Adversarial Example Games

Avishek Joey Bose[*]
Mila, McGill University
joey.bose@mail.mcgill.ca

Gauthier Gidel[*]
Mila, Université de Montréal
gauthier.gidel@umontreal.ca

Hugo Berard[*]
Mila, Université de Montréal
Facebook AI Research

Andre Cianflone
Mila, McGill University

Pascal Vincent[†]
Mila, Université de Montréal
Facebook AI Research

Simon Lacoste-Julien[†]
Mila, Université de Montréal

William L. Hamilton[†]
Mila, McGill University

Abstract

The existence of adversarial examples capable of fooling trained neural network classifiers calls for a much better understanding of possible attacks to guide the development of safeguards against them. This includes attack methods in the challenging *non-interactive blackbox* setting, where adversarial attacks are generated without any access, including queries, to the target model. Prior attacks in this setting have relied mainly on algorithmic innovations derived from empirical observations (e.g., that momentum helps), lacking principled transferability guarantees. In this work, we provide a theoretical foundation for crafting transferable adversarial examples to entire hypothesis classes. We introduce *Adversarial Example Games* (AEG), a framework that models the crafting of adversarial examples as a min-max game between a generator of attacks and a classifier. AEG provides a new way to design adversarial examples by adversarially training a generator and a classifier from a given hypothesis class (e.g., architecture). We prove that this game has an equilibrium, and that the optimal generator is able to craft adversarial examples that can attack any classifier from the corresponding hypothesis class. We demonstrate the efficacy of AEG on the MNIST and CIFAR-10 datasets, outperforming prior state-of-the-art approaches with an average relative improvement of 27.5% and 47.2% against undefended and robust models respectively.

1 Introduction

Adversarial attacks on deep neural nets expose critical vulnerabilities in traditional machine learning systems [55, 3, 64, 8]. In order to develop models that are robust to such attacks, it is imperative that we improve our theoretical understanding of different attack strategies. While there has been considerable progress in understanding the theoretical underpinnings of adversarial attacks in relatively permissive settings (e.g. *whitebox* adversaries; [53]), there remains a substantial gap between theory and practice in more demanding and realistic threat models.

In this work, we provide a theoretical framework for understanding and analyzing adversarial attacks in the highly-challenging ***No****n-interactive black****Box*** *adversary* (NoBox) setting, where the attacker has no direct access, including input-output queries, to the target classifier it seeks to fool. Instead,

[*]Equal Contribution, order chosen via randomization.
[†]Canada CIFAR AI Chair

Preprint. Under review.

the attacker must generate attacks by optimizing against some representative classifiers, which are assumed to come from a similar hypothesis class as the target.

The NoBox setting is a much more challenging setting than more traditional threat models, yet it is representative of many real-world attack scenarios, where the attacker cannot interact with the target model [15]. Indeed, this setting—as well as the general notion of transferring attacks between classifiers—has generated an increasing amount of empirical interest [25, 51, 73, 71]. The field, however, currently lacks the necessary theoretical foundations to understand the feasibility of such attacks.

Contributions. To address this theoretical gap, we cast NoBox attacks as a kind of *adversarial example game* (AEG). In this game, an attacker generates adversarial examples to fool a representative classifier from a given hypothesis class, while the classifier itself is trained to detect the correct labels from the adversarially generated examples. Our first main result shows that the Nash equilibrium of an AEG leads to a distribution of adversarial examples effective against *any* classifier from the given function class. More formally, this adversarial distribution is guaranteed to be the most effective distribution for attacking the hardest-to-fool classifiers within the hypothesis class, providing a worst-case guarantee for attack success against an arbitrary target. We further show that this optimal adversarial distribution admits a natural interpretation as being the distribution that maximizes a form of restricted conditional entropy over the target dataset, and we provide detailed analysis on simple parametric models to illustrate the characteristics of this optimal adversarial distribution. Note that while AEGs are latent games [30], they are distinct from the popular generative adversarial networks (GANs) [32]. In AEGs, there is *no* discrimination task between two datasets (generated one and real one); instead, there is a standard supervised (multi-class) classification task on an adversarial dataset.

Guided by our theoretical results we instantiate AEGs using parametric functions —i.e. neural networks, for both the attack generator and representative classifier and show the game dynamics progressively lead to a stronger attacker and robust classifier pairs. We empirically validate AEG on standard CIFAR and MNIST benchmarks and achieve state-of-the-art performance —compared to existing heuristic approaches— in nearly all experimental settings (e.g., transferring attacks to unseen architectures and attacking robustified models), while also maintaining a firm theoretical grounding.

2 Background and Preliminaries

Suppose we are given a classifier $f : \mathcal{X} \to \mathcal{Y}$, an input datapoint $x \in \mathcal{X}$, and a class label $y \in \mathcal{Y}$, where $f(x) = y$. The goal of an adversarial attack is to produce an adversarial example $x' \in \mathcal{X}$, such that $f(x') \neq y$, and where the distance[3] $d(x, x') \leq \epsilon$. Intuitively, the attacker seeks to fool the classifier f into making the wrong prediction on a point x', which is ϵ-close to a real data example x.

Adversarial attacks and optimality. A popular setting in previous research is to focus on generating *optimal* attacks on a single classifier f [13, 53]. Given a loss function ℓ, used to evaluate f, an adversarial attack is said to be optimal if,

$$x' \in \operatorname{argmax}_{x' \in \mathcal{X}} \ell(f(x'), y)\,, \quad \text{s.t.} \quad d(x, x') \leq \epsilon\,. \tag{1}$$

In practice, attack strategies that aim to realize (1) optimize adversarial examples x' directly using the gradient of f. In this work, however, we consider the more general setting of generating attacks that are optimal against entire hypothesis classes $\mathcal{F}$, a notion that we formalize below.

2.1 NoBox Attacks

Threat models specify the formal assumptions of an attack (e.g., the information the attacker is assumed to have access to), which is a core aspect of adversarial attacks. For example, in the popular *whitebox* threat model, the attacker is assumed to have full access to the model f's parameters and outputs [65, 33, 53]. In contrast, the *blackbox* threat model assumes restricted access to the model, e.g., only access to a limited number of input-out queries [18, 42, 57]. Overall, while they consider different access to the target model, traditional whitebox and blackbox attacks both attempt to generate adversarial examples that are optimal for a specific target (i.e., Equation 1).

In this paper, we consider the more challenging setting of ***no**n-interactive black**Box** (**NoBox**)* attacks, intending to generate successful attacks against an unknown target. In the NoBox setting, we assume

[3] We assume that the ℓ_∞ is used in this work, [33, 53] , but our results generalize to any distance d.

no interactive access to a target model; instead, we only assume access to a target dataset and knowledge of the function class to which a target model belongs. Specifically, the NoBox threat model relies on the following key definitions:

- **The target model** f_t. The adversarial goal is to attack some target model $f_t : \mathcal{X} \to \mathcal{Y}$, which belongs to an hypothesis class $\mathcal{F}$. Critically, the adversary has *no access* to f_t *at any time*. Thus, in order to attack f_t, the adversary must develop attacks that are effective against the entirety of $\mathcal{F}$.
- **The target examples** $\mathcal{D}$. The dataset $\mathcal{D}$ contains the examples (x, y) that attacker seeks to corrupt.
- **An hypothesis class** $\mathcal{F}$. As noted above, we assume that the attacker has access to a hypothesis class $\mathcal{F}$ to which the target model f_t belongs.[4] One can incorporate in $\mathcal{F}$ as much prior knowledge one has on f_t (e.g., the architecture, dataset, training method, or regularization), going from exact knowledge of the target $\mathcal{F} = \{f_t\}$ to almost no knowledge at all (e.g., $\mathcal{F} = \{f \in \text{DenseNets}\}$).
- **A reference dataset** $\mathcal{D}_{ref}$. The reference dataset $\mathcal{D}_{\text{ref}}$, which is *similar* to the training data of the target model (e.g., sampled from the same distribution) is used to reduce the size of the hypothesis class $\mathcal{F}$ (e.g., we know that the target model perfoms well at classification on $\mathcal{D}_{ref}$).
- **A representative classifier** f_c. Finally, we assume that the attacker has the ability to optimize a representative classifier f_c from the hypothesis class $\mathcal{F}$.

Given these four key components, we formalize the NoBox setting as follows:

Definition 1. *The NoBox threat model corresponds to the setting where the attacker (i) knows a hypothesis class $\mathcal{F}$ that the target model f_t belongs to, (ii) has access to a reference dataset $\mathcal{D}_{ref}$ that is similar to the the dataset used to train f_t (e.g., sampled from the same distribution), and (iii) can optimize a representative classifier $f_c \in \mathcal{F}$. The attacker has no other knowledge of—or access to—the target model f_t (e.g., no queries to f_t are allowed). The goal is, for the attacker, to use this limited knowledge to corrupt the examples in a given target dataset $\mathcal{D}$.*

Our definition of a NoBox adversary (Def. 1) formalizes similar notions used in previous work (e.g., see Def. 3 in [67]). Previous work also often refers to related settings as generating *blackbox transfer* attacks, since the goal is to attack the target model f_t while only having access to a representative classifier f_c [25, 51, 73].

Note, that our assumptions regarding dataset access are relatively weak. Like prior work, the attacker is given the target data (i.e., the examples to corrupt) as input, but this is constitutive of the task (i.e., we need access to a target example in order to corrupt it). Our only assumption is to have access to a reference dataset $\mathcal{D}_{\text{ref}}$, which is *similar* to the dataset used to train the target model. We do not assume access to the exact training set. A stronger version of this assumption is made in prior works on blackbox transfer, as these approaches must craft their attacks on a known source model which is pretrained on the same dataset as the target model [67].

