Meta-Learning the Search Distribution of Black-Box Random Search Based Adversarial Attacks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Adversarial attacks based on randomized search schemes have obtained state of the art results in black-box robustness evaluation recently. However, the efficiency of random search methods in different query budget regimes depends on manual design and heuristic tuning of the underlying proposal distributions. We propose a controller that can be plugged into random search methods and can adapt the proposal distribution online to the respective data and model under attack. We train the controller with gradient-based meta-learning end-to-end on models with white-box access. Once trained, learned controllers can be applied to unseen data and models with only black-box access. We show the effectiveness of our procedure for the specific case of Square Attack for the ℓ_∞-threat model. On unseen models, we achieve consistent improvements of up to 6% in robust accuracy with respect to existing hand-designed search distributions without any manual fine-tuning.

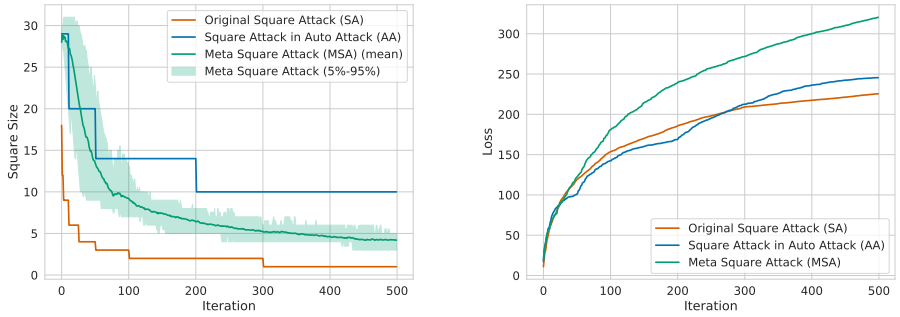
1 Introduction

It was demonstrated that despite their impressive performance in various tasks, neural networks are susceptible to small imperceptible perturbations in the input called *adversarial examples* [4]. This is a concerning issue for real-world deployment of deep learning approaches, especially in safety-critical domains such as autonomous driving [5]. But apart from practical concerns, adversarial examples serve as a tool to better understand the true nature of artificial neural networks and capture their inherent properties and differences from their biological counterparts [6].

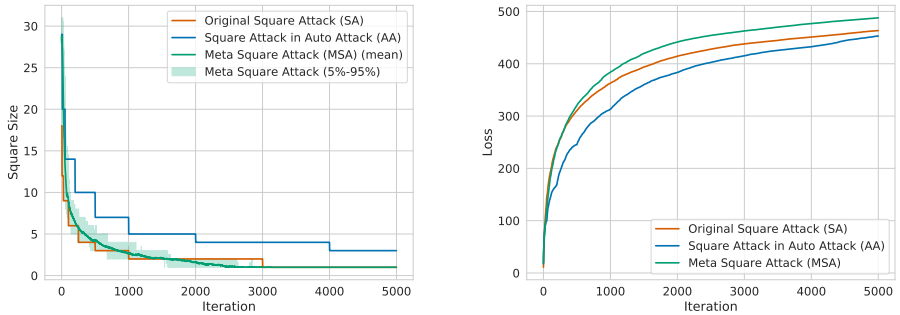
Evaluating adversarial robustness is usually formulated as a constrained optimization problem [4]. However, finding the exact solution is typically intractable [7, 8]. Therefore, in practice one often resorts to approximate methods called *adversarial attacks* that try to find adversarial examples within a small number of iterations. A number of different adversarial attacks were proposed [8–10] that can be distinguished into white- and black-box attacks. In white-box attacks, one assumes full access to the model architecture and weights [8]. However, as white-box attack typically rely on the gradient, gradient obfuscation, which does not eliminate the existence of adversarial examples but makes it significantly harder to find them gradient-based, can be a severe obstacle for certain attacks [11]. Therefore, black-box attacks that use only limited information from the model such as the class scores or the final decision often provide an additional perspective on the true robustness of the model [12–21] and are thus recommended for reliable robustness evaluation [2, 22].

A very promising direction in the field of black-box adversarial attacks are randomized search schemes for crafting adversarial examples [1, 23, 24]. Combining random search with specific update proposal distributions allows to achieve state-of-the-art black-box efficiency for different threat models such as ℓ_∞ and ℓ_2 [1], ℓ_1 [25], ℓ_0, adversarial patches, and adversarial frames [24]. Despite the conceptual simplicity of these methods, the main disadvantage of random search based

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) $T = 500$ iterations



(b) $T = 5000$ iterations

Figure 1: (Left) Schedules for Square Attack (SA) proposed by Andriushchenko et al. [1] (which differ for different query budgets) and Croce and Hein [2] (AA) compared to meta-learned Square Attack (MSA) proposed in this work. MSA meta-learns controllers for the square size schedule and the color distribution. The controllers adapt the schedule during the course of the attack on each particular image instead of using image-agnostic fixed schedules. Therefore we illustrate the mean square size and the $5-95$ percentile range for a given set of attacked images. (Right) Resulting loss of the different schedules for the model by Ding et al. [3] for 100 CIFAR10 images. MSA outperforms SA and AA significantly in terms of the achieved loss, see Table 1 for the robust accuracy.

methods is that the construction of a suitable proposal distribution requires significant manual design and is crucial for competitive performance (see also Figure 1).

In this work, we propose a method that allows to circumvent the fine-tuning and reduce the amount of manual design in random search based attacks. We use gradient-based meta-learning to automatically optimize controllers for schedules and proposal distribution on models with white-box access. After meta-training, the controllers can be plugged into a random search attack substituting manually designed schedules and proposal distributions. Importantly, once trained, the controllers do not require any gradient access and can thus be used in a fully black-box setting and without being affected by gradient obfuscation.

