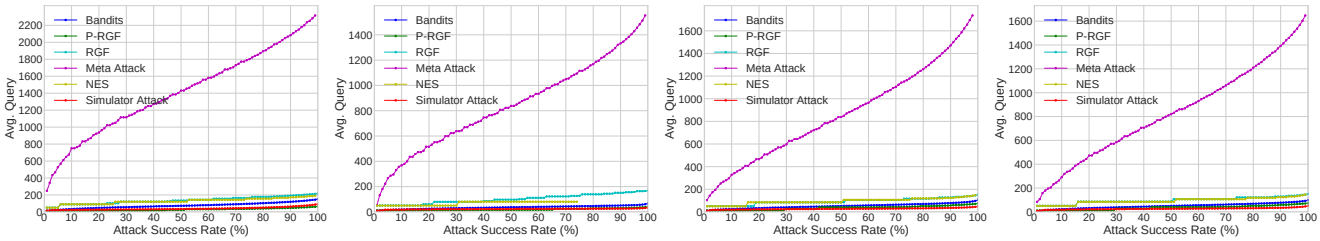


(g) attack PCL in CIFAR-100 (h) attack Adv Train in CIFAR-100 (i) attack ComDefend in TinyImageNet

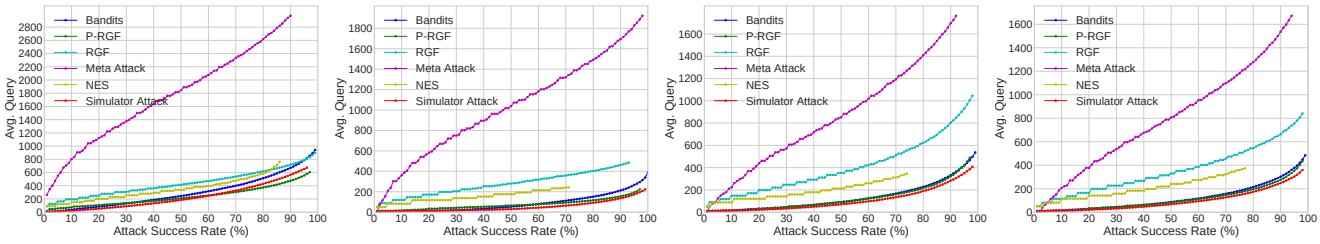


(j) attack Feature Distillation in TinyImageNet (k) attack PCL in TinyImageNet (l) attack Adv Train in TinyImageNet

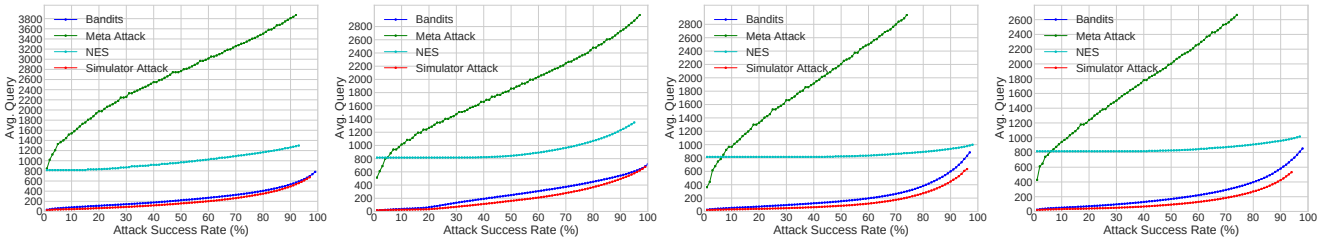
Figure 9: Comparisons of attack success rates at different maximum queries on defensive models with the ResNet-50 backbone. The experimental results are obtained by performing the untargeted attacks under ℓ_∞ norm.



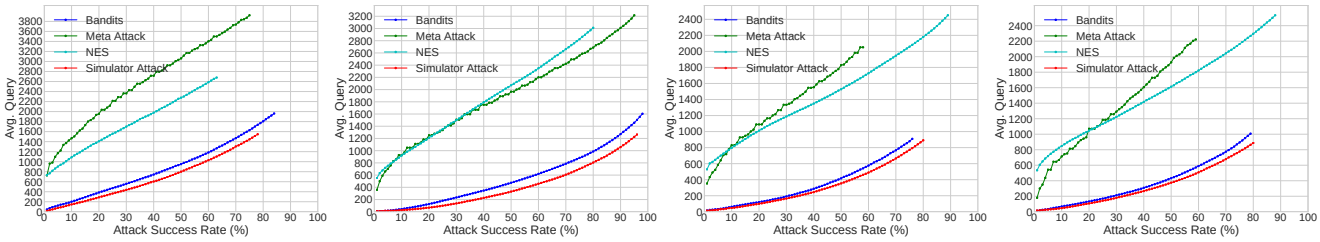
(a) untargeted ℓ_2 attack PyramidNet-272 (b) untargeted ℓ_2 attack GDAS (c) untargeted ℓ_2 attack WRN-28 (d) untargeted ℓ_2 attack WRN-40



(e) untargeted ℓ_∞ attack PyramidNet-272 (f) untargeted ℓ_∞ attack GDAS (g) untargeted ℓ_∞ attack WRN-28 (h) untargeted ℓ_∞ attack WRN-40

