AutoAttacker: A reinforcement learning approach for black-box adversarial attacks

Ilias Tsingenopoulos, Davy Preuveneers and Wouter Joosen
imec-DistriNet, KU Leuven
Department of Computer Science, Celestijnenlaan 200A, B-3001 Heverlee, Belgium
{firstname.lastname}@cs.kuleuven.be

***Abstract*—Recent research has shown that machine learning models are susceptible to adversarial examples, allowing attackers to trick a machine learning model into making a mistake and producing an incorrect output. Adversarial examples are commonly constructed or discovered by using gradient-based methods that require white-box access to the model. In most real-world AI system deployments, having complete access to the machine learning model is an unrealistic threat model. However, it is possible for an attacker to construct adversarial examples even in the black-box case – where one assumes solely a query capability to the model – with a variety of approaches each with its advantages and shortcomings.**

We introduce AutoAttacker, a novel reinforcement learning framework where agents learn how to operate around the black-box model by querying it, to effectively extract the underlying decision behaviour, and to undermine it successfully. AutoAttacker is a first of its kind framework that uses reinforcement learning and assumes nothing about the differentiability or structure of the underlying function and is thus robust towards common defenses like gradient obfuscation or adversarial training. Finally, without differentiable output, as in binary classification, most methods cease to operate and require either an approximation of the gradient, or another approach altogether. Our approach, however, maintains the capability to function when the output descriptiveness diminishes.

I. Introduction

Despite the extensive adoption and proliferation of machine learning models, and the ever-increasing literature on their robustness properties (through ad-hoc defenses or otherwise), they are still largely susceptible to adversarial examples. Adversarial examples are perturbed inputs that fool classifiers, and the task of discovering them has a bilateral nature in optimization defined as follows: Find the *minimum* perturbation under which the sample is classified as *something other than its actual class*. These adversarial examples can potentially be exploited in the real world [1]. For many commercial or proprietary systems, constructing adversarial examples has to be considered under a limited threat model. This has motivated black-box attacks that do not require access to explicit information about how the classifier decides and operates, i.e. model family, weights, gradients, or data used for training.

One common approach to attacking a classifier in this setting is to train a substitute network to emulate the original network and its decision behaviour, and to subsequently attack the substitute in a white-box manner with first-order methods [2] like gradient descent. Recent works note that adversarial examples constructed on substitute networks do not always transfer to the target model, especially when conducting targeted attacks [3] [4]. As a consequence the focus has instead shifted into constructing adversarial examples by estimating the gradient through the classifier with coordinate-wise finite difference methods [5] [6].

The main challenge addressed in this work is the fact that many adversarial attacks assume an unrealistic threat model. We therefore consider additional access and resource restrictions on the black-box model that characterize restrictions in real-world systems. To counter these restrictions a host of attacks with exploratory components has been proposed, e.g. by utilizing genetic algorithms [7]. But even in that case, information that was queried under a budget (i.e. imposing a limit on the amount of queries), is lost the moment we move to the next sample. We are looking into a way to combine the adversarial example generation together with reverse-engineering the decision boundary of the model under attack. For that reason, we propose a reinforcement learning framework for constructing adversarial perturbations conditional on sample input. In this way we ensure that the information queried from the black-box model is not squandered when we move to the next sample. In this work, the adversarial example discovery is staged as a game being played between the learning agent and the black-box environment, and enabled by the vast range of algorithms available in reinforcement learning, capable of operating even in binary input-output environments with non-differentiable elements.

As expected, different query capabilities and output descriptiveness imply different applicable techniques. Under the reinforcement learning framework, there are techniques that can deal with discrete input and output spaces, like Deep Q-Networks [8], and with continuous spaces, like Deep Deterministic Policy Gradient [9]. To demonstrate the potential of the framework, we formulate the attacks under a black-box model-as-environment: a Convolutional Neural Network trained on the MNIST dataset achieving 99.2% accuracy. The black-box model accepts images of numbers as continuous inputs, and outputs probabilities for each class.

