Adversarial Attack on Attackers: Post-Process to Mitigate Black-Box Score-Based Query Attacks

Sizhe Chen[1], **Zhehao Huang**[1], **Qinghua Tao**[2], **Yingwen Wu**[1], **Cihang Xie**[3], **Xiaolin Huang**[1]
[1]Department of Automation, Shanghai Jiao Tong University
[2]ESAT-STADIUS, KU Leuven
[3]Computer Science and Engineering, University of California, Santa Cruz

Abstract

The score-based query attacks (SQAs) pose practical threats to deep neural networks by crafting adversarial perturbations within dozens of queries, only using the model's output scores. Nonetheless, we note that if the loss trend of the outputs is slightly perturbed, SQAs could be easily misled and thereby become much less effective. Following this idea, we propose a novel defense, namely Adversarial Attack on Attackers (AAA), to confound SQAs towards incorrect attack directions by slightly modifying the output logits. In this way, (1) SQAs are prevented regardless of the model's worst-case robustness; (2) the original model predictions are hardly changed, *i.e.*, no degradation on clean accuracy; (3) the calibration of confidence scores can be improved simultaneously. Extensive experiments are provided to verify the above advantages. For example, by setting $\ell_\infty = 8/255$ on CIFAR-10, our proposed AAA helps WideResNet-28 secure 80.59% accuracy under Square attack (2500 queries), while the best prior defense (*i.e.*, adversarial training) only attains 67.44%. Since AAA attacks SQA's general greedy strategy, such advantages of AAA over 8 defenses can be consistently observed on 8 CIFAR-10/ImageNet models under 6 SQAs, using different attack targets, bounds, norms, losses, and strategies. Moreover, AAA calibrates better without hurting the accuracy.

1 Introduction

Deep Neural Networks (DNNs) are vulnerable to adversarial examples (AEs), where human-imperceptible perturbations added to clean samples can fool DNNs to give wrong predictions [1, 2]. Recently, such a threat is made practically feasible by the black-box score-based query attacks (SQAs) [3, 4, 5], as they only require the same information as users to craft efficient AEs. Users, for better judgments, need the model's prediction confidence indicated by DNNs' output scores, which is the only knowledge needed by SQAs to perform attacks. In contrast, white-box attacks [6, 7, 8] or transfer-based attacks [9, 10] require the gradients or training data of DNNs. Moreover, it has been shown that SQAs can achieve a non-trivial attack success rate by a reasonable number of queries, *e.g.*, dozens, compared to thousands of queries for decision-based query attacks [11, 12, 13]. Thus, such feasibility and effectiveness of SQAs are attracting increasing attention from defenders [14].

Defending against SQAs is a different goal compared to improving the *worst-case* robustness as commonly studied [15, 16, 17]. Because in *real-world scenarios* which SQAs are designed for, DNNs are treated as black boxes, in which the only interaction between models and users/attackers is the model's output scores. Thus, altering the scores is all defenders could do here, either in direct or indirect ways. Most existing defenses indirectly change outputs by optimizing the model [8, 18, 19] or pre-processing the inputs [14, 20], which, however, severely affect models' normal performance for different reasons. Training a robust model, *e.g.*, by adversarial training [8, 21], diverts the model's attention to learning AEs, yielding the so-called accuracy-robustness trade-off [22]. Randomizing

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

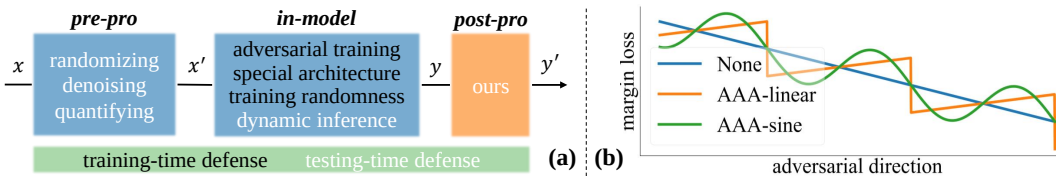

Figure 1: Compared to existing defenses on inputs or models, AAA post-processes to avoid SQAs as in (a). Our main idea is to show attackers the incorrect attack direction as in (b). Specifically, if we perturb the original undefended blue loss curve to the orange or green one, then attackers trying to decrease the loss would be mostly cheated away from their destiny, *i.e.*, the adversarial direction.

[14, 23, 24, 25, 26] and blurring [20, 27, 28, 29] inputs reduce the signal-noise ratio, inevitably hurdling accurate decision. Dynamic inference [19, 30] is time-consuming due to the test-time optimization on model. Thus, it is imperative to develop a user-friendly defense against SQAs. In this paper, we hereby consider a post-processing defense as demonstrated in Fig. 1 (a), which naturally enjoys the benefits as follows: (1) model's decision is hardly affected since good predictions have already been obtained; (2) model calibration can be simultaneously improved via post-processing [31, 32] to output accurate prediction confidence; (3) it can be flexibly used as a plug-in module for pre-trained models with negligible test-time computation overhead. Despite these merits, it remains unexplored, to the best of our knowledge, that by post-processing,

How to serve users while avoiding SQA attackers when they access the same output information?

Since SQAs are black-box attacks that find the adversarial direction to update AEs by observing the loss change indicated by DNN's output scores, we can perturb such scores directly to fool attackers into incorrect attack tracks. Following this idea, we propose the adversarial attack on attackers (AAA), which manipulates the loss trend so that attackers, trying to greedily update AEs following the original trend, are led to an incorrect path. Specifically, AAA directly optimizes DNN's logits to control the output loss to approximate certain designed curves so that attackers, seeing our confounding outputs, would be attacked to lose their direction. In Fig. 1 (b), we give two representative designs of the loss curve to achieve the above goal. One is to increase loss mostly and decrease it dramatically and shortly along the correct attack direction, and another is to oscillate the curve around the original blue line. Now that a post-processing module is adopted here, we could simultaneously use it to calibrate the model as commonly proposed [31, 33]. Thus, a simple plug-in post-module is all you need to both fool attackers and offer users accurate confidence.

By post-processing, AAA not only lowers the calibration error without hurting accuracy in all cases, but also efficiently prevents SQAs, *e.g.*, helping a WideResNet-28 [34] on CIFAR-10 [35] secure 80.59% accuracy under Square attack [5] (2500 queries), while the best prior defense (*i.e.*, adversarial training [36]) only attains 67.44%. Even if attackers try to guess defender's strategy (which requires heavy queries) and develop adaptive strategies, AAA could also confound both non-adaptive and adaptive attacks by the sine design. Because AAA attacks the general greedy update of SQAs, such advantages of AAA over 8 baseline defenses can be consistently observed on 8 CIFAR-10/ImageNet models under 6 SQAs, using different attack targets, bounds, norms, losses, and strategies, verifying AAA a user-friendly, effective, and generalizable defense. Our contributions are three-fold.

- We analyze current defenses from the view of SQA defenders and point out that a post-processing module forms not only effective but also user-friendly and plug-in defenses.
- We design a novel adversarial attack on attackers (AAA) defense that fools SQA attackers to incorrect attack directions by slightly perturbing the DNN output scores.
- We conduct comprehensive experiments showing that AAA outperforms the other 8 defenses in the accuracy, calibration, and protection from all tested 6 SQAs under various settings.

2 Related work

Query attacks are black-box attacks that only require the model's output information. Query attacks can be divided into score-based query attacks (SQAs) [3, 4, 5, 37, 38, 39, 40] and decision-based query attacks (DQAs) [11, 12, 13, 41]. SQAs greedily update AEs from original samples by

observing the loss change indicated by DNN's output scores, *i.e.*, logits or probabilities. Early SQAs try to estimate DNN's gradients by additional queries around the sample [38, 42]. Recently, it is validated query-efficient to perform fast pure random-search SQAs [4, 5]. It has also been proposed to use pre-trained surrogate models in SQAs [39, 43], which, however, demands unfeasible access to DNN's training sample as in transfer-based attacks [9, 10]. Thus, we do not focus on defending such SQAs as also in [14]. Besides SQAs, DQAs rely only on DNN's decisions, *e.g.*, the top-1 predictions, to generate AEs. Since DQAs could not perform the greedy update, they start crafting the AE from a different sample and keep DNN's prediction wrong during the attack. Currently, DQAs need thousands of queries to reach a non-trivial attack success rate [44] compared to dozens of times for SQAs [5] as compared in Appendix A, limiting their threats. Additionally, SQAs are mostly applicable and attackers do not have to resort to DQAs because, in the real world, DNNs must not only be accurate but also indicate when they tend to be incorrect [31] by outputting confidence scores.