3 Adversarial Example Games

In order to understand the theoretical feasibility of NoBox attacks, we view the attack generation task as a form of *adversarial game*. The players are the *generator* network g—which learns a conditional distribution over adversarial examples—and the representative classifier f_c. The goal of the generator network is to learn a conditional distribution of adversarial examples, which can fool the representative classifier f_c. The representative classifier f_c, on the other hand, is optimized to detect the true label y from the adversarial examples (x', y) generated by g. A critical insight in this framework is that the generator and the representative classifier are *jointly* optimized in a maximin game, making the generator's adversarial distribution at the equilibrium theoretically effective against *any* classifier from the hypothesis class $\mathcal{F}$ that f_c is optimized over. At the same time, we will see in Proposition 1 that the min and max in our formulation (AEG) can be switched. It implies that, while optimized, the model f_c converges to a *robust classifier* against any attack generated by the generator g [53, 70], leading to increasingly powerful attacks as the adversarial game progresses.

Framework. Given an input-output pair of target datapoints $(x, y) \sim \mathcal{D}$, the generator network g is trained to learn a distribution of adversarial examples $p_{\text{cond}}(\cdot|x, y)$ that—conditioned on an example

[4] Previous work [67] usually assumes to have access to the architecture of f_t; we are more general by assuming access to a hypothesis class $\mathcal{F}$ containing f_t; e.g., DenseNets can represent ConvNets.

to attack (x, y)—maps a prior distribution p_z on $\mathcal{Z}$ onto a distribution on $\mathcal{X}$. The classifier network f_c is simultaneously optimized to perform robust classification over the resulting distribution p_g defined in (2) (below). Overall, the generator g and the classifier f_c play the following, two-player zero-sum game:

$$\max_{g \in \mathcal{G}_\epsilon} \min_{f_c \in \mathcal{F}} \mathbb{E}_{(x,y)\sim\mathcal{D}, z\sim p_z}[\ell(f_c(g(x, y, z)), y)] =: \varphi(f_c, g), \tag{AEG}$$

where the generator $g \in \mathcal{G}_\epsilon$ is restricted by the similarity constraint $d(g(x, y, z), x) \leq \epsilon$, $\forall x, y, z \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$. Once the generator g is trained, one can generate adversarial examples against any classifier in $f_t \in \mathcal{F}$, without queries, by simply sampling $z \sim p_z$ and computing $g(x, y, z)$.

Connection with NoBox attacks. The NoBox threat model (Def. 1) corresponds to a setting where the attacker does not know the target model f_t but only a hypothesis class $\mathcal{F}$ such that $f_t \in \mathcal{F}$. With such knowledge, one cannot hope to be better than the *most pessimistic situation* where f_t is the best defender in $\mathcal{F}$. Our maximin formulation (AEG) encapsulates such a worst-case scenario, where the generator aims at finding attacks against the best performing f in $\mathcal{F}$.

Objective of the generator. When trying to attack infinite capacity classifiers—i.e., $\mathcal{F}$ contains any measurable function—the goal of the generator can be seen as generating the adversarial distribution p_g with the highest expected conditional entropy $\mathbb{E}_x[\sum_y p_g(y|x) \log p_g(y|x)]$, where p_g is defined as

$$(x', y) \sim p_g \Leftrightarrow x' = g(x, y, z)\,,\ (x, y) \sim \mathcal{D}\,,\ z \sim p_z \quad \text{with} \quad d(x', x) \leq \epsilon\,. \tag{2}$$

When trying to attack a specific hypothesis class $\mathcal{F}$ (e.g., a particular CNN architecture), the generator aims at maximizing a notion of restricted entropy defined implicitly through the class $\mathcal{F}$. Thus, the optimal generator in an (AEG) is primarily determined by the statistics of the target dataset $\mathcal{D}$ itself, rather any specifics of a target model. We formalize these high level concepts in §4.2.

Regularizing the Game. In practice, the target f_t is usually trained on a non-adversarial dataset and performs well at a standard classification task. In order to reduce the size of the class $\mathcal{F}$, one can bias the representative classifier f_c towards performing well on a standard classification task with respect to $\mathcal{D}_{\text{ref}}$, which leads to the following game:

$$\max_{g \in \mathcal{G}_\epsilon} \min_{f_c \in \mathcal{F}} \mathbb{E}_{(x,y)\sim\mathcal{D}, z\sim p_z}[\ell(f_c(g(x, y, z)), y)] + \lambda \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{ref}}}[\ell(f_c(x), y)] =: \varphi_\lambda(f, g). \tag{3}$$

Note that $\lambda = 0$ recovers (AEG). Such modifications in the maximin objective as well as setting the way the models are trained (e.g., optimizer, regularization, additional dataset) biases the training of the f_c and corresponds to an implicit incorporation of prior knowledge on the target f_t in the hypothesis class $\mathcal{F}$. We note that in practice, using a non-zero value for λ is essential to achieve the most effective attacks as the prior knowledge acts as a regularizer that incentivizes g to craft attacks against classifiers that behave well on data similar to $\mathcal{D}_{\text{ref}}$.

4 Theoretical results

When playing an adversarial example game, the generator and the representative classifier try to beat each other by maximizing their own objective. In games, a standard notion of optimality is the concept of Nash equilibrium [56] where each player cannot improve its objective value by unilaterally changing its strategy. The minimax result in Prop. 1 implies the existence of a Nash equilibrium for the game, consequently providing a well defined target for learning (we want to learn the equilibrium of that game). Moreover, a Nash equilibrium is a stationary point for gradient descent-ascent dynamics; we can thus hope for achieving such a solution by using a gradient-descent-ascent-based learning algorithm on (AEG).[5]

Proposition 1. *If ℓ is convex (e.g., cross entropy or mean squared loss), the distance $x \mapsto d(x, x')$ is convex for any $x' \in \mathcal{X}$, one has access to any measurable g respecting the proximity constraint in (2), and the hypothesis class $\mathcal{F}$ is convex, then we can switch min and max in (AEG), i.e.,*

$$\min_{f_c \in \mathcal{F}} \max_{g \in \mathcal{G}_\epsilon} \varphi_\lambda(f_c, g) = \max_{g \in \mathcal{G}_\epsilon} \min_{f_c \in \mathcal{F}} \varphi_\lambda(f_c, g) \tag{4}$$

[5]Note that, similarly as in practical GANs training, when the classifier and the generator are parametrized by neural networks, providing convergence guarantees for a gradient based method in such a nonconvex-nonconcave minimax game is an open question that is outside of the scope of this work.

Proof sketch. We first notice that, by (2) any g corresponds to a distribution p_g and thus we have,

$$\varphi(f_c, g) := \mathbb{E}_{(x,y)\sim\mathcal{D}, z\sim p_z}[\ell(f_c(g(x,y,z)), y)] = \mathbb{E}_{(x',y)\sim p_g}[\ell(f_c(x'), y)] =: \varphi(f_c, p_g)$$

Consequently, we also have $\varphi_\lambda(f_c, g) = \varphi_\lambda(f_c, p_g)$. By noting $\Delta_\epsilon := \{p_g \,:\, g \in \mathcal{G}_\epsilon\}$, we have that,

$$\min_{f_c\in\mathcal{F}} \max_{p_g\in\Delta_\epsilon} \varphi_\lambda(f_c, p_g) = \min_{f_c\in\mathcal{F}} \max_{g\in\mathcal{G}_\epsilon} \varphi_\lambda(f_c, g) \quad \text{and} \quad \max_{p_g\in\Delta_\epsilon} \min_{f_c\in\mathcal{F}} \varphi_\lambda(f_c, p_g) = \max_{g\in\mathcal{G}_\epsilon} \min_{f_c\in\mathcal{F}} \varphi_\lambda(f_c, g)$$

In other words, we can replace the optimization over the generator $g \in \mathcal{G}_\epsilon$ with an optimization over the set of possible adversarial distributions Δ_ϵ induced by any $g \in \mathcal{G}_\epsilon$. This equivalence holds by the construction of Δ_ϵ, which ensures that $\max_{g\in\mathcal{G}_\epsilon} \varphi_\lambda(f_c, g) = \max_{p_g\in\Delta_\epsilon} \varphi_\lambda(f_c, p_g)$ for any $f_c \in \mathbb{F}$. We finally use Fan's theorem [28] after showing that $(f_c, p_g) \mapsto \varphi_\lambda(f_c, p_g)$ is convex-concave (by convexity of ℓ and linearity of $p_g \mapsto \mathbb{E}_{p_g}$) and that Δ_ϵ is a compact convex set. In particular, Δ_ϵ is compact convex under the assumption that we can achieve any measurable g (detailed in §A). □

The convexity assumption on the hypothesis class $\mathcal{F}$, Prop. 1 applies in two main cases of interest: (i) infinite capacity, i.e., when $\mathcal{F}$ is any measurable function. (ii) linear classifiers with *fixed* features $\psi : \mathcal{X} \to \mathbb{R}^p$, i.e., $\mathcal{F} = \{w^\top \psi(\cdot)\,,\, w \in \mathbb{R}^{|\mathcal{Y}|\times p}\}$. This second setting is particularly useful to build intuitions on the properties of (AEG), as we will see in §4.1 and Fig. 1. The assumption that we have access to any measurable g, while relatively strong, is standard in the literature and is often stated in prior works as "if g has enough capacity" [33, Prop. 2]. Even if the class of neural networks with a fixed architecture do not verify the assumption of this proposition, the key idea is that neural networks are good candidates to approximate that equilibrium because they are universal approximators [38] and they form a set that is "almost convex" [30]. Proving a similar minimax theorem by only considering neural networks is a challenging problem that has been considered by Gidel et al. [30] in a related setting. It requires a fined grained analysis of the property of a certain neural network architecture and is only valid for approximate minimax. We consider such considerations outside of the scope of this work.

4.1 A simple setup: binary classification with logistic regression

Let us now consider a binary classification setup where $\mathcal{Y} = \{\pm 1\}$ and $\mathcal{F}$ is the class of linear classifiers with linear features, i.e $f_w(x) = w^\top x$. In this case, the payoff of the game (AEG) is,

$$\varphi(f_\omega, g) := \mathbb{E}_{(x,y)\sim\mathcal{D},\, z\sim p_z}[\log(1 + e^{-y\cdot w^\top g(x,y,z)})] \tag{5}$$

This example is similar to the one presented in [33]. However, our purpose is different since we focus on characterizing the optimal generator in (4). We show that the optimal generator can attack any classifier in $\mathcal{F}$ by shifting the means of the two classes of the dataset $\mathcal{D}$.

