POLICY-DRIVEN ATTACK: LEARNING TO QUERY FOR HARD-LABEL BLACK-BOX ADVERSARIAL EXAMPLES

Anonymous authors
Paper under double-blind review

ABSTRACT

To craft black-box adversarial examples, adversaries need to query the victim model and take proper advantage of its feedback. Existing black-box attacks generally suffer from high query complexity, especially when only the top-1 decision (i.e., the hard-label prediction) of the victim model is available. In this paper, we propose a novel hard-label black-box attack named *Policy-Driven Attack*, to reduce the query complexity. Our core idea is to learn promising search directions of the adversarial examples using a well-designed policy network in a novel reinforcement learning formulation, in which the queries become more sensible. Experimental results demonstrate that our method can significantly reduce the query complexity in comparison with existing state-of-the-art hard-label black-box attacks on various image classification benchmark datasets. Anonymous code and models for reproducing our results are available in the supplementary material.

1 INTRODUCTION

It is widely known that deep neural networks (DNNs) are vulnerable to *adversarial examples*, which are crafted via perturbing clean examples to cause the victim model to make incorrect predictions. In a white-box setting where the adversaries have full access to the architecture and parameters of the victim model, gradients w.r.t. network inputs can be easily calculated via back-propagation, and thus first-order optimization techniques can be directly applied to craft adversarial examples in this setting (Szegedy et al., 2014; Goodfellow et al., 2015; Carlini & Wagner, 2017; Madry et al., 2018; Rony et al., 2019). However, in black-box settings, input gradients are no longer readily available since all model internals are kept secret.

Over the past few years, the community has made massive efforts in developing black-box attacks. In order to gain high attack success rates, delicate queries to the victim model are normally required. Existing methods can be roughly categorized into *score-based attacks* (Chen et al., 2017; Ilyas et al., 2018; Nitin Bhagoji et al., 2018; Ilyas et al., 2019; Yan et al., 2019; Guo et al., 2019; Tu et al., 2019; Du et al., 2019; Li et al., 2019; Bai et al., 2020) and *hard-label attacks* (a.k.a, *decision-based attacks*) (Brendel et al., 2018; Cheng et al., 2019; Dong et al., 2019; Shi et al., 2019; Brunner et al., 2019; Chen et al., 2020; Rahmati et al., 2020; Li et al., 2020; Shi et al., 2020), based on the amount of information exposed to the adversaries from the output of victim model. When the prediction probabilities of the victim model are accessible, an intelligent adversary would generally prefer score-based attacks, while in a more practical scenario where only the top-1 class prediction is available, the adversaries will have to resort to hard-label attacks. Since less information is exposed from such feedback of the victim model, hard-label attacks often bare higher query complexity than that of score-based attacks, making their attack process costly and time intensive.

In this paper, we aim at reducing the query complexity of hard-label black-box attacks. We cast the problem of progressively refining the candidate adversarial example (by skillfully querying the victim model and analyzing its feedback) into a reinforcement learning formulation. At each iteration, we searched along a set of chosen directions to see whether there existed any new candidate adversarial example that was perceptually more similar to its benign counterpart, i.e., in the sense of requiring less distortion. A reward was assigned to each of such search directions (treated as actions), based on the amount of distortion reduction yielded after updating the adversarial example along that direction. Such a reinforcement learning formulation enabled us to learn the non-differentiable mapping from search directions to their potential of refining the current adversarial example, directly and

precisely. The policy network was expected to be capable of providing the most promising search direction for updating candidate adversarial examples and reducing the required distortion of the adversarial examples from their benign counterparts. As we will show, the proposed policy network can learn from not only the queries that had been performed following the evolving policy but also peer experience from other black-box attacks. As such, it is possible to pre-train the policy network on a small number of query-reward pairs obtained from the performance log of prior attacks (with or without policy) to the same victim model. Experimental results showed that our policy-driven attack (PDA) can achieve significantly lower distortions than existing state-of-the-arts under the same query count budgets.

2 Related Work

In this paper, we focus on the hard-label black-box setting where only the top-1 decision of the victim model is available. Since less information (of the victim model) is exposed after each query, attacks in this category generally require more model queries than those in the white-box or score-based settings. For example, an initial attempt named boundary attack (Brendel et al., 2018) could require $\sim$million queries before convergence. It proposed to start from an image that is already adversarial, and tried to reduce the distortion by walking towards the benign image along the decision boundary. Recent methods in this category focused more on gradient estimation which could provide more promising search directions, while relying only on top-1 class predictions. Ilyas et al. (2018) advocated to use NES (Wierstra et al., 2014; Salimans et al., 2017) to estimate the gradients over proxy scores, and then mounted a variant of PGD attack (Madry et al., 2018) with the estimated gradients. Towards improving the efficiency of gradient estimation, Cheng et al. (2019) and Chen et al. (2020) further introduced a continuous optimization formulation and an unbiased gradient estimation with careful error control, respectively. The gradients were estimated via issuing probe queries from the standard Gaussian distribution. To generate probes from some more powerful distributions, Dong et al. (2019) proposed to use the covariance matrix adaptation evolution strategy, while Shi et al. (2020) suggested to use customized distribution to model the sensitivity of each pixel. In contrast to these methods, our PDA proposes to use a policy network which is learned from prior intentions to advocate promising search directions to reduce the query complexity.