Adversarial defense mainly has two different goals. The first goal is to improve DNN's worst-case robustness [17], which demands DNNs to have no AE around clean samples bounded by a norm ball. Such robustness is originally evaluated by gradient-based white-box attacks [6, 7], but later, gradient obfuscation phenomenon is discovered [15], motivating evaluations to incorporate random noise and black-box attacks [16] in adaptive methods [16, 45, 46]. In this assessment, adversarial training (AT) [8, 21] is validated as the most effective defense, and other methods, *e.g.*, using more or augmented data [47, 48], designing special architecture [18, 49, 50], and inducing randomness in training [51, 52], all need collaborations with AT to achieve good performance. Another defense goal is to mitigate real-case adversarial threats, and the defense performance is evaluated by the feasible and query-efficient SQAs. A more robust model in the worst cases certainly protects itself in real cases. Besides, it is also possible to defend by dynamic inference [19, 30], and randomizing [14, 23, 24, 25], denoising [20, 27] or quantifying [28, 29] inputs. However, all aforementioned defenses on inputs or models exert a non-negligible impact on accuracy, calibration, or inference speed due to the focus on learning AEs in AT, the reduction of signal-noise ratio in pre-processing, or the costly test-time optimization in dynamic inference as compared meticulously in Section 3.

Model calibration performance is a higher demand for a good model besides high accuracy. Besides correct predictions, model calibration additionally requires DNNs to produce accurate confidence in their predictions [53]. For instance, exactly 80% of the samples predicted with 80% confidence should be correctly predicted. In this point of view, the expected calibration error (ECE) [32] is widely used to quantify the error between accuracy and confidence. Various methods have been proposed to improve DNN's calibration in training [33, 54, 55, 56] or by a post-processing module after training [31, 32, 57]. Among them, temperature scaling [32] is a simple but effective method [31], which divides all output logits by a single scalar tuned by a validation set so that the ECE in testing samples would be significantly reduced. Since division is a simple post-processing operation, calibration could be simultaneously achieved by AAA, the first post-processing defense. There have been methods to avoid attacks by calibration [56] or calibrate by attack [33], but the simultaneous improvement of defense and calibration has not been reported to the best of our knowledge.

3 Preliminaries and motivation

Before presenting the proposed method, we first introduce the key ideas of SQAs and analyze existing defenses. For a sample $\boldsymbol{x}$ labelled by y, an SQA on a DNN f generates an AE $\boldsymbol{x}'$ by minimizing the margin between logits [4, 5, 40, 58] as

$$\mathcal{L}(f(\boldsymbol{x}), y) = f_y(\boldsymbol{x}) - \max_{k \neq y} f_k(\boldsymbol{x}), \tag{1}$$

namely the margin loss [7]. For defenders without the label y, it is possible to calculate the unsupervised margin loss based only on the logits vector $\boldsymbol{z} = f(\boldsymbol{x})$ as

$$\mathcal{L}_{\mathrm{u}}(\boldsymbol{z}) \triangleq \mathcal{L}\left(f(\boldsymbol{x}), \hat{y}\right), \tag{2}$$

by assuming the model prediction $\hat{y}$ to be correct, since handling originally misclassified samples falls beyond the scope of defense [17]. For attackers with label y, it can be known that the attack succeeds if $\mathcal{L}(f(\boldsymbol{x}), y) < 0$. To quickly realize this, SQAs only update AEs if a query $\boldsymbol{x}_{\mathrm{q}}$ has a lower margin loss compared to the current best query $\boldsymbol{x}_k$, *i.e.*,

$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{x}_{\mathrm{q}}, & \mathcal{L}(f(\boldsymbol{x}_{\mathrm{q}}), y) < \mathcal{L}(f(\boldsymbol{x}_k), y), \\ \boldsymbol{x}_k, & \mathcal{L}(f(\boldsymbol{x}_{\mathrm{q}}), y) \geq \mathcal{L}(f(\boldsymbol{x}_k), y). \end{cases} \tag{3}$$

Existing SQAs [4, 5, 40] are quite effective, capable of halving the accuracy within 100 queries on CIFAR-10 when $\ell_\infty = 8/255$, posing significant and practical threats. Aware of this, the first defense specifically against SQAs has been recently proposed [14] to pre-process inputs by random noise. Besides, other existing defenses are also useful to avoid SQAs, among which the most representative ones are adversarial training (AT) [8, 21], dynamic inference [19, 30], and pre-processing [14, 28]. These defenses work in different mechanisms as illustrated in Fig. 1 (a) and Section 2. Here in Table 1, we present a comparison of their key characteristics related to the defense performance.

Table 1: Expectation and effects of current defenses (unpreferable effects are marked in red)

	expectation	AT [8, 21]	pre-pro [14, 28]	dyn-inf [19, 30]	AAA (ours)
accuracy	=	↓↓	↓	=	=
calibration	↑	/	↓	↓	↑
testing cost	=	=	=	↑↑↑	=
training cost	=	↑↑↑	=	=	=
acc under SQA	↑	↑↑	↑	↑	↑↑↑

Real-case defenders are expected to serve users and meanwhile avoid SQA attackers. The former demands good accuracy, calibration, and inference speed. While the latter requires good protection, and preferably, no additional training (denoted as "="). As a training-time defense on model, AT exerts a significant impact on accuracy [22], *e.g.*, >10% accuracy drop on ImageNet [49], with several-fold training computation and an indefinite impact (denoted as "/") on calibration. Pre-processing is a test-time defense on input, which avoids SQAs by reducing the input's signal-noise ratio but inevitably hurts normal performance. Recently, the dynamic inference is proposed as a test-time defense by optimizing the model, which shows no accuracy drop but dramatically increases the computation, *e.g.*, DENT [19] consumes >700% test-time calculation. Given the above discussion, it is imperative to develop a user-friendly and efficient method to effectively defend against SQAs.

4 Adversarial attack on attackers

Defending by post-processing naturally carries the advantages of good accuracy and calibration [31], no additional training and negligible test-time computation. Although it does not improve the commonly-studied worst-case robustness [16], its potential to defeat SQAs in real cases has not been explored yet to the best of our knowledge. By investigating current defenses from the perspective of SQA defenders in Sec. 3, it is interesting to find that in the real world, no matter whether defenders alter inputs, models, or outputs, what users/attackers get are just the changes in outputs, *i.e.*, the black-box setting. Thus, it is reasonable and preferable for SQA defenders to directly manipulate DNN's output scores, which has already become a standard practice in model calibration [31].

But how? Now that attackers conduct the adversarial attack on the model, why cannot we defenders actively perform an adversarial attack on attackers? If attackers search for the adversarial direction by scores, we can manipulate such scores to cheat them into an incorrect attack path. Following this idea, we develop the adversarial attack on attackers (AAA) to control what the SQA attackers base their actions on, *i.e.*, the loss (1) indicated by the DNN's output scores. Under such direct misleading, SQA attackers, following their original policy, would be guided to wherever we want them to go due to their greediness, seeing (3). But such change on outputs should be slight to ensure it is just attackers (observing loss trend) rather than users (observing outputs) that are misled. Therefore, the loss trend can not be reversed completely, but only be reversed locally in periodically-set intervals.

In each local interval, there are lots of designs to manipulate the loss trend. For example, we can let the loss along the adversarial direction (almost) always increase and only dramatically decrease between intervals, seeing the orange line of Fig. 1 (b). It is also possible to smooth the cross-interval loss change by making it oscillate like the green line in Fig. 1 (b), so that attackers could not easily guess defender's strategy by observing the loss drop. Either design cheats attackers by slight output modifications so that users get accurate scores. This is because we manually divide loss values into small intervals and handle them separately, which is realized by designing the misleading loss curve based on periodic *loss attractors* as

$$l_{\text{atr}} = (\text{floor}\,(l_{\text{org}}/\tau) + 1/2) \times \tau, \tag{4}$$

where $l_{\text{org}} = \mathcal{L}_{\text{u}}(\boldsymbol{z}_{\text{org}})$ is the original loss for the unmodified logits $\boldsymbol{z}_{\text{org}}$. τ is the period of attractor intervals, and "floor" denotes rounding down decimals to integers. Eq. (4) sets loss values from $[l_{\text{atr}} - \tau/2, l_{\text{atr}} + \tau/2]$ as an interval. For example, if $\tau = 6$, then according to Eq. (4), the closest loss attractor of logits with $l_{\text{org}} = 0 \rightarrow 6$ is $l_{\text{atr}} = 3$. Thus, the $1/2$ term is necessary to avoid setting $l_{\text{atr}} = 0$, *i.e.*, the decision boundary, which may flip model's decision.