Proposition 2. *If the generator is allowed to generate any ℓ_∞ perturbations. The optimal linear representative classifier is the solution of the following ℓ_1 regularized logistic regression*

$$w^* \in \arg\min_w \mathbb{E}_{(x,y)\sim\mathcal{D}}[\log(1 + e^{-y\cdot w^\top x + \epsilon\|w\|_1})]\,. \tag{6}$$

Moreover if ω^* has no zero entry, the optimal generator is $g^*(x, y) = x - y \cdot \epsilon\,\mathrm{sign}(w^*)$, is deterministic *and the pair (f_{w^*}, g^*) is a Nash equilibrium of the game* (5)*.*

A surprising fact is that, unlike in the general setting of Prop. 1, the generator in Prop.2 is deterministic (i.e., does not depend on a latent variable z).[6] This follows from the simple structure of classifiers in this class, which allow for a closed form solution for g^*. In general, one cannot expect to achieve an equilibrium with a deterministic generator. Indeed, with this example, our goal is simply to illustrate how the optimal generator can attack an entire class of functions with limited capacity: linear classifiers are mostly sensitive to the mean of the distribution of each class; the optimal generator exploits this fact by moving these means closer to the decision boundary.

[6] Note also that one can generalize Prop. 2 to a perturbation with respect to a general norm $\|\cdot\|$, in that case, the ϵ-regularization for the classifier would be with respect to the dual norm $\|\cdot\|_* := \max_{\|u\|\leq 1}\langle\cdot, u\rangle$. E.g., as previously noted by Goodfellow et al. [33], ℓ_∞ adversarial perturbation leads to a ℓ_1 regularization.

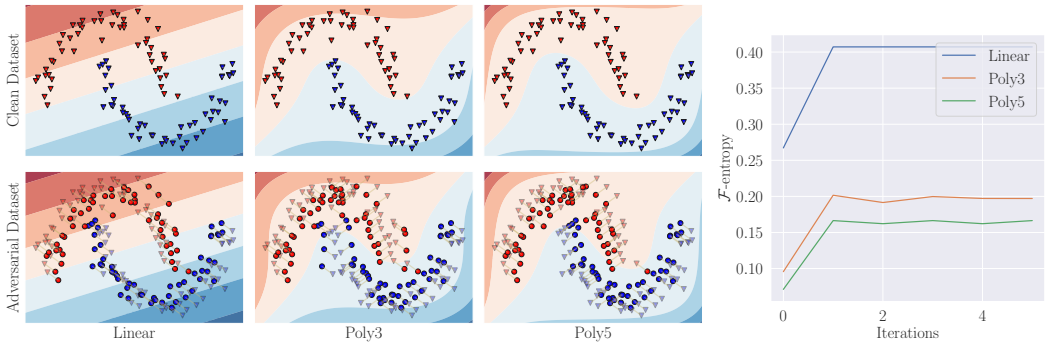

Figure 1: Illustration of Proposition 3 for three classes of classifiers in the context of logistic regression for the two moon dataset of scikit-learn [60] with linear and polynomial (of degree 3 and 5) features. **Left:** Scatter plot of the clean or adversarial dataset and the associated optimal decision boundary. For the adversarial dataset, each corresponding clean example is represented with a ▲/▲ and is connected to its respective adversarial example •/•. **Right:** value of the $\mathcal{F}$-entropy for the different classes as a function of the number of iterations.

4.2 General multi-class classification

In this section, we show that, for a given hypothesis class $\mathcal{F}$, the generated distribution achieving the global maximin against $f_c \in \mathcal{F}$ can be interpreted as the distribution with the highest $\mathcal{F}$-entropy. For a given distribution p_g, its $\mathcal{F}$-entropy is the minimum expected risk under p_g one can achieve in $\mathcal{F}$.

Definition 2. *For a given distribution $(x, y) \sim p_g$ we define the $\mathcal{F}$-entropy of p_g as*

$$H_{\mathcal{F}}(p_g) := \min_{f_c \in \mathcal{F}} \mathbb{E}_{(x,y)\sim p_g}[\ell(f_c(x), y)] \qquad \textit{where } \ell \textit{ is the cross entropy loss.} \tag{7}$$

Thus $\mathcal{F}$-entropy quantifies the amount of "classification information" available in p_g using the class of classifiers $\mathcal{F}$. If the $\mathcal{F}$-entropy is large, $(x, y) \sim p_g$ cannot be easily classified with a function f_c in $\mathcal{F}$. Moreover, it is an upper-bound on the *expected conditional entropy* of the distribution p_g.

Proposition 3. *The $\mathcal{F}$-entropy is a decreasing function of $\mathcal{F}$, i.e., for any $\mathcal{F}_1 \subset \mathcal{F}_2$,*

$$H_{\mathcal{F}_1}(p_g) \geq H_{\mathcal{F}_2}(p_g) \geq H_y(p_g) := \mathbb{E}_{x\sim p_x}[H(p_g(\cdot|x))]\,.$$

where $H(p(\cdot|x)) := \sum_{y\in\mathcal{Y}} p(y|x) \ln p(y|x)$ is the entropy of the conditional distribution $p(y|x)$.

For a given class $\mathcal{F}$, the solution to an (AEG) game can be seen as one which finds an adversarial distribution of maximal $\mathcal{F}$-entropy regularized towards attacking the particular model f_s,

$$\max_{g\in\mathcal{G}_\epsilon} \min_{f_c\in\mathcal{F}} \varphi_\lambda(f_c, g) = \max_{g\in\mathcal{G}_\epsilon} H_{\mathcal{F}}(p_g) + \lambda \mathbb{E}_{p_g}[\ell(f_s(x), y)]\,. \tag{8}$$

This alternative perspective on the game (AEG) shares similarities with the divergence minimization perspective on GANs [40]. However, while in GANs it represents a divergence between two distributions, in (AEG) this corresponds to a notion of entropy.

A high-level interpretation of $\mathcal{F}$-entropy maximization is that it implicitly defines a metric for distributions which are challenging to classify with only access to classifiers in $\mathcal{F}$. Overall, the optimal generated distribution p_g can be seen as the most adversarial dataset against the class $\mathcal{F}$.

Properties of the $\mathcal{F}$-entropy. We illustrate the idea that the optimal generator and the $\mathcal{F}$-entropy depend on the hypothesis class $\mathcal{F}$ using a simple example. To do so, we perform logistic regression (5) with linear and polynomial (of degree 3 and 5) features (respectively called Linear, Poly3, and Poly5) on the two moon dataset of scikit-learn [60]. Note that we have Linear $\subset$ Poly3 $\subset$ Poly5. For simplicity, we consider a deterministic generator $g(x, y)$ that is realized by computing the maximization step via 2D grid-search on the ϵ neighborhood of x. We train our models by successively fully solving the minimization step and the maximization step in (5).

We present the results in Figure 1. One iteration corresponds to the computation of the optimal classifier against the current adversarial distribution p_g (also giving the value of the $\mathcal{F}$-entropy), followed by the computation of the new optimal adversarial p'_g against this new classifier. The left plot illustrates the fact that the way of attacking a dataset depends on the class considered. For instance, when considering linear classifiers, the attack is a uniform translation on all the data-points of the same class. While when considering polynomial features, the optimal adversarial dataset pushes the the corners of the two moons closer together. In the right plot, we can see an illustration of Proposition 3, where the $\mathcal{F}$-entropy takes on a smaller value for larger classes of classifiers.

5 Attacking in the Wild: Experiments and Results

We investigate the application of our AEG framework to produce adversarial examples against MNIST and CIFAR-10 classifiers. First we investigate our performance in a challenging NoBox setting where we must attack an unseen target model with knowledge of only its hypothesis class (i.e., architecture) and a sample of similar training data (§5.1). Following this, we investigate how well AEG attacks transfer across architectures (§5.2), as well as AEG's performance attacking robust classifiers (§5.3).

Experimental setup. We perform all attacks, including baselines, with respect to the ℓ_∞ norm constraint with $\epsilon = 0.3$ for MNIST and $\epsilon = 0.03125$ for CIFAR-10. For AEG models, we train both generator (g) and representative classifier (f_c) using stochastic gradient descent-ascent with the ExtraAdam optimizer [29] and held out target models, f_t, are trained offline using SGD with Armijo line search [69]. Full details of our model architectures, including hyperparameters, employed in our AEG framework can be found in Appendix §D.

Baselines. Throughout our experiments we rely on four standard blackbox transfert attack strategies adapted to the NoBox setting: the Momentum-Iterative Attack (MI-Attack) [24], the Input Diversity (DI-Attack) [73], the Translation-Invariant (TID-Attack) [25] and the Skip Gradient Method (SGM-Attack) [71]. For fair comparison, we inherit all hyperparameter settings from their respective papers. Note that SGM-attack is only defined with architectures that contain skip connections (e.g. ResNets).

5.1 NoBox Attacks on a Known Architecture Class but Unknown Train Set

We first evaluate the AEG framework in a NoBox setting, where we know only the architecture of the target model and have only access to a sample of similar training data (but not the exact training data of the target model). To simulate having access to a similar (but not identical dataset) as the target model, for each dataset we create random equally-sized splits of the data (10000 examples per splits). Within each split we use one fold to train the split classifier which acts as the representative classifiers for all attackers who are then evaluated their ability to fool the remaining split classifiers on unseen target examples $\mathcal{D}$. For the MNIST dataset we consider LeNet classifier [47], while for CIFAR-10 we consider ResNet-18 [37]. Table 5.1 shows the results of our experiments on this task, averaged across all splits and folds. We see that our AEG approach achieves state-of-the-art results, either outperforming or matching (within a 95% confidence interval) all baselines in both settings. Note that this task is significantly more challenging than many prior blackbox attack setups, which assume access to the full training data of the target model.[7]

Dataset	MI-Attack	DI-Attack	TID-Attack	SGM-Attack	AEG (Ours)
MNIST	87.5 ± 2.7	**89.5 ± 2.5**	85.4 ± 2.8 †	N/A	**89.5 ± 3.2**
CIFAR-10 (Res18)	56.8 ± 1.2 †	84.0 ± 1.5 †	9.1 ± 1.6 †	60.5 ± 1.5 †	**87.0 ± 2.1**

Table 1: Attack success rates, averaged across target models with 95% confidence intervals shown. † signifies a statistically significant result as determined by the paired T-test when compared to AEG.