The remainder of this paper is structured as follows: In section II, we provide the necessary background to adversarial examples and the black-box setting. Section III presents our reinforcement learning based approach. We present experimental results with our framework in section IV. Finally, we conclude

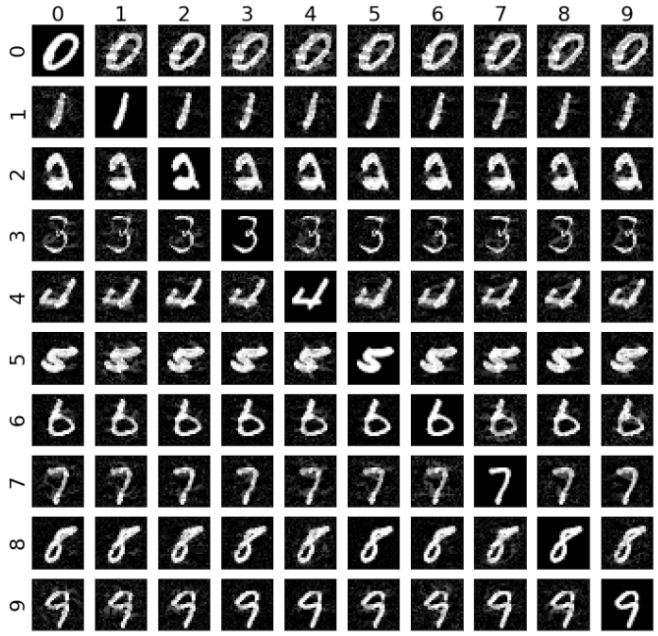

Fig. 1. MNIST adversarial examples. Row label is the true label and column label is the target label.

this paper summarizing our main contributions and findings in section V.

II. Background and related work

Adversarial examples are inputs that have been minimally perturbed in order to induce a targeted or untargeted misclassification from the model. Depending on the underlying threat model, the ease and success of creating these adversarial examples varies. Most often it decreases as we move from (a) the white-box case where we assume complete knowledge of the underlying model, that is the model family, architecture, parameters, and data used for training, to (b) the black-box case where we have no knowledge of the above and only the following capability remains: to submit queries to the model and record the output. The reason that this phenomenon is observed is that creating adversarial examples is commonly posed as an optimization problem. More specifically, having white-box access to the model entails the ability to calculate the gradient of the output w.r.t. the input in closed form. Even when gradients are not readily applied, there is always a workaround as long as the adversary has access to the model:

- *deep neural networks*: use backpropagation of errors and the chain rule
- *non-differentiable components*: make linear approximations of these components
- *stochastic elements*: use the reparameterization trick

The less precise the attacker's knowledge of the inner model workings gets, the more the attacker is forced to use approximations of the actual gradient, or to even abandon the optimization approach altogether.

A. White-Box attacks

In the white-box scenario, adversaries have complete knowledge of and access to the model. Assuming continuous function components, e.g. most activation functions in deep neural networks, the adversary is able to compute the exact gradient of the function which has been shown in the literature to enable a host of powerful attacks. In one of the first works to document this non-robust behaviour, Szegedy et al. [10] argue that the reason for the existence of adversarial examples is a question of robust learning and they attribute their occurrence to blind-spots in the decision boundaries of neural networks. In order to generate adversarial examples they use a box-constrained L-BFGS algorithm to solve the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & ||\delta||_2 \\ \text{subject to} \quad & f(x+\delta) = t \\ & x + \delta \in [0,1]^m \end{aligned} \tag{1}$$

where $f : \mathbb{R}^m \to \{1, .., k\}$ is the function that classifies the input as a discrete label. $t \in \{1, ., ., k\}$ is the target class and δ is the added noise.

