LURING OF TRANSFERABLE ADVERSARIAL PERTURBATIONS IN THE BLACK-BOX PARADIGM

Anonymous authors
Paper under double-blind review

ABSTRACT

The growing interest for adversarial examples, i.e. maliciously modified examples which fool a classifier, has resulted in many defenses intended to detect them, render them inoffensive or make the model more robust against them. In this paper, we pave the way towards a new approach to improve the robustness of a model against black-box transfer attacks. A removable additional neural network is included in the target model, and is designed to induce the *luring effect*, which tricks the adversary into choosing false directions to fool the target model. Training the additional model is achieved thanks to a loss function acting on the logits sequence order. Our method only needs to have access to the predictions of the target model, and does not require a labeled dataset. We explain the luring effect thanks to the notion of robust and non-robust useful features and perform experiments on MNIST, SVHN and CIFAR10 to characterize and evaluate this phenomenon. Additionally, we discuss two simple prediction schemes, and verify experimentally that our approach can be used as a defense to efficiently thwart an adversary using state-of-the-art attacks and allowed to perform large perturbations.

1 INTRODUCTION

Neural networks based systems have been shown to be vulnerable to adversarial examples (Szegedy et al., 2014), i.e. maliciously modified inputs that fool a model at inference time. Many directions have been explored to explain and characterize this phenomenon (Schmidt et al., 2018; Ford et al., 2019; Ilyas et al., 2019; Shafahi et al., 2019) that became a growing concern and a major curb on the deployment of Machine Learning (ML) models. In response, many defenses have been proposed to protect the integrity of ML systems (Madry et al., 2018; Zhang et al., 2019; Cohen et al., 2019; Hendrycks et al., 2019; Carmon et al., 2019). While these existing approaches focus very predominantly on an adversary in the white-box setting, we present an innovative way to defend against realistic attacks, not sufficiently covered in the literature yet, namely transfer black-box attacks (Papernot et al., 2017). As ML-based online API are likely to become increasingly widespread, and regarding the massive deployment of edge models in a large variety of devices, several instances of a model may be deployed in systems with different environment and security properties. Thus, the black-box paradigm needs to be extensively studied to efficiently protect systems in many critical domains.

Considering a target model M that a defender aims at protecting against adversarial examples, we propose a method which allows to build the model T, an augmented version of M, such that adversarial examples do not transfer from T to M. Importantly, training T only requires to have access to M, meaning that no labeled data set is required, so that our approach can be implemented at a low cost for any already trained model. T is built by augmenting M with an additional component P (i.e. $T = M \circ P$) taking the form of a neural network trained with a specific loss function with logit-based constraints. From the observation that transferability of adversarial perturbations between two models occurs because they rely on similar non-robust features (Ilyas et al., 2019), we design P such that (1) the augmented network exploits useful features of M and that (2) non-robust features of T and M are either different or require different perturbations to reach misclassification towards the same class. Our deceiving method is conceptually new as it does not aim at making M relying more on robust-features as with proactive schemes (Madry et al., 2018; Zhang et al., 2019), nor tries to

anticipate perturbations which directly target the non-robust features of M as with reactive defense schemes (Meng & Chen, 2017; Hwang et al., 2019).

Our contributions are as follows:

- We present an innovative approach to thwart transferability between two models, which we name the luring effect.
- We propose an implementation of the luring effect which fits any pre-trained model and does not require a label data set. An additional neural network is preprended to the target model and trained with a specific loss function that acts on the logits sequence order.
- We experimentally characterize the luring effect and discuss its potentiality for black-box defense strategies.

For reproducibility purposes, the code is available at `https://anonymous.4open.science/r/3c64e745-927d-4f51-b187-583e64586ff6/`.

2 LURING ADVERSARIAL PERTURBATIONS

2.1 NOTATIONS

We consider a classification task where input-label pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ are sampled from a distribution $\mathcal{D}$. $|\mathcal{Y}| = C$ is the cardinality of the labels space. A neural network model $M_\phi : \mathcal{X} \to \mathcal{Y}$, with parameters ϕ, classifies an input $x \in \mathcal{X}$ to a label $M(x) \in \mathcal{Y}$. $\mathcal{L}\big(M_\phi(x), y\big)$ is the cross-entropy loss for M_ϕ with respect to an input $x \in \mathcal{X}$ and the ground-truth label $y \in \mathcal{Y}$. The pre-softmax output function of M_ϕ (the logits) is denoted as $\ell : \mathcal{X} \to \mathbb{R}^C$. For the sake of readability, the model M_ϕ is simply noted as M, except when necessary.

2.2 CONTEXT: ADVERSARIAL EXAMPLES IN THE BLACK-BOX SETTING

Black-box settings are realistic use-cases since many models are deployed (in the cloud or embedded in mobile devices) within secure environments and accessible through open or restrictive API. Contrary to the white-box paradigm where the capacity of an adversary allows the use of existing gradient-based attacks (Goodfellow et al., 2015; Carlini & Wagner, 2017; Chen et al., 2018; Dong et al., 2018; Madry et al., 2018; Wang et al., 2019), an attacker in a black-box setting only accesses the output label, confidence scores, or logits from the target model. He can still take advantage of gradient-free methods (Uesato et al., 2018; Guo et al., 2019; Su et al., 2019; Brendel et al., 2018; Ilyas et al., 2018; Chen et al., 2020). However, practically, the number of queries requires to mount the attack is prohibitive and may be flagged as suspicous (Chen et al., 2019; Li et al., 2020). In that case, the adversary may take advantage of the transferability property (Papernot et al., 2017) by crafting adversarial examples on a *substitute model* and then transfering them to the target model.

2.3 OBJECTIVES AND DESIGN

Our objective is to find a new way to make models more robust against transferable black-box adversarial perturbation without expensive (and sometimes prohibitive) training cost as it the case in adversarial training approaches. Our main idea is based on the classical deception-based approach for network security (e.g. honeypots) and can be summarized as follow: *rather than try to prevent an attack, let's fool the attacker*. Our approach relies on a network $P : \mathcal{X} \to \mathcal{X}$, pasted to the already trained target network M before the input layer, such as the augmented model will answer $T(x) = M \circ P(x)$ when fed with input x. The additional component P is designed and trained to reach a twofold objective:

- *Prediction neutrality:* adding P does not alter the decision for a clean example x, i.e. $T(x) = M \circ P(x) = M(x)$;
- *Adversarial luring:* according to an adversarial example x' crafted to fool T, M does not output the same label as T ($M \circ P(x') \neq M(x')$) and, in the best case, x' is inefficient ($M(x') = y$).

To explain the intuition of our method, we follow the feature-based framework proposed in Ilyas et al. (2019) where a *feature* f is a function from $\mathcal{X}$ to $\mathbb{R}$, that M has learned to perform its predictions. Considering a binary classification task, f is said to be *ρ-useful* ($\rho > 0$) if it satisfies Equation 1a. Moreover, f is *γ-useful robust* if under the worst perturbation δ chosen in a predefined set of allowed perturbations Δ, f stays γ-useful under this perturbation (Eq. 1b):

$$\text{(a)} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[y \cdot f(x)\right] > \rho \qquad \text{(b)} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\inf_{\delta\in\Delta} y \cdot f(x+\delta)\right] > \gamma \tag{1}$$

A ρ-useful feature f is said to be a *non-robust feature* if f is not robust for any $\gamma \geq 0$. An adversary which aims at fooling a model will thus perform perturbations to the inputs to influence the useful features which are not robust with respect to the perturbation he is allowed to apply. We denote $\mathcal{F}^*_M$ the set of ρ-useful features learned by M. We consider a set of allowed perturbations Δ and $\gamma > 0$ such that we note $\mathcal{F}^{*,R}_M$ and $\mathcal{F}^{*,NR}_M$ respectively the set of γ-robust and non-robust features learned by M (relatively to Δ). An adversary that aims at fooling $M \circ P$ will alter function compositions of the form $f \circ P$ with $f \in \mathcal{F}^*_M$. These function compositions are the non-robust useful features of $M \circ P$, whose set is denoted $\mathcal{F}^{*,NR}_{M\circ P}$.