Periodically-set loss attractors divide logits with different l_{org} into intervals, so that we could manipulate the loss trend in each interval to form our designed loss curve that cheats attackers. The target loss values l_{trg} in the two misleading curves $l_{\text{trg_lnr}}, l_{\text{trg_sin}}$ in Fig. 1 (b) can be expressed as

$$\begin{aligned} l_{\text{trg_lnr}} &= l_{\text{atr}} - \alpha \times (l_{\text{org}} - l_{\text{atr}}) \\ l_{\text{trg_sin}} &= l_{\text{org}} - \alpha \times \tau \sin(\pi(1 - 2(l_{\text{org}} - l_{\text{atr}})/\tau)). \end{aligned} \tag{5}$$

In the former linear design, when l_{org} decreases from $l_{\text{atr}} + \tau/2$ to $l_{\text{atr}} - \tau/2$ as the sample approximates the decision boundary, we output loss that increases from $l_{\text{atr}} - \alpha \times \tau/2$ to $l_{\text{atr}} + \alpha \times \tau/2$ to fool attackers that this is an incorrect attack direction. In contrast, AAA-sine outputs an increasing loss when l_{org} decreases from around $l_{\text{atr}} + \tau/4$ to $l_{\text{atr}} - \tau/4$. Although we do not always reverse the loss trend here, it is sufficient to resist SQAs. Plus, AAA-sine enjoys an additional advantage to fool attackers that try to guess our defense strategy due to the smooth loss transition across intervals.

Although the periodic design has already altered output confidence very slightly, we could also step further to improve the precision of confidence because post-processing logits is a standard practice in model calibration [32, 57]. Thus, simultaneous defense and calibration is obtainable in a single AAA module by controlling the loss while encouraging the output confidence $\sigma(\boldsymbol{z}) = \max(\text{softmax}(\boldsymbol{z}))$, *i.e.*, the maximum probability after softmax, to approach the calibrated one p_{trg} as

$$\min_{\boldsymbol{z}} \quad \| \mathcal{L}_{\text{u}}(\boldsymbol{z}) - l_{\text{trg}} \|_1 + \beta \cdot \| \sigma(\boldsymbol{z}) - p_{\text{trg}} \|_1, \tag{6}$$

which is a straightforward design to fulfill two purposes. The first term encourages the perturbed logits z to have a loss $L_{\text{u}}(z)$ close to the target value l_{trg}, forming the misleading loss curve for SQAs. The second item motivates the output confidence $\sigma(z)$ close to the calibrated one p_{trg} so that users get accurate confidence scores. β balances between the above two goals. Despite its simplicity, Eq. (6) has to be solved by optimization because the exponential operation in softmax makes Eq. (6) a transcendental equation without closed-form solutions. Luckily, optimizing low-dimensional logits is not costly [32, 57]. By optimizing Eq. (6) for κ iterations, DNN's output is both accurate and misleading, achieving two seemingly contradictory goals simultaneously.

The calibrated confidence p_{trg} is obtainable by various model calibration methods [32, 57]. Among them, temperature scaling [32] is simple but effective [31], which divides all logits by a scalar T as $p_{\text{trg}} = \sigma(\boldsymbol{z}_{\text{org}}/T)$. The temperature T is tuned by a validation set to minimize the calibration error, and here, such tuning is conducted when the optimization (6) is also performed for κ iterations so that we can find the temperature that suits AAA best. After that, T is fixed for inference in AAA.

We summarize our algorithm in Alg. 1. AAA first calculates the original loss l_{org}, according to which sets the target loss l_{trg} that forms the misleading loss curve. Then AAA optimizes the logits to reach l_{trg}, and also the target confidence p_{trg}, which is obtained by dividing the original logits using a pre-tuned temperature T. The overall procedure is in test-time with only optimization on the logits, making AAA a computation-efficient, plug-in, and model-agnostic method with good accuracy, calibration, and defense against SQAs.

Algorithm 1 Adversarial Attack on Attackers

Input: the logits $\boldsymbol{z}_{\text{org}}, T, \tau, \alpha, \beta, \kappa$.
Output: post-processed logits $\boldsymbol{z}$

1: get original loss $l_{\text{org}} = \mathcal{L}_{\text{u}}(\boldsymbol{z}_{\text{org}})$ by (2)
2: set target loss l_{trg} by (5)
3: set target confidence $p_{\text{trg}} = \sigma(\boldsymbol{z}_{\text{org}}/T)$
4: initialize $\boldsymbol{z} = \boldsymbol{z}_{\text{org}}$ and optimize it for (6)
5: **return** $\boldsymbol{z}$

5 Experiments

5.1 Setup

We evaluate AAA along with 8 defense baselines, including random noise defense (RND [33]), adversarial training (AT [36, 59, 49]), dynamic inference (DENT [19]), training randomness (PNI [51]), and ensemble (TRS [60]). Results of AT with extra data [22]) are in Appendix A. For AAA, we mostly experiment with the linear design, because the two designs of AAA share most good

characteristics. AAA-sine could fool adaptive attacks, which is costly to design in real world as discussed in Sec 5.6. AAA-linear uses $\alpha = 1, \tau = 6, \beta = 5, \kappa = 100$, and an Adam optimizer [61] with learning rate $0.1, \beta_1 = 0.9, \beta_2 = 0.999$. AAA-sine uses $\alpha = 0.7$ and inherits other hyper-parameters in study of adaptive attacks. To calibrate simultaneously, we perform temperature scaling using 1K/5K samples for CIFAR-10 testing set / ImageNet validation set to and make them disjoint with the attack samples as much as possible. Other hyper-parameters are in Appendix E.

The defenses are assessed by 6 state-of-the-art SQAs, which are random search methods including Square [5], SignHunter [40], and SimBA [4], and gradient estimation methods including NES [58], and Bandits [3]. Square attack [5] selects a decreasing size of random square perturbations. SignHunter [40] estimates the sign of gradient and flips corresponding perturbations. SimBA [4] randomly samples perturbations given an orthonormal basis. NES [58] adopts the Natural Evolutionary Strategies for gradient estimation. Bandits [3] jointly uses the time and data-dependent gradient prior. Some SQAs use pre-trained models [39, 43], which, however, demands unfeasible access to DNN's training sample. To evaluate real-case defenses against training-based attacks, we alternatively consider a practical SQA QueryNet [62] based on model stealing. Results of DQAs and SQAs using other than margin loss are put in Appendix A. We mostly perform untargeted ℓ_∞ attacks, but we also test the targeted and ℓ_2 attacks under different bounds to observe AAA's generalization.

We use 8 DNNs, which are mostly WideResNets [34] as in [17]. The pre-trained models of PNI [51] / TRS [60] are ResNet-20 [63], and we also test ResNeXt-101 [64] in ImageNet. Other studied DNNs come from RobustBench [17] and torchvision [65] as specified in Appendix E. AT models are tested using the same bound as in AT, if not otherwise stated. We use all 10K CIFAR-10 testing samples. For ImageNet, we randomly-select 1K validation samples from all 1K classes respectively (1 image in 1 class) to eliminate the class bias as in [62]. All images are rescaled to $[0, 1]$, and the ImageNet ones are resized to 224×224. Before feeding them to DNNs, we quantify images to 8-bit, imitating the real-case 8-bit image setting [62]. Experiments are performed on an NVIDIA Tesla A100 GPU (a GPU with 4GB+ memory works). Our code is available at `https://github.com/Sizhe-Chen/AAA`.

As defenders, we are concerned about DNN's remaining accuracy after it being attacked by SQAs for a certain query times. Thus, we report such *SQA adversarial accuracy* after 100 and 2500 queries, reflecting DNN's performance under mild and extreme SQAs. We include the average query times, another metric commonly used by attackers, in Appendix C. To measure the calibration, the expected calibration error (ECE) [32] is commonly used. ECE divides all N testing samples into M bins, and each bin contains samples with confidence ranging from the mth quantile to the $(m+1)^{\text{th}}$ quantile of $[0, 1]$. Then ECE is calculated by the difference between accuracy and confidence as $\frac{1}{N}\sum_{m=1}^{M}\left|\sum_{i\in B_m}\mathbf{1}(\hat{y}_i = y_i) - \sum_{i\in B_m}\hat{p}_i^{\hat{y}_i}\right|$, where $\hat{y}_i$ is the predicted label of the ith sample in the mth bin B_m, and $\hat{p}_i^{\hat{y}_i}$ represents the probability confidence of this prediction.