One of the earliest approaches for generating adversarial examples is the Fast Gradient Sign Method and its iterative version [11]. FGSM uses the gradient of the training loss ($\nabla_x J$) with respect to the input for crafting adversarial examples. Let x_o and x_t denote the original and adversarial examples respectively, and let t denote the target class. As an attack it is constrained under the L_∞ norm and is crafted in the following manner:

$$x' = x - \epsilon \cdot sign(\nabla_x J(x, t)) \tag{2}$$

where ϵ specifies the L_∞ constraint between x and x', and $sign(\nabla_x J)$ is the sign of the gradient. In [12] the iterative version was introduced, where FGSM is applied iteratively with a finer distortion, followed by an ϵ-ball clipping. Madry et al. [13] reformulated the multi-step FGSM as projected gradient descent (PGD), while they maintain L_∞ constraints:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in C \\ & y_{t+1} = \quad x_t - \epsilon \nabla f(x_t) \\ & x_{t+1} = \quad \arg\min_{x \in C} ||y_{t+1} - x|| \end{aligned} \tag{3}$$

where x_t is the sample, ϵ the update step, and x_{t+1} is the perturbed sample.

Carlini & Wagner [14] and Elastic Net Attacks [15] do not use the training loss directly. Instead they designed an L_2 regularized loss function based on the logit layer representation of the DNN. It is formulated is as follows:

$$\begin{aligned} \text{minimize} \quad & c \cdot f(x, t) + ||x - x_0||_2^2 \\ \text{subject to} \quad & x \in [0,1]^n \end{aligned} \tag{4}$$

where $f(x, t)$ is the logit layer loss function. They have defined a separate function f for each L_0, L_2, and L_{inf} attack,

but all contain a parameter k that regulates the necessary margin between the predicted probability of the target class and that of the rest. Parameter k is effectively controlling how strong the adversarial examples are, with the accompanied increase in distortion. What puts Elastic Net Attacks apart from the C&W attack is that they incorporate L_1 minimization in the loss and thus perform elastic-net regularization. This has been shown to generate more robust, transferable adversarial examples [16] [3].

B. Black-Box attacks

As a black-box we describe the environment where the attacker has no access to the specifics of the underlying model, like the model family, parameters, and the data used to train, but still has the capability to submit queries and record the model output. This threat model is the most pervasive in the literature regarding black-box attacks [5] [3] [17]. It thus follows naturally that constructing adversarial examples under the black-box case shares a lot of common ground with the literature of optimizing black-box functions [18]. We can group the different approaches in 3 principal categories.

- **Transferable Attacks**: the attacker builds a substitute model and attacks it in white-box fashion.
- **Gradient Estimation**: the attacker attempts to use gradient-free optimization techniques to approximate the gradient of the black-box function.
- **Exploratory Attacks**: the attacker is delivering random payloads to the model and records the results, quite akin to fuzzing in penetration testing.

*1) **Transferable Attacks**:* Transferability is an intriguing property that adversarial examples exhibit. It maintains that adversarial examples generated against a specific model can also be misclassified by other models. In [19] Ling et al. run a comprehensive evaluation of transferability from one model to another. There is an expansive part of the literature that employs the queries in order to train a substitute model. The property of transferability has been studied exhaustively and it has been shown that it holds at its strongest when examples are transferred between models of the same family, e.g. a DNN to DNN attack. The principle of operation in this case is that the substitute model will learn an approximately similar decision boundary as the one in the black-box model.

Training model substitutes, generating adversarial examples on them, and then relying on the transferability property has certain advantages and drawbacks. It can be used to generate adversarial examples without the requirement to access the black-box model again. If the attacker is interested in learning how to generate a multitude of adversarial examples, it is query efficient, and it also can be combined with generative models in order to generate perturbations conditional on input. But as it is based solely on the transferability property, it has lower success rate than gradient estimation, dependent on source and target model family [20]. This property diminishes as outputs get less informative and the attacker cannot make reasonable assumptions about the underlying model family and architecture [6].