We note that some attacks in the score-based setting also proposed to train DNN models for assisting the generation of black-box queries (Li et al., 2019; Du et al., 2019; Bai et al., 2020). Our method is naturally different from them in problem settings (score-based vs hard-label) and problem formulations. To the best of our knowledge, our work is the first to incorporate reinforcement learning into the black-box scenario for performing attacks, and we advocate the community to consider more about this principled formulation in the future. In addition to the novel reinforcement-based formulation, we also introduced a specific architecture for the policy network which enjoys superior generalization performance, while these methods adopted off-the-shelf auto-encoding architectures.

3 Our Policy-Driven Attack

We study the problem of attacking an image classifier in the hard-label setting. The goal of adversaries is to perturb an benign image $\boldsymbol{x} \in \mathbb{R}^n$ to fool a k-way victim classifier $f : \mathbb{R}^n \to \mathbb{R}^k$ into making an incorrect decision: $\arg\max_i f(\boldsymbol{x}')_i \neq y$, where $\boldsymbol{x}'$ is the adversarial example generated by perturbing the benign image and y is the true label of $\boldsymbol{x}$. The adversaries would generally prefer adversarial examples $\boldsymbol{x}'$ with smaller distortions $\|\boldsymbol{x} - \boldsymbol{x}'\|_2$ achieved using less queries, since these properties make the attack less suspicious and also save the cost. In this section, we first briefly review some backgrounds that motivate our method (in Section 3.1), and then detail our reinforcement learning formulation (in Section 3.2) and the architecture of our policy network (in Section 3.4).

3.1 Motivations

Most recent hard-label attacks followed a common pipeline of searching from a starting point which was already an adversarial image[1] yet not close enough to the benign one. Unlike the white-box and score-based black-box setting in which the input gradients can be calculated and used as the most effective perturbation direction, in the concerned hard-label setting, outputs of the victim model only

[1] In practice, it is performed by randomly sampling until the adversarial constraint is satisfied, i.e., it is not classified as y by the victim model, or by directly choosing from the adversarial class.

flip on the decision boundary while keeping constant away from the boundary, making it difficult to evaluate different directions almost everywhere. In this context, the search of promising perturbation directions were restricted into the regions near the decision boundary, since these regions are arguably more informative, and binary search was used to reach the decision boundary efficiently.

Let us take a very recent attack named HopSkipJumpAttack (Chen et al., 2020) as an example. Given the current estimation $\boldsymbol{x}'_s$ of the adversarial example at each iteration, HopSkipJumpAttack first performed binary search to project it onto the decision boundary of the victim model. Denote $\boldsymbol{x}'$ as the updated example that was on the decision boundary already, HopSkipJumpAttack then sampled many probes around $\boldsymbol{x}'$ from an isotropic Gaussian distribution, and issued the probes to the victim model as queries. The feedback of the victim model was utilized to estimate the gradient direction at $\boldsymbol{x}'$, and it was updated along this direction to obtain a new estimation of the adversarial example. It was repeated many times until the query budget was exhausted.

In comparison with boundary attack (Brendel et al., 2018), HopSkipJumpAttack was in general far more query-efficient, though a large number of queries had to be consumed for probing the local geometrics of the decision boundary of the victim model. Its superiority came from using the estimated gradient directions as the update directions, which motivated us to explore even better update directions at each iteration of the attack. As will be shown in Section 3.2, drawing on some geometric insights, we found that the gradient directions are in fact not the optimal search directions in the framework of HopSkipJumpAttack. We also found that the task of performing hard-label black-box attack could be naturally cast into a reinforcement learning task, thus we attempted to explore the possibility of developing a model-based method for predicting the most promising search directions for attacks. Feedback from the victim provides supervision and thus the models in our reinforcement learning framework can be trained/fine-tuned on the fly during each attack process, such that little query is required once the model has be well-trained.

3.2 Attack as Reinforcement Learning Problem

Algorithm 1 Policy-Driven Attack Algorithm

Input: the environment $\mathcal{E}(\boldsymbol{x}, y, f)$; the target label y', initial adversarial image $\boldsymbol{x}'_1 \in \mathbb{R}^n$ which lies on the decision boundary; the policy network g.
Output: an adversarial example $\boldsymbol{x}'$.
Initialize the step index $t \leftarrow 1$.
while the query count limit not reached **do**
    // Determine the baseline l_t to rank the distortion reduction of different actions
    $\boldsymbol{\mu}_t \leftarrow g(\boldsymbol{x}'_t, y, y'), \boldsymbol{z} \leftarrow \mathrm{BS}(\boldsymbol{x}'_t + \delta \cdot \frac{\boldsymbol{\mu}_t}{\|\boldsymbol{\mu}_t\|_2}, \boldsymbol{x}, f)$, where $\mathrm{BS}(\cdot,\cdot,\cdot)$ performs binary search
    Set the distortion reduction of $\boldsymbol{z}$ as baseline: $l_t \leftarrow \max\{\|\boldsymbol{x}'_t - \boldsymbol{x}\|_2 - \|\boldsymbol{z} - \boldsymbol{x}\|_2, l_{\min}\}$