We consider the proposed methodology for the case of Square Attack for the ℓ_∞ threat model [1]. We meta-train controllers for square size and color on an adversarially trained [26] ResNet18 [27] model with white-box access and apply them to several different models from the RobustBench model zoo [28] that we treat as black-boxes. Since query efficiency is of crucial importance in black-box adversarial attacks, we also study the method for different query regimes ranging from several hundreds to several thousands. Depending on the query regime and the attacked model we obtain up to 6% improvement with respect to the baseline schedules proposed by Andriushchenko et al. [1] and Croce and Hein [2]. In short, we make the following contributions:

- We establish a theoretical framework for meta-learning the search distribution of random search based adversarial attacks (Section 3).

- We implement and study the proposed framework for the Square Attack with the ℓ_∞ threat model [1]. We test our Meta-learned Square Attack on CIFAR10 on a subset of models from the RobustBench model zoo [28] in different query budget regimes. We achieve consistent improvements of up to 6% in robust accuracy with respect to the hand-designed search distributions of Andriushchenko et al. [1] (Table 1) without the need for any manual fine-tuning.

2 Related Work

Black-box adversarial attacks: Existing approaches to black-box robustness evaluation can be divided into several groups based on their mode of interaction with the target model. Transfer-based attacks [17, 29] rely on having a model with white-box access similar to the target one. If this is the case, one can generate adversarial perturbations for this substitute model and transfer them in a black-box setting. A downside of such approaches is that similarity between the models is a crucial factor. Without it the efficiency of transfer attacks is drastically reduced [29].

Decision-based attacks [13, 20, 30] applied to classifiers assume that one can submit queries to the model and receive the predicted class label. Score-based attacks [1, 12–19] consider broader access to the model and assume the attack receives predicted scores for all the classes. Prior work [15, 31, 32] use this information to estimate a gradient and apply gradient descent. Alzantot et al. [19] propose a derivative-free approach based on genetic algorithms. Several works use empirical observation that successful ℓ_∞ perturbations are positioned at the corners of the color cube, therefore instead of continuous optimization one can reduce the problem to a discrete one [1, 12, 24, 33]. Huang et al. [21] formulate search over structured space as a time-varying contextual bandit problem and solve it using Bayesian optimization. Depending on the goal of the attacker one can also consider targeted black-box attacks that aim not only to change the true label to some other, but try to change it into a specified one [34].

Meta-learning and adversarial robustness: The closely related fields of meta-learning [35] and learning to optimize [36] have been employed in the field of adversarial robustness. The idea of learned optimizer [37, 38] was applied to finding adversarial examples in white-box [39] and black-box [40] settings. Meta-learning [41, 42] was also used to improve zeroth-order gradient estimation and allow better query efficiency of black-box attacks [43]. Besides that there is the recent work [44] on automating the existing AutoAttack [2] framework for robustness evaluation. Meta-learning has also been used as part of adversarial training [39, 45] for increasing adversarial robustness.

Square Attack: Square Attack (SA) was proposed by Andriushchenko et al. [1] and combines classical random search with a heuristic design of the update rule. This design depends on the geometry of the perturbation set and therefore differs for ℓ_∞ and ℓ_2 attacks and was later extended also to ℓ_1 [25]. As the adversarial attack problem is highly non-convex, a good initialization can significantly improve query efficiency. SA uses a stripe initialization motivated by empirical findings [46]. The design principle of the attack is based on the observation that the strongest perturbations are usually found on the boundary of the feasible set [18]. For the ℓ_∞ case, updates are sampled as squares parametrized by square size, square color, and position of the square. Performance of SA depends heavily on how color, position and square size are chosen, which requires manual design.

The square size schedules employed by previous work are relatively sophisticated (indicating non-trivial manual design): Andriushchenko et al. [1] proposed a schedule parametrized by $p^0 \in [0, 1]$ (the fraction of image pixels to be modified by a square in the first query) and the total query budget T, where p^t is halved at $\{0.1, 0.5, 2, 10, 20, 40, 60, 80\}\%$ of the total query budget. For evaluation on CIFAR10, Andriushchenko et al. [1] suggest different values of p^0 and report $p^0 = 0.3$ as a default choice. Croce and Hein [2] proposed a different schedule for SA to be used in AutoAttack: they use $p^0 = 0.8$ and $T = 5000$ but fix the halving points of p^t as if $T = 10000$. An illustration of the two schedules for $T = 500$ and $T = 5000$ can be seen in Figure 1.

The second aspect that characterizes SA is the distribution from which position and color of the next square are sampled. For positions, the distribution is uniform over all positions for which a square of a given size would be fully contained in the input. For colors, SA always generates points on on the boundary of the perturbation set (corners of the color cube) and uses a uniform distribution over the 2^c different colors (with c being number of channels, typically $c = 3$).

As discussed above, SA either relies on extensive manual design (for the square size schedules) or resorts to simple baseline choices if manual design is not conducted (such as the uniform distributions over colors and positions). Both is undesirable since it corresponds either to high effort or suboptimal performance. In this work, we show that meta-learning SA can achieve strong performance (see Table 1) with very little manual design and identify non-trivial patterns that increase attack efficiency (see Section 4.3).

3 Method

In the following, we briefly introduce the formulation of the optimization problem for an adversarial attack, rephrase it in the from of a meta-learning problem, and introduce our method for learning the search distribution of a specific random search based black-box adversarial attack, namely Square Attack [1]. We denote this method as meta-learned Square Attack (MSA).

3.1 Adversarial Robustness Evaluation

Let $f : [0,1]^d \mapsto \mathcal{P}(Y)$ be a classifier which maps a d-dimensional input x to $\mathcal{P}(Y) \in [0,1]^K$, which denotes the set of probability distributions over the K discrete possible outcomes Y (the classes). For a label $y \in Y$, a loss function $l : Y \times \mathcal{P}(Y) \mapsto \mathbb{R}$, a perturbation set S, and operator a, we define the robustness evaluation problem as

$$V(f, x, y) = \max_{\xi \in S} l(f(a(x, \xi)), y) \tag{1}$$

For $S = \{\xi \mid ||\xi||_p \leq \epsilon\}$ and $a(x, \xi) = \max(\min(x + \xi, 1), 0)$, one obtains the standard ℓ_p ball threat model. Assuming that loss function l and operator a are fixed, we denote $L(f, x, y, \xi) := l(f(a(x, \xi)), y)$ as a functional that one needs to maximize in robustness evaluation.