5.2 NoBox Attacks Across Distinct Architectures

We now consider NoBox attacks where we do not know the architecture of the target model but where the training data is known—a setting previously referred to as blackbox transfer [67]. For evaluation, we use CIFAR-10 and train 10 instances of VGG-16 [62], ResNet-18 (RN-18) [37], Wide ResNet (WR) [75], DenseNet-121 (DN-121) [39] and Inception-V3 architectures (Inc-V3) [66]. Here, we optimize the attack approaches against a single pre-trained classifier from a particular architecture and then evaluate their attack success on classifiers from distinct architectures averaged over 5 instantiations. Our findings when using ResNet-18, DenseNet-121 and the VGG-16 as the source architecture are provided in Table 2. Overall we find that AEG beats all other approaches and lead to a new state of the art. In particular AEG outperforms the best baseline in each setting by an average of 18.4% across the different source architectures with individual average gains of 6.2%, 22.6%, and 26.5 when using a RN-18 model, DN-121, and VGG-16 source models respectively.

[7] We include results on a more permissive settings with access to the full training data in Appendix C.1

Source	Attack	VGG-16	RN-18	WR	DN-121	Inc-V3
	Clean	11.2 ± 0.9	13.1 ± 2.0	6.8 ± 0.7	11.2 ± 1.4	9.9 ± 1.3
RN-18	MI-Attack	63.9 ± 1.3	74.6 ± 0.4	63.1 ± 1.2	72.5 ± 1.3	67.9 ± 1.6
	DI-Attack	77.4 ± 1.7	90.2 ± 0.8	74.0 ± 1.0	87.1 ± 1.3	$\mathbf{85.8 \pm 0.8}$
	TID-Attack	21.6 ± 1.3	26.5 ± 2.2	14.0 ± 1.5	22.3 ± 1.6	19.8 ± 0.9
	SGM-Attack	68.4 ± 1.8	79.5 ± 0.5	64.3 ± 1.6	73.8 ± 1.0	70.6 ± 1.7
	AEG (Ours)	$\mathbf{89.0 \pm 2.1}$	$\mathbf{96.8 \pm 0.7}$	$\mathbf{80.9 \pm 2.4}$	$\mathbf{91.6 \pm 1.7}$	$\mathbf{87.2 \pm 1.6}$
DN-121	MI-Attack	54.3 ± 1.1	62.5 ± 0.9	56.3 ± 1.3	66.1 ± 1.5	65.0 ± 1.3
	DI-Attack	61.1 ± 1.9	69.1 ± 0.8	61.9 ± 1.1	77.1 ± 1.2	71.6 ± 1.6
	TID-Attack	21.7 ± 1.2	23.8 ± 1.5	14.0 ± 1.4	21.7 ± 1.1	19.3 ± 1.2
	SGM-Attack	51.6 ± 0.7	60.2 ± 1.3	52.6 ± 0.9	64.7 ± 1.6	61.4 ± 1.3
	AEG (Ours)	$\mathbf{90.5 \pm 1.6}$	$\mathbf{95.9 \pm 1.4}$	$\mathbf{80.3 \pm 2.3}$	$\mathbf{95.9 \pm 1.4}$	$\mathbf{90.6 \pm 2.4}$
VGG-16	MI-Attack	49.9 ± 0.1	50.0 ± 0.2	46.7 ± 0.4	50.4 ± 0.6	50.0 ± 0.3
	DI-Attack	65.1 ± 0.1	64.5 ± 0.2	58.8 ± 0.6	64.1 ± 0.3	60.9 ± 0.6
	TID-Attack	26.2 ± 0.6	24.0 ± 0.6	13.0 ± 0.2	20.8 ± 0.7	18.8 ± 0.2
	AEG (Ours)	$\mathbf{94.2 \pm 1.2}$	$\mathbf{93.7 \pm 1.6}$	$\mathbf{77.1 \pm 1.1}$	$\mathbf{92.3 \pm 1.7}$	$\mathbf{86.5 \pm 1.3}$

Table 2: Error rates on $\mathcal{D}$ for average NoBox architecture transfer attacks with $\epsilon = 0.03125$

5.3 NoBox Attacks Against Robust Classifiers

We now test the ability of our AEG framework to attack target models that have been robustified using adversarial and ensemble adversarial training [53, 67]. For evaluation against PGD adversarial training, we use the public models as part of the MNIST and CIFAR-10 adversarial examples challenge [8]. For ensemble adversarial training, we follow the approach of Tramèr et al. [67] (see Appendix D.4). We report our results in Table 3 and average the result of stochastic attacks over 5 runs. We find that AEG achieves state-of-the-art performance in all settings, proving an average improvement in success rates of 14.3% on ensemble-robustified MNIST models, 2.8% on PGD-robustified MNIST models, and 12.9% across all robustified models on CIFAR-10.

Dataset	Defence	Clean	MI-Att†	DI-Att	TID-Att	SGM-Att †	AEG (Ours)
MNIST	A_{ens4}	0.8	43.4	42.7	16.0	N/A	**65.0**
	B_{ens4}	0.7	20.7	22.8	8.5	N/A	**50.0**
	C_{ens4}	0.8	73.8	30.0	9.5	N/A	**80.0**
	D_{ens4}	1.8	84.4	76.0	81.3	N/A	**86.7**
	Madry-Adv	0.8	2.0	3.1	2.5	N/A	**5.9**
CIFAR-10	$RN\text{-}18_{ens3}$	16.8	17.6	21.6	33.1	19.9	**52.2**
	WR_{ens3}	12.8	18.4	20.6	28.8	18.0	**49.9**
	$DN\text{-}121_{ens3}$	21.5	20.3	22.7	31.3	21.9	**41.4**
	$Inc\text{-}V3_{ens3}$	14.8	19.5	42.2*	30.2*	35.5*	**47.5**
	Madry-Adv	12.9	17.2	16.6	16.6	16.0	**21.6**

Table 3: Error rates on $\mathcal{D}$ for NoBox known architecture attacks against Adversarial Training and Ensemble Adversarial Training. * Attacks were done using WR. † Deterministic attack.

6 Related Work

In addition to non-interactive blackbox adversaries we compare against, there exists multiple hybrid approaches that combine crafting attacks on surrogate models which then serve as a good initialization point for queries to the target model [57, 61, 41]. Other notable approaches to craft blackbox transfer attacks learning ghost networks [48], transforming whitebox gradients with small ResNets [49], and transferability properties of linear classifiers and 2-layer ReLu Networks [17]. There is also a burgeoning literature of using parametric models to craft adversarial attacks such as the Adversarial Transformation Networks framework and its variants [4, 72]. Similar in spirit to our approach many

[8] `https://github.com/MadryLab/[x]_challenge`, for `[x]` in {`cifar10`, `mnist`}. Note that our threat model is more challenging than these challenges as we use non-robust source models.

attacks strategies benefit from employing a latent space to craft attacks [76, 68, 9]. However, unlike our work, these strategies cannot be used to attack entire hypothesis classes.

Adversarial prediction games between a learner and a data generator have also been studied in the literature [11], and in certain situations correspond to a Stackelberg game Brückner and Scheffer [10]. While similar in spirit, our theoretical framework is tailored towards crafting adversarial attacks against a fixed held out target model in the novel NoBox threat model and is a fundamentally different attack paradigm. Finally, Erraqabi et al. [27] also investigate an adversarial game framework as a means for building robust representations in which an additional discriminator is trained to discriminate adversarial example from natural ones, based on the representation of the current classifier.

7 Conclusion

In this paper, we introduce the Adversarial Example Games (AEG) framework which provides a principled foundation for crafting adversarial attacks in the NoBox threat model. Our work sheds light on the existence of adversarial examples as a natural consequence of restricted entropy maximization under a hypothesis class and leads to an actionable strategy for attacking all functions taken from this class. Empirically, we observe that our approach leads to state-of-the-art results when generating attacks on MNIST and CIFAR-10 in a number of challenging NoBox attack settings. Our framework and results point to a promising new direction for theoretically-motivated adversarial frameworks. However, one major challenge is scaling up the AEG framework to larger datasets (e.g., ImageNet), which would involve addressing some of the inherent challenges of saddle point optimization [5]. Investigating the utility of the AEG framework for training robustified models is another natural direction for future work.

Broader Impact

Adversarial attacks, especially ones under more realistic threat models, pose several important security, ethical, and privacy risks. In this work, we introduce the NoBox attack setting, which generalizes many other blackbox transfer settings, and we provide a novel framework to ground and study attacks theoretically and their transferability to other functions within a class of functions. As the NoBox threat model represents a more realistic setting for adversarial attacks, our research has the potential to be used against a class of machine learning models in the wild. In particular, in terms of risk, malicious actors could use approaches based on our framework to generate attack vectors that compromise production ML systems or potentially bias them toward specific outcomes.

As a concrete example, one can consider creating transferrable examples in the physical world, such as the computer vision systems of autonomous cars. While prior works have shown the possibility of such adversarial examples —i.e., adversarial traffic signs, we note that there is a significant gap in translating synthetic adversarial examples to adversarial examples that reside in the physical world [45]. Understanding and analyzing the NoBox transferability of adversarial examples to the physical world—in order to provide public and academic visibility on these risks—is an important direction for future research. Based on the known risks of designing new kinds of adversarial attacks—discussed above—we now outline the ways in which our research is informed by the intent to mitigate these potential societal risks. For instance, our research demonstrates that one can successfully craft adversarial attacks even in the challenging NoBox setting. It raises many important considerations when developing robustness approaches. A straightforward extension is to consider our adversarial example game (AEG) framework as a tool for training robust models. On the theoretical side, exploring formal verification of neural networks against NoBox adversaries is an exciting direction for continued exploration. As an application, ML practitioners in the industry may choose to employ new forms of A/B testing with different types of adversarial examples, of which AEG is one method to robustify and stress test production systems further. Such an application falls in line with other general approaches to red teaming AI systems [10] and verifiability in AI development. In essence, the goal of such approaches, including adversarial examples for robustness, is to align AI systems' failure modes to those found in human decision making.