    // Collect actions and rewards, and update the policy network
    Sample M actions: $\boldsymbol{a}_{t,i} \sim \mathcal{N}(\boldsymbol{\mu}_t, \sigma_t \boldsymbol{I}), \quad i \in \{1, 2, \ldots, M\}$
    Assign rewards $r_{t,i}$ to each actions with our mechanism
    Update the policy network using one-step REINFORCE on $(\boldsymbol{a}_{t,i}, r_{t,i})$

    // Update adversarial image using predicted direction
    $\boldsymbol{\mu}_t \leftarrow g(\boldsymbol{x}'_t, y, y')$ if $\|\boldsymbol{z} - \boldsymbol{x}\|_2 \leq \|\boldsymbol{x}'_t - \boldsymbol{x}\|_2$, otherwise $\boldsymbol{\mu}_t \leftarrow \boldsymbol{a}_{t,i^*}$, $i^* = \arg\max_i r_{t,i}$
    $\boldsymbol{x}'_{t+1} \leftarrow \mathrm{BS}(\boldsymbol{x}'_t + \epsilon \cdot \frac{\boldsymbol{\mu}_t}{\|\boldsymbol{\mu}_t\|_2}, \boldsymbol{x}, f)$

    // Update other variables
    Cut σ_t by half if all M rewards are zeros: $\sigma_{t+1} \leftarrow 0.5\sigma_t$; else keep it: $\sigma_{t+1} \leftarrow \sigma_t$
    $t \leftarrow t + 1$
end while
return final adversarial image $\boldsymbol{x}'_t$

In this paper, we consider both targeted attacks and untargeted attacks. Given a benign example $\boldsymbol{x}$, its label y, and the victim model f, an environment $\mathcal{E}(\boldsymbol{x}, y, f)$ is naturally formed. The adversary play the role of agent, trying to interact with the environment by issuing queries and collecting feedback, under some policy. The current example $\boldsymbol{x}'_t$ on the decision boundary of the victim model (or called the candidate adversarial example) represents the state at each timestamp t. The agent use

a learnable policy network g which will be carefully introduced in Section 3.4 to guide its actions, and the action is to update the candidate adversarial example such that less distortions are required to fool the victim model. The action here incorporates searching along a promising direction $\boldsymbol{a}_t/\|\boldsymbol{a}_t\|$ where $\boldsymbol{a}_t \in \mathbb{R}^n$ is sampled from an isotropic Gaussian distribution whose mean vector is given by the policy network $\boldsymbol{\mu}_t = g(\boldsymbol{x}'_t, y, y') \in \mathbb{R}^n$ and its covariance matrix is given by $\boldsymbol{\Sigma} = \sigma \boldsymbol{I} \in \mathbb{R}^{n \times n}$, in which the value of $\sigma \in \mathbb{R}$ is set to be gradually increased as the attack on each sample progresses and $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ indicates the $n \times n$ identity matrix. With $\boldsymbol{a}_t$, the agent searches along its direction $\boldsymbol{a}_t/\|\boldsymbol{a}_t\|$ to see whether any better candidate adversarial example can be found, and in this step, binary search shall also be performed in our PDA, just like in HopSkipJumpAttack (Chen et al., 2020). After all these steps, as will be carefully introduced in Section 3.3, a reward $r_t \in \mathbb{R}$ based on the performance of each action and the corresponding $\boldsymbol{a}_t$ is given to the agent for updating the parameter of the policy network. Figure 1 demonstrates the reward assignment mechanism of our method. For untargeted attacks, $\boldsymbol{x}'_t$ should be on the decision boundary where one side is the ground-truth label y and the other side could be regarded as the "target label". All details of our PDA are summarized in Algorithm 1.

Powered by the reinforcement learning framework, we can use policy gradient algorithms to learn the policy network g and to generate promising search directions in a direct way. For simplicity, we use the one-step REINFORCE (Williams, 1992) in the sequel of this paper and leave the exploration on more advanced policy gradient algorithms to future work.

3.3 Reward and Action

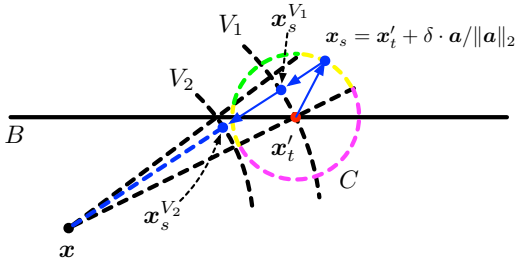

Figure 1: The reward assignment mechanism of our method. Arcs with color magenta, yellow, and green corresponded to actions with reward 0, 1, and 2, respectively.