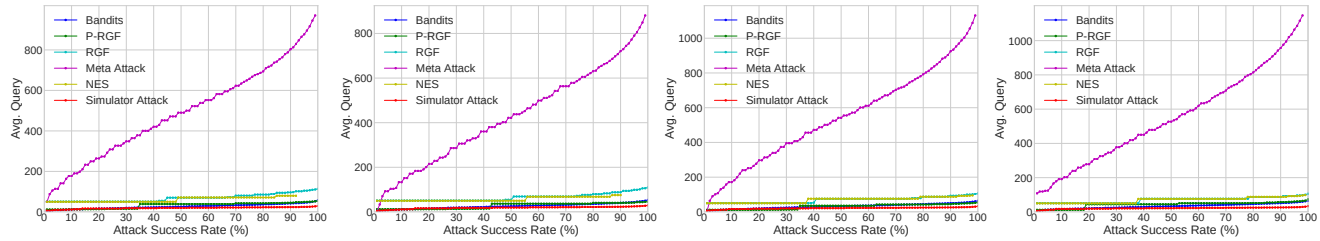


(i) targeted ℓ_2 attack PyramidNet-272 (j) targeted ℓ_2 attack GDAS (k) targeted ℓ_2 attack WRN-28 (l) targeted ℓ_2 attack WRN-40

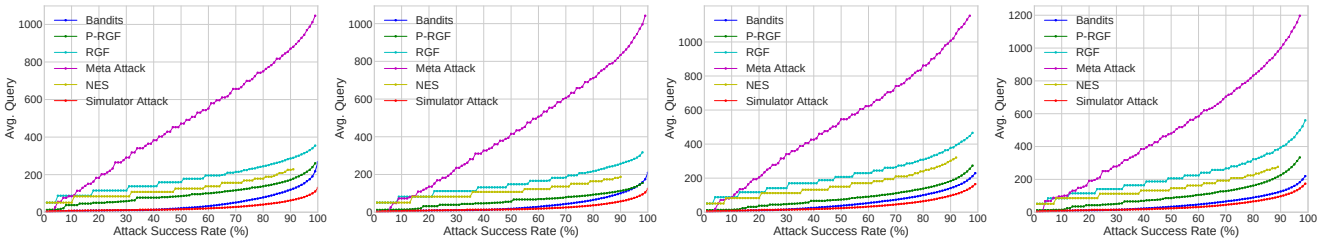


(m) targeted ℓ_∞ attack PyramidNet-272 (n) targeted ℓ_∞ attack GDAS (o) targeted ℓ_∞ attack WRN-28 (p) targeted ℓ_∞ attack WRN-40

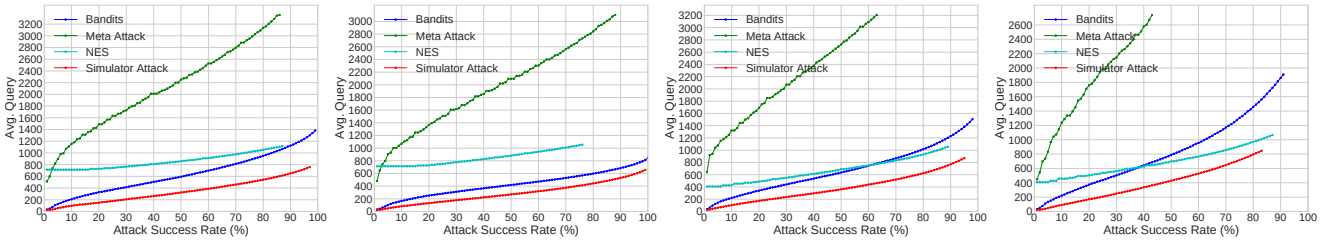
Figure 10: Comparisons of the average query per successful image at different desired success rates in CIFAR-10 dataset.



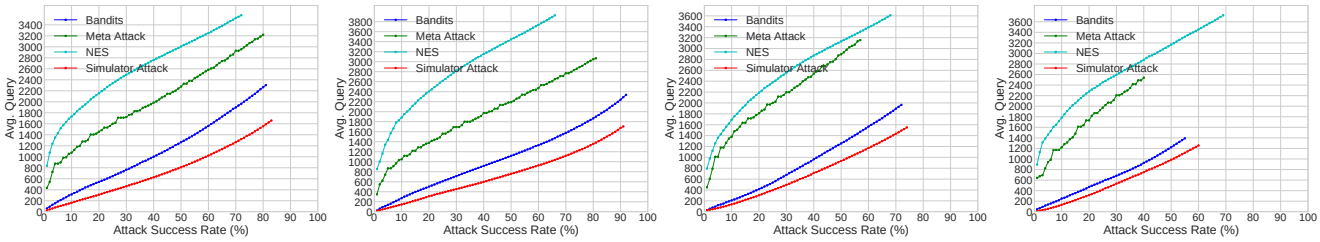
(a) untargeted ℓ_2 attack PyramidNet-272 (b) untargeted ℓ_2 attack GDAS (c) untargeted ℓ_2 attack WRN-28 (d) untargeted ℓ_2 attack WRN-40