Acknowledgements

The authors would like to acknowledge Olivier Mastropietro, Chongli Qin and David Balduzzi for helpful discussions as well as Sebastian LaChapelle, Pouya Bashivan, Yanshuai Cao, Gavin Ding, Ioannis Mitliagakas, and Nadeem Ward for reviewing early drafts of this work. This work is partially supported by the Canada CIFAR AI Chair Program (held at Mila), NSERC Discovery Grant RGPIN-2019-05123 (held by Will Hamilton at McGill), NSERC Discovery Grant RGPIN-2017-06936, and a Google Focused Research award (both held at U. Montreal by Simon Lacoste-Julien). Joey Bose was also supported by an IVADO PhD fellowship, Gauthier Gidel by a Borealis AI fellowship and by the Canada Excellence Research Chair in "Data Science for Real-Time Decision-making" (held at Polytechnique by Andrea Lodi), and Andre Cianflone by a NSERC scholarship. Simon Lacoste-Julien and Pascal Vincent are CIFAR Associate Fellows in the Learning in Machines & Brains program. Finally, we thank Facebook for access to computational resources.

References

[1] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[2] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. Square attack: a query-efficient black-box adversarial attack via random search. *arXiv preprint arXiv:1912.00049*, 2019.

[3] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. In *ICML*, 2018.

[4] S. Baluja and I. Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

[5] H. Berard, G. Gidel, A. Almahairi, P. Vincent, and S. Lacoste-Julien. A closer look at the optimization landscapes of generative adversarial networks. In *ICLR*, 2020.

[6] S. Bhambri, S. Muku, A. Tulasi, and A. Balaji Buduru. A survey of black-box adversarial attacks on computer vision models. *arXiv*, pages arXiv–1912, 2019.

[7] P. Billingsley. *Convergence of probability measures*. John Wiley & Sons, 1999.

[8] A. J. Bose and P. Aarabi. Adversarial attacks on face detectors using neural net based constrained optimization. In *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2018.

[9] A. J. Bose, A. Cianflone, and W. Hamiltion. Generalizable adversarial attacks using generative models. *arXiv preprint arXiv:1905.10864*, 2019.

[10] M. Brückner and T. Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 547–555, 2011.

[11] M. Brückner, C. Kanzow, and T. Scheffer. Static prediction games for adversarial learning problems. *The Journal of Machine Learning Research*, 13(1):2617–2654, 2012.

[12] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. 2018.

[13] N. Carlini and D. Wagner. Magnet and" efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017.

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

[15] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[16] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[17] Z. Charles, H. Rosenberg, and D. Papailiopoulos. A geometric perspective on the transferability of adversarial directions. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1960–1968, 2019.

[18] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

[19] F. Croce and M. Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. *arXiv preprint arXiv:1907.02044*, 2019.

[20] F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv preprint arXiv:2003.01690*, 2020.

[21] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018.

[22] G. W. Ding, L. Wang, and X. Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[23] G. W. Ding, Y. Sharma, K. Y. C. Lui, and R. Huang. MMA training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HkeryxBtPB.

[24] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[25] Y. Dong, T. Pang, H. Su, and J. Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4312–4321, 2019.

[26] J. Du, H. Zhang, J. T. Zhou, Y. Yang, and J. Feng. Query-efficient meta attack to deep neural networks. *arXiv preprint arXiv:1906.02398*, 2019.

[27] A. Erraqabi, A. Baratin, Y. Bengio, and S. Lacoste-Julien. A3t: Adversarially augmented adversarial training. *arXiv preprint arXiv:1801.04055*, 2018.

[28] K. Fan. Minimax theorems. *Proceedings of the National Academy of Sciences of the United States of America*, 1953.

[29] G. Gidel, H. Berard, G. Vignoud, P. Vincent, and S. Lacoste-Julien. A variational inequality perspective on generative adversarial networks. In *ICLR*, 2019.

[30] G. Gidel, D. Balduzzi, W. M. Czarnecki, M. Garnelo, and Y. Bachrach. Minimax theorem for latent games or: How i learned to stop worrying about mixed-nash and love neural nets. *arXiv preprint arXiv:2002.05820*, 2020.

[31] Z. Gong, W. Wang, and W.-S. Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1705.04960*, 2017.

[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[33] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[34] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[35] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

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

[37] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[38] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 1991.

[39] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[40] G. Huang, H. Berard, A. Touati, G. Gidel, P. Vincent, and S. Lacoste-Julien. Parametric adversarial divergences are good task losses for generative modeling. *arXiv preprint arXiv:1708.02511*, 2017.

[41] Z. Huang and T. Zhang. Black-box adversarial attack with transferable model-based embedding. *arXiv preprint arXiv:1911.07140*, 2019.

[42] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Query-efficient black-box adversarial examples. *arXiv preprint arXiv:1712.07113*, 2017.

[43] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

[44] A. Ilyas, L. Engstrom, and A. Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *arXiv preprint arXiv:1807.07978*, 2018.

[45] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[46] L. Jiang, X. Ma, S. Chen, J. Bailey, and Y.-G. Jiang. Black-box adversarial attacks on video recognition models. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 864–872, 2019.

[47] Y. LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 2015.

[48] Y. Li, S. Bai, Y. Zhou, C. Xie, Z. Zhang, and A. Yuille. Learning transferable adversarial examples via ghost networks. *arXiv preprint arXiv:1812.03413*, 2018.

[49] Y. Li, S. Bai, C. Xie, Z. Liao, X. Shen, and A. L. Yuille. Regional homogeneity: Towards learning transferable universal adversarial perturbations against defenses. *arXiv preprint arXiv:1904.00979*, 2019.

[50] Y. Li, L. Li, L. Wang, T. Zhang, and B. Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. *arXiv preprint arXiv:1905.00441*, 2019.

[51] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[52] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[53] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[54] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[55] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[56] J. Nash. Non-cooperative games. *Annals of mathematics*, 1951.

[57] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

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

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

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011.

[61] Y. Shi, S. Wang, and Y. Han. Curls & whey: Boosting black-box adversarial attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6519–6527, 2019.

[62] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[63] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

[64] L. Sun, M. Tan, and Z. Zhou. A survey of practical adversarial example attacks. *Cybersecurity*, 1(1):9, 2018.

[65] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[66] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[67] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[68] C.-C. Tu, P. Ting, P.-Y. Chen, S. Liu, H. Zhang, J. Yi, C.-J. Hsieh, and S.-M. Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[69] S. Vaswani, A. Mishkin, I. Laradji, M. Schmidt, G. Gidel, and S. Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *Advances in Neural Information Processing Systems*, pages 3727–3740, 2019.

[70] A. Wald. Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, 1945.

[71] D. Wu, Y. Wang, S.-T. Xia, J. Bailey, and X. Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. *arXiv preprint arXiv:2002.05990*, 2020.

[72] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

[73] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2730–2739, 2019.

[74] H. Xu, Y. Ma, H. Liu, D. Deb, H. Liu, J. Tang, and A. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *arXiv preprint arXiv:1909.08072*, 2019.

[75] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[76] Z. Zhao, D. Dua, and S. Singh. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017.

Adversarial Example Games Supplementary Materials

A Proofs for Section 4 (Theoretical results)

Proposition 1. *If ℓ is convex (e.g., cross entropy or mean squared loss), the distance $x \mapsto d(x, x')$ is convex for any $x' \in \mathcal{X}$, one has access to any measurable g respecting the proximity constraint in (2), and the hypothesis class $\mathcal{F}$ is convex, then we can switch min and max in (AEG), i.e.,*

$$\min_{f_c \in \mathcal{F}} \max_{g \in \mathcal{G}_\epsilon} \varphi_\lambda(f_c, g) = \max_{g \in \mathcal{G}_\epsilon} \min_{f_c \in \mathcal{F}} \varphi_\lambda(f_c, g) \tag{4}$$

Proof. Let us recall that the payoff φ is defined as

$$\max_{g \in \mathcal{G}_\epsilon} \min_{f \in \mathcal{F}} \mathbb{E}_{(x,y)\sim p_{data}, z\sim p_z}[\ell(f(g(x, y, z)), y)] =: \varphi(f, g) \tag{9}$$

Let us consider the case where $\mathcal{X}$ is finite as a warm-up. In practice, this can be the case if we consider that for instance one only allow a finite number of values for the pixels, e.g. (integers between 0 and 255 for CIFAR-10). In that case we have that

$$\mathbb{P}_{adv}(x, y) = \sum_{x' \in \mathcal{X}} \mathbb{P}_{data}(x', y)\mathbb{P}_g(x|x', y) \quad \text{where} \quad d(x, x') > \epsilon \Rightarrow \mathbb{P}_g(x|x', y) = 0\,. \tag{10}$$

Assuming that one can achieve any $\mathbb{P}_g(\cdot|x', y)$ respecting the proximity constraint, the set $\{\mathbb{P}_{adv}\}$ is convex and compact. It is compact because closed and bounded in finite dimension and convex because of the linear dependence in $\mathbb{P}_g$ in (10) (and the fact that if $\mathbb{P}_{g_1}$ and $\mathbb{P}_{g_2}$ respect the constraints then $\lambda\mathbb{P}_{g_1} + (1 - \lambda)\mathbb{P}_{g_2}$ does it too).