Another approach in training substitute models for black-box attacks involves Generative Adversarial Networks. In [21] Hu et al. attack a malware detector in black-box fashion. In order to achieve this, they employ the standard GAN architecture, where the generator takes as input both a malware sample and a noise vector and the generated result is submitted to the black-box detector as a query. The training data of the substitute discriminator consist of adversarial malware examples from the generator, and benign programs from an additional benign dataset collected by malware authors. The pivotal notion compared to typical substitute training is the following: The ground-truth labels of the training data are not used to train the discriminator. Instead the goal of the substitute discriminator is to fit the black-box detector, and this is achieved by replacing the ground-truth labels with the detector's output.

*2) **Gradient Estimation**:* Gradient estimation is a fundamentally different approach. One of the first to propose it in the context of black-box attacks were Chen et al. [5], where they compute the gradient approximation by using the finite difference method instead of actual back propagation through the unknown network. Given a sample x_o to calculate a single gradient with the finite difference method, for the Inception-v3 network it requires $299 \times 299 \times 3 \times 2 = 536.406$ queries to it. This number of queries is for a single perturbation step, for a single sample. It is evident that this approach, while seminal in the black-box attack literature and an important proof-of-concept, is fairly naive and it does not scale at all. Following work like [6] and [22] expand on it, employing probability priors to reduce the required amount of queries per successful adversarial example construction.

We can summarize the advantages and drawbacks of gradient estimation techniques as follows: They have a high success rate as their approximation is based directly on the specific black-box function behaviour, they operate irrespective to the model family being attacked, and they are still useful, albeit with an increase in required number of queries, when the output gets less informative. But they usually have a high query budget, they construct tailored, one-use perturbations and they squander what they learn.

The projected gradient descent (PGD) method as described above, is designed to be used in the context of white-box attacks. That is, in the setting where the adversary has full access to the gradient $\nabla_x L(x, y)$ of the loss function of the attacked model. In the corresponding and more realistic black-box setting, the adversary has access only to an oracle that returns, for a given input (x, y), only the output of the model and subsequently the value of the adversarial loss function the attacker has defined on that output. As Ilyas et al. [22] have shown, one can estimate the gradient using sequential queries. The finite difference method is still being employed as a primitive to approximate the gradient, where it estimates the directional derivative $D_v f(x) = \langle \nabla_x f(x), v \rangle$ of some function f at a point x in the direction of a vector v as

$$D_v f(x) = \langle \nabla_x f(x), v \rangle \approx (f(x + \delta v) - f(x))/\delta \quad (5)$$

The step size $\delta > 0$ governs the quality of the gradient estimate. Decreasing δ gives more accurate estimates but also decreases reliability, due to precision and noise issues. In practice, δ is a tunable parameter. It follows that the attacker is able to find the d components of the gradient by estimating the inner products of the gradient with all the standard basis vectors $b_1, .., b_d$:

$$\nabla_x L(x, y) \approx \sum_{k=1}^{d} b_k \langle \nabla_x L(x, y), b_k \rangle \quad (6)$$

*3) **Exploratory Attacks**:* The idea of exploration is fairly common in search based optimization techniques, where the goal is to get a fair understanding about the data space and then emphasize only on the promising directions. In [23] Sethi et al. propose the Seed-Explore-Exploit framework for generating adversarial samples. The attack starts with the Seed phase where it acquires a legitimate sample and a malicious sample to form the seed set D'_{Seed}. It is followed by the Explore phase, where the goal is to obtain diverse information about the space of legitimately classified samples by using the Gram-Schmidt process for orthonormalising a set of vectors. Finally at the Exploit phase, the information gathered is used to generate the attack samples by what they define as Anchor Point attacks, where they stochastically try perturbations on them, and Reverse Engineering attack where they attempt to infer the decision surface of the black-box model.

Another approach in exploratory attacks is based on genetic algorithms which are population-based gradient-free optimization strategies and are inspired by the process of natural selection. In [7] Alzantot et al. iteratively evolve a population of candidate adversarial solutions towards better ones. In each generation, the quality of population members is evaluated using a fitness function. Fitter adversarial examples are more likely to be selected for "breeding" the next generation. Successive generations are generated through a combination of the crossover technique, where characteristics from two parent solutions are intermixed to produce a child solution, and the mutation technique, where it accounts for the necessary stochastic element required for exploration.