Since exact maximization of Equation 1 is intractable in the general case [7], we consider a (potentially non-deterministic) procedure $\mathcal{A}_\omega(L, f, x, y)$ called adversarial attack. This attack $\mathcal{A}_\omega$ is parametrized by hyperparameters ω and designed with the intention that $\xi^\omega \sim \mathcal{A}_\omega(L, f, x, y)$ with $\xi^\omega \in S$ becomes an approximate solution (tight lower bound) of $V(f, x, y)$, that is $V(f, x, y) - L(f, x, y, \xi^\omega)$ should become small (in expectation). By optimizing the hyperparameters of the attack, one can achieve a tighter lower bound of $V(f, x, y)$ via $\max_\omega \mathbb{E}_{\xi^\omega \sim \mathcal{A}_\omega(L,f,x,y)} L(f, x, y, \xi^\omega)$. Unfortunately, this maximization is still intractable typically, for instance when f is a black-box (no gradient information is available), the number of queries to f per data (x, y) is limited, or $\mathcal{A}_\omega$ has high variance.

3.2 Black-box Adversarial Robustness Evaluation as a Meta-learning Problem

We now frame adversarial robustness evaluation in the query-restricted black-box setting as a *meta-learning* problem. We follow the taxonomy proposed in the recent survey on meta-learning by Hospedales et al. [35].

First, we formulate our **meta-objective** (the specification of the goal of meta-learning): we assume classifiers to be evaluated are governed by a distribution $f \sim \mathcal{F}$ and data by a distribution $(x, y) \sim \mathcal{D}$. Our meta-objective is to find parameters ω^* of the attack $\mathcal{A}_\omega$ that maximize the lower bound $L(f, x, y, \xi^\omega)$ of $V(f, x, y)$ in expectation across models $f \sim \mathcal{F}$, data $(x, y) \sim \mathcal{D}$, and the stochastic attack $\xi^\omega \sim \mathcal{A}_\omega$:

$$\omega^* = \underset{\omega}{\operatorname{argmax}} \mathop{\mathbb{E}}_{f \sim \mathcal{F}} \mathop{\mathbb{E}}_{(x,y) \sim \mathcal{D}} \mathop{\mathbb{E}}_{\xi^\omega \sim \mathcal{A}_\omega(L,f,x,y)} L(f, x, y, \xi^\omega) \tag{2}$$

Here, the expensive optimization of ω^* is amortized across models and data. More specifically, we assume finite sets of data $D = \{(x_i, y_i)_{i=1}^N \mid (x_i, y_i) \sim \mathcal{D}\}$ and classifiers $F = \{f_j \mid f_j \sim \mathcal{F}\}$ are available. Moreover, we assume the $f_j \in \mathcal{F}$ allow white-box access and quasi-unrestricted number of queries. The sets D and F can be used during *meta-training* of the attack, that is the objective during meta-training becomes $\omega^* = \arg\max_\omega R(F, D, \omega)$ with $R(F, D, \omega) = \sum_{f \in F} \sum_{(x,y) \in D} L(f, x, y, \xi^\omega)$ and $\xi^\omega \sim \mathcal{A}_\omega(L, f, x, y)$. However, the ultimate goal of meta-learning is to apply $\mathcal{A}_{\omega^*}$ during *meta-testing* to unseen $(x, y) \sim \mathcal{D}$ and unseen $f \sim \mathcal{F}$ that allow only

black-box and query-limited access, and maximize $\mathbb{E}_{\xi^\omega \sim \mathcal{A}_\omega(L,f,x,y)} L(f, x, y, \xi^\omega)$, that is: the attack needs to generalize across models and data.

Next, we define the **meta-representation**, that is how the adversarial attack $\mathcal{A}_\omega$ is designed and parametrized such that generalization across $f \sim \mathcal{F}$ is effective. In this work we focus on random search based adversarial attacks for black-box robustness evaluation since they have achieved strong results in prior work and are amenable to meta-learning.

Let $\mathcal{A}_\omega$ be a random search based attack with a query budget limited by T. Then an adversarial perturbation $\xi^\omega = \xi^T \sim \mathcal{A}_\omega(L, f, x, y)$ is obtained using the following iterative procedure:

$$\xi^0 \sim \mathcal{D}^0; \quad \xi^{t+1} = \underset{\xi \in \{\xi^t, \Pi_\mathcal{S}(\xi^t + \delta^{t+1})\}}{\operatorname{argmax}} L(f, x, y, \xi); \quad \delta^{t+1} \sim \mathcal{D}_\omega(t, \xi^0, \delta^0, \ldots, \xi^t, \delta^t), \tag{3}$$

where $\Pi_\mathcal{S}$ corresponds to the projection onto the perturbation set $\mathcal{S}$.

That is, we assume a fixed distribution $\mathcal{D}^0$ for initializing the perturbation ξ^0 but a meta-learnable $\mathcal{D}_\omega$ for the update proposals δ^{t+1}. Importantly, $\mathcal{D}_\omega$ depends on the entire attack trajectory up to step t. Since this trajectory contains implicitly information on the classifier f when applied to data (x, y), our meta-learned random search attack can adapt to the classifier and data at hand. We provide more details on $\mathcal{D}_\omega$ for the specific case of Square attack [1] in Section 3.3.