For the non finite input case, we can consider that the generator is a random variable defined on the probability space $(\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}, \mathcal{B}, \mathbb{P}_{(x,y)} \times \mathbb{P}_z := \mathbb{P})$ where $\mathcal{B}$ is the Borel σ-algebra, $\mathbb{P}_{(x,y)}$ the probability on the space of data and $\mathbb{P}_z$ the probability on the latent space. Then the adversarial distributions p_g we consider is the pushforward distributions $\mathbb{P} \circ G^{-1}$ with G such that,

$$G(x, y, z) = (g(x, y, z), y) \quad \text{and} \quad d(g(x, y, z), x) \leq \epsilon\,,\ \forall x, y, z \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}\,. \tag{11}$$

It implies that $\Delta_\epsilon := \{p_g \,:\, g \in \mathcal{G}_\epsilon\} = \{\mathbb{P} \circ G^{-1} \,|\, G \text{ measurable satisfying (11)}\}$. By definition of the payoff φ and (2), we have that

$$\varphi(f_c, g) := \mathbb{E}_{(x,y)\sim\mathcal{D}, z\sim p_z}[\ell(f_c(g(x, y, z)), y)] = \mathbb{E}_{(x',y)\sim p_g}[\ell(f_c(x'), y)] =: \varphi(f_c, p_g)$$

and thus it lead to the equivalence,

$$\min_{f_c \in \mathcal{F}} \max_{p_g \in \Delta_\epsilon} \varphi_\lambda(f_c, p_g) = \max_{p_g \in \Delta_\epsilon} \min_{f_c \in \mathcal{F}} \varphi_\lambda(f_c, p_g) \iff \min_{f_c \in \mathcal{F}} \max_{g \in \mathcal{G}_\epsilon} \varphi_\lambda(f_c, g) = \max_{g \in \mathcal{G}_\epsilon} \min_{f_c \in \mathcal{F}} \varphi_\lambda(f_c, g)$$

Recall that we assumed that one has access to any measurable g that satisfies (11), the minimax problem (4). Let us show that under this assumption the set Δ_ϵ is convex and compact.

Δ_ϵ is convex: let us consider $\mathbb{P} \circ G_1^{-1}$ and $\mathbb{P} \circ G_2^{-1}$ we have that

$$\lambda\mathbb{P} \circ G_1^{-1} + (1 - \lambda)\mathbb{P} \circ G_2^{-1} = \mathbb{P} \circ G_3^{-1} \tag{12}$$

where $g_3(x, y, z) = \delta(z)g_1(x, y, z) + (1 - \delta(z))g_2(x, y, z)$ and where $\delta \sim Ber(\lambda)$. Note that g_3 satisfies (11) by convexity of $x \mapsto d(x, x')$.

Δ_ϵ is compact: By using Skorokhod's representation theorem [7] we can show that this set is closed and thus compact (as closed subsets of compact sets are compact) using the weak convergence of measures as topology.

Thus Δ_ϵ is a convex compact Haussdorf space and in both cases ($\mathcal{X}$ finite and infinite) and we can apply Fan's Theorem.

Theorem 1. *[28, Theorem 2] Let U be a compact and convex Hausdorff space and V an arbitrary convex set. Let φ be a real valued function on $U \times V$ such that for every $v \in V$ the function $\varphi(\cdot, v)$ is lower semi-continuous on U. If φ is convex-concave then,*

$$\min_{u \in U} \sup_{v \in V} \varphi(u, v) = \sup_{v \in V} \min_{u \in U} \varphi(u, v) \tag{13}$$

Note that we do not prove this result for neural networks. (□

Proposition 2. *If the generator is allowed to generate any ℓ_∞ perturbations. The optimal linear representative classifier is the solution of the following ℓ_1 regularized logistic regression*

$$w^* \in \arg\min_w \mathbb{E}_{(x,y)\sim\mathcal{D}}[\log(1+e^{-y\cdot w^\top x+\epsilon\|w\|_1})]\,. \tag{6}$$

Moreover if ω^* has no zero entry, the optimal generator is $g^*(x,y) = x - y\cdot\epsilon\operatorname{sign}(w^*)$, is deterministic *and the pair (f_{w^*}, g^*) is a Nash equilibrium of the game (5).*

Proof. We prove the result here for any given norm $\|\cdot\|$. Let us consider the loss for a given pair (x,y)

$$\log(1+e^{y(w^\top g(x,y)+b)}) \tag{14}$$

then by the fact that $x\mapsto\log(1+e^x)$ is increasing, maximizing this term for $\|g(x,y)-x\|_\infty\leq\epsilon$, boils down to solve the following maximization step,

$$\max_{\delta\,,\,\|\delta\|\leq\epsilon} y(w^\top\delta) = \|w\|_* \tag{15}$$

Particularly, for the ℓ_∞ norm we get

$$\arg\max_{\delta\,,\,\|\delta\|_\infty\leq\epsilon} y(w^\top\delta) = \epsilon y\operatorname{sign}(w)\,. \tag{16}$$

and

$$\max_g \mathbb{E}_{(x,y)\sim p_{data}}[\log(1+e^{y(w^\top g(x,y)+b)})] = \mathbb{E}_{(x,y)\sim p_{data}}[\log(1+e^{y(w^\top x+b)+\epsilon\|w\|_1})] \tag{17}$$

To show that (f^*,g^*) is a Nash equilibrium of the game (5), we first notice that by construction

$$\varphi(f^*,g^*) = \min_{f\in\mathcal{F}}\max_g\varphi(f,g) \tag{18}$$

where $\mathcal{F}$ is the class of classifier with linear logits. We then just need to notice that for all $f\in\mathcal{F}$ we have,

$$\varphi(f,g^*) = \mathbb{E}_{(x,y)\sim p_{data}}[\log(1+e^{-y(w^\top x+b)+\epsilon w^\top\operatorname{sign}(w^*)})] \tag{19}$$

That is a convex problem in (w,b). Since, in a neighborhood of w^* we have that $w^\top\operatorname{sign}(w^*) = \|w\big|_{\operatorname{supp}(w^*)}\|_1$. Thus by assuming that w^* is full support and since w^* minimize (6) we have that

$$\nabla_w\varphi(w^*,b^*,g^*) = \nabla_w\mathbb{E}_{(x,y)\sim p_{data}}[\log(1+e^{-y(w^\top x+b)+\epsilon\|w\|_1})] = 0 \tag{20}$$

Finally, by convexity of the problem (19) we can conclude that w^* is a minimizer of $\varphi(\cdot,g^*)$. To sum-up we have that

$$\varphi(f^*,g)\leq\varphi(f^*,g^*)\leq\varphi(f,g^*) \tag{21}$$

meaning that (f^*,g^*) is a Nash equilibrium of (5). □

Proposition 3. *The $\mathcal{F}$-entropy is a decreasing function of $\mathcal{F}$, i.e., for any $\mathcal{F}_1\subset\mathcal{F}_2$,*

$$H_{\mathcal{F}_1}(p_g)\geq H_{\mathcal{F}_2}(p_g)\geq H_y(p_g) := \mathbb{E}_{x\sim p_x}[H(p_g(\cdot|x))]\,.$$

where $H(p(\cdot|x)) := \sum_{y\in\mathcal{Y}}p(y|x)\ln p(y|x)$ is the entropy of the conditional distribution $p(y|x)$.

Proof. In the case where $f^*(x) := p(y|x)\in\mathcal{F}$, since f^* a minimizer of the expected cross entropy loss over the class of any function, we have that

$$H_y(p) := \min_{f\in\mathcal{X}^\mathcal{Y}}\mathbb{E}_{(x,y)\sim p}[\ell(f(x),y)] = \mathbb{E}_x[H(p(\cdot|x))] \tag{22}$$

Lemma 1. *Given a data distribution $(x,y)\sim p_{adv}$ a minimizer of the cross entropy loss is*

$$p_{adv}(y|\cdot)\in\arg\min_f\mathbb{E}_{(x,y)\sim p_{adv}}[\ell(f(x),y)]\,. \tag{23}$$

Proof. Let us start by noticing that,

$$\min_f \mathbb{E}_{(x,y)\sim p}[\ell(f(x), y)] = \mathbb{E}_{x\sim p_x} \min_{q=f(x)} \mathbb{E}_{y\sim p(\cdot|x)}[\ell(q, y)] \tag{24}$$

Using the fact that ℓ is the cross-entropy loss we get

$$\mathbb{E}_{y\sim p(\cdot|x)}[\ell(q, y)] = \mathbb{E}_{y\sim p(\cdot|x)}[-\sum_{i=1}^{K} y_i \ln(q_i)] = -\sum_{i=1}^{K} p_i \ln(q_i) \tag{25}$$

where we noted $p_i = p(y = i|x)$. Noticing that since q_i is a probability distribution we have $\sum_{i=1}^{K} q_i = 1$, we have,

$$\mathbb{E}_{y\sim p(\cdot|x)}[\ell(q, y)] = -\sum_{i=1}^{K-1} p_i \ln(q_i) - p_K \ln(1 - \sum_{i=1}^{K-1} q_i) \tag{26}$$

we can then differentiate this loss with respect to $q_i \geq 0$ and get,

$$\frac{\partial \mathbb{E}_{y\sim p(\cdot|x)}[\ell(q, y)]}{\partial q_i}(q) = -\frac{p_i}{q_i} + \frac{p_K}{q_K} \tag{27}$$

We can finally notice that $q_i = p_i$ is a feasible solution. □

□

B Experimental Details

The experiments are subject to different source of variations, in all our experiments we try to take into account those source of variations when reporting the results. We details the different source of variations for each experiments and how we report them in the next section.

B.1 Source of variations

NoBox attacks on a known architecture class we created a random split of the MNIST and CIFAR10 dataset, and trained a classifier on each splits. To evaluate each method we then use one of the classifiers as the source classifier and all the other classifiers as the targets we want to attack. We then compute the mean and standard deviation of the attack success rates across all target classifiers. To take into account the variability in the results that comes from using a specific classifier as the source model, we also repeat the evaluation by changing the source model. We report the average and 95% interval (assuming the results follow a normal distribution) in Table 5.1 by doing macro-averaging over all evaluations.