Figure 1 illustrates how we assign the scalar reward r given a candidate adversarial example $\boldsymbol{x}'_t$ and an action $\boldsymbol{a}_t$. The decision boundary is illustrated by a horizontal straight line (denoted by B) in the figure, the benign counterpart $\boldsymbol{x}$ is assumed to be below B, and the circle C centered at $\boldsymbol{x}'_t$ with a small radius δ shows all possible locations after jumping along the directions of some actions by δ from $\boldsymbol{x}'_t$. As described earlier, the reward r should be assigned based on the amount of potential distortion reduction brought by $\boldsymbol{a}_t$, which is denoted by d_t hereinafter. A direct evaluation can be achieved by jumping along the direction of $\mathbf{a}_t$ first and then projecting the updated example back onto the decision boundary via binary search, to see how much improvement is obtained. However, since we evaluate multiple actions $\boldsymbol{a}_{t,i}$ simultaneously at an iteration (see Algorithm 1) and binary search needs to be performed for each of them, and the overall process would be prohibitively (query-)expensive. On this point, to efficiently assess the performance of an action $\boldsymbol{a}_{t,i}$, we instead evaluate whether the reduction of distortion by taking a particular action can exceed particular baselines. Concretely, we first evaluate $\boldsymbol{\mu}_t = g(\boldsymbol{x}'_t, y, y')$ as an action directly by using binary search in a way as just described. Suppose that it can reduce the required adversarial distortion by l_t, then we setup two levels of baselines $\|\boldsymbol{x} - \boldsymbol{x}'_t\|_2 - \beta_1 \cdot l_t$ and $\|\boldsymbol{x} - \boldsymbol{x}'_t\|_2 - \beta_2 \cdot l_t$ to see whether other actions can lead to adversarial examples with closer distance (than these baselines) from the benign example $\boldsymbol{x}$, in which $\beta_1 = 0$ and $\beta_2 = 0.25$. As shown in Figure 1, for an action $\boldsymbol{a} \in \{\boldsymbol{a}_{t_i}\}$ other that $\boldsymbol{\mu}_t$, we first obtain $\boldsymbol{x}'_t + \delta \cdot \boldsymbol{a}/\|\boldsymbol{a}\|_2$ and then move it towards $\boldsymbol{x}$ to see how much reward it can obtain. The two arcs V_1 and V_2 indicates where the same progress as the two baselines can be achieved, thus we can further project $\boldsymbol{x}'_t + \delta \cdot \boldsymbol{a}/\|\boldsymbol{a}\|_2$ onto the arcs to see if the projections (i.e, $\boldsymbol{x}_s^{V_1}$ and $\boldsymbol{x}_s^{V_2}$) are still adversarial. It can be seen that $\boldsymbol{x}_s^{V_1}$ is still adversarial yet $\boldsymbol{x}_s^{V_2}$ is not. We assign a reward 1 to such an action $\boldsymbol{a}$. If both the projections are still adversarial we shall assign a reward of 2, and if neither of them is adversarial, zero reward is assign. Such a way reduces the number of query for assessing each action to only 2 (just for $\boldsymbol{x}_s^{V_1}$ and $\boldsymbol{x}_s^{V_2}$) and makes our PDA more query-efficient.

3.4 ARCHITECTURE OF THE POLICY NETWORK

As described in Section 3.2, the goal of our policy network is to predict a direction based on which the optimal adversarial example/candidate can be easily found. Its input is the current example on the decision boundary of the victim model (together with other useful information that is available to the agent if needed) and its output is expected to be a promising search direction that shares the same dimension with its input. Naïve architecture designs of the policy network include the conventional auto-encoders and U-Net (Ronneberger et al., 2015). However, our experimental results suggest that such off-the-shelf auto-encoding architectures often offer degraded performance (see Section 4.3 for more details). We note that predicting an promising search direction is discrepant from the computer vision tasks for which these architectures are widely applied (e.g., predicting a segmentation map). Specifically, a segmentation map often aligns with the visual contents of the input image, while the promising search directions might be less correlated with the semantics of the input examples. We reckon it can be more beneficial to incorporate domain knowledge about adversarial attacks into the architecture of the policy network g.

On this point, we propose a new architecture for the policy network for our PDA. First, we know from HopSkipJumpAttack that the gradient direction at $\boldsymbol{x}'_t$, although not theoretically optimal, can provide strong empirical performance when serving as the update direction. Therefore, designing an architecture which could output the gradient vector at the point of input seems to be an appropriate option for the policy network g. Formally, it takes the candidate adversarial example $\boldsymbol{x}'_t$, the ground-truth label y of the benign example, and the target label y' (if a targeted attack is performed) as input, and mapped them to a search direction in $\mathbb{R}^n$. In this spirit, the policy network is designed to own an internal classifier $h : \mathbb{R}^n \to \mathbb{R}^k$, which performs a k-way classification. The number k can be the same as the number of prediction classes of the black-box victim model if the adversary has such information. We hope the internal classifier h can learn to distill knowledge from the victim model if possible, as such we can use the input gradient of h as a descent search direction. Following the logit-diff loss developed for the white-box setting in Carlini & Wagner (2017)'s work, we propose to use:

$$g(\boldsymbol{x}'_t, y, y') = \nabla_{\boldsymbol{x}'_t} h(\boldsymbol{x}'_t)_{y'} - \nabla_{\boldsymbol{x}'_t} h(\boldsymbol{x}'_t)_y + \boldsymbol{b}, \tag{1}$$

as the output of the policy network, where a learnable bias vector $\boldsymbol{b} \in \mathbb{R}^n$ is introduced to improved the capacity and flexibility of the network. The forward process of such a policy network is basically a back-propagation process of the internal classifier h, and if the decision boundary of the internal classifier h is aligned with that of the victim model, the output of the policy network should be the gradient direction of the victim model. Since the model is parameterized and has sufficient capacity, it can also learn to explore even better search directions other than the gradient directions.

3.5 (OPTIONAL) PRE-TRAINING OF THE POLICY NETWORK.