(e) untargeted ℓ_∞ attack PyramidNet-272 (f) untargeted ℓ_∞ attack GDAS (g) untargeted ℓ_∞ attack WRN-28 (h) untargeted ℓ_∞ attack WRN-40

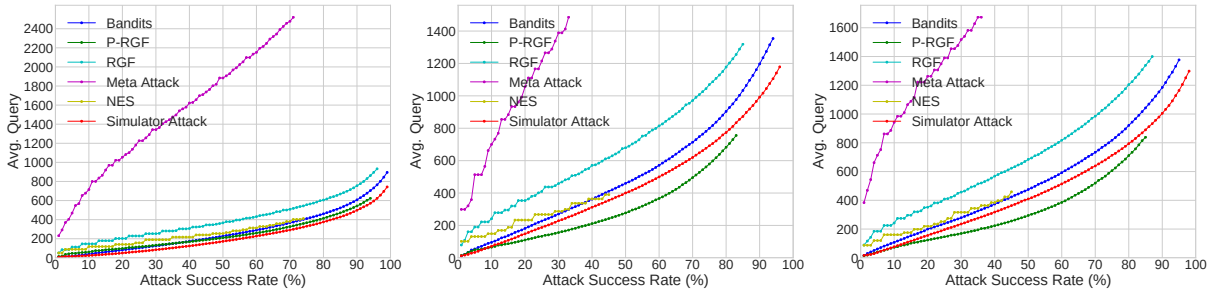


(i) targeted ℓ_2 attack PyramidNet-272 (j) targeted ℓ_2 attack GDAS (k) targeted ℓ_2 attack WRN-28 (l) targeted ℓ_2 attack WRN-40

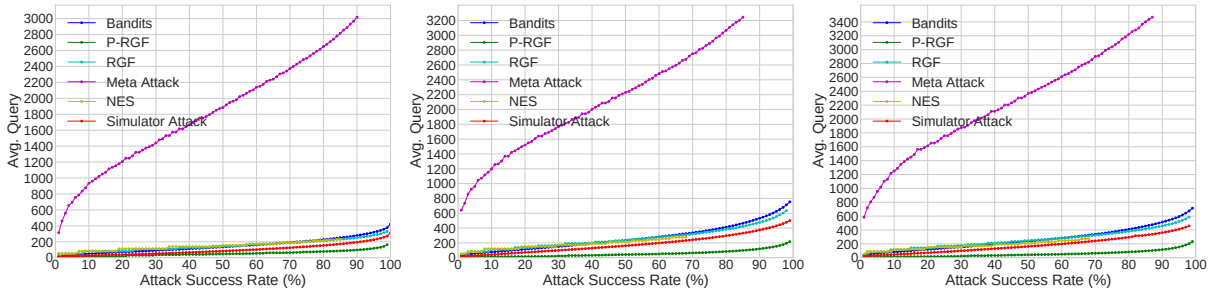


(m) targeted ℓ_∞ attack PyramidNet-272 (n) targeted ℓ_∞ attack GDAS (o) targeted ℓ_∞ attack WRN-28 (p) targeted ℓ_∞ attack WRN-40

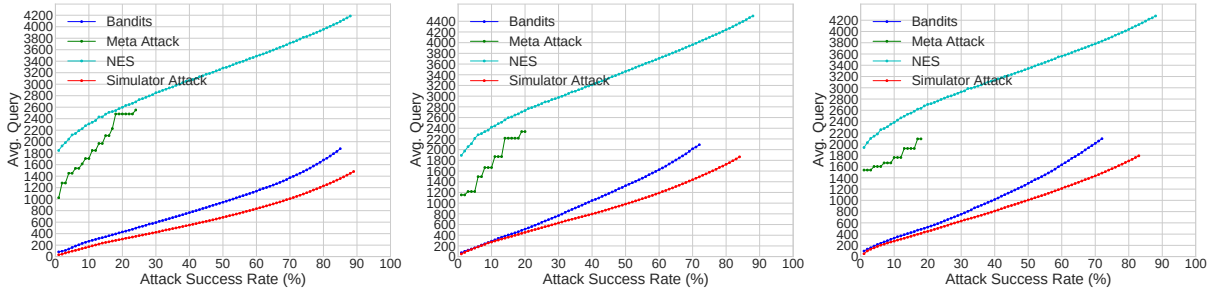
Figure 11: Comparisons of the average query per successful image at different desired success rates in CIFAR-100 dataset.



(a) untargeted ℓ_∞ attack DenseNet-121 (b) untargeted ℓ_∞ attack ResNeXt-101(32×4d) (c) untargeted ℓ_∞ attack ResNeXt-101(64×4d)