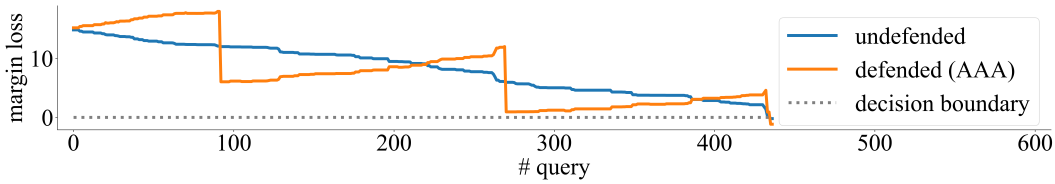

Figure 2: The AAA(linear)-defended and undefended margin loss value when attacking the undefended WideResNet-28 [34] in RobustBench [17] by Square attack [5] ($\ell_\infty = 8/255$) using the 9953th CIFAR-10 test sample (other samples have similar trends as in Appendix B). AAA fools attackers precisely as shown by the tiny symmetric oscillations of two lines.

5.2 Visual illustration of AAA

We first visually illustrate the mechanism and effects of AAA. AAA, besides fooling attackers, also calibrates the model, forming the loss curve as in Fig. 2. Slightly different from the ideal defense in Fig. 1 (b) is that the orange curve is lowered for calibration. However, calibration does not hurdle fooling attackers, *i.e.*, the orange line is mostly going opposite to the blue line in a precisely symmetric manner. Also, two lines cross the decision boundary at the same time, meaning that AAA does not change the decision (the negligible increase of AAA accuracy comes from randomness).

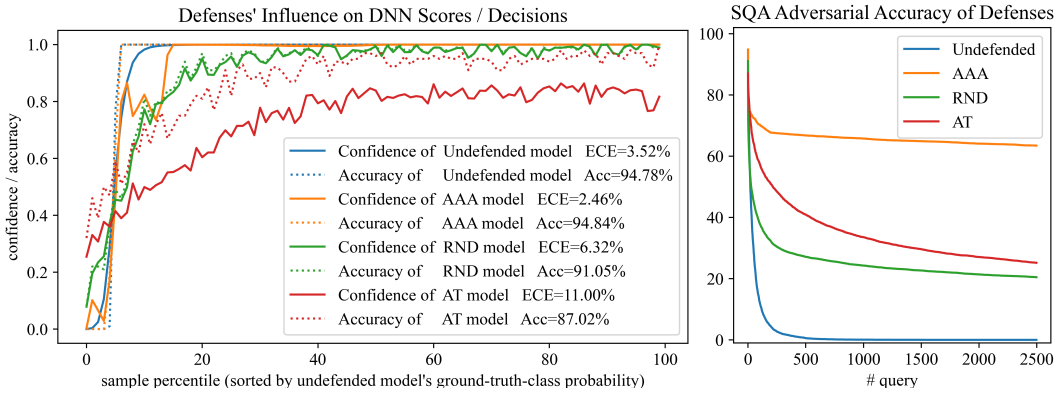

Figure 3: The left figure illustrates the change of output scores for different defenses. We first sort 10K testing samples in ascending order according to their ground-truth-class probability predicted by the undefended model (blue line), and then divide them into 100 bins, so that the left bins in the figure stand for low-confidence or misclassified samples, and vice versa. Then for samples in each bin, we plot the confidence (solid lines) and accuracy (dotted line) for AT [36], RND [33], and AAA-linear . Compared to AT and RND, AAA alters scores most slightly without influencing the accuracy. The right figure shows DNN's accuracy under Square attack [5] when $\ell_2 = 2.5$, indicating that AAA outperforms alternatives in defending SQAs by a large margin with also the highest clean accuracy.

Besides the loss, it is also necessary to observe the output probabilities of AAA, which is displayed in Fig. 3 (left). AAA, indicated by the orange lines, changes the scores slightly without hurting accuracy (dotted lines) compared to RND [33] and AT [36]. By such small modifications, AAA not only improves the calibration but also prevents SQAs by misleading them into incorrect directions. As displayed by the right part of Fig. 3, AAA is super effective in avoiding SQAs, even if it is a query-efficient attack working at a large bound. Specifically, AAA preserves a standard trained DNN to have >60% accuracy after 2500 queries, doubling the performance of AT and tripling that for RND.

Table 2: The defense performance under attacks (#query $= 100/2500$)

Model	Metric / Attack	None	AT [36, 59, 49]	RND [14]	AAA-linear
CIFAR-10	ECE (%)	3.52	11.00	6.32	**2.46**
$\ell_\infty = \frac{8}{255}$	Acc (%)	94.78	87.02	91.05	**94.84**
	Square [5]	39.38 / 00.09	78.30 / 67.44	60.83 / 49.15	**81.36 / 80.59**
Wide-	SignHunter [40]	41.14 / 00.04	78.87 / 66.79	61.02 / 47.82	**79.41 / 76.71**
ResNet-	SimBA [4]	53.04 / 03.95	84.21 / 75.85	76.39 / 64.34	**88.86 / 83.36**
28 [34]	NES [58]	83.42 / 12.24	85.92 / 81.01	86.23 / 68.19	**90.62 / 85.95**
	Bandit [3]	69.86 / 41.03	**83.62** / 76.25	70.44 / 41.65	80.86 / **78.36**
ImageNet	ECE (%)	5.42	5.03	5.79	**4.30**
$\ell_\infty = \frac{4}{255}$	Acc (%)	77.11	66.30	75.32	**77.17**
	Square [5]	52.27 / 09.25	59.20 / 51.11	58.67 / 50.54	**63.13 / 62.51**
Wide-	SignHunter [40]	53.05 / 13.88	59.47 / 56.22	59.36 / 52.98	**62.35 / 56.80**
ResNet-	SimBA [4]	71.79 / 20.90	65.64 / 47.60	66.36 / 63.27	**74.16 / 67.14**
50 [34]	NES [58]	77.11 / 64.93	66.30 / 64.38	71.33 / 66.05	**77.12 / 67.06**
	Bandit [3]	71.33 / 65.77	65.30 / 63.98	65.15 / 61.38	**72.15 / 70.53**
ImageNet	ECE (%)	8.37	**5.74**	8.93	7.38
$\ell_\infty = \frac{4}{255}$	Acc (%)	**78.21**	63.94	76.75	**78.21**
	Square [5]	54.51 / 11.73	57.99 / 51.47	58.56 / 48.20	**66.32 / 65.77**
Res-	SignHunter [40]	54.12 / 13.30	59.14 / 56.49	58.02 / 52.50	**62.72 / 59.60**
NeXt-	SimBA [4]	70.86 / 24.64	59.40 / 57.53	68.31 / 66.16	**74.06 / 67.26**
101 [64]	NES [58]	**78.21** / 66.24	63.94 / 62.51	73.37 / **69.30**	**78.21** / 68.51
	Bandit [3]	72.19 / 67.73	63.18 / 62.26	67.69 / 64.62	**72.91 / 71.48**

5.3 Numerical results of AAA

We report the main numerical results in Table 2, where RND and AAA-linear are directly implemented in the undefended model denoted as "None". The AT method for the three models are PSSiLU [36], vanilla AT [59], and feature denoising [49], respectively. According to Table 2, AAA-linear not only consistently reduces ECE by >12%, but also does not hurt clean accuracy. In contrast, the AT models lose >7% accuracy and RND also endures a drop in accuracy and calibration. Under SQAs, AAA-linear preserves a significantly higher accuracy than the undefended model, which is totally destroyed. The most threatening SQAs in two datasets more than halve the CIFAR-10 model's accuracy within 100 queries and degrade ImageNet ones to <15% by 2500 times. However, the AAA model remains >75% and >55% accuracy in extreme cases. AT and RND are useful in mitigating SQAs, but it is AAA-linear that tops the defense performance in almost all cases.

Besides AT and RND, diverse defenses have also been proposed, and it would be interesting to see their results. DENT [19] optimizes the model in test-time, trying to learn the AE distribution. PNI [51] injects noise during training, making the learned weights less sensitive to input perturbations. TRS [60] ensembles three models with low attack transferability between each other. They are originally developed for gradient-based attacks, but also provide some protection against SQAs. As displayed in Table 3, however, they are not comparable to AAA-linear in real cases regarding the accuracy, calibration, and defense performance. Here, we also test a strong SQA QueryNet, which uses three architecture-alterable models to steal the DNN. Due to its utilization of large-scale testing samples, QueryNet greatly hurts DNNs, but AAA is still the defense that protects the model best.