Note that the learning of the policy network can bare from large sampling complexity and may even fail to converge if its initial outputs are completely unable to cut the distortion. Just like for playing the game of go (Silver et al., 2016), we can optionally (pre-)train the policy network in a supervised manner to make the initial actions more reasonable in the reinforcement learning process, such that the samples collected in the follow-up steps are more informative and the issue can be relieved. In this section, we introduce how such pre-training can be performed for the concerned task. Recall that the goal of the policy network is to predict promising search directions for the candidate adversarial examples, thus we can construct a set $\mathbb{S}$ by collecting the intermediate results of any prior attacks to the victim model, or we can also use a simplified policy and collect its suggested actions for constructing $\mathbb{S}$ for pre-training.

Given a small dataset $\mathbb{D} = \{(\boldsymbol{x}, y)\}$ which consists of benign examples and their ground-truth labels, it is easier to first run our PDA without learning a policy network, i.e., using an input-independent policy for it. Concretely, we can instead sample each direction $\boldsymbol{a}''_t$ from the distribution $\mathcal{N}(\boldsymbol{b}, \sigma_t \boldsymbol{I})$ at the timestamp t, where $\boldsymbol{b}$ here is still learnable, just like in Eq. (1). We found that such a simplified policy tends to learn a search direction that is very similar to the gradient direction. This simplified reinforcement learning problem where the policy network g is now absent shows more stable training performance and each of its attack trajectories can be effectively used to pre-train the policy network, which is formulated as:

$$\mathbb{T}_{\boldsymbol{x}} \triangleq \{(\boldsymbol{x}''_t, \boldsymbol{x}, y, y''_t, \boldsymbol{a}''_t) \,|\, t \in \{1, 2, \ldots, m_{\boldsymbol{x}}\}\}, \tag{2}$$

where $m_{\boldsymbol{x}}$ is the total number of iterations, and $\boldsymbol{x}_t''$, y_t'', $\boldsymbol{a}_t''$ are the candidate adversarial example, the target label, and the suggested search direction by the simplified policy at each iteration, with a common timestamp t, respectively. To improve the compactness of the sample set for pre-training, we suggest a simple post-processing strategy to discard the tuples in $\mathbb{T}_{\boldsymbol{x}}$ with less informative candidate adversarial examples: for $i \geq 2$, we can discard the i-th tuple in $\mathbb{T}_{\boldsymbol{x}}$ if $\|\boldsymbol{x}_i'' - \boldsymbol{x}\|_2 > 0.99\|\boldsymbol{x}_j'' - \boldsymbol{x}\|_2$, where j indicates the index of any previous tuple that were decided not to be discarded. The set that contains the remaining tuples is denoted by $\mathbb{T}_{\boldsymbol{x}}^r$, and the final pre-training set $\mathbb{S}$ is constructed using this sort of set gathered from all attack trajectories using the simplified policy, i.e.,

$$\mathbb{S} = \bigcup_{(\boldsymbol{x},y)\in\mathbb{D}} \mathbb{T}_{\boldsymbol{x}}^r. \tag{3}$$

Then it is how to pre-train the policy network given $\mathbb{S}$ for better initialization. First, according to the design of g as introduced in Section 3.4, it is natural to encourage the internal classifier h to perform as a classifier, and then, probably more importantly, the output of the policy network are encouraged to somehow align with the considered effective search directions (i.e., $\boldsymbol{a}_t''$) found by the aforementioned simplified policy. On this point, we introduce the cosine similarity S together with a regularizer Ψ which incorporates the classification loss to achieve the two goals, making the pre-training loss as:

$$L = \frac{1}{|\mathbb{S}|} \sum_{(\boldsymbol{x}'',\boldsymbol{x},y,y'',\boldsymbol{a}'')\sim\mathbb{S}} -S\left(g(\boldsymbol{x}'', y, y''), \boldsymbol{a}''\right) + \lambda \cdot \Psi(h(\boldsymbol{x}), h(\boldsymbol{x}''), y, y''), \tag{4}$$

where $S(\cdot,\cdot)$ calculates the cosine similarity between its two input vectors, λ is the coefficient for regularization, and Ψ serves as a regularizer which is given by:

$$\Psi(h(\boldsymbol{x}), h(\boldsymbol{x}''), y, y'') = CE(h(\boldsymbol{x}), y) + \frac{1}{2}CE(h(\boldsymbol{x}''), y) + \frac{1}{2}CE(h(\boldsymbol{x}''), y''), \tag{5}$$

where $CE(\cdot,\cdot)$ calculates the cross-entropy loss given logits and labels. Since $\boldsymbol{x}''$ is on the decision boundary of the victim model f, f must assign 0.5 probability to both the benign class y and the adversarial class y'' at $\boldsymbol{x}''$, which interprets the terms $\frac{1}{2}CE(h(\boldsymbol{x}''), y)$ and $\frac{1}{2}CE(h(\boldsymbol{x}''), y'')$.

4 EXPERIMENTS

Table 1: Mean ℓ_2 distortions for performing untargeted attacks with different query budgets.