Based on the observations that transferability of adversarial perturbations between two models occurs because these models rely on similar non-robust features (Ilyas et al., 2019), we consider $f \circ P \in \mathcal{F}^{*,NR}_{M\circ P}$, and we derive two possibilities which ensure the lowest transferability between $M \circ P$ and M, regarding the robustness of f. If f is robust, $f \in \mathcal{F}^{*,R}_M$, that means the adversarial perturbations from Δ is sufficient to *flip* the augmented feature $f \circ P$ but is not efficient to directly impact f. This case is the optimal one, since the adversarial example is unsuccessful on the target model ($M \circ P(x') \neq y$ and $M(x') = y$). On the other hand, if $f \in \mathcal{F}^{*,NR}_M$ (i.e. both f and $f \circ P$ are non-robust), restraining the transferability means that the additional model P impacts the way useful features vary with respect to input alterations so that the adversarial perturbation lead to two different labels. We encompass these two cases with what we called the luring effect, illustrated in Figure 1. The adversary is tricked into modifying input values in some way to flip useful and non-robust features of $M \circ P$ *and* these modifications are either without effect on the useful features of M, or flip the non-robust features of M in a different way.

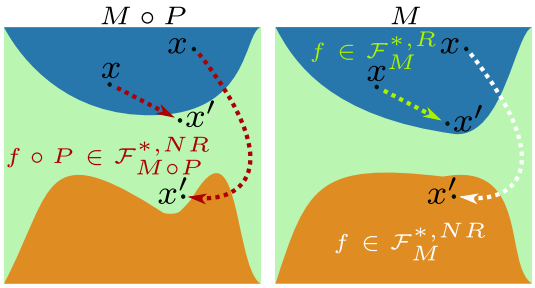

Figure 1: Luring effect. (left) x' fools $M \circ P$ by flipping a non-robust feature $f \circ P$ (toward the green class).(right) However, f can be a robust feature learned by M, then M is not fooled (still in the blue class), or a non-robust feature learned by M but switched differently than $f \circ P$ (toward the orange class).

2.4 Training the Luring Component

To reach the two objectives cited in 2.3, we propose to train P with constraints based on the predicted labels order. For $x \in \mathcal{X}$, let α and β be the labels corresponding to the first and second highest confidence score for the model M. The training of P is achieved with a new loss function that constraints α to (still) be the first class predicted by $M \circ P$ (prediction neutrality) and to make the confidence gap between α and β the highest as possible for $M \circ P$ (adversarial luring). Conceptually, as the *direction* of confidence towards classes is forced to be structurally different for $M \circ P$ and M, we hypothesize that useful features of the two classifiers should be different and behave differently to adversarial perturbations. Despite its simplicity, as it will be verified experimentally, this idea is effective to have $M \circ P$ and M behaving similarly for clean examples but differently for adversarial examples.

The *luring loss*, designed to induce this behavior, is given in Equation 2 and the complete training procedure is detailed in Algorithm 1. The parameters of P are denoted by θ, $x \in \mathcal{X}$ is an input image and M is the target model. M has already been trained and its parameters are frozen during

the process. ℓ denotes the logits of M and $\ell\, P_i^\theta(x)$ the logits of $M \circ P$ for input x and class i. The classes a and b correspond to the second maximum value of ℓ and $\ell\, P^\theta$ respectively.

$$\mathcal{L}(\theta, x, M) = -\lambda\big(\ell\, P^\theta_{M(x)}(x) - \ell\, P^\theta_a(x)\big) + max\big(0, \ell\, P^\theta_b(x) - \ell\, P^\theta_{M(x)}(x)\big) \tag{2}$$

The first term of Equation 2 optimizes the gap between the logits of $M \circ P$ corresponding to the first and second biggest unscaled confidence score given by M (i.e. $M(x)$ and a). This part formalizes the goal of changing the direction of confidence between $M \circ P$ and M. The second term is compulsory to reach a good classification since the first part only does not ensure that $\ell\, P^\theta_{M(x)}(x)$ is the highest logit value (prediction neutrality). The parameter $\lambda > 0$, called the *luring coefficient*, allows to control the trade-off between ensuring good accuracy and shifting confidence direction.

Algorithm 1 Training of the luring component

Input: trained model M, training steps K, learning rate η, batch size B, luring coefficient λ
Output: luring component P.

```
ℓ P_c^θ(x) denotes the logits of M ∘ P for input x and class c
Randomly initialize the parameters θ of P
for step = 1 ... K: do
    {x_1, x_2, ..., x_B}: batch of training set examples
    for i = 1 ... B: do
        (a^i,b^i) ← (class of the 2^nd max value of ℓ(x_i),class of the 2^nd max value of ℓ P^θ(x_i))
        L(θ, x_i, M) ← −λ(ℓ P^θ_{M(x_i)}(x_i) − ℓ P^θ_{a^i}(x_i)) + max(0, ℓ P^θ_{b^i}(x_i) − ℓ P^θ_{M(x_i)}(x_i))
    end for
    θ ← θ − η Σ_{i=1}^{B} ∇_θ L(θ, x_i, M)/B
end for
return P
```

3 Characterization of the luring effect

3.1 Objective

Our first experiments are dedicated to the characterization of the luring effect by (1) evaluating our objectives in term of transferability and (2) isolating it from other factors that may influence transferability. For that purpose, we compare our approach to the following architectures, training procedures and loss functions:

- ***Stack* model**: $M \circ P$ is retrained as a whole with the cross-entropy loss. *Stack* serves as a first baseline to measure the transferability between the two architectures of $M \circ P$ and M.
- ***Auto* model**: P is an auto-encoder trained separately with binary cross-entropy loss. *Auto* serves as a second baseline of a component resulting in a *neutral* mapping from $\mathbb{R}^d$ to $\mathbb{R}^d$.
- ***C_E* model**: P is trained with the cross-entropy loss between the confidence score vectors $M \circ P(x)$ and $M(x)$ in order to mimic the decision of the target model M. This model serves as a comparison between our loss and a loss function which does not aim at maximizing the gap between the confidence scores.

We perform experiments on three datasets: MNIST (Lecun et al., 1998), SVHN (Netzer et al., 2011) and CIFAR10 (Krizhevsky, 2009). For MNIST, M has the same architecture as in (Madry et al., 2018). For SVHN and CIFAR10, we follow an architecture inspired from VGG (Simonyan & Zisserman, 2015). Detailed architectures and training setup for models M and P are presented in Appendix A and B. Table 8 in Appendix C gathers the test set accuracy and agreement rate between each augmented model and M (the augmented model and M agree on the ground-truth label). We observe that our approach has a limited impact on the test accuracy with a relative decrease of 1.71%, 4.26% and 4.48% for MNIST, SVHN and CIFAR10 respectively.