NoBox attacks across distinct architectures For each architecture we trained 10 different models. When evaluating against a specific architecture we evaluate against all models of this architecture. In Table 2, we report the mean and standard deviation of the error rates across all models.

NoBox Attacks against robust classifiers For this experiment we could only trained a single robust target model per architecture because of our computational budget. The only source of variations is thus due to the inherent stochasticity of each method. Evaluating this source of randomness would requires to run each method several times, unfortunately this is quite expensive and our computational budget didn't allow for it. In Table 3, we thus only report a single number per architecture.

C Additional results

C.1 Quantitative Results

We now provide additional results in the form of whitebox and blackbox query attacks adapted to the NoBox evaluation protocol for Known-Architecture attacks which is the experimental setting in

Q1. For whitebox attacks we evaluate APGD-CE and APGD-DLR [20] which are improvements over the powerful PGD attack [53]. When ensembled with another powerful perturbation minimizing whitebox attack FAB [19] and the query efficient blackbox Square attacks [2] yields the current SOTA attack strategy called AutoAttack [20] [20]. Additionally, we compare with two parametric blackbox query approaches that both utilize a latent space in AutoZoom [68] and $\mathcal{N}$Attack [50]. To test transferability of whitebox and blackbox query attacks in the NoBox known architecture setting we give generous iteration and query budgets (10x the reported settings in the original papers) when attacking the source models, but only a single query for each target model. It is interesting to note that APGD variant whitebox attacks are significantly more effective than query based blackbox attacks but lack the same effectiveness of NoBox baselines. We hypothesize that the transferability of whitebox attacks may be due to the fact that different functions learn similar decision boundaries but different enough such that minimum distortion whitebox attacks such as FAB are ineffective.

		MNIST	CIFAR-10
Whitebox	AutoAttack*	84.4 ± 5.1	91.0 ± 1.9
	APGD-CE	95.8 ± 1.9	97.5 ± 0.7
	APGD-DLR	83.9 ± 5.4	90.7 ± 2.1
	FAB	5.4 ± 2.2	10.4 ± 1.7
Blackbox-query	Square	60.9 ± 10.3	21.9 ± 2.8
	$\mathcal{N}$-Attack	9.5 ± 3.2	56.7 ± 8.9
Non-Interactive Blackbox	MI-Attack	93.7 ± 1.1	**99.9** ± 0.1
	DI-Attack	95.9 ± 1.6	**99.9** $\pm$ **0.1**
	TID-Attack	92.8 ± 2.7	19.7 ± 1.5
	SGM-Attack	N/A	**99.8** ± 0.3
	AEG (Ours)	**98.9** $\pm$ **1.4**	98.5 ± 0.6

Table 4: Test error rates for average blackbox transfer over architectures at $\epsilon = 0.3$ for MNIST and $\epsilon = 0.03125$ for CIFAR-10 (higher is better)

Source	Attack	VGG-16	RN-18	WR	DN-121	Inc-V3
	Clean	11.2 ± 0.9	13.1 ± 2.0	6.8 ± 0.7	11.2 ± 1.4	9.9 ± 1.3
WR	MI-Attack	67.8 ± 3.01	86.0 ± 1.7	**99.9** $\pm$ **0.1**	89.0 ± 2.6	88.2 ± 1.4
	DI-Attack	68.3 ± 2.4	88.5 ± 2.1	**99.9** $\pm$ **0.1**	91.2 ± 1.6	**91.5** $\pm$ **1.8**
	TID-Attack	23.1 ± 1.8	25.9 ± 1.3	20.6 ± 1.0	23.6 ± 1.2	21.9 ± 1.7
	SGM-Attack	69.1 ± 2.1	88.6 ± 2.0	**99.6** $\pm$ **0.4**	90.7 ± 1.9	86.8 ± 2.2
	AEG (Ours)	40.8 ± 3.22	70.6 ± 4.9	98.5 ± 0.6	**88.2** $\pm$ **4.6**	**89.6** $\pm$ **1.8**
	AEG (New)	**86.2** $\pm$ **1.8**	**94.1** $\pm$ **1.5**	81.1 ± 1.1	**93.1** $\pm$ **1.7**	89.2 ± 1.8

Table 5: Error rates on $\mathcal{D}$ for average NoBox architecture transfer attacks with $\epsilon = 0.03125$ with Wide-ResNet architecture

C.2 Qualitative Results

As a sanity check we also provide some qualitative results about the generated adversarial attacks. In Figure 2 we show the 256 attacked samples generated by our method on MNIST. In Figure 3 we show on the left the 256 CIFAR samples to attack, and on the right the perturbations generated by our method amplified by a factor 10.

D Implementation Details

D.1 AEG Architecture

The high-level architecture of our AEG framework is illustrated in Figure 4. The generator takes the input x and encode it into $\psi(x)$, then the generator uses this encoding to compute a probability vector $p(\psi(x))$ in the probability simplex of size K, the number of classes. Using this probability vector the network then samples a categorical variable z according to a multinomial distribution of parameter $p(\psi(x))$. Intuitively, this category may correspond to a target for the attack. Gradient,

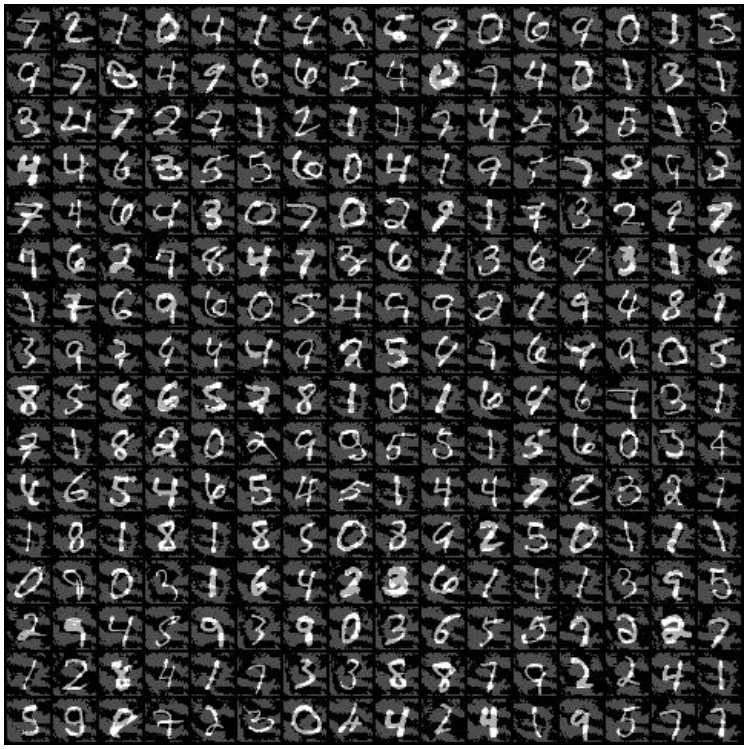

Figure 2: Attacks generated on MNIST by our method.

is backprogated across this categorical variable using the gamble-softmax trick [45, 52]. Finally, the decoder takes as input $\psi(x)$, z and the label y to output an adversarial perturbation δ such that $\|\delta\| \leq \epsilon$. In order to generate adversarial perturbations over images that obey ϵ-ball constraints, we employ a scaled tanh output layer to scale the output of the generator to $(0, 1)$, subtract the clean images, and finally apply an elementwise multiplication by ϵ. We then compute $\ell(f(x + \delta), y)$ where f is the critic and ℓ the cross entropy loss. We solve the game using the ExtraAdam optimizer [29] with a learning rate of 10^{-3}. We allow the generator to update its parameters several times on the same batch of examples before updating the critic. In particular we update the generator until it is able to fool the critic or it reaches some fixed number of iterations. We set this max number of iterations to 20 in all our experiments. We also find that biasing the critic update various forms of adversarial training consistently leads to the most effective attack. We reconcile this phenomenon by noting that through adversarial training the critic itself becomes a robust model which provides a richer learning signal to the generator. Furthermore, the elegance of the AEG frameworks allows the practitioner to further bias the optimization process of the critic —and consequently the generator— through picking and choosing effective robustness techniques such as training with PGD adversarial examples generated at a prior timestep.

D.2 Generator Architecture

The architecture we used for the encoder and the decoder is described in Table 7 and 8. For MNIST we used a standard convolutional architecture and for CIFAR-10 we used a ResNet architecture.

D.3 Baseline Implementation Details

The principal baselines used in the main paper include the Momentum-Iterative Attack (MI-Attack) [24], the Input Diversity (DI-Attack)[73], the Translation-Invariant (TID-Attack) [25] and the Skip Gradient Method (SGM-Attack) [71]. As Input Diversity and Translation invariant are approaches that generally can be combined with existing attack strategies we choose to use the powerful Momentum-Iterative attack as our base attack. Thus the DI-Attack consists of random input transformations when using an MI-Attack adversary while the TID-attack further adds a convolutional kernel ontop of the

d-**ResBlock**
Input:x
Forward for computing $F(x)$:
Reflection pad (1)
conv. (ker: 3×3, $d \to d$; stride: 1; pad: 1)
Batch Normalization
ReLU
Reflection pad (1)
conv. (ker: 3×3, $d \to d$; stride: 1; pad: 1)
Batch Normalization
Output:$x + F(x)$

Table 6: ResNet blocks used for the ResNet architectures (see Table 7) for the Generator. Each ResNet block contains skip connection (bypass), and a sequence of convolutional layers, normalization, and the ReLU non–linearity.

Encoder
Input: $x \in \mathbb{R}^{3\times32\times32}$
Reflection Padding (3)
conv. (ker: 7×7, $32 \to 63$; stride: 1; pad: 0)
Batch Normalization
ReLU
conv. (ker: 3×3, $63 \to 127$; stride: 2; pad: 0)
Batch Normalization
ReLU
conv. (ker: 3×3, $127 \to 255$; stride: 2; pad: 0)
Batch Normalization
ReLU
255-ResBlock
255-ResBlock
255-ResBlock

Decoder
Input: $(\psi(x), z, y) \in \mathbb{R}^{256\times8\times8}$
256-ResBlock
256-ResBlock
256-ResBlock
Transp. conv. (ker: 3×3, $256 \to 128$; stride: 2; pad: 0)
Batch Normalization
ReLU
Transp. conv. (ker: 3×3, $128 \to 64$; stride: 2; pad: 0)
Batch Normalization
ReLU
ReflectionPadding(3)
conv. (ker: 7×7, $64 \to 32$; stride: 1; pad: 0)
Tanh

Table 7: Encoder and Decoder for the convolutional generator used for the MNIST dataset.