Table 3: Various defenses under strong attacks (#query $= 100/2500$, CIFAR-10, $\ell_\infty = 8/255$)

Metric / Attack	None	DENT [19]	PNI [51]	TRS [60]	AAA-linear
ECE (%)	3.52	5.20	3.09	3.64	**2.46**
Acc (%)	94.78	94.80	81.91	88.64	**94.84**
Square [5]	39.38 / 00.09	62.01 / 35.07	57.69 / 45.47	56.26 / 23.91	**81.36 / 80.59**
QueryNet [40]	13.50 / 00.03	39.35 / 20.75	44.06 / 34.69	16.43 / 08.49	**50.01 / 49.63**

5.4 Generalization of AAA

Aside from the untargeted ℓ_∞ attacks, we also conduct targeted attacks, ℓ_2 attacks under different bounds to study the generalization of AAA. A targeted attack is successful only if an AE is mispredicted as a pre-set class, which, is randomly chosen from incorrect classes for each sample here. And ℓ_2 attacks bound the perturbations by ℓ_2 norm, which is reported to fool DNNs better [10]. Here we additionally validate AAA's plug-in advantage by combining it with AT. Note that what differs from AAA and most existing defenses [49, 50, 51] combined with AT is that AAA has already achieved excellent defense performance, so such a combination is feasible but not necessary. We choose Square attack [5] to perform the above evaluations because it is the most effective SQA as tested in Table 2. The results are presented in Table 4, where all models are exactly the same as in Table 2 (CIFAR-10) without tuning hyper-parameters of defense to fairly evaluate the generalization of different methods. The results of ℓ_2 AT model and other ℓ_∞ AT models are put in Appendix A.

Table 4: Generalization of AAA tested by Square attack [5] (#query $= 100/2500$, CIFAR-10)

Metric / Attack	None	AAA-linear	AT [36]	AT-AAA-linear
ECE (%)	3.52	**2.46**	11.00	**10.56**
Acc (%)	94.78	**94.84**	**87.02**	**87.02**
untargeted $\ell_\infty = 8/255$	39.38 / 00.09	**81.36 / 80.59**	78.30 / 67.44	**80.80 / 80.13**
targeted $\ell_\infty = 8/255$	75.59 / 02.84	**92.05 / 91.62**	85.75 / 82.72	**86.22 / 86.13**
untargeted $\ell_2 = 0.5$	81.53 / 18.75	**92.66 / 92.63**	84.26 / 78.97	**85.12 / 84.31**
untargeted $\ell_2 = 2.5$	12.77 / 00.01	**70.35 / 63.46**	57.88 / 25.19	**74.03 / 73.72**

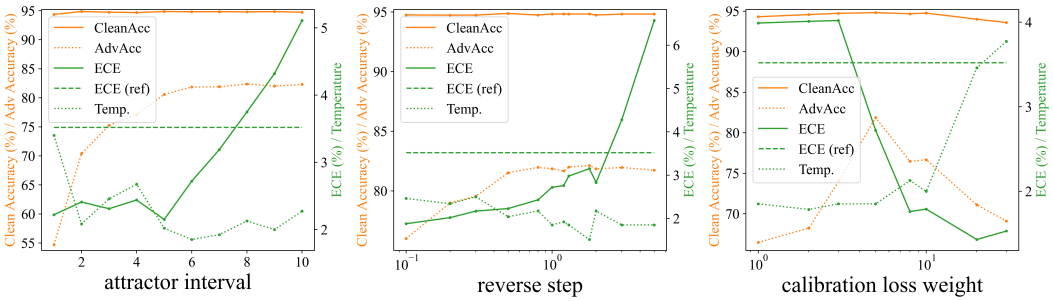

Figure 4: The influence of AAA hyper-parameters (the attractor interval τ in (4), the reverse step α in (5), and the calibration loss weight β in (6)) to accuracy (left axis), ECE (right axis), SQA adversarial accuracy (left axis), and temperature (right axis). The dashed ECE (ref) means the undefended model.

In the difficult targeted attack setting, the undefended model remains only 2.84% accuracy after 2500 queries, which is approximately just the accuracy drop of the AAA model. For ℓ_2 attacks, AAA-linear is still capable of mitigating threats without hurting users, and its superiority is more outstanding as the attack bound becomes larger. AT models, although robust, suffer from attacks under a large or different norm ball [56]. Thus, its defense effects decrease as SQA alters the setting, seeing the bottom line. Defended by AAA, however, this drawback would be greatly avoided. An ℓ_∞ AT model, even under $\ell_2 = 2.5$ attack after 2500 queries, is hardly influenced, *i.e.*, increasing query times after 100 queries hardly better the attack performance, discouraging SQA attackers.

5.5 Hyper-parameters of AAA

The reported amazing results in various settings are all obtained using fixed parameters that are heuristically selected, and it would be necessary to see how each of them affects the results. The attractor interval τ in (4) decides the period of the margin loss attractors, and a larger τ divides the losses into fewer intervals so that attackers are harder to jump out of one of them. The reverse step α in (5) controls the reverse degree, and if it increases, the modified loss curve would be steeper oppositely along the adversarial direction, emphasizing the defense. The calibration loss weight β in (6) directly balances defense and calibration in optimization. Here we still consider the accuracy, ECE, and SQA adversarial accuracy (under 100 queries). Plus, we study an additional metric, the temperature T of logits rescaling tuned with AAA-linear.

The results are shown in Fig. 4. The adversarial accuracy far exceeds the undefended model (39.38%) and mostly surpasses AT (67.44%). The clean accuracy is hardly impacted and the ECE is mostly below the undefended model (the green dashed line). Thus, AAA's good performance is insensitive to hyper-parameters, even if tuned in logarithmic scale. Regarding the trend, an intuitive conclusion is that a larger α, a larger τ, or a smaller β that highlights defense more v.s. calibration would thereby increase the adversarial accuracy and ECE. Interestingly, the temperature mostly decreases as the defense is emphasized (the logits are divided by a smaller value), indicating that the model tends to output lower confidence to defend, consistent with the situation in AT [17].

We also study the influence of optimization times κ in AAA to see the balance between the amount of computation and defense performance. AAA's runtime is estimated by inferring 10K CIFAR-10 samples in a WideResNet28 model on an NVIDIA GeForce RTX 2080Ti GPU. The results are shown in Table 5. As we see, optimizing low-dimensional logits is not costly, and good defense and calibration results are also obtainable by 60 to 80 iterations, which costs less time. Since optimizing logits is independent of model size, model owners could determine AAA runtime precisely.

Table 5: Influence of the optimization times in AAA (100-query Square attack on CIFAR-10)

κ	0	20	40	60	80	100
ECE	3.52	2.87	2.81	2.66	2.53	2.53
Adv-Acc	39.38	79.29	80.92	81.37	81.28	81.36
inference time per sample (ms)	1.016	1.034	1.088	1.099	1.143	1.163

5.6 Adaptive attacks of AAA

Designing adaptive attacks, though possible, is costly and easy to bypass in real-world scenarios. Because here, attackers and defenders are in a double-blind relationship, i.e., attackers do not know the model, including the defense strategy. Thus, the discovery process of defense strategies for developing adaptive attacks would require additional queries. After that, attackers also have to devote considerable manual efforts to creatively deduce what defenders actually do. In this regard, AAA-linear has already imposed a great hurdle to adaptive attackers.

But still, it is interesting to see that after unveiling defender's strategy by great effort, how adaptive attackers could break the AAA model. There are two straightforward adaptive attacks against AAA: going bidirectional or opposite to the original attack direction, which are both based on the exceptional loss change that attackers observe in AAA-linear to decide the update direction. However, adaptive attackers are also easy to fool because the dramatic drop of loss across intervals could be smoothed, e.g. by using a sine function to design the target loss as the second equation in (5). In this way, neither direction of search is likely to figure out the defense strategy, seeing results below.

Table 6: AAA under adaptive attacks (100 queries)

Defense	None	AAA-linear	AAA-sine
Square	39.38	81.36	78.34
bi-Square	57.09	62.91	76.69
op-Square	94.78	57.31	76.41

The bidirectional / opposite search weakens AAA-linear from 81.36% to 62.91% / 57.31%, but AAA-sine that ascends and descends the loss along the attack direction misleads both non-adaptive and adaptive SQAs. By a sine l_{trg}, the defended adversarial accuracy for the non-adaptive attack is kept at 78.34% while that under adaptive attacks is improved to 76.69% / 76.41%. Thus, defenders could easily mitigate even adaptive SQAs following our idea to fool attackers.