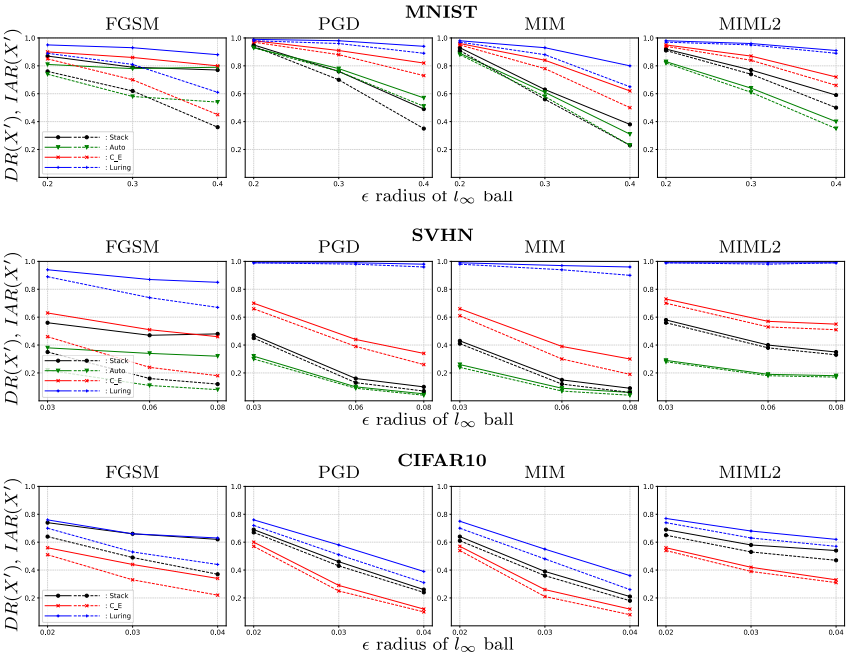

Figure 2: Disagreement Rate (solid line) and Inefficient Adversarial examples Rate (dashed line) for different attacks.

3.2 Attack setup and metrics

For the characterization of the luring effect, we attack the model $M \circ P$ of the four approaches and transfer to M *only* the adversarial examples that are successful for these four models. We define the *disagreement rate*, noted $DR(X')$, that represents the rate of successful adversarial examples on $M \circ P$ for which M and $M \circ P$ do not agree. To measure the best case where the luring effect leads to unsuccessful adversarial examples when transferred to M, we note $IAR(X')$ an *inefficient adversarial examples rate* that represents the proportion of successful adversarial examples on $M \circ P$ but not on M. For both metrics (see Equation 3), the higher, the better the luring effect to protect the target model M against transferable attacks.

$$DR(X') = \frac{\sum_{X'} \mathbf{1}_{M o P(x') \neq y, M \circ P(x') \neq M(x')}}{\sum_{X'} \mathbf{1}_{M \circ P(x') \neq y}} \quad IAR(X') = \frac{\sum_{X'} \mathbf{1}_{M \circ P(x') \neq y, M(x') = y}}{\sum_{X'} \mathbf{1}_{M \circ P(x') \neq y}} \tag{3}$$

We use the gradient-based attacks FGSM (Goodfellow et al., 2015), PGD (Madry et al., 2018), and MIM (Dong et al., 2018) in its l_∞ (MIM) and l_2 (MIML2) versions. We used three l_∞ perturbation budgets (ϵ values): 0.2, 0.3 and 0.4 for MNIST, 0.03, 0.06 and 0.08 for SVHN, and 0.02, 0.03 and 0.04 for CIFAR10. We detail the parameters used to run these attacks in Appendix D.1.

3.3 Results and complementary analysis

We report DR and IAR in Figure 2. The highest performance with our loss for every ϵ and the fact that both metrics decrease much slower as ϵ increases compared to the other architectures, show that our training method is efficient at inducing the luring effect.

As a complementary analysis, we investigate the magnitude of the adversarial perturbations. Results show that our approach leads to similar l_∞ and l_2 distortion scales than the other methods. This indicates that our approach truly impacts the underlying useful features and does not only imply different distortions scales needed to fool both M and $M \circ P$. The complete analysis presented in Appendix D.2.

We also find that for MNIST, the l_0 distortion (i.e. the cardinality of the pixels impacted by the adversarial attack) is significantly higher with our method as presented in Figure 3a. That points

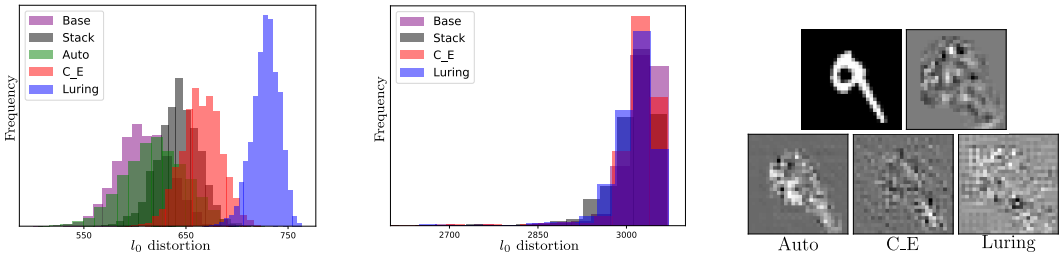

Figure 3: l_0 adversarial distortion for MNIST *(a)* and CIFAR10 *(b)*. Saliency maps for MNIST *(c)*. Top: clean image and input gradients $\nabla_x \mathcal{L}(M(x), y)$. Bottom: mapping gradients $\nabla_x P(x)$ for 3 augmented models.

out P leverages more different useful features, which are easily identifiable for MNIST because of the basic structure of the images. This effect can also be demonstrated thanks to saliency maps (i.e. analyzing $\nabla_x P(x)$), as illustrated in Figure 3c: for the Luring model, P is sensitive to many additional *useless* background pixels compared to M. Note that for *Auto*, modifying P's mapping consists almost exclusively on modifying pixels correctly correlated with the true label.

For more complex inputs, such as in SVHN and CIFAR10, we observe that the l_0 distortion is approximately the same for all the architectures (Figure 3b). However, for these two datasets, we analyze the influence of P's mapping thanks to the logits variations. We find that logits between M and $M \circ P$ vary more differently with respect to input modifications when P is trained with our approach, which confirms that our luring loss enables to reach our second objective (adversarial luring). The detailed analysis in presented in Appendix D.3.

4 Using the Luring Effect as a Defense

By fooling an adversary on the way to target a black-box model, the *luring effect* can be seen as a defense mechanism on its own or as a complement of state-of-the-art approaches. In this section we discuss the way to use our approach as a protection.

4.1 Threat Model

Attacker Our work sets in the black-box paradigm, i.e. we assume that the adversary has no access to the inner parameters of the target model M nor the augmented model $M \circ P$. This setting is classical when dealing with cloud-based API or edge neural networks in secure embedded systems where users have only access to input/output information with different querying abilities and output precision (i.e. scores or labels). More precisely:

- The *adversary goal* is to craft (untargeted) adversarial examples on the model he has access to, namely, the augmented model $T = M \circ P$.
- The *adversarial knowledge* corresponds to an adversary having a black-box access to a ML system $\mathbb{S}_A$ containing the protected model $T = M \circ P$. He can query it (without any limit on the number of queries) and we assume that he is able to get the logits outputs (see 4.2). The target model M stays completely hidden and the adversary is not aware of any potential protections.
- Classically, the *adversarial capability* is an upper bound ϵ of the perturbation $\|x' - x\|_\infty$.