The GenAttack algorithm gets the original image x and the target classification label t chosen by the attacker, and it computes an adversarial image x_{adv} such that the model classifies x_{adv} as t and $\|x - x_{adv}\|_\infty \leq \delta_{max}$. Then each population members' fitness is evaluated repeatedly, until a successful example is found. The fitness function reflects the optimization objective, where in this case is the output score given to the target class label directly. The authors claim that it actually improves efficiency when they jointly motivate the decrease in the probability of other classes.

C. Motivation for this work

The above landscape of black-box attacks motivates our work, which attempts to bring together the advantages of the approaches and circumvent the disadvantages, under one framework/attack methodology. So the main question being posed in this work is the following: Can we efficiently produce a tailored perturbation (or series thereof) conditional on input, that does not make any assumptions about the underlying model and can at the same time inform future perturbations?

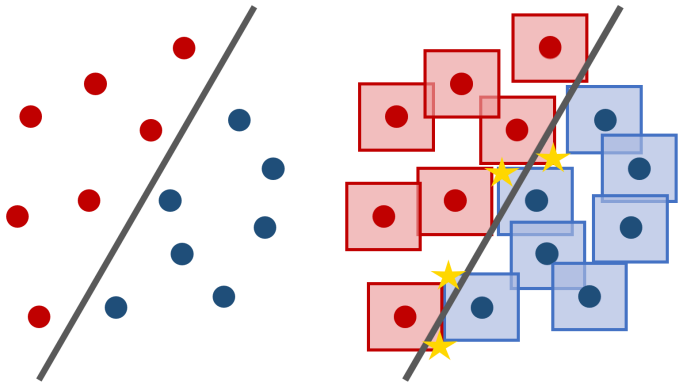

Fig. 2. Left: A linearly separable set of points. Right: Perturbations lying within the L_∞-box around the points can position them on the other side of the decision boundary. The yellow stars are adversarial examples that will be misclassified.

III. Approach

Gradient based optimization has been key to many recent advances in machine learning and the progressively deeper variants. In most actual threat model cases, the mapping, loss, or adversarial function is either not accessible or not differentiable (or both). In reinforcement learning for example, the function being optimized is unknown to the agent and is treated as a black box [24]. Reappropriating the reinforcement learning framework for attacking black-box models, otherwise seen as optimizing unknown adversarial functions, is the key concept of our approach.

A reinforcement learning framework consists of an environment that represents the unknown world, in our case the black-box model, and an agent that takes actions, receives observations from the environment, a reward for the action taken, and information about the new state. The reward informs the agent of how good or bad was the action taken, and the observation indicates the next state in the environment. The environment and the agent interact for an amount of turns t until a predefined goal is reached, and the agent attempts to figure out the best actions to take or the optimal way to behave in the environment in order to reach the goal.

A. Environment

In a black-box environment, the agent is trying to find the action, or to be more specific the series of actions, that will push the example to the other side of the decision boundary of the model. A visual representation of the goal

is shown in Figure 2. The game being played is navigating a N-dimensional hypercube if the perturbation constraint is L_∞, or a N-dimensional hypersphere maze if the perturbation constraint is L_2. The model output is used as the sole feedback on how well the agent is doing, i.e. where is the perturbed sample located w.r.t. the adversarial goal. But not every black-box environment falls under the same threat model. There is a continuum of output descriptiveness under the black-box scope that enables different attacker capabilities. We assume that query limits are constituent of all black-box threat models and we employ a slightly modified taxonomy of the one used by Ilyas et al. in [6] that reflects access and resource restrictions in real-world systems:

1) **Logit setting.** In the logit setting, the attacker has access to the logits or the probabilities of each class as been produced by the model. Example: The Clarifai NSFW (Not Safe for Work) detection API is a binary classifier that outputs $P(\text{NSFW}|x)$ for a submitted image x.
2) **Partial-information setting.** In the partial-information setting, the model outputs a subset of all the possible classes. This aggravates the problem of discovering adversarial examples as the target class might not belong (initially) to this subset and any learning algorithm is deprived of direct information on how to improve. Example: A non-binary sentiment analysis API that outputs a variable number of detected sentiments.
3) **Binary setting.** In the binary setting, the model outputs a binary decision on each of the k possible classes. This setting is the most difficult to tackle as the output is non-differentiable and thus gradient methods not directly applicable. Example: A malware detection API that returns true of false.