Dataset	Victim Model	Method	@100	@500	@1K	@5K	10K	@25K
MNIST	CNN	Boundary Attack	10.179	9.970	9.692	8.960	3.670	1.850
		HopSkipJumpAttack	7.267	3.576	2.709	1.843	1.721	1.641
		Ours	**2.307**	**2.138**	**2.097**	**1.842**	**1.717**	**1.603**
CIFAR-10	CNN	Boundary Attack	9.625	8.649	8.463	5.442	1.408	0.395
		HopSkipJumpAttack	4.567	1.520	0.911	0.378	0.304	**0.261**
		Ours	**0.640**	**0.546**	**0.507**	**0.358**	**0.298**	0.263
	WRN	Boundary Attack	9.268	8.095	7.972	3.639	0.664	0.287
		HopSkipJumpAttack	3.041	1.052	0.649	**0.268**	**0.213**	0.179
		Ours	**0.700**	**0.515**	**0.453**	0.273	0.217	**0.174**
	ResNet50 Adv.	Boundary Attack	10.672	10.273	10.217	9.064	4.998	2.378
		HopSkipJumpAttack	7.980	5.453	4.353	2.521	2.097	1.793
		Ours	**2.638**	**2.454**	**2.365**	**1.926**	**1.733**	**1.570**
ImageNet	ResNet-18	Boundary Attack						
		HopSkipJumpAttack	23.250	9.980	6.303	2.086	1.442	1.056
		Ours						

In this section, we evaluate the effectiveness of our method on three datasets: MNIST (LeCun et al., 2010), CIFAR-10 (Krizhevsky & Hinton, 2009), and ImageNet (Russakovsky et al., 2015). We compared our method with Boundary Attack (Brendel et al., 2018) and HopSkipJumpAttack (Chen et al., 2020) in both untargeted and targeted settings, and the ℓ_2 distortions required given a specific query budget are evaluated, as in recent related work (Brendel et al., 2018; Chen et al., 2020; Li et al., 2020). For comprehensive comparison, we report the distortions at {100, 500, 1K, 5K, 10K, 25K} query budgets in all experiments in the paper. The required mean distortions for untargeted and targeted attacks are reported in Table 1 and Table 2, respectively, and the median distortions are reported in the appendices. For targeted attack, we only show results on MNIST and CIFAR-10 which are faster to be evaluated. The target label is set to $y' = y + 1 \mod 10$ where y is the true label for the clean image. All experiments are conducted on NVIDIA GTX 2080 Ti GPUs with PyTorch (Paszke et al., 2017).

Table 2: Mean ℓ_2 distortions for performing targeted attacks with different query budgets.

Dataset	Victim Model	Method	@100	@500	@1K	@5K	10K	@25K
MNIST	CNN	Boundary Attack	5.559	5.486	5.480	5.417	4.087	2.638
		HopSkipJumpAttack	5.291	4.480	3.786	2.644	2.423	**2.268**
		Ours	**3.072**	**2.772**	**2.712**	**2.423**	**2.355**	2.314
CIFAR-10	CNN	Boundary Attack	11.194	10.075	9.381	7.933	2.778	0.750
		HopSkipJumpAttack	8.864	3.763	2.118	0.724	**0.549**	**0.448**
		Ours	**1.426**	**1.207**	**0.930**	**0.691**	0.589	0.481
	WRN	Boundary Attack						
		HopSkipJumpAttack						
		Ours						
	ResNet50 Adv.	Boundary Attack						
		HopSkipJumpAttack						
		Ours						

4.1 Experimental Setup

Victim models and h. For MNIST, we use the same architecture as in Carlini & Wagner (2017)'s work [2], which contains four convolutional layers and three fully connected layers and shows a test error rate of 0.59%, since it is widely adopted as a victim model in many related work. For CIFAR-10, we consider three victim models: (a) a CNN whose architecture is similar to the one considered on MNIST, which achieves a test error rate of 22.03% [3]; (b) a WRN-28-10 (Zagoruyko & Komodakis, 2016) with a test error rate of 4.03% [4]; (c) a ResNet-50 with a test error rate of 18.38% collected from the robustness package (Engstrom et al., 2019), which was adversarially trained under ℓ_2 PGD attacks ($\epsilon = 1.0$). On ImageNet, we adopt a ResNet-18 (He et al., 2016) from the PyTorch official model zoo as the victim model, which shows a top-1 error rate of 30.24% on the official validation set. As for the policy network g, we adopt a VGG-13 (Simonyan & Zisserman, 2015) architecture as its internal classifier h for attacking all these victim models, i.e., the architecture of g and h is by no means similar to that of any of the victim models.

Implementation details. We mostly test our PDA with pre-training of the policy network g, and its performance without pre-training will be presented in the appendices. When pre-training is to be performed, we should construct a training dataset $\mathbb{S}$ for it. On MNIST and CIFAR-10, we randomly sample 5,000 images from their official test set to construct $\mathbb{S}$, which are confirmed to be correctly classified by all the victim models. On ImageNet, we similarly gather 50,000 images from an auxiliary dataset called ImageNetV2 and the official ImageNet validation set to construct $\mathbb{S}$. For each gathered image $\mathbf{x}$, we sample $m_{\mathbf{x}} = 500$ actions at each iteration t (see Eq. (2)). Another 500 images are also collected to form the validation set for tuning all hyper-parameters for each of the three datasets (i.e., MNIST, CIFAR-10, and ImageNet). We choose the SGD optimizer without momentum, and we use a learning rate 0.003.