Defender For each query, the defender has access to two outputs: $M(x)$ and $M \circ P(x)$. Therefore, two different inference schemes are viable according to the goals and characteristics of the system to defend, noted $\mathbb{S}_D$.

- *(inner transferability scenario)* If $\mathbb{S}_D$ is equivalent (or close) to $\mathbb{S}_A$, the goal of the defender is to take advantage of the luring effect to detect an adversarial examples by comparing the two inference outputs $M(x)$ and $T(x)$. An appropriate metric is the rate of adversarial

Table 1: AC_{MoP}, AC_M and DAC for different source model architectures. SVHN (top), CIFAR10 (down).

SVHN		STACK			AUTO			C_E			LURING		
	ϵ	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC
SPSA	0.03	0.10	0.54	0.56	0.06	0.37	0.38	0.06	0.67	0.68	0.0	**0.96**	**0.97**
	0.06	0.01	0.21	0.24	0.0	0.10	0.11	0.0	0.37	0.42	0.0	**0.96**	**0.96**
	0.08	0.0	0.13	0.15	0.0	0.06	0.06	0.0	0.23	0.28	0.0	**0.94**	**0.96**
ECO	0.03	0.06	0.42	0.44	0.14	0.48	0.49	0.18	0.66	0.68	0.20	**0.97**	**0.98**
	0.06	0.0	0.11	0.12	0.06	0.09	0.11	0.1	0.35	0.39	0.1	**0.86**	**0.88**
	0.08	0.0	0.03	0.07	0.06	0.09	0.09	0.08	0.29	0.32	0.09	**0.84**	**0.86**
MIM-W	0.03	0.04	0.32	0.35	0.01	0.20	0.21	0.03	0.41	0.45	0.11	**0.81**	**0.87**
	0.06	0.0	0.06	0.09	0.0	0.03	0.05	0.0	0.10	0.18	0.0	**0.58**	**0.71**
	0.08	0.0	0.03	0.06	0.0	0.01	0.02	0.0	0.06	0.13	0.0	**0.48**	**0.67**

CIFAR10		STACK			C_E			LURING		
	ϵ	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC
SPSA	0.02	0.01	0.75	0.78	0.06	0.68	0.71	0.12	**0.78**	**0.82**
	0.03	0.0	0.57	0.62	0.01	0.45	0.49	0.02	**0.59**	**0.64**
	0.04	0.0	0.38	0.42	0.0	0.22	0.24	0.01	**0.42**	**0.50**
ECO	0.02	0.15	0.7	0.71	0.14	0.6	0.63	0.16	**0.81**	**0.87**
	0.03	0.09	0.44	0.59	0.09	0.36	0.42	0.2	**0.65**	**0.67**
	0.04	0.05	0.29	0.29	0.04	0.28	0.34	0.09	**0.35**	**0.44**
MIM-W	0.02	0.0	0.49	0.52	0.02	0.4	0.43	0.03	**0.58**	**0.64**
	0.03	0.0	0.24	0.28	0.0	0.15	0.19	0.0	**0.30**	**0.39**
	0.04	0.0	0.13	0.16	0.0	0.05	0.1	0.0	**0.18**	**0.28**

examples which are either detected or well-predicted by M, as expressed in Equation 4 and noted DAC for *Detection Adversarial Accuracy*.

$$\text{DAC} = 1 - \frac{\sum_{x \in X'} \mathbf{1}_{M \circ P(x') = M(x'), M(x') \neq y}}{|X'|} \tag{4}$$

- *(distant transferability scenario)* If $\mathbb{S}_D$ is a secure black-box system with limited access, that only contains the target model M, the goal of the defender is to thwart attacks crafted from $\mathbb{S}_A$. For this specific scenario, the defender may only rely on the fact that the luring effect likely leads to unsuccessful adversarial examples ($M(x') = y$). The appropriate metric is the classical adversarial accuracy (AC) that is the standard accuracy measured on the adversarial set X' and noted AC_{MoP} and AC_M respectively for $M \circ P$ and M.

An illustration of these two scenarios is presented in Appendix E.

4.2 ATTACKS

In order to evaluate our defense with respect to the threat model, we attack $M \circ P$ with strong gradient-free attacks:

- **SPSA** attack (Uesato et al., 2018) ensures to consider the strongest adversary in our black-box setting as it allows the adversary to get the logits of $M \circ P$.
- Coherently to the first scenario ($\mathbb{S}_A = \mathbb{S}_D$) with an adversary that has no querying limitation, **ECO** (Moon et al., 2019) is a strong score-based gradient-estimation free attack.

To perform an even more strict evaluation, and to anticipate future gradient-free attacks, we report the best results[1] obtained with the state-of-the-art transferability designed gradient-based attacks **MIM**, **DIM**, **MIM-TI** and **DIM-ti** (Dong et al., 2018; Xie et al., 2018; Dong et al., 2019), under the name **MIM-W**. The parameters used to run these attacks are presented in Appendix F.

4.3 RESULTS

The results are presented in Table 1 for SVHN and CIFAR10 (results for MNIST are close to SVHN and are presented in Appendix G). For CIFAR10, since no autoencoder we tried reaches correct test set accuracy, we exclude the *Auto* model. First, we notice that AC_{MOP} almost always equal 0 with

[1]Here *best* is from the adversary point of view.

iterative attacks[2]. This is an indication that the defense component P does not induce a form of gradient masking (Athalye et al., 2018).

Remarkably on SVHN, for $\epsilon = 0.08$ (the largest perturbation), and for the worst-case attacks, adversarial examples tuned to achieve the best transferability only reduce AC_M to 0.48 against our approach (compared to 0 for the other architectures). The robustness benefits are more observable on SVHN and MNIST but the results on CIFAR10 are particularly promising in the scope of a defense scheme that only requires a pre-trained model. Indeed, for the common l_∞ perturbation value of 0.03, the worst DAC value for the *C_E* and *Luring* models are respectively 0.19 (against DIM attack) and 0.39 (against DIM-TI attack).

5 COMPATIBILITY WTIH IMAGENET AND ADVERSARIAL TRAINING

We scale our approach on the ImageNet data set (Deng et al., 2009). Results are presented in Appendix I.1. As training models on ImageNet is computationally expensive, we only present results for the C_E method and our approach. Moreover, we consider the strong attacks **MIM-W**. The worst AC_M and DAC values equal for our approach, while...
In Appendix I.2 we illustrate our approach in the case of M is trained with adversarial training.

6 RELATED WORK

Defenses in the black-box context are weakly covered in the litterature compared to many approaches focused on white-box attacks. More particularly, very few approaches deal with query-based black-box attacks such as the recent BlackLight (Li et al., 2020). To our knowledge, our work is the first to exploit the idea of deceiving an adversary taking advantage of the transferability property. However, a honeypot-based approach has been recently proposed (Shan et al., 2019) with a *trapdoored model* to detect adversarial examples. Even if the threat model (white-box) is different from ours, it is interestingly mentioned that transferability between the target and the trapdoor-protected model is very poor. By fitting the trapdoor approach in our black-box threat model we highlight complementary performances in term of detection of adversarial examples, meaning that an overall deception-based strategy is a promising direction for future work. The detailed analysis is presented in Appendix H.