The **meta-optimizer** (how we optimize the meta-objective) in our case assumes that both the loss function l and $\mathcal{A}_\omega$ are (stochastic) differentiable with respect to ω or we can find differentiable relaxations (as we will discuss in Section 3.3). Thus the meta-parameters ω can be optimized using stochastic gradient descent on mini-batches $B \subseteq D$ based on the (stochastic) gradient

$$g = \nabla_\omega R(F, D, \omega) = \sum_{f_j \in F} \sum_{(x_i, y_i) \in B \subseteq D} \nabla_\omega L(f_j, x_i, y_i, \xi_{i,j}), \tag{4}$$

where $\xi_{i,j} \sim \mathcal{A}_\omega(L, f_j, x_i, y_i)$. However, due to the stochasticity of $\mathcal{A}_\omega$ induced by $\mathcal{D}^0$ and $\mathcal{D}_\omega$, the discrete argmin in the update step of $\mathcal{A}_\omega$, and the length T of the unrolled optimization (often in the order of hundreds to thousands queries), g would have very high variance and typically one would also face issues with vanishing or exploding gradients. To address this, we propose using a greedy alternative instead:

$$g = \frac{1}{T} \sum_{f_i} \sum_{(x_j, y_j)} \sum_{t=1}^{T-1} \nabla_\omega L(f_i, x_j, y_j, \Pi_\mathcal{S}(\xi^t + \delta^{t+1})). \tag{5}$$

Importantly, even though ξ^t depends on ω for $t > 0$, we do not propagate gradients with respect to ω through ξ^t, that is we set $\nabla_\omega \xi^t := 1$. By this, the gradient corresponds to optimizing $\mathcal{D}_\omega$ in a myopic way, such that proposals $\delta^{t+1} \sim \mathcal{D}_\omega(t, \xi^0, \delta^0, \ldots, \xi^t, \delta^t)$ are trained to maximally increase the immediate loss in step $t + 1$. While this introduces a bias of acting myopic and greedy, it works reasonably well in practice.

We can now rewrite $\nabla_\omega L(f_i, x_j, y_j, \xi^t + \delta^{t+1}) = \nabla_{\delta^{t+1}} L(f_i, x_j, y_j, \xi^t + \delta^{t+1}) \nabla_\omega \delta^{t+1}$. We note that $\nabla_\omega \delta^{t+1}$ can be a very high variance estimate of $\nabla_\omega \mathcal{D}_\omega(t, \xi^0, \delta^0, \ldots, \xi^t, \delta^t)$ depending on the stochasticity of $\mathcal{D}_\omega$. However, as it is an unbiased estimate and we average over very many steps T and models and data, g is still a sufficiently good estimate in practice.

3.3 Meta-learned Square Attack

In this section, we demonstrate how the proposed meta-learning approach can be applied to Square Attack (SA) [1] with ℓ_∞ threat model. We denote the resulting meta-learned attack as *Meta-learned Square Attack* (MSA). We keep $\mathcal{D}^0$ as the stripe initialization from SA and focus on meta-learning $\mathcal{D}_\omega$ as it governs all but the first step. To recapitulate (see also Section 2), sampling $\delta^{t+1} \sim \mathcal{D}(t)$ in SA proceeds by computing a square size (its width in pixels) $s_t = \pi^s(t) \in \{1, \ldots, s_{max}\}$ and sampling a position $(p_x, p_y) \sim \pi^p(s) \in \{1, \ldots, s_{max} - s\}^2$ and a color $c \sim \pi^c \in \{c_1, \ldots c_m\}$. In SA, π^s is a heuristic schedule that depends on t (and differs in prior work [1, 2]) and both π^p and π^c are uniform distributions. δ^{t+1} is then chosen to be zero everywhere except for a square of size s at position p with color c. Possible colors c_i correspond to the eight corners of the RGB hypercube with ℓ_∞ norm 2ϵ (this choice guarantees that δ^{t+1} "overwrites" ξ^t within the square after projection

on $\mathcal{S}$). That is $c_i \in (\pm 2\epsilon, \pm 2\epsilon, \pm 2\epsilon)$, while s_{max} is chosen maximally with the constraint that all sampled squares must not exceed the image dimensions. We keep π^p as uniform distribution, but meta-learn the controllers $\pi^s_{\omega_s}$ and $\pi^c_{\omega_c}$ with parameters $\omega = (\omega_s, \omega_c)$.

Square size controller $\pi^s_{\omega_s}$. We design $\pi^s_{\omega_s}$ as a multi-layer perceptron (MLP) with parameters ω_s. The MLP outputs a scalar value $s' \in \mathbb{R}$ and we map this value to the actual square size via $s = \sigma(s') \cdot (s_{max} - 1) + 1$ with $\sigma(x) = 1/(1 + e^{-x})$. During meta-testing, we round the continuous $s \in [1, s_{max}]$ to a discrete value $\lfloor s \rceil \in \{1, \ldots, s_{max}\}$. As this would block gradient flow during meta-training, we relax the square sampling in SA such that it supports continuous square sizes. This relaxed sampling consists of sampling a standard square of size $\mathrm{odd}(s)^2$ with color c_i and $\mathrm{odd}(s) = 2 \cdot \lfloor \frac{s-1}{2} \rfloor + 1$ being the largest odd integer smaller than s. Thereupon, pixels in δ^t in the 4-neighborhood of the square are filled by $\frac{s - \mathrm{odd}(s)}{2} c_i$ and by $\left(\frac{s - \mathrm{odd}(s)}{2}\right)^2 c_i$ in the 8-neighborhood pixels that are not in 4-neighborhood (see illustration in appendix). Importantly, this kind of relaxed sampling is only conducted during meta-training and not in the final evaluation during meta-testing.

We provide two scalar inputs to the MLP $\pi^s_{\omega_s}$: (a) information regarding the current step size t encoded as $\log_2(\frac{t}{5000} + 1)$. This encoding ensures that the input stays in the range $[0, 1]$ for $t \leq 5000$. (b) Let $r^{t+1} = H\left(L(f, x, y, \Pi_{\mathcal{S}}(\xi^t + \delta^{t+1})) - L(f, x, y, \xi^t)\right) \in \{0, 1\}$ be an indicator of whether adding δ^{t+1} at time $t+1$ improved the loss (for H being the Heaviside step-function). The MLP gets as second input at time t the value $R^t = \gamma R^{t-1} + (1 - \gamma) r^t / r^0$ with $R^0 = 1$. Here γ is a decay term that controls how quickly past experience is "forgotten" and $r^0 = 0.25$ is a constant whose purpose is to ensure that the MLP's second input has a similar scale as the first (namely in $[0, 1]$). Intuitively, (a) allows to schedule square sizes based on time step of the attack (this information is also used in SA itself) while (b) allows adapting the square size based on the recent success frequency R^t of proposals (for instance reducing the square size if few proposals were successful recently). Thus (b) allows the schedule to adapt to classifier f and data (x, y) at hand.