After a pre-trained policy network is obtained, the final performance of our PDA is evaluated based on a set of 1,000 clean images disjoint from the training/validation set described in the above paragraph (i.e., these 1,000 images are not used in training the policy network and tuning hyper-parameters). At this stage, we sample only $m_{\mathbf{x}} = 25$ actions at each iteration for faster convergence in all experiments. We use the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.0001 and set λ (in Eq. (4)) to be 0.003. To achieve better trade-offs between exploration and exploitation, we initialize σ in the sampling Gaussian distribution to be 0.003, and scale it at each iteration if necessary, to make sure that the ratio of the average output of the policy network and σ lies in the range of [0.01, 0.5]. σ is cut by half if all sampled actions at an iteration receive zero reward. The step size ϵ during attack was set as $0.4\|\boldsymbol{x} - \boldsymbol{x}_t'\|_2$, and the geometric regression strategy suggested by Chen et al. (2020) is also applied.

To make a fair comparison, we also sample 25 probs around each example on the decision boundary for HopSkipJumpAttack, which yield even better performance than its default setting which used 100 probs. Other hyper-parameters for performing boundary attack and HopSkipJumpAttack are kept as in their original papers. Following Brendel et al. (2018) and Chen et al. (2020), the starting adversarial examples are obtained by randomly sampling from a uniform distribution in the input space $[0, 1]^n$ until the adversarial criterion is met. Once generated, the starting points are shared among all the compared attacks.

[2] Pre-trained weights: `https://github.com/IBM/Autozoom-Attack`

[3] Pre-trained weights also from: `https://github.com/IBM/Autozoom-Attack`

[4] Pre-trained weights: `https://github.com/bearpaw/pytorch-classification`

4.2 COMPARISON WITH THE STATE-OF-THE-ARTS

Table 1 compares our PDA with the state-of-the-art methods for performing hard-label black-box attacks. We considered a threat model where a large number of adversarial examples are required to be generated in a hard-label black-box manner, such that the query for pre-training could be omitted. It can be easily seen that in general our method outperforms its competitors, especially in the earlier stage of attacks, which is of importance when lower query budget are permitted. In particular, with only 100 queries, our method leads to only one-sixth to one-third distortions when compared with HopSkipJumpAttack which is the second best. With a larger query budget of 500, HopSkipJumpAttack still leads to 1.5 to 3.0 times larger distortion than our method. More interestingly, on CIFAR-10, when attacking the ResNet-50 guarded with adversarial training, which is proved to be one of the most powerful defenses, the superiority of our PDA is in fact more significant. Such a observation is considered consistent with a phenomenon discovered in prior work (Yu et al., 2019; Zhang & Wang, 2019) which shows that adversarially trained models often have less sharp peaks and cliffs on the decision boundary, making traditional gradient estimation methods more difficult. Comparison of the state-of-the-arts with our PDA where no pre-training was performed is given in the appendices.

4.3 ABLATION STUDY

We perform ablation study on how the architecture design of the policy network would affect the performance of our PDA. More specifically, we train policy networks with several different architectures on the same set $\mathbb{S}$ and then attempt to attack the CNN model on CIFAR-10 based on these policy networks. In addition to our design (as introduced in Section 3.4) which leverages the gradient of a internal classifier h, we mostly consider the U-Net (Ronneberger et al., 2015) which is a popular option for the learning models for assisting adversarial attacks. We compare our proposed architecture with several U-Nets with different configurations. Details of their architectures can be found at `https://github.com/milesial/Pytorch-UNet`. For fair comparison, we use the same learning rate scheduler (as described in Section 4.1) for them. Note that, for U-Net models, the cross-entropy regularization is not applied, since they do not perform classification and there is no logit for computing such a loss. The performance of different policy networks in terms of the mean l_2 distortion is reported in Table 3. Obviously, we can see that the proposed architecture outperformed U-Net significantly in the framework of our PDA.

Table 3: Comparison of choosing different architectures for the policy network. Mean ℓ_2 distortions for performing untargeted attacks were evaluated.

Architecture	@100	@500	@1K	@5K	@10K	@25K
U-Net (small)	1.023	0.791	0.613	0.374	0.300	0.276
U-Net (medium)	0.871	0.772	0.587	0.362	**0.288**	0.273
U-Net (large)	0.882	0.741	0.589	0.365	0.308	**0.262**
Ours	**0.640**	**0.546**	**0.507**	**0.358**	0.298	0.263

5 CONCLUSION

Existing hard-label black-box attacks often suffer from very high query complexity. In this paper, we have introduced a model-based method (i.e., PDA) for learning from past queries and model feedback, based on a reinforcement learning formulation of the attack. We have developed a novel architecture for the policy network that is designed to suggest promising search directions for the optimal adversarial examples. Moreover, it has been demonstrated that such a policy network can be effectively pre-trained using prior attacking logs on the same victim model. Experimental results on various victim models (including both naturally and adversarially trained ones) trained on different datasets (including MNIST, CIFAR-10, and ImageNet) suggest that the proposed PDA significantly outperforms existing state-of-the-arts in terms of query efficiency. Although queries were also spent for pre-training the policy network (e.g., 125 million queries on MNIST and CIFAR-10 and 1.25 billion for ImageNet), these "initial investments" would benefit all later attacks, and thus could be diluted and being relatively cheap if the pre-trained policy network is used to attack many images.