Encoder
Input: $x \in \mathbb{R}^{28\times28}$
conv. (ker: 3×3, $1 \to 64$; stride: 3; pad: 1)
LeakyReLU(0.2)
Max Pooling (stride: 2)
conv. (ker: 3×3, $64 \to 32$; stride: 2; pad: 1)
LeakyReLU(0.2)
Max Pooling (stride: 2)

Decoder
Input: $(\psi(x), z, y) \in \mathbb{R}^{64\times2\times2}$
Transp. conv. (ker: 3×3, $64 \to 32$; stride: 2; pad: 1)
LeakyReLU(0.2)
Max Pooling stride 2
Reflection Padding (3)
Transp. conv. (ker: 5×5, $32 \to 16$; stride: 3; pad: 1)
LeakyReLU(0.2)
Max Pooling stride 2
Transp. conv. (ker: 2×2, $16 \to 1$; stride: 2; pad: 1)
Tanh

Table 8: Encoder and Decoder for the ResNet generator used for the MNIST dataset.

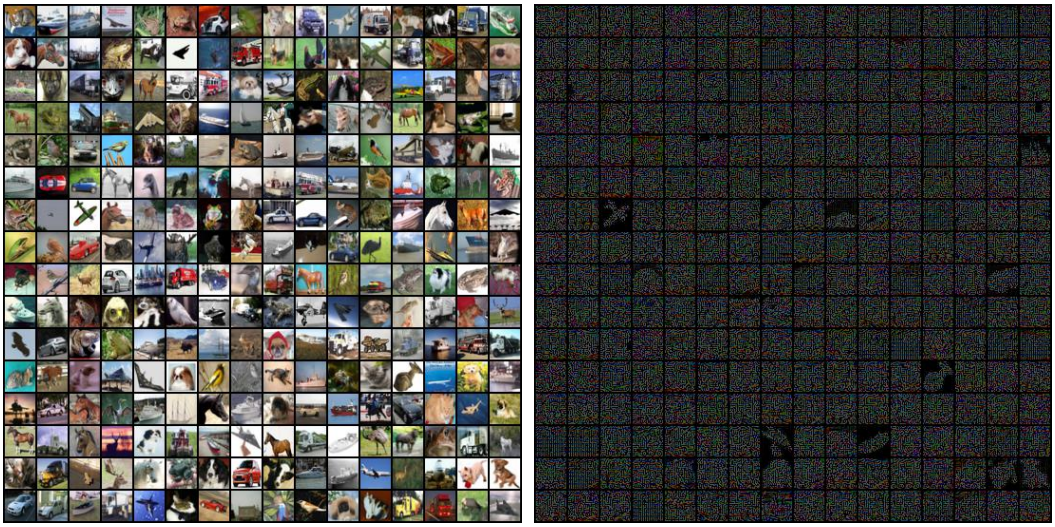

Figure 3: **Left:** CIFAR examples to attack. **Right:** Pertubations generated by our method amplified by a factor 10. An interesting observation is that the generator learns not to attack the pixel where the background is white.

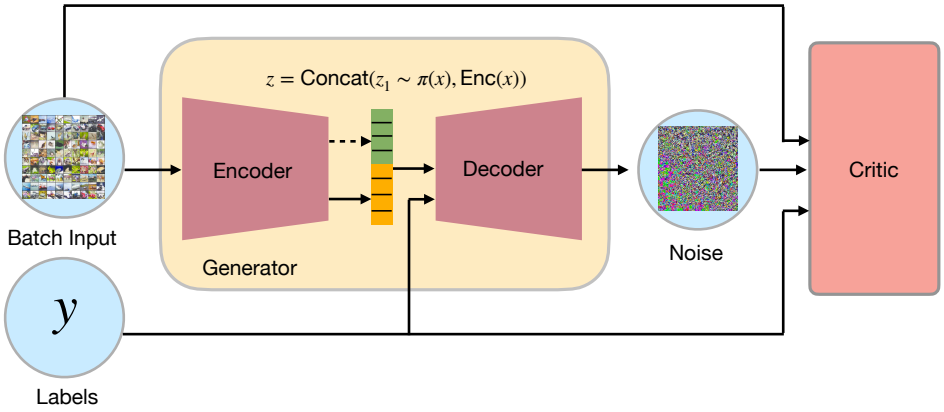

Figure 4: AEG framework architecture

DI-Attack. We base our implementions using the AdverTorch [22] library and adapt all baselines to this framework using original implementations where available. In particular, when possible we reused open source code in the Pytorch library [59] otherwise we re-implement existing algorithms. We also inherit most hyperparameters settings when reporting baseline results except for number steps used in iterated attacks. We find that most iterated attacks benefit from additional optimization steps when attacking MNIST and CIFAR-10 classifiers. Specifically, we allot a 100 step budget for all iterated attacks which is often a five to ten fold increase than the reported setting in all baselines.

D.4 Ensemble Adversarial Training Architectures

We ensemble adversarially train our models in accordance with the training protocol outlined in [67]. For MNIST models we train a standard model for 6 epochs, and an ensemble adversarial model using adversarial examples from the remaining three architectures for 12 epochs. The specific architectures for Models A-D are provided in Table. 8. Similarly, for CIFAR-10 we train both the standard model and ensemble adversarial models for 50 epochs. For computationally efficiency we randomly sample two out of three held out architectures when ensemble adversarially training the source model.

A	B	C	D
Conv(64, 5, 5) + Relu	Dropout(0.2)	Conv(128, 3, 3) + Tanh	FC(300) + Relu
Conv(64, 5, 5) + Relu	Conv(64, 8, 8) + Relu	MaxPool(2,2)	Dropout(0.5)
Dropout(0.25)	Conv(128, 6, 6) + Relu	Conv(64, 3, 3) + Tanh	FC(300) + Relu
FC(128) + Relu	Conv(128, 6, 6) + Relu	MaxPool(2,2)	Dropout(0.5)
Dropout(0.5)	Dropout(0.5)	FC(128) + Relu	FC(300) + Relu
FC + Softmax	FC + Softmax	FC + Softmax	Dropout(0.5)
			FC(300) + Relu
			Dropout(0.5)
			FC + Softmax

Table 9: MNIST Ensemble Adversarial Training Architectures)

E Further Related Work

Adversarial attacks can be classified under different threat models, which impose different access and resource restrictions on the attacker [1]. The whitebox setting, where the attacker has full access to the model parameters and outputs, thus allowing the attacker to utilize gradients based methods to solve a constrained optimization procedure. This setting is more permissive than the semi-whtiebox and the blackbox setting, the latter of which the attacker has only access to the prediction [57, 58] or sometimes the predicted confidence [36]. In this paper, we focus on a challenging variant of the conventional blackbox threat model which we call the NoBox setting which further restricts the attacker by *not* allowing any query from the target model. While there exists a vast literature of adversarial attacks, we focus on ones that are most related to our setting and direct the interested reader to comprehensive surveys for adversarial attacks and blackbox adversarial attacks [6, 16].

Whitebox Attacks. The most common threat model for whitebox adversarial examples are l_p-norm attacks, where $p \in \{2, \infty\}$ is the choice of norm ball used to define the attack budget. One of the earliest gradient based attacks is the Fast Gradient Sign Method (FGSM) [33], which computes bounded perturbations in a single step by computing the signed gradient of the loss function with respect to a clean input. More powerful adversaries can be computed using multi-step attacks such as DeepFool [55] which iteratively finds the minimum distance over perturbation direction needed to cross a decision boundary. For constrained optimization problems the Carlini-Wagner (CW) attack [14] is a powerful iterative optimization scheme which introduces an attack objective designed to maximize the distance between the target class and the most likely adversarial class. Similarly, projected gradient descent based attacks has been shown to be the strongest class of adversaries for l_2 and l_∞ norm attacks [53] and even provides a natural way of robustifying models through adversarial training. Extensions of PGD that fix failures due to suboptimal step size and problems of the objective function include AutoPGD-CE (APGD-CE) and AutoPGD-DLR (APGD-DLR) and leads to the state of the art whitebox attack in AutoAttack [20] which ensembles two other strong diverse and parameter free attacks.

Blackbox Attacks. Like whitebox attacks the adversarial goal for a blackbox attacker remains the same with the most common threat model also being l_p norm attacks. Unlike, whitebox attacks the adversarial capabilities of the attacker is severely restricted rendering exact gradient computation impossible. In lieu of exact gradients, early blackbox attacks generated adversarial examples on surrogate models in combination with queries to the target model [57]. When given a query budget gradient estimation is an attractive approach with notable approaches utilizing black box optimization schemes such as Finite Differences [18], Natural Evolutionary Strategies [43, 46], learned priors in a bandit optimization framework [44], meta-learning attack patterns [26], and query efficient

Defenses. In order to protect against the security risk posed by adversarial examples there have been many proposed defense strategies. Here we provide a non-exhaustive list of such methods. Broadly speaking, most defense approaches can be categorized into either robust optimization techniques, gradient obfuscation methods, or adversarial example detection algorithms [74]. Robust optimization techniques aim to improve the robustness of a classifier by learning model parameters by incorporating adversarial examples from a given attack into the training process [53, 67, 23]. On the other hand obfuscation methods rely on masking the input gradient needed by an attacker to construct adversarial examples [63, 12, 35, 21].

In adversarial example detection schemes the defender seeks to sanitize the inputs to the target model by rejecting any it deems adversarial. Often this involves training auxiliary classifiers or differences in statistics between adversarial examples and clean data [34, 54, 31].