6 Conclusion, impacts and limitations

We develop a novel defense against score-based query attacks (SQAs). Our main idea is to actively attack the attackers (AAA), misleading them into incorrect attack directions. AAA achieves that by post-processing DNN's logits while enforcing the new output confidence to be calibrated, making AAA a deterministic plug-in test-time defense with improvements in calibration and accuracy by costing negligible computation overhead. Compared to alternative defenses, AAA is effective in mitigating SQAs according to our study on 8 defenses, 6 SQAs, and 8 DNNs under various settings.

As a defense in real-world applications, AAA greatly mitigates the adversarial threat without requiring huge computational burden. For example, in autonomous driving or supervision systems, the post-processing defense module could be directly implemented in pre-trained models. Despite the low cost, the benefits of adopting AAA are profound. In most cases, users would be provided with a more accurate confidence score so that they know better when the model tends to fail. In adversarial cases, SQAs, the most threatening attack in real cases, would be effectively prevented.

AAA is developed to especially prevent SQAs. Thus, defending other types of attacks is beyond our scope. For example, AAA does not improve the worst-case robustness evaluated in white-box settings [16, 46] where attackers have complete knowledge of the model (the AutoAttack [16] robust accuracy would be increased in an undesirable manner by AAA). Also, AAA is not applicable to avoiding transfer-based attacks and decision-based query attacks, which are either unfeasible or inefficient in the real world, because AAA induces only a negligible impact on the decision boundary.

Acknowledgments and Disclosure of Funding

This work was partially supported by National Natural Science Foundation of China (61977046), Research Program of Shanghai Municipal Science and Technology Committee (22511105600), and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102). Cihang Xie is supported by a gift from Open Philanthropy. The authors are grateful to the anonymous reviewers for their insightful comments.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2015.

[3] A. Ilyas, L. Engstrom, and A. Madry, "Prior convictions: Black-box adversarial attacks with bandits and priors," in *International Conference on Learning Representations (ICLR)*, 2019.

[4] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *International Conference on Machine Learning (ICML)*, 2019, pp. 2484–2493.

[5] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *the European Conference on Computer Vision (ECCV)*, 2020, pp. 484–501.

[6] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2574–2582.

[7] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *the IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[8] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.

[9] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2730–2739.

[10] S. Chen, Z. He, C. Sun, and X. Huang, "Universal adversarial attack on attention and the resulting dataset DAmageNet," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022, pp. 2188–2197.

[11] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *International Conference on Learning Representations (ICLR)*, 2018.

[12] M. Cheng, T. Le, P.-Y. Chen, H. Zhang, J. Yi, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," in *International Conference on Learning Representations (ICLR)*, 2018.

[13] J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A query-efficient decision-based attack," in *the IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1277–1294.

[14] Z. Qin, Y. Fan, H. Zha, and B. Wu, "Random noise defense against query-based black-box attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 7650–7663.

[15] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International Conference on Machine Learning (ICML)*, 2018, pp. 274–283.

[16] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *International Conference on Machine Learning (ICML)*, 2020, pp. 2206–2216.

[17] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, "Robustbench: a standardized adversarial robustness benchmark," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[18] Y. Li, Z. Yang, Y. Wang, and C. Xu, "Neural architecture dilation for adversarial robustness," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 29 578–29 589.

[19] D. Wang, A. Ju, E. Shelhamer, D. Wagner, and T. Darrell, "Fighting gradients with gradients: Dynamic defenses against adversarial attacks," *arXiv preprint arXiv:2105.08714*, 2021.

[20] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer, "Deflecting adversarial attacks with pixel deflection," in *the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8571–8580.

[21] F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. D. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *6th International Conference on Learning Representations (ICLR)*, 2018.

[22] R. Rade and S.-M. Moosavi-Dezfooli, "Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off," in *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

[23] H. Salman, M. Sun, G. Yang, A. Kapoor, and J. Z. Kolter, "Denoised smoothing: A provable defense for pretrained classifiers," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 21 945–21 957.

[24] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. L. Yuille, "Mitigating adversarial effects through randomization," in *International Conference on Learning Representations (ICLR)*, 2018.

[25] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," in *the European Conference on Computer Vision (ECCV)*, 2018, pp. 369–385.

[26] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *the IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 656–672.

[27] Z. Liu, Q. Liu, T. Liu, N. Xu, X. Lin, Y. Wang, and W. Wen, "Feature distillation: Dnn-oriented JPEG compression against adversarial examples," in *the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 860–868.

[28] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2018.

[29] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," in *International Conference on Learning Representations (ICLR)*, 2018.

[30] Y.-H. Wu, C.-H. Yuan, and S.-H. Wu, "Adversarial robustness via runtime masking and cleansing," in *International Conference on Machine Learning (ICML)*, 2020, pp. 10 399–10 409.

[31] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning (ICML)*, 2017, pp. 1321–1330.

[32] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *the AAAI conference on Artificial Intelligence (AAAI)*, 2015, pp. 2901–2907.

[33] Y. Qin, X. Wang, A. Beutel, and E. Chi, "Improving calibration through the relationship with adversarial robustness," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 14 358–14 369.

[34] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conference (BMVC)*, 2016, pp. 87.1–87.12.

[35] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," 2009.

[36] S. Dai, S. Mahloujifar, and P. Mittal, "Parameterizing activation functions for adversarial robustness," *arXiv preprint arXiv:2110.05626*, 2021.

[37] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *the ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 506–519.

[38] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 15–26.

[39] S. Cheng, Y. Dong, T. Pang, H. Su, and J. Zhu, "Improving black-box adversarial attacks with a transfer-based prior," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 10 934–10 944.

[40] A. Al-Dujaili and U.-M. O'Reilly, "Sign bits are all you need for black-box attacks," in *International Conference on Learning Representations (ICLR)*, 2019.

[41] M. Cheng, S. Singh, P. H. Chen, P.-Y. Chen, S. Liu, and C.-J. Hsieh, "Sign-opt: A query-efficient hard-label adversarial attack," in *International Conference on Learning Representations (ICLR)*, 2019.

[42] A. N. Bhagoji, W. He, B. Li, and D. Song, "Practical black-box attacks on deep neural networks using efficient query mechanisms," in *the European Conference on Computer Vision (ECCV)*, 2018, pp. 154–169.

[43] Y. Guo, Z. Yan, and C. Zhang, "Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 3820–3829.

[44] D. C. R. Viet Quoc Vo, Ehsan Abbasnejad, "Ramboattack: A robust query efficient deep neural network decision exploit," *Network and Distributed Systems Security (NDSS) Symposium*, 2022.

[45] C. Yao, P. Bielik, P. Tsankov, and M. Vechev, "Automated discovery of adaptive attacks on adversarial defenses," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 26 858–26 870.

[46] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 1633–1645.

[47] S. Gowal, S.-A. Rebuffi, O. Wiles, F. Stimberg, D. A. Calian, and T. A. Mann, "Improving robustness using generated data," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 4218–4233.

[48] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. A. Mann, "Data augmentation can improve robustness," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 29 935–29 948.

[49] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 501–509.

[50] Y. Fu, Q. Yu, Y. Zhang, S. Wu, X. Ouyang, D. Cox, and Y. Lin, "Drawing robust scratch tickets: Subnetworks with inborn robustness are found within randomly initialized networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 13 059–13 072.

[51] Z. He, A. S. Rakin, and D. Fan, "Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack," in *the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 588–597.

[52] H. Salman, J. Li, I. Razenshteyn, P. Zhang, H. Zhang, S. Bubeck, and G. Yang, "Provably robust deep learning via adversarially trained smoothed classifiers," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 11 292–11 303.

[53] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *International Conference on Machine learning (ICML)*, 2005, pp. 625–632.

[54] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[55] S. Thulasidasan, G. Chennupati, J. A. Bilmes, T. Bhattacharya, and S. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 13 911–13 922.

[56] D. Stutz, M. Hein, and B. Schiele, "Confidence-calibrated adversarial training: Generalizing to unseen attacks," in *International Conference on Machine Learning (ICML)*, 2020, pp. 9155–9166.

[57] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers," in *the AAAI conference on Artificial Intelligence (AAAI)*, 2001, pp. 609–616.

[58] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *International Conference on Machine Learning (ICML)*, 2018, pp. 2142–2151.

[59] H. Salman, A. Ilyas, L. Engstrom, A. Kapoor, and A. Madry, "Do adversarially robust imagenet models transfer better?" in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 3533–3545.

[60] Z. Yang, L. Li, X. Xu, S. Zuo, Q. Chen, P. Zhou, B. Rubinstein, C. Zhang, and B. Li, "TRS: Transferability reduced ensemble via promoting gradient diversity and model smoothness," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 17 642–17 655.