REFERENCES

Yang Bai, Yuyuan Zeng, Yong Jiang, Yisen Wang, Shu-Tao Xia, and Weiwei Guo. Improving query efficiency of black-box adversarial attack. In *ECCV*, 2020.

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

Thomas Brunner, Frederik Diehl, Michael Truong Le, and Alois Knoll. Guessing smart: Biased sampling for efficient black-box adversarial attacks. In *ICCV*, pp. 4958–4966, 2019.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, 2017.

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

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

Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *ICLR*, 2019.

Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. Efficient decision-based black-box adversarial attacks on face recognition. In *CVPR*, pp. 7714–7722, 2019.

Jiawei Du, Hu Zhang, Joey Tianyi Zhou, Yi Yang, and Jiashi Feng. Query-efficient meta attack to deep neural networks. *arXiv preprint arXiv:1906.02398*, 2019.

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

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

Chuan Guo, Jacob R Gardner, Yurong You, Andrew G Wilson, and Kilian Q Weinberger. Simple black-box adversarial attacks. In *ICML*, 2019.

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

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *ICML*, 2018.

Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *ICLR*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Huichen Li, Xiaojun Xu, Xiaolu Zhang, Shuang Yang, and Bo Li. Qeba: Query-efficient boundary-based blackbox attack. In *CVPR*, pp. 1221–1230, 2020.

Yandong Li, Lijun Li, Liqiang Wang, Tong Zhang, and Boqing Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In *ICML*, pp. 3866–3876, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *ECCV*, 2018.

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

Ali Rahmati, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Huaiyu Dai. Geoda: a geometric framework for black-box adversarial attacks. In *CVPR*, pp. 8446–8455, 2020.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Jérôme Rony, Luiz G Hafemann, Luiz S Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *CVPR*, pp. 4322–4330, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Yucheng Shi, Siyu Wang, and Yahong Han. Curls & whey: Boosting black-box adversarial attacks. In *CVPR*, pp. 6519–6527, 2019.

Yucheng Shi, Yahong Han, and Qi Tian. Polishing decision-based adversarial noise with a customized sampling. In *CVPR*, pp. 1030–1038, 2020.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *AAAI*, 2019.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *JMLR*, 15(1):949–980, 2014.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Ziang Yan, Yiwen Guo, and Changshui Zhang. Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks. In *NIPS*, pp. 3820–3829, 2019.

Fuxun Yu, Zhuwei Qin, Chenchen Liu, Liang Zhao, Yanzhi Wang, and Xiang Chen. Interpreting and evaluating neural network robustness. In *IJCAI*, pp. 4199–4205. AAAI Press, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. In *NeurIPS*, pp. 1831–1841, 2019.

A MEDIAN DISTORTIONS

In this section we reported the median distortions over test images at different query count budgets under the untargeted setting in Table 4.

Table 4: Median distortion at different query budgets with ℓ_2 constraint under the untargeted setting.

Dataset	Victim Model	Method	@100	@500	@1K	@5K	10K	@25K
MNIST	CNN	Boundary Attack	10.108	9.946	9.662	9.098	3.604	1.825
		HopSkipJumpAttack	7.114	3.505	2.664	1.813	1.693	1.616
		Ours	**2.263**	**2.112**	**2.097**	**1.830**	**1.709**	**1.579**
CIFAR-10	CNN	Boundary Attack	8.825	7.985	7.970	5.000	1.189	0.332
		HopSkipJumpAttack	3.629	1.188	0.739	0.321	0.262	0.227
		Ours	**0.543**	**0.463**	**0.429**	**0.298**	**0.249**	**0.218**
	WRN	Boundary Attack	8.740	7.778	7.765	3.456	0.607	0.263
		HopSkipJumpAttack	2.638	0.927	0.589	0.248	0.199	0.168
		Ours	**0.578**	**0.448**	**0.400**	**0.244**	**0.195**	**0.158**
	ResNet50 Adv.	Boundary Attack	9.303	8.881	8.869	8.647	5.071	2.296
		HopSkipJumpAttack	7.569	5.439	4.362	2.424	1.972	1.672
		Ours	**2.545**	**2.396**	**2.301**	**1.806**	**1.624**	**1.464**
ImageNet	ResNet-18	Boundary Attack						
		HopSkipJumpAttack	22.371	7.547	4.113	1.385	1.016	0.796
		Ours						

B PERFORMANCE WITHOUT PRE-TRAINING

In this section we study the performance when pre-training is not applied. We only show results on MNIST with CNN as the victim model which is faster to be evaluated. In this case, our policy network does not own the internal classifier h, and instead it directly uses the bias $\boldsymbol{b}$ as the output. Table 5 shows the performance in terms of mean distortion under the untargeted setting.

Table 5: Mean ℓ_2 distortions for performing untargeted attacks with different query budgets. Our policy network is not pre-trained.

Dataset	Victim Model	Method	@100	@500	@1K	@5K	10K	@25K
MNIST	CNN	Boundary Attack	10.179	9.970	9.692	8.960	3.670	1.850
		HopSkipJumpAttack	7.267	3.576	2.709	1.843	1.721	1.641
		Ours (w.o. pre-training)						
		Ours	**2.307**	**2.138**	**2.097**	**1.842**	**1.717**	**1.603**

We see without pre-training our PDA achieves comparable performance with competitive methods, and as expected, the pre-training could boost the performance significantly.