B. Reinforcement Learning

A common occurrence in black-box environments is sparse and delayed rewards, where an agent has to operate under constant uncertainty regarding its performance and uninformative feedback. It is deemed advantageous to employ a learning framework that can operate in this fashion, like reinforcement learning, for the simple reason that the reward of discovering a successful adversarial example might require a number of action steps to come. We express black-box adversarial attacks as a sequential, non-cooperative game learned by an agent. The motivation behind this choice is twofold. With regard to previous work, there are two key aspects that we consider unaddressed:

1) While in [22] Ilyas et al. put priors on successive queries and intersample dependencies, the "episode" is over as soon as a successful example is found. The information that these queries gleaned under a budget, is lost when we move to the next sample. There should be a sufficiently expressive learning framework for creating perturbations conditional on input and with the ability to generalize on unseen instances.
2) While the approach of gradient approximation relies on how close the gradient is compared to the actual one, it still is an optimization approach which means it preserves the same pitfall: Being stuck in an inadequate local minimum and with insufficient exploration to overcome it. As a multitude of attack techniques rely on FGSM as a building block, which is performing gradient ascent on the model loss function, an obvious defense at training time would be to select among the various local minima the one with the smallest ϵ-close maximum. An attacker utilizing a purely gradient based policy will be stuck in a suboptimal maximum.

Therefore, we claim that posing the problem as a game environment is intuitive, in the sense that reinforcement learning as a framework provides a combination of exploitation, that is following the gradient from the information already queried from the black-box, and exploration, that is guiding the stochastic component of its operation through shaped noise and/or probability priors. A RL agent parameterized by neural networks and with experience replay (ER) has the ability to simultaneously learn tailored actions and generalize. We provide an approximate correspondence between terms used in adversarial machine learning context and reinforcement learning:

- Perturbation $\longrightarrow$ Action
- Perturbation Steps $\longrightarrow$ Episode
- Input & Output & Target $\longrightarrow$ State
- Adversarial Goal $\longrightarrow$ Rewards
- Deterministic Model $\longrightarrow$ Model-Free
- Feature Engineering $\longrightarrow$ State Representation
- Domain Knowledge $\longrightarrow$ Reward & Noise Shaping

The procedure of discovering a successful adversarial example by querying a black-box model can be formulated as a Markov Decision Process (MDP) equipped with a state space S, an action space A, a transition function $P(s'|s,a)$, a reward function $R(s,a) : S \times A \to \mathbb{R}$, and a discount factor γ. We construct s as a tuple containing an ingestion of input x, the original class o, the target class t, and the model output l. Given s and a, s' is conditionally independent of all previous states and actions; in other words, the state transitions satisfy the Markov property and this formulation can be considered a Markov Decision Process.

We consider a standard reinforcement learning setup consisting of an agent interacting with an environment E in discrete timesteps. The environment in our case is the topological space of the black-box model's decision surface, constrained on the N-dimensional input space. Initially the adversarial goal $O \to T$ is set that maps the original classes to target ones. The black-box model is queried with the unperturbed example, and the output is recorded to construct the initial state s. The game of discovering an adversarial example is played in episodic fashion: At each timestep t the agent receives the state s_t, takes an action a_t and receives a scalar reward r_t. In this