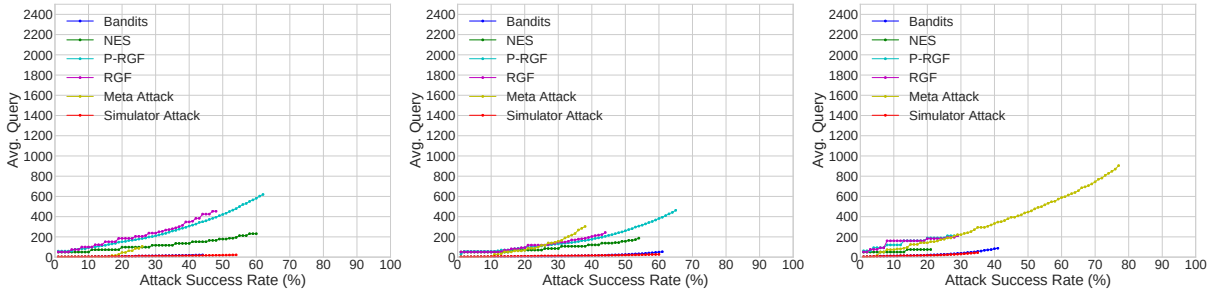


(d) untargeted ℓ_2 attack DenseNet-121 (e) untargeted ℓ_2 attack ResNeXt-101(32×4d) (f) untargeted ℓ_2 attack ResNeXt-101(64×4d)

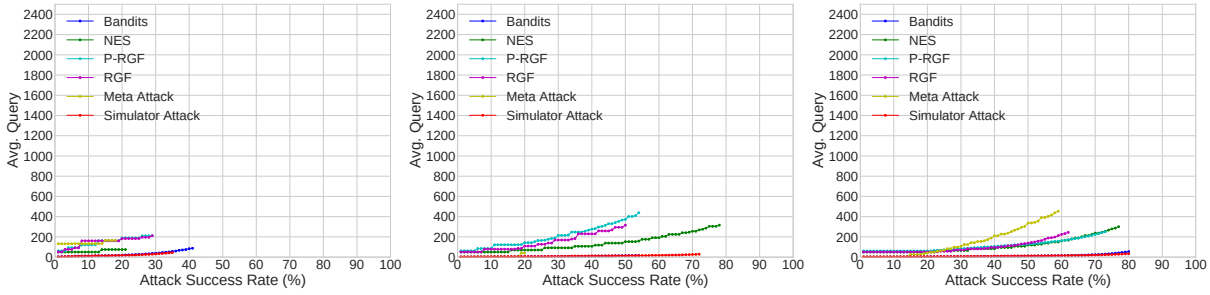


(g) targeted ℓ_2 attack DenseNet-121 (h) targeted ℓ_2 attack ResNeXt-101(32×4d) (i) targeted ℓ_2 attack ResNeXt-101(64×4d)

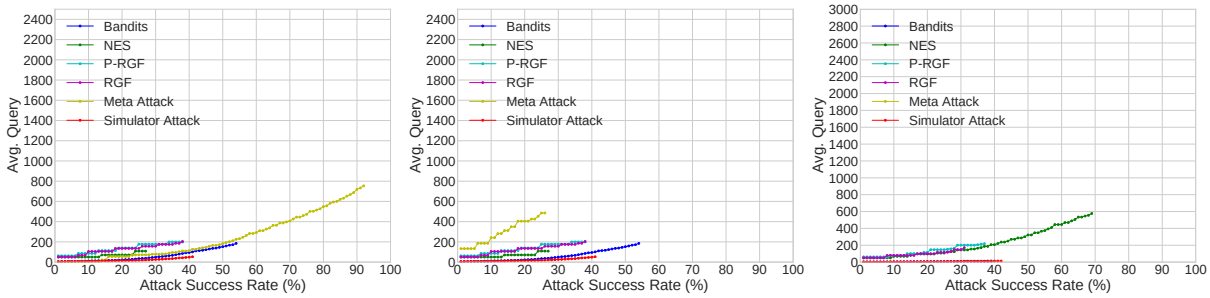
Figure 12: Comparisons of the average query per successful image at different desired success rates in TinyImageNet dataset.



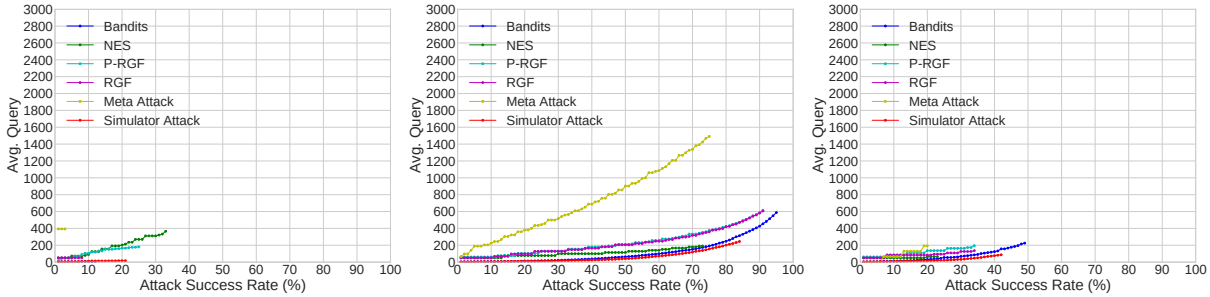
(a) attack ComDefend in CIFAR-10 (b) attack Feature Distillation in CIFAR-10 (c) attack PCL in CIFAR-10



(d) attack Adv Train in CIFAR-10 (e) attack ComDefend in CIFAR-100 (f) attack Feature Distillation in CIFAR-100