Color controller $\pi^c_{\omega_c}$. We design the color controller $\pi^c_{\omega_c}$ as a categorical distribution $c^t \sim \mathrm{Cat}(\alpha^t_1, \ldots, \alpha^t_m)$, where each $\alpha^t_i \in \mathbb{R}$ is predicted by an MLP with weights ω_c. The m MLPs for the α^t_i share weights ω_c but differ in their inputs. In order to differentiate through the categorical distribution at meta-train time, we reparametrize the categorical distribution with Gumbel-softmax and draw discrete (hard) samples in the forward pass but treat them as soft samples in the backward pass [47, 48]. Additionally, we ensure that every color c_i is sampled at least with probability p^c_{min} by sampling with probability $m \cdot p^c_{min}$ from a uniform distribution rather than from the learned categorical distribution. This ensures continuous exploration of all m colors.

The m MLPs get two inputs: (a) information regarding the current step size t encoded as $\log_2(\frac{t}{5000} + 1)$ (same encoding as for the step size controller and also same for all m MLPs). (b) Information regarding the recent success frequency of proposals δ^t based on squares of the respective colors ($c^t = c_i$): $R^t_i = \begin{cases} \gamma R^{t-1}_i + (1 - \gamma) r^t / r^0 & \text{if } c^t = c_i \\ R^{t-1}_i & \text{otherwise} \end{cases}$. This second input allows the controller to learn, e.g., to sample colors more often which resulted in higher success frequency recently.

4 Experiments

We perform an empirical evaluation of Meta-learned Square Attack (MSA). We consider the data distribution $\mathcal{D}$ of CIFAR10 [49] images and classifier distribution $\mathcal{F}$ consisting of robust classifiers for the ℓ_∞-threat model as approximated by the RobustBench model zoo [28]. We compare performance of MSA in 4 different query budget regimes with manually designed schedules for Square Attack proposed by Andriushchenko et al. [1] (denoted by SA) and Croce and Hein [2] (denoted by AA). Section 4.2 summarizes results and Section 4.3 analyzes the behavior learned by the controllers.

4.1 Meta-Training and Controller Design

We meta-train the controller on a single robust model[1] $f \sim \mathcal{F}$ with white-box access (the "source model"). The source model was designed such that attackers could easily and cheaply acquire it

[1] Meta-training on more than one source model could improve generalization across models, but we found that even meta-training on a single model generalizes sufficiently well.

Table 1: Results for ℓ_∞-Square Attack with $\epsilon = 8/255$ on CIFAR10 on 1000 test images for 6 models from RobustBench. We compare the Square Attack (SA) schedule [1], the modified schedule from AutoAttack (AA) [2], and our meta-learned square-size schedule MSA_s. Moreover, uniform (Uni) color sampling and our learned color distribution MSA_c are compared. We report mean and standard error of robust accuracy for different queries budgets across 5 runs with different random seeds. The clean and robust accuracy of the models is taken from `https://robustbench.github.io/`.

Model	Accuracy (%)		Square	Color	Query budget			
	Clean	Robust	size		500	1000	2500	5000
Wong et al. [53]	83.34	43.21	SA	Uni	69.7±0.15	63.5±0.10	55.1±0.04	50.8±0.08
			AA	Uni	69.5±0.21	63.9±0.10	57.4±0.07	53.6±0.06
			MSA_s	Uni	**63.9±0.11**	59.8±0.10	54.0±0.16	51.1±0.08
			MSA_s	MSA_c	**63.9±0.12**	**59.1±0.09**	**53.0±0.16**	**49.8±0.08**
Ding et al. [3]	84.36	41.44	SA	Uni	68.7±0.20	63.2±0.28	57.8±0.13	54.9±0.17
			AA	Uni	66.6±0.18	62.2±0.14	57.5±0.12	55.0±0.20
			MSA_s	Uni	62.4±0.15	59.4±0.09	56.1±0.10	54.6±0.06
			MSA_s	MSA_c	**62.2±0.14**	**59.1±0.16**	**55.9±0.15**	**54.1±0.15**
Engstrom et al. [54]	87.03	49.25	SA	Uni	72.8±0.19	67.4±0.21	59.9±0.17	56.3±0.07
			AA	Uni	71.9±0.1	67.9±0.14	61.6±0.12	58.0±0.06
			MSA_s	Uni	67.9±0.12	64.2±0.15	58.9±0.05	56.4±0.12
			MSA_s	MSA_c	**67.8±0.12**	**63.4±0.18**	**58.2±0.06**	**55.9±0.04**
Gowal et al. [55]	89.48	62.76	SA	Uni	80.6±0.09	76.7±0.06	70.8±0.14	67.5±0.07
			AA	Uni	80.0±0.17	76.8±0.11	72.2±0.1	69.2±0.08
			MSA_s	Uni	**76.9±0.05**	73.7±0.05	69.8±0.13	67.6±0.04
			MSA_s	MSA_c	**76.9±0.07**	**73.4±0.13**	**69.0±0.08**	**67.2±0.04**
Carmon et al. [56]	89.69	59.53	SA	Uni	79.0±0.15	76.0±0.14	68.2±0.07	**65.4±0.09**
			AA	Uni	78.0±0.10	74.5±0.11	69.6±0.04	67.1±0.05
			MSA_s	Uni	**74.4±0.09**	70.8±0.06	67.5±0.07	65.6±0.07
			MSA_s	MSA_c	74.6±0.10	**70.3±0.07**	**67.0±0.07**	**65.4±0.08**
Huang et al. [57]	83.48	53.34	SA	Uni	72.3±0.1	66.6±0.16	60.5±0.09	57.3±0.08
			AA	Uni	70.6±0.10	66.5±0.07	61.2±0.10	58.5±0.11
			MSA_s	Uni	66.4±0.12	63.4±0.08	59.2±0.07	57.4±0.15
			MSA_s	MSA_c	**66.1±0.10**	**62.9±0.15**	**58.7±0.05**	**56.8±0.08**

themselves: the model has ResNet18 [27] architecture and was trained on the CIFAR10 training set using adversarial training [26] with the advertorch [50] package. Adversarial training was done using ℓ_∞-PGD attack with $\epsilon = 8/255$, fixed step size of 0.01, and 20 steps. We used 200 epochs for training and Adam [51] optimizer with learning rate 0.001.