7 CONCLUSION

We propose a conceptually innovative approach to improve the robustness of a model against transfer black-box adversarial perturbations that basically relies on a deception strategy. Inspired by the notion of robust and non-robust features, we derive and characterize the *luring effect*, which is implemented via a decoy network built upon the target model, and a loss designed to fool the adversary into targeting different non-robust features than the ones of the target model. Importantly, this approach only relies on the logits of target model and does not require a labeled data set.
We show that our approach can be used as a defense and that a defender may exploit two prediction schemes to detect adversarial examples or enhance the adversarial robustness. Experiments on MNIST, SVHN and CIFAR10 show that exploiting the luring effect enables to successfully thwart an adversary using state-of-the-art optimized attacks even with large adversarial perturbations.

[2]It does not equal exactly 0 sometimes because parameters for these attacks are tuned relatively to the adversarial goal: lowering our defense efficiency. Thus, the lowest performances of our defense sometimes do not correspond to the lowest accuracy of $M \circ P$ on the adversarial examples.

REFERENCES

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 2018.

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

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, and John C. Duchi. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, 2019.

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, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *The Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Steven Chen, Nicholas Carlini, and David Wagner. Stateful detection of black-box adversarial attacks. *arXiv preprint arXiv:1907.05587*, 2019.

Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, pp. 1310–1320, 2019.

J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Nicolas Ford, Justin Gilmer, and Ekin D. Cubuk. Adversarial examples are a natural consequence of test error in noise. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, pp. 2280–2289, 2019.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*. Wiley Online Library, 2015.

Chuan Guo, Jacob R. Gardner, Yurong You, Andrew G. Wilson, and Kilian Q. Weinberger. Simple black-box adversarial attacks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, 2019.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, 2019.

Uiwon Hwang, Jaewoo Park, Hyemi Jang, Sungroh Yoon, and Nam Ik Cho. Puvae: A variational autoencoder to purify adversarial examples. *IEEE Access*, 7:126582–126593, 2019.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 2018.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pp. 125–136, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Huiying Li, Shawn Shan, Emily Wenger, Jiayun Zhang, Haitao Zheng, and Ben Y. Zhao. Blacklight: Defending black-box adversarial attacks on deep neural networks. *arXiv preprint arXiv:2006.14042*, 2020.

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

Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 135–147. ACM, 2017.

Seungyong Moon, Gaon An, and Hyun Oh Song. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 4636–4645, 2019.

Yuval Netzer, Tiejie Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519. ACM, 2017.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pp. 5014–5026, 2018.

Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? In *International Conference on Learning Representations*, 2019.

Shawn Shan, Emily Wenger, Bolun Wang, Bo Li, Haitao Zheng, and Ben Y. Zhao. Using honeypots to catch adversarial attacks on neural networks. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2019.

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

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

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

Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pp. 5025–5034, 2018.

Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. Enhancing gradient-based attacks with symbolic intervals. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, 2019.

Cihang Xie, Zhishuai Zhang, Jianyu Wang, Yuyin Zhou, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, 2019.

A SETUP FOR BASE CLASSIFIERS

The architectures of the models trained on MNIST, SVHN and CIFAR10 are detailed respectively in tables 2, 3 and 4. BN, MaxPool(u,v), UpSampling(u,v) and Conv(f,k,k) denote respectively batch normalization, max pooling with window size (u,v), upsampling with sampling factor (u,v) and 2D convolution with f filters and kernel of size (k,k).

For MNIST, we used 5 epochs, a batch size of 28 and the Adam optimizer with a learning rate of 0.01. For SVHN, we used 50 epochs, a batch size of 28 and the Adam optimizer with a learning rate of 0.01. For CIFAR10, we used 200 epochs, a batch size of 32, and the Adam optimizer with a piecewise learning rate of 0.1, 0.01 and 0.001 after respectively 80 and 120 epochs.

Table 2: MNIST base classifier architecture. Epochs: 5. Batch size: 28. Optimizer: Adam with learning rate 0.01.

ARCHITECTURE
CONV(32,5,5) + RELU
MAXPOOL(2,2)
CONV(64,5,5) + RELU
MAXPOOL(2,2)
DENSE(1024) + RELU
DENSE(10) + SOFTMAX

Table 3: SVHN base classifier architecture. Epochs: 5. Batch size: 28. Optimizer: Adam with learning rate 0.01.

ARCHITECTURE
CONV(64,3,3) + BN + RELU
CONV(64,3,3) + MAXPOOL(2,2) + BN + RELU
CONV(128,3,3) + BN + RELU
CONV(128,3,3) + MAXPOOL(2,2) + BN + RELU
CONV(256,3,3) + BN + RELU
CONV(256,3,3) + MAXPOOL(2,2) + BN + RELU
DENSE(1024) + BN + RELU
DENSE(1024) + BN + RELU
DENSE(10) + SOFTMAX

Table 4: CIFAR10 base classifier architecture. Epochs: 200. Batch size: 32. Optimizer: Adam with learning rate starting at 0.1, decreasing to 0.01 and 0.001 respectively after 80 and 120 epochs.

ARCHITECTURE
CONV(128,3,3) + BN + RELU
CONV(128,3,3) + MAXPOOL(2,2) + BN + RELU
CONV(256,3,3) + BN + RELU
CONV(256,3,3) + MAXPOOL(2,2) + BN + RELU
CONV(512,3,3) + BN + RELU
CONV(512,3,3) + MAXPOOL(2,2) + BN + RELU
DENSE(1024) + BN + RELU
DENSE(1024) + BN + RELU
DENSE(10) + SOFTMAX

B TRAINING SETUP FOR DEFENSE COMPONENTS

The architectures for the defense component P on MNIST, SVHN and CIFAR10 are detailed respectively in Tables 5, 6 and 7. BN, $MaxPool(u, v)$, $UpSampling(u, v)$ and $Conv(f, k, k)$ denote respectively batch normalization, max pooling with window size (u, v), upsampling with sampling

Table 5: Defense component architecture for MNIST.

ARCHITECTURE
CONV(16,3,3) + RELU
MAXPOOL(2,2)
CONV(8,3,3) + RELU
MAXPOOL(2,2)
CONV(8,3,3) + RELU
CONV(8,3,3) + RELU
UPSAMPLING(2,2)
CONV(8,3,3) + RELU
UPSAMPLING(2,2)
CONV(16,3,3)
UPSAMPLING(2,2)
CONV(1,3,3) + SIGMOID

factor (u, v) and 2D convolution with f filters and kernel of size (k, k). The detailed parameters used to perform training are also reported.

MNIST:

- *Stack* Epochs: 5. Batch size: 28. Optimizer: Adam with learning rate 0.001
- *Auto* Epochs: 50. Batch size: 128. Optimizer: Adam with learning rate 0.001
- *C_E* Epochs: 64. Batch size: 64. Optimizer: Adam with learning rate starting at 0.001, decreasing to 0.0002 and 0.0004 respectively after 45 and 58 epochs
- *Luring* Epochs: 64. Batch size: 64. Optimizer: Adam with learning rate starting at 0.001, decreasing to 0.0002 and 0.0004 respectively after 45 and 58 epochs

Table 6: Defense component architecture for SVHN.