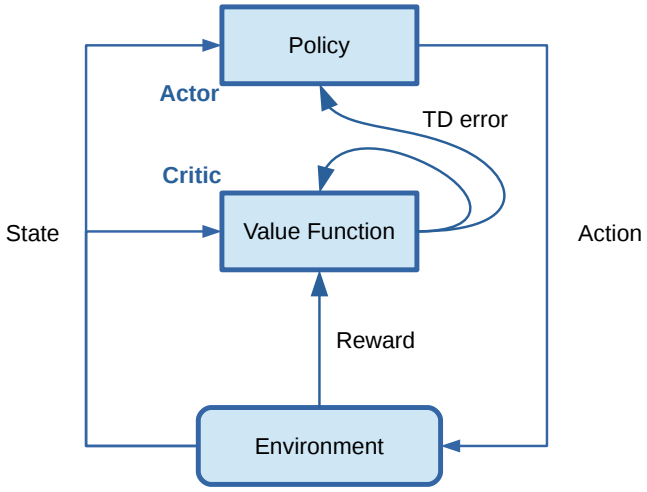

Fig. 3. Actor-Critic architecture. By bootstrapping with temporal difference, the critic loss guides the actor learning. Source: [25]

work the actions taken are real-valued $a_t \in \mathbb{R}^N$, where N is the dimensionality of the perturbation vector. As the tuple describing the state contains all necessary information about it, we operate under the assertion that the environment is fully-observable.

Agent behavior is defined by a policy π that maps states to either a probability distribution over the actions $\pi : S \rightarrow P(A)$, or deterministically to specific actions $\pi(S) = A$. The environment, E, may also behave in a stochastic manner as in the adversarial ML literature it can be an important defense and improve the overall robustness towards adversarial examples. Zhou et al. [26] evaluate defenses with a game-theoretic approach, and as defensive distillation [27], obfuscated gradients [28] and adversarial training [13] have been shown to be ultimately circumventable, the authors show that a promising guarantee of robustness is to have an array n of diverse models under the black-box hood and for each query choose a random one with probability $1/n$. In that case, the reinforcement learning framework is capable of adapting as it can intuitively model stochastic state changes. This path is deemed beyond of the scope of this paper though, and is appointed as future work.

While Deep Q-Networks have exhibited impressive performance in many environments [8], it is not possible to straightforwardly apply Q-learning to continuous action spaces, as in the case of image perturbations. For this reason, we opt for an Actor-Critic architecture, as depicted in Fig. 3, where both actor and critic are parameterized by a neural network. The critic network is approximating the Q-function, where it maps states and actions to Q values $Q(s, a)$ and the actor network is approximating the policy $\pi(s, a)$. In continuous spaces though, finding the policy requires an optimization of a_t every timestep; this optimization is too slow to be practical with large, unconstrained function approximators and nontrivial action spaces. For this reason we adapt to our purposes the Deep Deterministic Policy Gradient (DDPG) algorithm defined by Lillicrap et al. [9]. It is an extension of the DPG algorithm [29] that specifies the current policy by deterministically mapping states to a specific action. As is the case with Q-learning, the critic $Q(s, a)$ is learned using the Bellman equation.

Algorithm 1: AutoAttacker Algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ;
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$;
Initialize replay buffer R;
for *episode = 1, M* **do**
  Initialize the noise process H for action exploration, based on the adversarial shape;
  Receive initial state s_1 by querying the black-box model;
  for *t = 1, T* **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + N_t$ according to the current policy and exploration noise;
    Perturb by action a_t and observe new state s_{t+1} and compute reward r_t;
    Store transition (s_t, a_t, r_t, s_{t+1}) in buffer R;
    Sample a random minibatch of N transitions (s_t, a_t, r_t, s_{t+1}) from R;
    Use Bellman equation to set: $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$;
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$;
    Update the actor policy using the policy gradient:; $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$;
    Update the target networks:; $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$;
    $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$;
  end
end

Introducing non-linear function approximators means that convergence is no longer guaranteed. Introducing them in a black-box environment also implies that hyper-parameter tuning will by definition be constrained and heavily rely on domain knowledge. However, such approximators are essential in order to learn and generalize on large state spaces. One challenge when using neural networks for reinforcement learning is that most optimization algorithms assume that the samples are independently and identically distributed. To decorrelate the learning process from the bias introduced by successive queries to the black-box model, we store experience tuples $[s, a, r, s']$ to the replay buffer and train both actor and critic networks by drawing minibatches from there. DDPG is an off-policy algorithm, so the replay buffer can be large, allowing the algorithm to benefit from learning across a set of uncorrelated transitions, or even a completely different adversarial environment configuration. The process is shown

in Algorithm 1.