For both square size and color controllers, we use MLP architectures with 2 hidden layers, 10 neurons each, and ReLU activations. We purposefully did not finetune the MLP architecture. Meta-training was run on a set D consisting of 1000 images from CIFAR10 test set (different from the ones used in evaluation of controllers in the next subsection) and Square Attack with a query budget of 1000 iterations. Therefore, controller behaviour on query regimes higher than 1000 are obtained by extrapolation of the behaviour learned for 1000 iterations. Both controllers were trained simultaneously for 10 epochs using Adam optimizer with batch size 100 and cosine step size schedule [52] with learning rate 0.03. The total loss improvement over the attack was used as meta-loss that we optimized in meta-training. We always run the attack on all images for the full budget T, even after the model was fooled on some images, since removing images from the attacked batch would cause discontinuity in the meta-loss. All computations including meta-training and evaluation of the controllers were performed on a single Nvidia Tesla V100-32GB GPU.

4.2 Evaluation on RobustBench models

We perform an evaluation on 1000 CIFAR10 test images and a subset of 6 robust models from the RobustBench model zoo [28]. We do not evaluate on the whole CIFAR10 test set because part of it was used during meta-training the controllers.

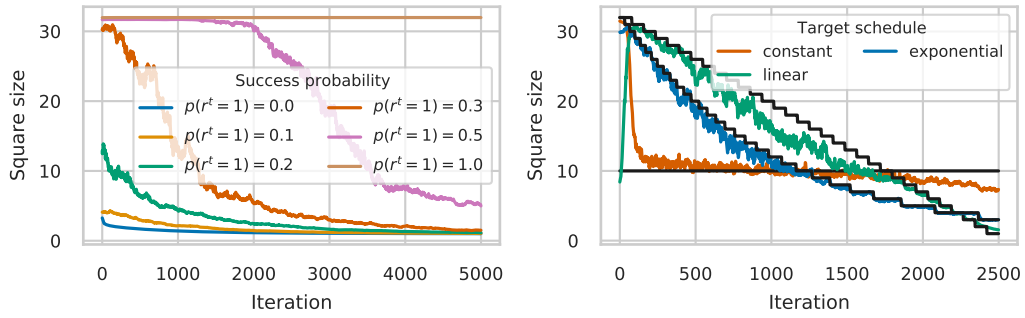


Figure 2: Illustration of the learned square-size controller MSA_s (averaged over 25 runs). (Left) Square-size schedule over time for fixed success probability $p(r^t = 1)$. (Right) Observed square-size schedules compared to different idealized target schedules (unknown to the controller) with $p(r^t = 1) = 0.1$ for squares size above the target schedule (black line) and $p(r^t = 1) = 0.4$ otherwise.

The results provided in Table 1 indicate that using our meta-learned controllers allows to consistently improve the obtained robust accuracy estimate. In particular, we observe that both the full controller MSA_s+MSA_c and its ablated version MSA_s+Uni improve over the baseline AA+Uni in all considered settings. Improvement over both SA and AA baselines is more significant for small query regimes (500, 1000). Here we see consistent improvement from both controllers by several percent points. In a big query regime of 5000, which corresponds to an extrapolation regime since MSA was only meta-trained for 1000 iteration, we see that SA slightly outperforms the schedule-only controller MSA_s+Uni but remains dominated by MSA_s+MSA_c in all cases except [56] where they achieve equal performance (note that 5000 is the maximal query budget in AutoAttack [2]).

As for the comparison between MSA_s+MSA_c and MSA_s+Uni, we observe that MSA_c provides consistent improvement over Uni of around 0.5 percent points for all query budgets except 500. For a query budget of 500, both controllers show close performance with MSA_s+Uni outperforming MSA_s+MSA_c in one case. This shows that the effect of color controller is more consistent in larger query regimes.

4.3 Analysis of Learned Controllers

As the learned controllers are black-boxes (implemented by MLPs), it is non-trivial to understand their realized strategy. We present some analysis of the controllers' internal strategy based on their empirical behavior.

Figure 2 illustrates the behaviour of the the square-size controller MSA_s. The left plot shows how the controller chooses the step-size over time if the success probability $p(r^t = 1)$ is constant. As implemented also by the heuristic schedules SA and AA, square size decays over time. However, the decay pattern depends heavily on the success probability $p(r^t = 1)$, with slower decay for larger $p(r^t = 1)$. This property is not implemented by heuristic schedules, but makes sense intuitively: high $p(r^t = 1)$ corresponds to a situation where the current perturbation ξ^t can be improved relatively easily by coarse-grained changes implemented by the current square sizes; in this case it makes sense to first get the coarse-grained structure "right", before proceeding to fine-grained details that can be captured by small squares.

The right plot illustrates observed schedules for idealized (and untypical) target schedules: these target schedules are unknown to the controller and are encoded in the success probabilities by setting $p(r^t = 1) = 0.4$ for square sizes smaller or equal to the value of the target schedules and to $p(r^t = 1) = 0.1$ otherwise. This abrupt change of the success probabilities and the shape of the target schedules "constant" and "linear" are very unlike the behavior of the attacks during meta-training; nevertheless the empirical schedules by the controller follow the target behavior reasonably good, indicating that the learned square-size controller generalizes well.