ARCHITECTURE
CONV(128,3,3) + BN + RELU
MAXPOOL(2,2)
CONV(256,3,3) + BN + RELU
MAXPOOL(2,2)
CONV(512,3,3) + BN + RELU
CONV(1024,3,3) + BN + RELU
MAXPOOL(2,2)
CONV(512,3,3) + BN + RELU
CONV(512,3,3) + BN + RELU
UPSAMPLING(2,2)
CONV(256,3,3) + BN + RELU
UPSAMPLING(2,2)
CONV(128,3,3) + BN + RELU
UPSAMPLING(2,2)
ONV(64,3,3) + BN + RELU
CONV(3,3,3) + BN + SIGMOID

SVHN:

- *Stack* Epochs: 20. Batch size: 256. Optimizer: Adam with learning rate 0.001
- *Auto* Epochs: 5. Batch size: 128. Optimizer: Adam with learning rate 0.001
- *C_E* Epochs: 210. Batch size: 256. Optimizer: Adam with learning rate starting at 0.0001, decreasing to 0.00001 and 0.000008 respectively after 126 and 168 epochs
- *Luring* Epochs: 210. Batch size: 256. Optimizer: Adam with learning rate starting at 0.0001, decreasing to 0.00001 and 0.000008 respectively after 126 and 168 epochs. Dropout is used

Table 7: Defense component architecture for CIFAR10.

ARCHITECTURE
CONV(128,3,3) + BN + RELU
CONV(256,3,3) + BN + RELU
MAXPOOL(2,2)
CONV(512,3,3) + BN + RELU
CONV(1024,3,3) + BN + RELU
CONV(512,3,3) + BN + RELU
CONV(512,3,3) + BN + RELU
CONV(256,3,3) + BN + RELU
UPSAMPLING(2,2)
CONV(128,3,3) + BN + RELU
CONV(64,3,3) + BN + RELU
CONV(3,3,3) + BN + SIGMOID

CIFAR10:

- *Stack* Epochs: 200. Batch size: 32. Optimizer: Adam with learning rate starting at 0.1, decreasing to 0.01 and 0.001 respectively after 80 and 120 epochs
- *C_E* Epochs: 216. Batch size: 256. Optimizer: Adam with learning rate starting at 0.00001, decreasing to 0.000005 and 0.0000008 respectively after 154 and 185 epochs. Dropout is used
- *Luring* Epochs: 216. Batch size: 256. Optimizer: Adam with learning rate starting at 0.00001, decreasing to 0.000005 and 0.0000008 respectively after 154 and 185 epochs. Dropout is used

C TEST SET ACCURACY AND AGREEMENT RATES

For the luring parameter, we set $\lambda = 1$ for MNIST and SVHN, and $\lambda = 0.15$ for CIFAR10. For CIFAR10, since no autoencoder we tried reaches correct test set accuracy, we exclude the *Auto* model.

Table 8: Test set accuracy and agreement (augmented and target model agree on the ground-truth label) between each augmented model and the target model M, noted BASE.

MODEL	DATA SET					
	MNIST		SVHN		CIFAR10	
	TEST	AGREE	TEST	AGREE	TEST	AGREE
BASE	0.991	–	0.961	–	0.893	–
STACK	0.98	0.976	0.925	0.913	0.902	0.842
AUTO	0.971	0.969	0.95	0.943	–	–
C_E	0.982	0.977	0.919	0.907	0.860	0.834
LURING	0.974	0.969	0.920	0.917	0.853	0.822

D LURING EFFECT ANALYSIS

D.1 ATTACK PARAMETERS

For MIML2, we report results when adversarial examples are clipped to respect the threat model with regards to ϵ. An illustration of a clean image and its adversarial counterpart for the maximum perturbation allowed is presented in Figure 4. We note that the ground-truth label is still clearly recognizable. For PGD, MIM and MIML2, the number of iterations is set to 1000, the step size to 0.01 and μ to 1.0 (MIM and MIML2). For MIML2, the l_2 bound is set to 30 on MNIST and 2 on SVHN and CIFAR10.

Figure 4: (top) Clean image, (bottom) adversarial example for the maximum perturbation allowed (left to right: $\epsilon = 0.4, 0.08, 0.04$).

D.2 Analysis of the l_2 and l_∞ distortions

We investigate the impact of our additional component on the distance between a clean and an adversarial example, compared to the other architectures. We measure the l_2 means of the adversarial perturbations needed to find an adversarial example, and verify that our approach is globally at the same level than the other architectures.

For that purpose, an appropriate and recommended attack is the Carlini & Wagner l_2 attack (hereafter, CWL2) (Carlini & Wagner, 2017) that searches for the closest adversarial examples in terms of the l_2 distortion. We run CWL2 with the following parameters: 10 binary search steps, learning rate of 0.1, initial constant of 0.5 and 500 iterations for MNIST, SVHN and CIFAR10. These parameters have been chosen such that increasing the number of iterations does not lower the final l_2 distortion. For each data set, we run the attack on $1,000$ test set examples correctly classified by all the architectures for both M and $M \circ P$.

Table 9: Average l_2 and l_∞ distortions between clean and adversarial examples generated with the CWL2 attack for the target model M and the augmented models $M \circ P$

	MNIST		SVHN		CIFAR10	
Model	Mean l_2	Mean l_∞	Mean l_2	Mean l_∞	Mean l_2	Mean l_∞
Target	0.96	0.27	0.32	0.04	0.42	0.04
Stack	0.87	0.25	0.34	0.05	0.4	0.03
Auto	0.94	0.35	0.29	0.04	–	–
C_E	0.92	0.34	0.29	0.04	0.41	0.04
Luring	0.94	0.31	0.31	0.04	0.39	0.04

The average l_2 and l_∞ distortions are reported in Table 9 and show that there is no significant difference between our approach and the other architectures. This is an additional observation that our *luring loss* allows to train an augmented model which causes the *luring effect* by predominantly targeting different useful non-robust features rather than artificially modifying the scale of the adversarial distortion needed to cause misclassification.

D.3 Logits analysis

As a complementary analysis on the impact of the mapping induced by the additional compoment P, we analyse how P acts on the variation of the logits with respect to the input features ($\nabla_{x_i} \ell(x)$). For that purpose, we first pick 1000 test set examples correctly classified by all the models. Then, for each augmented model and each class c, we compute the number Ω_c of pixels x_d that lead to opposite effect on prediction between M and $M \circ P$ (see Equation 5):

$$\Omega_c = \left| \left\{ d : \frac{\partial \ell_c}{\partial x_d} \cdot \frac{\partial \ell \, P_c}{\partial x_d} < 0 \right\} \right| \tag{5}$$

Next, we look at the proportion of these examples for which Ω_c is higher for the Luring model than for the other ones. Results for each class are presented in Table 10. For both SVHN and CIFAR10, these proportions are strictly higher than 80% whatever the class. This is a supplementary indication toward the fact that – with respect to the allowed adversarial perturbation altering x – our approach is more relevant than other architectures (that also perform a mapping within $\mathcal{X}$) to *flip* useful and non-robust features of $M \circ P$ and M towards different labels.

Table 10: Rate, over , 1000 examples, of examples for which logits vary more differently between M and $M \circ P$, for the *Luring* model against the other models.