[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[62] S. Chen, Z. Huang, Q. Tao, and X. Huang, "Querynet: Attack by multi-identity surrogates," *arXiv preprint arXiv:2105.15010*, 2021.

[63] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[64] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1492–1500.

[65] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 8024–8035.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the third paragraph in Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the second paragraph in Sec. 6 for potential impacts. Negative ones have not been identified since we aim at defense.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] Our contributions are mostly empirical.
   (b) Did you include complete proofs of all theoretical results? [N/A] Our contributions are mostly empirical.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] All the code would be released with detailed instructions and annotations. Core code of AAA is shown in Appendix D. The data are described in experimental setup in Sec. 5.1.
   (b) Did you specify all the training details (*e.g.*, data splits, hyperparameters, how they were chosen)? [Yes] Details are presented in Sec. 5.1 and Appendix E
   (c) Did you report error bars (*e.g.*, with respect to the random seed after running experiments multiple times)? [Yes] Error bars and computation amount are reported in Appendix C.
   (d) Did you include the total amount of compute and the type of resources used (*e.g.*, type of GPUs, internal cluster, or cloud provider)? [Yes] Our devices are described in Sec. 5.1, and the amount of computation is quantified in Appendix C.
4. If you are using existing assets (*e.g.*, code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We state the source of DNNs in experimental setup in Sec. 5.1.
   (b) Did you mention the license of the assets? [Yes] Most used code assets are under MIT License. Square attack is under BSD 3-Clause License. ImageNet is under a custom research and non-commercial license.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We would release our code as an asset.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Our empirical studies are based on public datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] Our used public datasets generally do not contain personally identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] Our experiments involve no human subjects and crowdsourcing.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] Our experiments involve no human subjects or crowdsourcing.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] Our experiments involve no human subjects or crowdsourcing.

A More numerical results

Decision-based attacks. In Table 7, we present results concerning a SOTA decision-based attack, RamBo [44] (denoted as "RB"). As shown in the first two columns, compared to Square [5] (denoted as SQ), RamBo could hardly impact the undefended DNN even after 2500 queries under a large $\ell_2 = 1.5$ bound. Thus, it is reasonable for us to target at mitigating black-box SQAs in real cases.

ℓ_2 attacks. Seeing the right four columns of Table 7, one could observe that when the attack bound increases, both ℓ_∞ and ℓ_2 AT models are impacted much more significantly. Moreover, AAA's superiority is enhanced as the attack becomes stronger.

Table 7: The adversarial accuracy under RamBo (RB) [44] and Square (SQ) [5] (#query $= 2500$)

Bound	None-RB	None-SQ	ℓ_∞ AT [36]-SQ	ℓ_2 AT [22]-SQ	RND-SQ	AAA-SQ
$\ell_2 = 0.5$	94.78	18.75	78.97	85.64	66.18	**92.63**
$\ell_2 = 1.0$	94.74	02.22	66.79	76.34	59.09	**90.01**
$\ell_2 = 1.5$	94.10	00.31	50.97	61.64	46.52	**82.38**
$\ell_2 = 2.0$	69.50	00.02	36.24	45.22	33.35	**72.99**
$\ell_2 = 2.5$	10.38	00.01	25.19	28.56	20.47	**63.46**

Attack using different losses. Although attackers generally greedily update based on the margin (of logits) loss [5, 40, 58], it is possible for them to choose other loss options such as minimizing the probability margin and maximizing the cross-entropy loss. The results in Table. 8 show that despite the choice of AAA to reverse the margin loss, it could prevent attacks using other loss types.

The plug-in advantage of AAA. AAA, as a plug-in post-processing defense, is embeddable into any defense that increases the model's robustness. As shown in Table 8, AAA dramatically decreases the ECE of AT models without impacting the accuracy. Moreover, the already good defense performance of AT models is further boosted by AAA.

Table 8: The defense performance under Square attack [5] (#query $= 100/2500$)

	CIFAR-10 ($\ell_\infty = 8/255$) WideResNet34 [34]		ImageNet ($\ell_\infty = 4/255$) WideResNet50 [34]	
Metric / Loss	AT [22]	AT-AAA-linear	AT [59]	AT-AAA-linear
ECE	18.96	**5.93**	5.03	**2.64**
Acc	**91.47**	**91.47**	**66.30**	**66.30**
logits-margin	83.22 / 69.67	**84.68 / 82.92**	59.20 / 51.12	**60.83 / 59.73**
probability-margin	82.90 / 69.38	**84.41 / 82.67**	59.21 / 50.59	**60.33 / 57.88**
cross-entropy	83.93 / 71.17	**84.55 / 82.57**	60.13 / 52.84	**60.53 / 58.01**

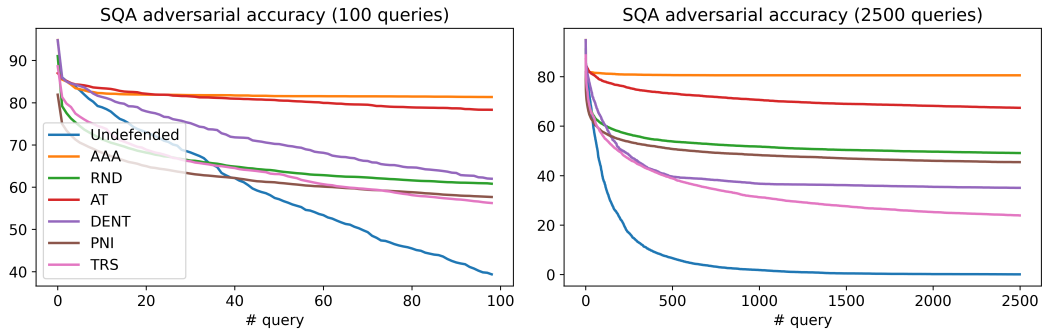

Figure 5: The adversarial accuracy under Square attack [5] when $\ell_\infty = 8/255$, indicating that AAA outperforms alternatives in defending SQAs by a large margin with also the highest clean accuracy.

B More visual illustrations

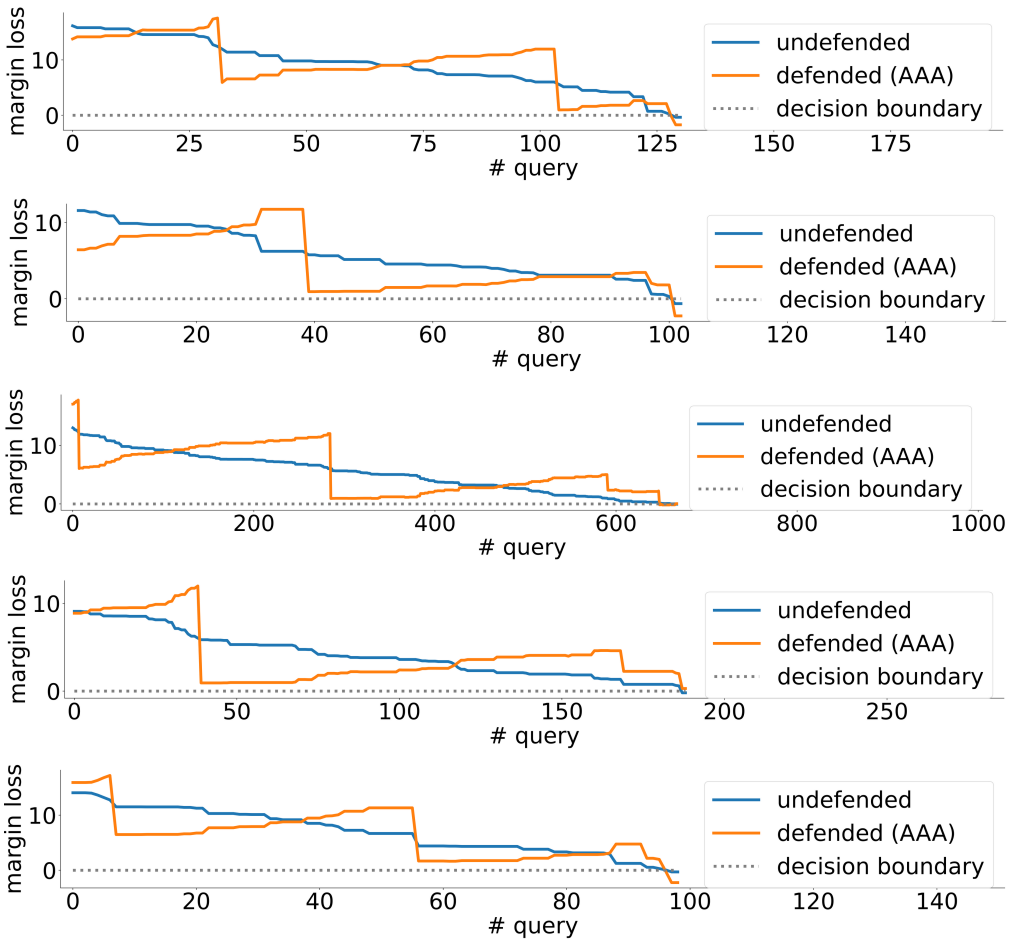

Figure 6: The AAA-defended and undefended margin loss value when attacking the undefended WideResNet-28 [34] in RobustBench [17] by Square attack [5] ($\ell_\infty = 8/255$) using the $1^{\text{rd}}, 2^{\text{nd}}, 5^{\text{th}}, 7^{\text{th}}, 8^{\text{th}}$ CIFAR-10 test sample.