Figure 3 illustrates the empirical behavior of the color controller MSA_c when attacking the model by Ding et. al. [3]. Shown are histograms over 500 images for the frequency of specific colors being

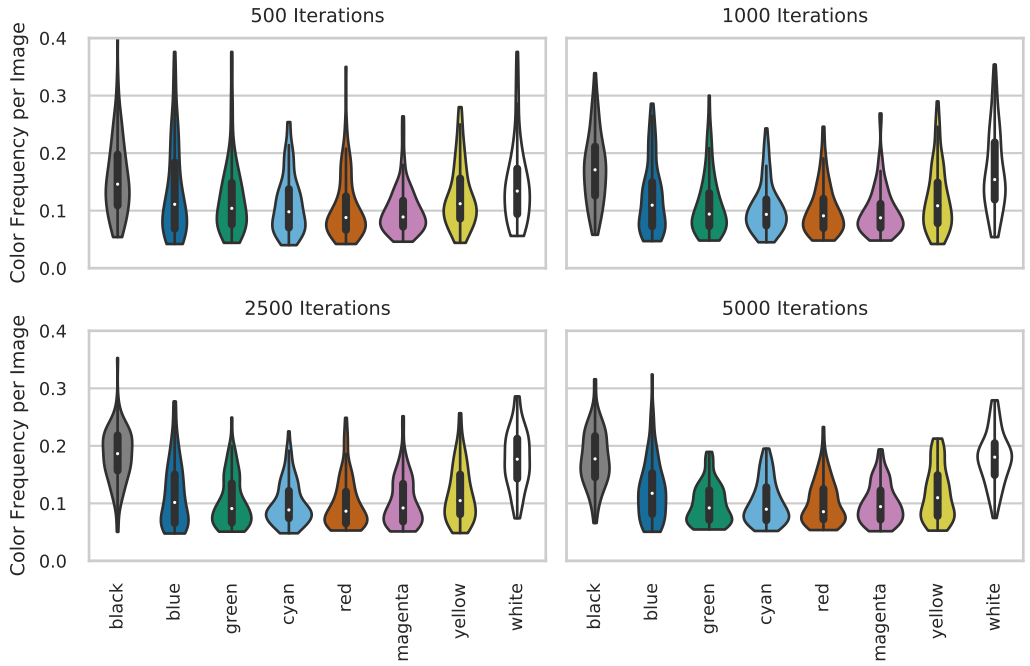


Figure 3: Illustration of color frequency histogram of 500 images after 500, 1000, 2500, and 5000 iterations of MSA_c. A clear preference towards sampling black and white more often than other colors becomes apparent after 1000 iterations.

sampled up to the respective iteration. Prior work like SA and AA maintained a uniform distribution of colors. However, the learned controller shows a clear preference for sampling black and white more often than uniform ($p \approx 0.18$ vs. $p = 0.125$ for uniform), blue and yellow approximately with $p = 0.125$, and the other colors less often than uniform. Since the color controller depends on the success rates R^t of colors, this behavior is not hard-coded into the controller but identified on-the-fly during the attack (so behavior can differ for models with different vulnerabilities). As evidenced by the results in Table 1, this adaptive behavior of the color controller improves attack efficiency on all tested models.

5 Conclusion

In this work we propose a theoretical framework for meta-learning search distributions that help to improve efficiency of random search based black-box adversarial attacks. We implement and investigate this framework for Square Attack with ℓ_∞-perturbations. Our experimental results support the hypothesis that learned adaptive controllers allow to improve the attack performance and make it more competitive for different query budgets. Future directions are applying our framework to other random-search based attacks and threat models as well as learning controllers for sampling positions or even geometric primitives (going beyond squares).

Ethical and Societal Impact This work contributes to the field of adversarial attacks, which can be used for benign purposes like reliably evaluating the robustness of ML systems as well as malign ones such as identifying and exploiting weaknesses of these systems. Increasing the query-efficiency and success rate of black-box attacks can amplify safety and security concerns in domains such as highly automated driving or robotics. Meta-learned square attack itself allows more efficient generation of imperceptible image distortions that may allow bypassing automated (black-box) content-control systems for filtering images with violent, pornographic, or otherwise offensive content. We are optimistic that the research on adversarial attacks contributes to developing defenses and mitigation strategies that outweigh these risks in the medium term.

References

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *ECCV*, 2020.

[2] F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

[3] Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. MMA training: Direct input space margin maximization through adversarial training. In *ICLR*, 2020.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[5] Anurag Ranjan, Joel Janai, Andreas Geiger, and Michael J. Black. Attacking optical flow. In *Proceedings International Conference on Computer Vision (ICCV)*, pages 2404–2413. IEEE, November 2019. URL `https://flowattack.is.tue.mpg.de/`. ISSN: 2380-7504.

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

[7] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *CAV*, 2017.

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

[9] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[10] Anirban Chakraborty, M. Alam, Vishal Dey, A. Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv: 1810.00069*, 2018.

[11] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

[12] A. Al-Dujaili and U.-M. O'Reilly. There are no bit parts for sign bits in black-box attacks. In *ICLR*, 2020.

[13] J. Chen, M. I Jordan, and Wainwright M. J. HopSkipJumpAttack: a query-efficient decision-based attack. *arXiv preprint arXiv:1904.02144*, 2019.

[14] S. N. Shukla, A. K. Sahu, D. Willmott, and Z. Kolter. Black-box adversarial attacks with Bayesian optimization. *arXiv preprint arXiv:1909.13857*, 2019.

[15] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. *ICML*, 2018.

[16] Z. Yan, Y. Guo, and C. Zhang. Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks. In *NeurIPS*, 2019.

[17] S. Cheng, Y. Dong, T. Pang, H. Su, and J. Zhu. Improving black-box adversarial attacks with a transfer-based prior. In *NeurIPS*, 2019.