	CLASS									
	1	2	3	4	5	6	7	8	9	10
SVHN	0.81	0.85	0.86	0.89	0.86	0.82	0.83	0.84	0.83	0.82
CIFAR	0.86	0.88	0.88	0.87	0.88	0.87	0.89	0.86	0.88	0.85

E THREAT MODELS ILLUSTRATION

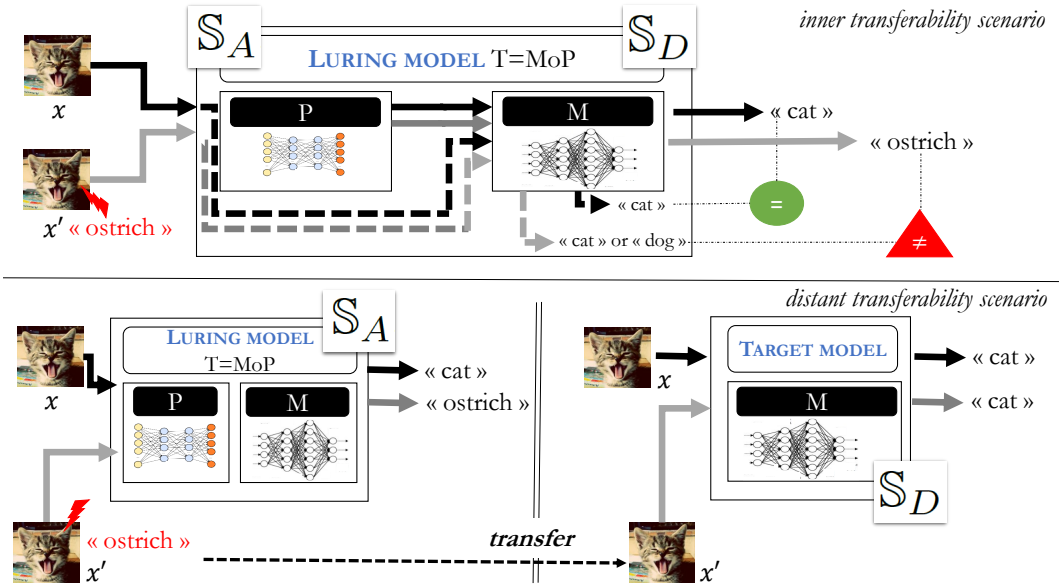

Figure 5: Illustration of the black-box threat model with two scenarios. (top) In the *inner transferability scenario* the system to attack and to defend is the same. Because the luring effect induces different behavior when facing adversarial perturbation, the defender is able to detect adversarial examples by comparing $M(x)$ and $T(x)$. (bottom) In the *distant transferability scenario*, the system to defend may suffer from transferable adversarial examples but the defender takes advantage of the weak transferability between T and M.

F ATTACK PARAMETERS

All the attacks (SPSA, ECO, MIM, DIM, MIM-TI, DIM-TI) are performed on 1000 correctly classified test set examples, with parameter values tuned for the highest transferability results. More precisely, we searched for parameters leading to the lowest AC_M and DAC values.

F.1 GRADIENT-FREE ATTACKS

For the SPSA attack, the learning rate is set to 0.1, the number of iterations is set to 100 and the batch size to 128. For the ECO attack, for the three datasets, the number of queries is set to $20,000$. We did not perform early-stopping as it results in less transferable adversarial examples. The block size is set to 2, 8 and 4 respectively for MNIST, SVHN and CIFAR10.

F.2 GRADIENT-BASED ATTACKS

For DIM and DIM-TI, the optimal p value was searched in $\{0.1, 0.2, \cdots, 1.0\}$. We obtained the optimal values of 1.0, 1.0 and 0.6 respectively on MNIST, SVHN and CIFAR10. For MIM and its variants, the number of iterations is set to 1000, 500 and 100 respectively on MNIST, SVHN and CIFAR10, and $\mu = 1.0$. The optimal kernel size required by MIM-TI and DIM-TI was searched in $\{(3,3),(5,5),(10,10),(15,15)\}$. For the MIM-TI and DIM-TI, the size of the kernel resulting in the lowest AC_M and DAC values reported in Section I.2 are presented in tables 11, 12 and 13 respectively for MNIST, SVHN and CIFAR10.

Table 11: MNIST. Kernel size for the MIM-TI and DIM-TI attacks.

ATTACK	ARCHITECTURE	ϵ VALUE		
		0.3	0.4	0.5
MIM-TI	STACK	5×5	5×5	5×5
	AUTO	10×10	5×5	5×5
	C_E	10×10	10×10	5×5
	LURING	5×5	5×5	10×10
DIM-TI	STACK	5×5	5×5	5×5
	AUTO	5×5	5×5	5×5
	C_E	10×10	10×10	10×10
	LURING	5×5	5×5	5×5

Table 12: SVHN. Kernel size for the MIM-TI and DIM-TI attacks.

ATTACK	ARCHITECTURE	ϵ VALUE		
		0.03	0.06	0.08
MIM-TI	STACK	5×5	5×5	5×5
	AUTO	5×5	5×5	5×5
	C_E	5×5	10×10	5×5
	LURING	10×10	10×10	10×10
DIM-TI	STACK	5×5	5×5	5×5
	AUTO	5×5	5×5	5×5
	C_E	5×5	5×5	5×5
	LURING	10×10	10×10	10×10

Table 13: CIFAR10. Kernel size for the MIM-TI and DIM-TI attacks.

ATTACK	ARCHITECTURE	ϵ VALUE		
		0.02	0.03	0.04
MIM-TI	STACK	3×3	3×3	3×3
	AUTO	3×3	3×3	3×3
	C_E	3×3	10×10	3×3
	LURING	3×3	3×3	3×3
DIM-TI	STACK	3×3	3×3	3×3
	AUTO	3×3	3×3	3×3
	C_E	3×3	3×3	3×3
	LURING	3×3	3×3	3×3

Table 14: MNIST. AC_{MoP}, AC_M and DAC for different source model architectures.

		STACK			AUTO			C_E			LURING		
	ϵ	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC
SPSA	0.2	0.0	0.96	0.97	0.03	0.95	0.95	0.0	0.97	0.98	0.14	**0.99**	**0.99**
	0.3	0.0	0.86	0.89	0.03	0.90	0.92	0.0	0.94	0.95	0.05	**0.96**	**0.97**
	0.4	0.0	0.72	0.77	0.0	0.85	0.87	0.0	0.88	0.91	0.02	**0.95**	**0.96**
ECO	0.2	0.03	0.86	0.92	0.03	0.86	0.88	0.01	0.91	0.93	0.05	**0.99**	**1.0**
	0.3	0.02	0.56	0.65	0.03	0.68	0.7	0.01	0.8	0.87	0.02	**0.91**	**0.96**
	0.4	0.01	0.35	0.54	0.03	0.36	0.46	0.01	0.45	0.48	0.03	**0.77**	**0.79**
MIM-W	0.2	0.0	0.79	0.82	0.0	0.81	0.82	0.0	0.85	0.88	0.19	**0.92**	**0.93**
	0.3	0.0	0.31	0.45	0.0	0.35	0.45	0.0	0.43	0.57	0.13	**0.69**	**0.75**
	0.4	0.0	0.07	0.24	0.0	0.07	0.17	0.0	0.13	0.31	0.07	**0.34**	**0.45**

G RESULT FOR MNIST

H RELATED WORK

In Shan et al. (2019), the authors introduce trapdoors to detect white-box adversarial examples attacks in order to trick the adversary's attack into converging towards specific adversarial directions, which later enables a straightforward detection. To this end, to defend a model against targeted adversarial examples towards label y_t, the clean dataset of input-label pairs (x, y) is augmented with *trapdoored* pairs $(x + \Delta, y_t)$, where Δ is a specific trigger. Then, the target model is trained by minimizing the cross-entropy between the clean and trapdoored input-label pairs to obtain the *trapdoored model*. An input x is detected as a targeted adversarial example thanks to a threshold-based test, that compares a "*neuron activation signature*" of this input against that of the embedded trapdoors. Note that the authors extend their approach to untargeted adversarial examples. We refer to the article (Shan et al., 2019) for more details.