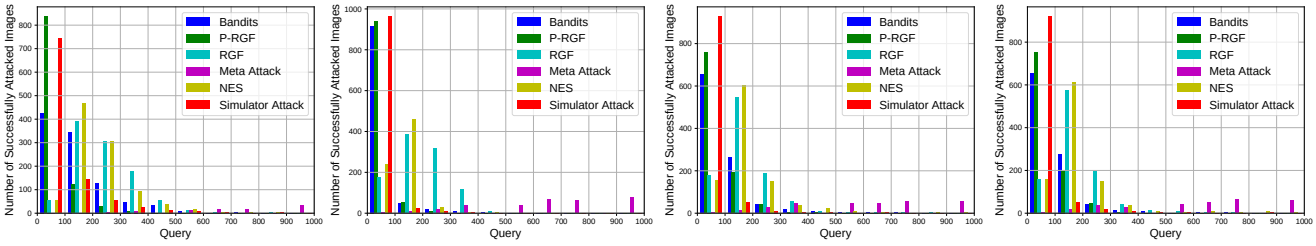


(g) attack PCL in CIFAR-100 (h) attack Adv Train in CIFAR-100 (i) attack ComDefend in TinyImageNet

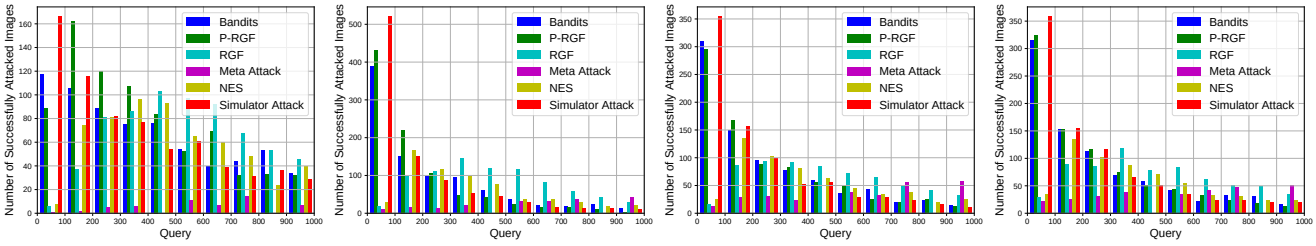


(j) attack Feature Distillation in TinyImageNet (k) attack PCL in TinyImageNet (l) attack Adv Train in TinyImageNet

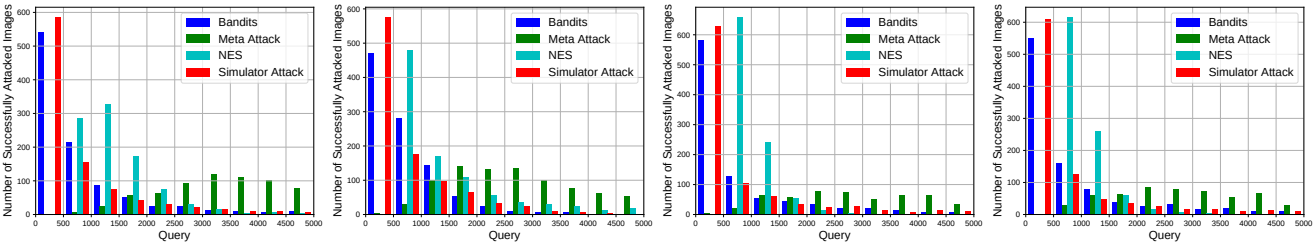
Figure 13: Comparisons of the average query per successful image at different desired success rates on defensive models with the backbone of ResNet-50. The experimental results are obtained by performing the untargeted attacks under ℓ_∞ norm.



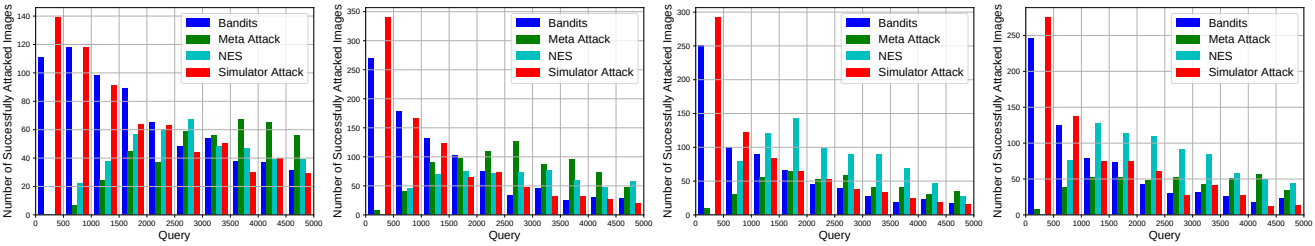
(a) untargeted ℓ_2 attack PyramidNet-272 (b) untargeted ℓ_2 attack GDAS (c) untargeted ℓ_2 attack WRN-28 (d) untargeted ℓ_2 attack WRN-40



(e) untargeted ℓ_∞ attack PyramidNet-272 (f) untargeted ℓ_∞ attack GDAS (g) untargeted ℓ_∞ attack WRN-28 (h) untargeted ℓ_∞ attack WRN-40