[18] M. Seungyong, A. Gaon, and O. S. Hyun. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In *ICML*, 2019.

[19] M. Alzantot, Y. Sharma, S. Chakraborty, and M. Srivastava. Genattack: practical black-box attacks with gradient-free optimization. *Genetic and Evolutionary Computation Conference (GECCO)*, 2019.

[20] Huichen Li, Xiaojun Xu, Xiaolu Zhang, Shuang Yang, and Bo Li. Qeba: Query-efficient boundary-based blackbox attack. In *CVPR*, 2020.

[21] Zhichao Huang, Yaowei Huang, and Tong Zhang. Corrattack: Black-box adversarial attack with structured search, 2020.

[22] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *arXiv:2002.08347*, 2020.

[23] C. Guo, J. R Gardner, Y. You, A. G. Wilson, and K. Q. Weinberger. Simple black-box adversarial attacks. In *ICML*, 2019.

[24] Francesco Croce, Maksym Andriushchenko, Naman D. Singh, Nicolas Flammarion, and Matthias Hein. Sparse-rs: a versatile framework for query-efficient sparse black-box adversarial attacks. *arXiv:2006.12834*, 2020.

[25] Francesco Croce and Matthias Hein. Mind the box: l_1-apgd for sparse adversarial attacks on image classifiers. *arXiv preprint arXiv:2103.01208*, 2021.

[26] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Valdu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[27] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[28] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark, 2020.

[29] Zhichao Huang and Tong Zhang. Black-box adversarial attack with transferable model-based embedding. In *ICLR*, 2020.

[30] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*, 2018.

[31] J. Uesato, B. O'Donoghue, A. Van den Oord, and P. Kohli. Adversarial risk and the dangers of evaluating against weak attacks. In *ICML*, 2018.

[32] A. N. Bhagoji, W. He, B. Li, and D. Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *ECCV*, 2018.

[33] L. Meunier, J. Atif, and O. Teytaud. Yet another but more efficient black-box adversarial attack: tiling and evolution strategies. *arXiv preprint, arXiv:1910.02244*, 2019.

[34] Giuseppe Ughi, Vinayak Abrol, and Jared Tanner. An empirical study of derivative-free-optimization algorithms for targeted black-box attacks in deep neural networks, 2020.

[35] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv: 2004.05439*, 2020.

[36] Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark. *arXiv: 2103.12828*, 2021.

[37] Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, D. Pfau, Tom Schaul, and N. D. Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, 2016.

[38] Yutian Chen, Matthew W. Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matt Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. In *ICML*, 2017.

[39] Yuanhao Xiong and Cho-Jui Hsieh. Improved adversarial training via learned optimizer. In *ECCV*, 2020.

[40] Yangjun Ruan, Yuanhao Xiong, Sashank Reddi, Sanjiv Kumar, and Cho-Jui Hsieh. Learning to learn by zeroth-order oracle. In *ICLR*, 2020.

[41] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[42] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018.

[43] Jiawei Du, Hu Zhang, Joey Tianyi Zhou, Yi Yang, and Jiashi Feng. Query-efficient meta attack to deep neural networks. In *ICLR*, 2020.

[44] Chengyuan Yao, Pavol Bielik, Petar Tsankov, and Martin Vechev. Automated discovery of adaptive attacks on adversarial defenses. *arXiv:2102.11860*, 2021.

[45] Jan Hendrik Metzen, Nicole Finnie, and Robin Hutmacher. Meta adversarial training. *CoRR*, abs/2101.11453, 2021. URL `https://arxiv.org/abs/2101.11453`.

[46] D Yin, R. G. Lopes, J. Shlens, E. D Cubuk, and J. Gilmer. A Fourier perspective on model robustness in computer vision. In *NeurIPS*, 2019.

[47] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparametrization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

[48] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*, 2017.

[49] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.

[50] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[51] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

[52] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=Skq89Scxx`.

[53] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020.

[54] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. URL `https://github.com/MadryLab/robustness`.

[55] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv: 2010.03593*, 2021.

[56] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, and John C. Duchi. Unlabeled data improves adversarial robustness. In *NeurIPS*, 2019.

[57] Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. In *NeurIPS*, 2020.

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] Main limitations of our work are that we assume the availability of source models and data from the target distribution (see Section 3). Moreover, we discuss that the learned controllers are black-boxes and thus more difficult to interpret than manually designed controllers (see Section 4.3).

(c) Did you discuss any potential negative societal impacts of your work? [Yes] Please refer to "Ethical and Societal Impact" in Section 5.

(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] We do not provide theoretical results.

(b) Did you include complete proofs of all theoretical results? [N/A] We do not provide theoretical results.

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)? [N/A] We are clarifying currently if code along with instructions can be submitted as part of the supplementary material. We are planning an open source release of our code in the following months (subject to code clearance by our organization).

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.1

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Main results in Table 1 were obtained by running evaluation 5 times with 5 different random seeds. Means and standard errors are reported.

(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] As stated in 4.1, all computations including meta-training and evaluation of the controllers were performed on a single Nvidia Tesla V100-32GB GPU. We can not report the total amount of compute since we did not track it. However, our organization is carbon neutral so that all its activities including compute on the GPU clusters on which the experiments have been performed do no longer leave a carbon footprint.

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] Our work uses 6 pre-trained models from the model zoo provided by RobustBench [28] (`https://robustbench.github.io/`). We cite this source as well as authors of the methods used to train each model (see Table 1). Moreover, we use the CIFAR10-data and cite it.

(b) Did you mention the license of the assets? [Yes] For licensing information see supplementary material

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A] We are clarifying currently if source model can be submitted as part of the supplementary material (subject to code clearance by our organization).

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] As stated in supplementary material, in this work we only use the data published under formal licenses.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] As stated in supplementary material, to the best of our knowledge data used in this project does not contain any 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] No crowdsourcing was used and no research with human subjects was conducted

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] No crowdsourcing was used and no research with human subjects was conducted

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] No crowdsourcing was used and no research with human subjects was conducted