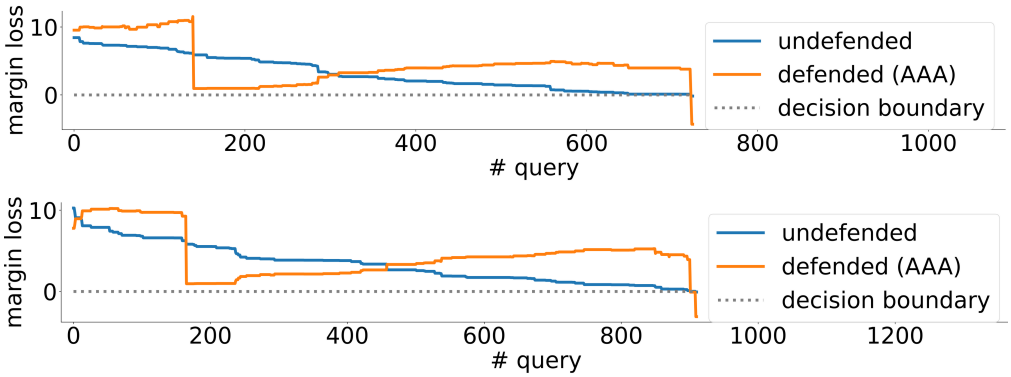

Figure 7: The AAA-defended and undefended margin loss value when attacking the undefended WideResNet-50 [34] in RobustBench [17] by Square attack [5] ($\ell_\infty = 4/255$) using the $2^{\text{nd}}, 7^{\text{th}}$ ImageNet test sample.

C Error analysis

Results of multiple runs. We run the main experiments in Table 2 for 5 times using different random seeds for Square attack, and report the results in Table 9. Since AAA is a deterministic method without randomness, its defense performance is very constant.

Average query times. Attackers generally use the average query times (AQ) to measure attacks, which is reported in Table 9. Here we record the AQ of all query samples to reflect the real attack cost. AAA hurdles the attack very much, seeing the large AQ and adversarial accuracy.

Computation. We report the FLOPs of each model in Table 2. Since the only calculation of AAA is to post-process logits, the computational overhead is negligible (<0.01 GFLOPs). The total amount of required calculation could be obtained by multiplying FLOPs with AQ using 10K samples.

Table 9: The defense performance under Square attack [5]

Dataset Model	CIFAR-10 WideResNet-28 [34]	ImageNet WideResNet-50 [34]	ImageNet ResNeXt-101 [64]
Acc (%)	94.84	77.17	78.32
ECE (%)	2.46	4.30	7.38
FLOPs (G)	5.24	11.43	16.48
Bound (ℓ_∞)	$8/255$	$4/255$	$4/255$
AdvAcc-100	81.77 ± 0.24	63.77 ± 0.40	66.63 ± 0.48
AdvAcc-2500	81.00 ± 0.25	62.89 ± 0.27	66.13 ± 0.44
AQ-100	87.00 ± 0.26	84.71 ± 0.51	86.76 ± 0.52
AQ-2500	2138.12 ± 6.39	2050.67 ± 5.62	2121.86 ± 14.48

D Core code

We present the core part of AAA python code (PyTorch) below, where a_i stands for the attractor interval τ in (4), reverse_step is α in (5), and calibration_loss_weight is β in (6).

```
logits = cnn(x_curr)
logits_ori = logits.detach()
p_target = F.softmax(logits_ori / temperature, dim=1).max(1)[0]

value, index_ori = torch.topk(logits_ori, k=2, dim=1)
margin_ori = value[:, 0] - value[:, 1]
attractor = ((margin_ori / a_i).ceil() - 0.5) * a_i
l_target = attractor - reverse_step * (margin_ori - attractor)

mask1 = torch.zeros(logits.shape, device=device)
mask1[torch.arange(logits.shape[0]), index_ori[:, 0]] = 1
with torch.enable_grad():
 logits.requires_grad = True
 optimizer = torch.optim.Adam([logits], lr=optimizer_lr)

 for i in range(num_iter):
  prob = F.softmax(logits, dim=1)
  loss_c = ((prob * mask1).max(1)[0] - p_target).abs().mean()
  value, index = torch.topk(logits, k=2, dim=1)
  margin = value[:, 0] - value[:, 1]
  loss_d = (margin - l_target).abs().mean()
  loss = loss_d + loss_c * calibration_loss_weight
  optimizer.zero_grad(); loss.backward(); optimizer.step()
```

E Detailed experimental settings

Table 10: The used models

Defense	Dataset	Architecture	Source	ID
None	CIFAR-10	WideResNet-28	RobustBench	Standard
PSSiLU (AT)	CIFAR-10	WideResNet-28	RobustBench	Dai2021Parameterizing
HAT (AT)	CIFAR-10	WideResNet-34	RobustBench	Rade2021Helper_extra
PNI (AT)	CIFAR-10	ResNet-20	Official	PNI-W (channel-wise)
TRS	CIFAR-10	ResNet-20	Official	/
None	ImageNet	WideResNet-50	TorchVision	wide_resnet50_2
AT	ImageNet	WideResNet-50	RobustBench	Salman2020Do_50_2
None	ImageNet	ResNeXt-101	TorchVision	resnext101_32x8d
FD (AT)	ImageNet	ResNeXt-101	Official	ResNeXt101_DenoiseAll

Defenses. The detailed information of all our used models is shown in Table 10. The official repositories of PNI, TRS, and FD are `https://github.com/elliothe/CVPR_2019_PNI`, `https://github.com/AI-secure/Transferability-Reduced-Smooth-Ensemble`, and `https://github.com/facebookresearch/ImageNet-Adversarial-Training`, respectively. AAA, RND, and DENT are directly implemented on the undefended model. RND adds the random noise with variance 0.02 to input samples as recommended in [14]. In DENT, we follow the original work to optimize the model for 6 iterations using the tent loss and Adam optimizer (lr$= 0.001$). The pre-trained AT/PNI model comes from RobustBench / official repository. We train the TRS model (ensemble 3 models) using the default coeff, lambda, and scale in the official code.

Attacks. All the attacks are adapted from the official repositories with original hyper-parameters. SimBA and Bandit are implemented from `https://github.com/cg563/simple-blackbox-attack` and `https://github.com/MadryLab/blackbox-bandits`, respectively. SignHunter and NES are both from `https://github.com/ash-aldujaili/blackbox-adv-examples-signhunter`. Square and QueryNet are both from the implementation in `https://github.com/AllenChen1998/QueryNet`. The detailed hyper-parameters of attacks are outlined in Table 11.

Table 11: Hyper-parameters for other attacks

Method	Hyperparameter	CIFAR-10	ImageNet
SimBA [4]	d (dimensionality of 2D frequency space)	32	32
	order (order of coordinate selection)	random	random
	ϵ (step size per iteration)	0.2	0.2
SignHunter [40]	δ (finite difference probe)	8 ([0,255])	0.05 ([0,1])
NES [58]	δ (finite difference probe)	2.55	0.1
	η (image l_p learning rate)	2	0.02
	q (# finite difference estimations / step)	20	100
Bandit [3]	δ (finite difference probe)	0.1	0.1
	η (image l_p learning rate)	0.01	0.01
	τ (online convex optimization learning rate)	0.01	0.01
	Tile size (data-dependent prior)	50	50
	ζ (bandit exploration)	1.0	1.0
Square [5]	p (initial probability to change coordinate)	0.05	0.05
QueryNet [62]	Number of batches (NAS training)	500	/
	batch size (NAS training)	128	/
	Number of layers (NAS surrogate models)	6, 8, 10	/