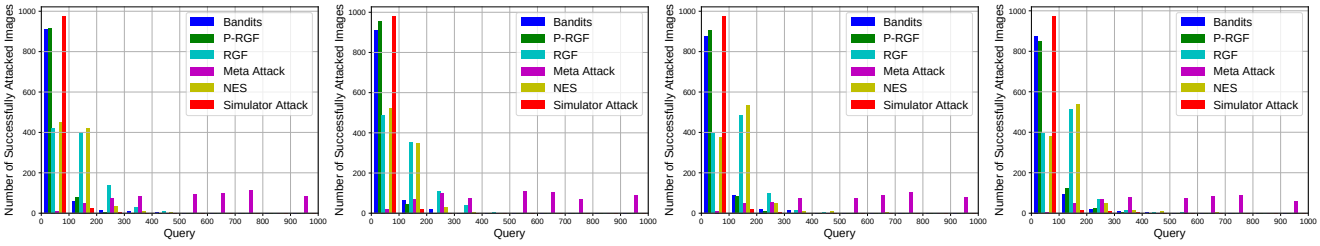


(i) targeted ℓ_2 attack PyramidNet-272 (j) targeted ℓ_2 attack GDAS (k) targeted ℓ_2 attack WRN-28 (l) targeted ℓ_2 attack WRN-40

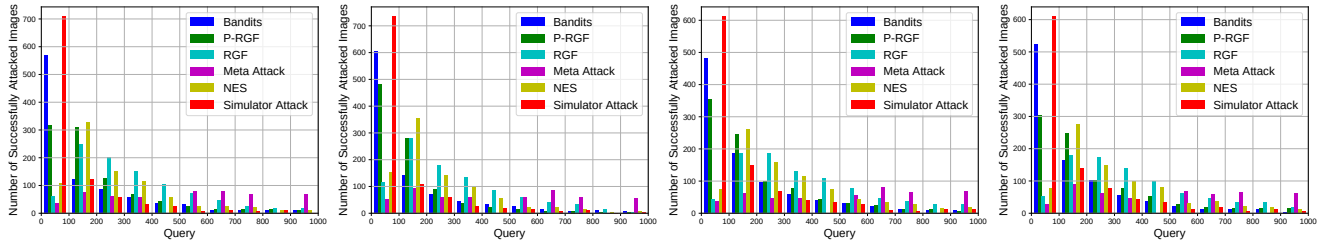


(m) targeted ℓ_∞ attack PyramidNet-272 (n) targeted ℓ_∞ attack GDAS (o) targeted ℓ_∞ attack WRN-28 (p) targeted ℓ_∞ attack WRN-40

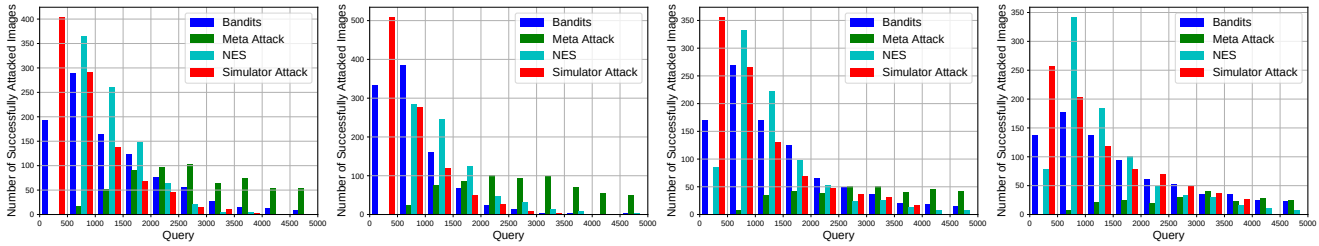
Figure 14: The histogram of query number in the CIFAR-10 dataset.



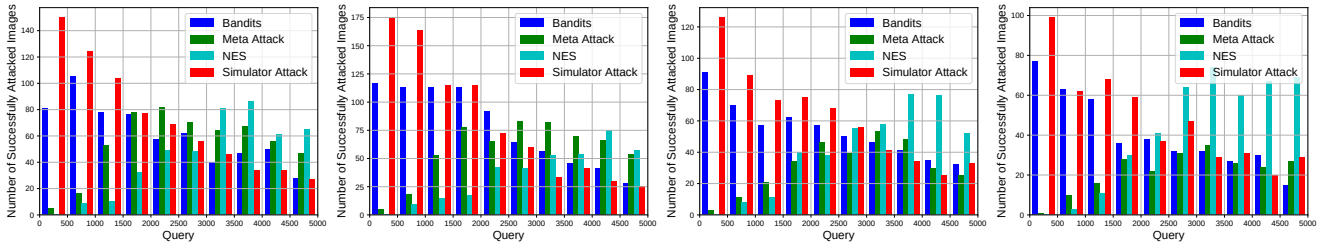
(a) untargeted ℓ_2 attack PyramidNet-272 (b) untargeted ℓ_2 attack GDAS (c) untargeted ℓ_2 attack WRN-28 (d) untargeted ℓ_2 attack WRN-40



(e) untargeted ℓ_∞ attack PyramidNet-272 (f) untargeted ℓ_∞ attack GDAS (g) untargeted ℓ_∞ attack WRN-28 (h) untargeted ℓ_∞ attack WRN-40