Table 15: Optimal AC_{MoP}, AC_M and DAC values (among FGSM, MIM, DIM, MIM-TI and DIM-TI) for the Trapdoor and Luring approaches.

		TRAPDOOR			LURING		
	ϵ	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC
MNIST	0.2	0.06	0.82	**0.97**	0.19	**0.92**	0.93
	0.3	0.0	0.54	**0.94**	0.13	**0.69**	0.75
	0.4	0.0	0.27	**0.87**	0.07	**0.34**	0.45
SVHN	0.03	0.1	0.48	0.58	0.11	**0.81**	**0.87**
	0.06	0.07	0.38	0.48	0.0	**0.58**	**0.71**
	0.08	0.01	0.23	0.38	0.0	**0.48**	**0.67**
CIFAR10	0.02	0.0	**0.71**	**0.84**	0.03	0.58	0.64
	0.03	0.0	**0.58**	**0.77**	0.0	0.30	0.39
	0.04	0.0	**0.45**	**0.78**	0.0	0.16	0.27

In (Shan et al., 2019), the authors mention that transferability between the target and the trapdoored model is very poor. Based on the code provided by the authors[3], we train trapdoored models with the architectures we used for M. We observe that the transferability is hight from M to the trapdoor version but, on the contrary, the transferability is weak when adversarial examples are crafted on the trapdoored model and transferred to the target model. For the latter scenario, we report in Table 15 the worst AC_M and DAC values for FGSM, MIM, DIM, MIM-TI and DIM-TI along with those of our method. The parameters used to run the attacks are the same as described in Section I.2. These results would correspond to a defense similar to ours where the public model is not the augmented model $M \circ P$, but the trapdoored model trained from M. The threat model stays the same, corresponding to an adversary which wants to transfer adversarial examples from the public model to the target model M.Results in Table 15 on MNIST and CIFAR10 for DAC indicate that using trapdoors could allow for a more efficient detection of adversarial examples, especially for

[3]https://github.com/Shawn-Shan/dnnbackdoor

large adversarial perturbations. Results for the AC_M values on the three data sets also corroborate the idea that both methods could benefit from each other.

I IMAGENET AND ADVERSARIAL TRAINING

I.1 RESULTS ON IMAGENET

The ImageNet dataset (Deng et al., 2009) consists of color images with $1,281,167$ and $50,000$ samples in the training and validation set respectively. For our experiments, the target model M consists of a MobileNetV2 model (Sandler et al., 2018), taking inputs of size $224 \times 224 \times 3$. The architecture for the defense component is described in Table 16.

Table 16: Defense component architecture for ImageNet.

ARCHITECTURE
CONV(128,3,3) + BN + RELU
MAXPOOL(2,2)
CONV(256,3,3) + BN + RELU
MAXPOOL(2,2)
CONV(512,3,3) + BN + RELU
MAXPOOL(2,2)
CONV(512,3,3) + BN + RELU
CONV(256,3,3) + BN + RELU
UPSAMPLING(2,2)
CONV(128,3,3) + BN + RELU
UPSAMPLING(2,2)
CONV(64,3,3) + BN + RELU
CONV(3,3,3) + BN + SIGMOID

For the *Luring* and *C_E* approaches, the additional component P is trained for 100 epochs, with a bath size of 256, using a momentum of 0.9 with learning rate starting at 0.01, decreasing to 0.001 after 80 epochs. The λ parameter of the loss is set to 1.0. For the *Stack* approach, the additional component P is trained for 100 epochs, with a bath size of 256, using a momentum of 0.9 with learning rate starting at 0.05, decreasing to 0.005 after 80 epochs. The λ parameter of the loss is set to 1.0.

The attacks are performed on $1,000$ well-classified images from the validation set.

Table 17: ImageNet. AC_{MoP}, AC_M and DAC for different source model architectures.

		STACK			C_E			LURING		
	ϵ	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC
SPSA	0.2	0.0	0.96	0.97	0.0	0.97	0.98	0.14	**0.99**	**0.99**
	0.3	0.0	0.86	0.89	0.0	0.94	0.95	0.05	**0.96**	**0.97**
	0.4	0.0	0.72	0.77	0.0	0.88	0.91	0.02	**0.95**	**0.96**
MIM-W	0.2	0.0	0.79	0.82	0.0	0.85	0.88	0.19	**0.92**	**0.93**
	0.3	0.0	0.31	0.45	0.0	0.43	0.57	0.13	**0.69**	**0.75**
	0.4	0.0	0.07	0.24	0.0	0.13	0.31	0.07	**0.34**	**0.45**

I.2 COMPATIBILITY WITH ADVERSARIAL TRAINING

As a simple illustration, we present the performances of the *luring* approach when the target model M is trained with adversarial training (Madry et al., 2018). We train the target model M with adversarial training. The l_∞ bound constraint for PGD is set to 0.3 for MNIST and 0.03 for both SVHN and CIFAR10. During training, the PGD is performed for 40 steps, and a step size of 0.1 for MNIST. The PGD is performed for 10 steps with a step size of 0.008 for both SVHN and CIFAR10. We report in Table 18 the accuracy and agreement (augmented and target model agree

on the ground-truth label) between the target model M and its augmented version T, whether M is trained classically or with adversarial training. In Table 19 we report the AC_M and DAC values corresponding to the same threat model as defined in Section , against the strong **MIM-W** attack. Moroever, we report the accuracy of the model M in a white-box setting against the PGD attack with 40 iterations and a step size of 0.01, denoted as $AC_{M,wb}$.

Table 18: Test set accuracy and agreement (augmented and target model agree on the ground-truth label) between an augmented luring model T and a target model M. Base and Base–Luring denote respectively a target model trained classically and the augmented model trained with the luring defense. AdvTrain and AdvTrain–Luring denote respectively a target model trained with adversarial training and the augmented model trained with the luring defense.

Model	Data set					
	MNIST		SVHN		CIFAR10	
	Test	Agree	Test	Agree	Test	Agree
Base	0.991	–	0.961	–	0.893	–
Base–Luring	0.97	0.97	0.92	0.92	0.85	0.82
AdvTrain	0.99	–	0.93	–	0.79	–
AdvTrain–Luring	0.96	0.96	0.87	0.85	0.78	0.76

Table 19: $\text{AC}_{M,wb}$, AC_M and DAC values (among FGSM, MIM, DIM, MIM-TI and DIM-TI) for the luring approach. Base–Luring and AdvTrain–luring denote the cases where the model M is trained respectively classically and with adversarial training.

		Luring			AdvTrain–Luring		
	ϵ	$\text{AC}_{M,wb}$	AC_M	DAC	$\text{AC}_{M,wb}$	AC_M	DAC
MNIST	0.3	0.0	0.69	0.75	0.82	0.97	0.98
SVHN	0.03	0.0	0.81	0.87	0.45	0.898	0.911
CIFAR10	0.03	0.0	0.30	0.39	0.41	0.85	0.88