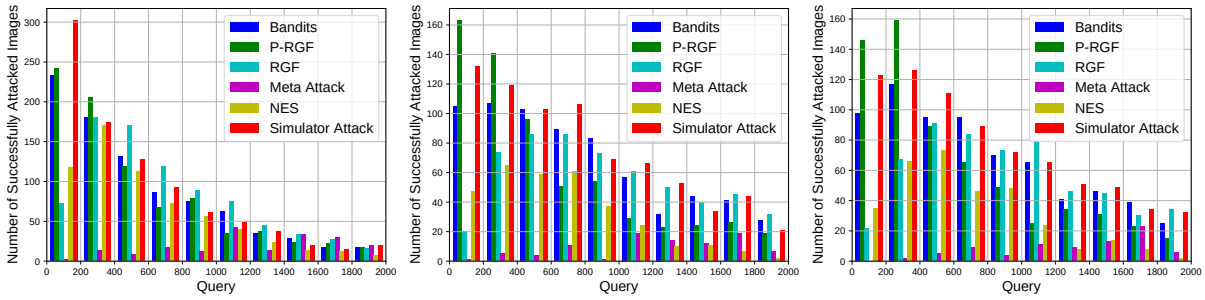


(i) targeted ℓ_2 attack PyramidNet-272 (j) targeted ℓ_2 attack GDAS (k) targeted ℓ_2 attack WRN-28 (l) targeted ℓ_2 attack WRN-40

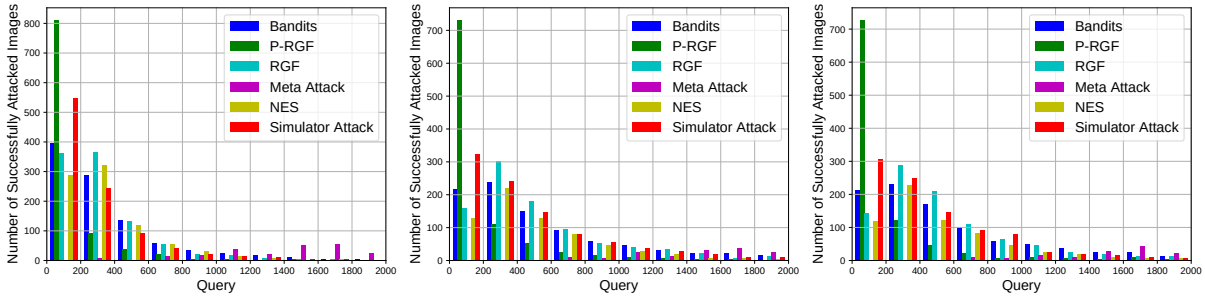


(m) targeted ℓ_∞ attack PyramidNet-272 (n) targeted ℓ_∞ attack GDAS (o) targeted ℓ_∞ attack WRN-28 (p) targeted ℓ_∞ attack WRN-40

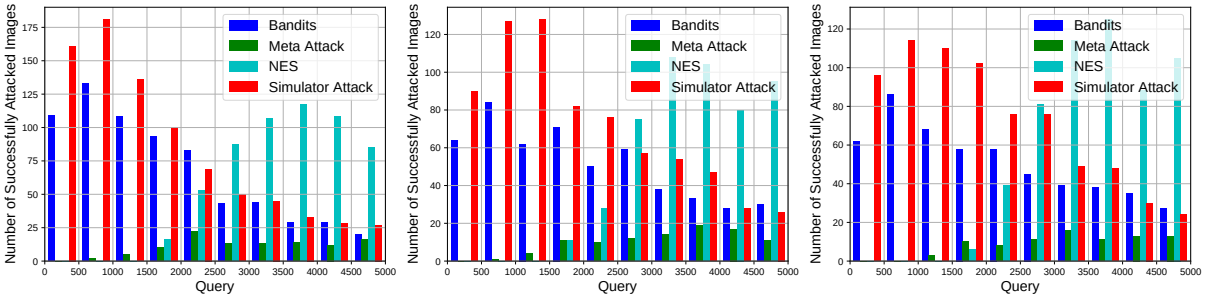
Figure 15: The histogram of query number in the CIFAR-100 dataset.



(a) untargeted ℓ_∞ attack DenseNet-121 (b) untargeted ℓ_∞ attack ResNext-101(32×4d) (c) untargeted ℓ_∞ attack ResNext-101(64×4d)

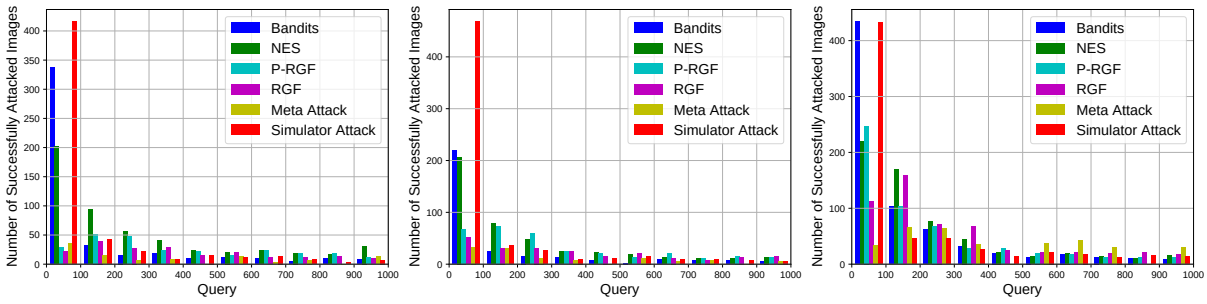


(d) untargeted ℓ_2 attack DenseNet-121 (e) untargeted ℓ_2 attack ResNext-101(32×4d) (f) untargeted ℓ_2 attack ResNext-101(64×4d)

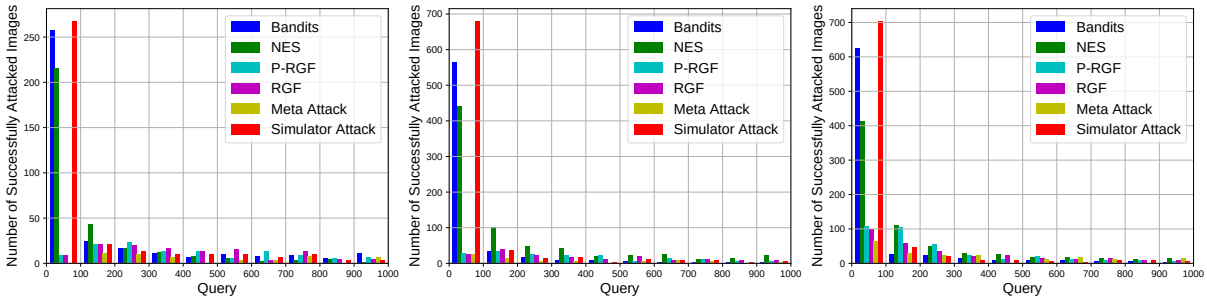


(g) targeted ℓ_2 attack DenseNet-121 (h) targeted ℓ_2 attack ResNext-101(32×4d) (i) targeted ℓ_2 attack ResNext-101(64×4d)

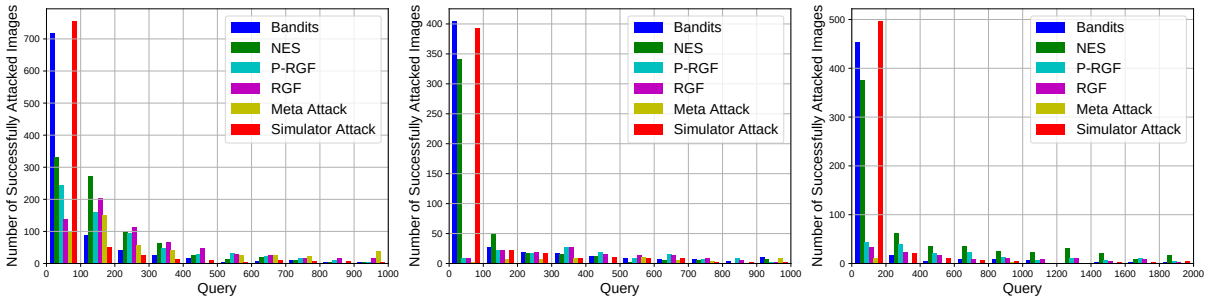
Figure 16: The histogram of query number in the TinyImageNet dataset.



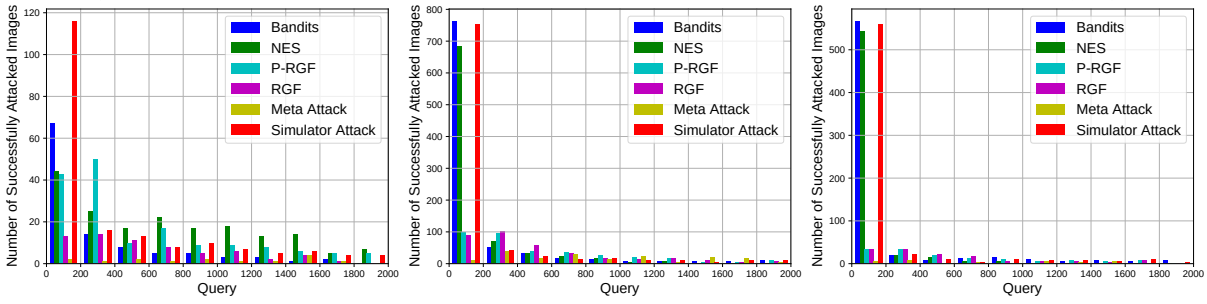
(a) attack ComDefend in CIFAR-10 (b) attack Feature Distillation in CIFAR-10 (c) attack PCL in CIFAR-10



(d) attack Adv Train in CIFAR-10 (e) attack ComDefend in CIFAR-100 (f) attack Feature Distillation in CIFAR-100



(g) attack PCL in CIFAR-100 (h) attack Adv Train in CIFAR-100 (i) attack ComDefend in TinyImageNet



(j) attack Feature Distillation in TinyImageNet (k) attack PCL in TinyImageNet (l) attack Adv Train in TinyImageNet

Figure 17: The histogram of query number on defensive models with the backbone of ResNet-50. The experimental results are obtained by performing the untargeted attacks under ℓ_∞ norm.