C. Implementation

In our experiments, we use a 300-dimensional continuous state space, comprised of the following features:

- A 2D-convolution feature map generated from the digit image
- A 1-hot vector of the target class
- The output layer of the black-box model

The action space is a 784-dimensional vector that is straightforwardly applied as a perturbation on the input image. The actor network has the following architecture:

1) Input Layer: 300-dimensional state vector
2) Hidden Layer: Dense of size 512
3) Hidden Layer: Dense of size 512
4) Output Layer: 784-dimensional action vector

The critic network has the following architecture:

1) Input Layer: 300-dimensional state vector & 784-dimensional action vector
2) Hidden Layer: Dense of size 512
3) Hidden Layer: Dense of size 512
4) Output Layer: single neuron containing the Q-Value

IV. Evaluation

We evaluate AutoAttacker by running experiments attacking a state-of-the-art MNIST classification model that achieves 99.2% accuracy on the test set. Results are still preliminary and we mainly outline the testing methodology and the obstacles lying ahead.

To incentivize exploration, and guide the agent towards the desired adversarial goal, we introduce shaped noise which is added after each actor network decision. The noise is based on the uniform distribution $U(0,1)$ and is calculated as:

$$H(t) = \frac{U(0,1) \cdot (x_T - x_O)}{t} \tag{7}$$

where x_T is a random sample from the target class and x_O is the original sample respectively, and t is the incremental number of steps within the episode. Noise in the case of untargeted attacks is plainly uniform.

The reward function describes how the agent ought to behave, stipulating what we want the agent to accomplish and facilitating speed of convergence and not getting stuck in local minima. Thus designing a reward function that can perform, guide the agent's learning and be well-behaved takes a degree of domain knowledge, insights about monotonicity, and ingenuity. We have devised several different reward functions by permuting a number of fundamental components:

$$\begin{aligned} r_1 &= w_1 F(x')_T \\ r_2 &= w_2 (F(x')_T - F(x)_T) \\ r_3 &= w_3 (F(x')_T - F(x')_O) \\ r_4 &= w_4 (max(0, (F(x')_T - max(F(x')_{i \neq T})))) \\ r_5 &= w_5 \|a\|_2 \\ r_6 &= -c \end{aligned} \tag{8}$$

Fig. 4. MNIST adversarial examples generated by AutoAttacker

where w_n are weights for the components, $F(x')_T$ is the output logit for the target class T, $F(x)_T$ is the same logit from previous step, $F(x')_O$ is the output logit of the original class O, $F(x')_{i \neq T}$ is the output logits of every other class but the original one, a is the action taken, and c is a positive constant. We discover that the most robust and consistently performing reward function $R(s, a, s') \to \mathbb{R}$ is $R = r_1 + r_4 - r_6/r_5$.

A. Experimental setup

We adhere to the following experiment methodology: the black-box MNIST classification model is trained on the training split, and we submit one sample per episode from the testing split. Each episode is run until a successful adversarial example is found, or 100 steps have passed. We consider 100 to be a reasonable depth for successive moves and a sufficient delineation for a box-constrained path. The evaluation is done over two types of adversarial goals: targeted and untargeted attacks.

In the targeted attack case, in order to define the adversarial goal we elect to rotate the ground truth class one spot to the right. This means that 1 is mapped to 2, 2 to 3, and so on. We chose to do so in order to enforce consistency between differently parameterized runs and reproducibility of results. As long as the source-target class mapping is random but consistent between each experiment, it should be a sufficiently good approximation of the performance.

In the untargeted attack case, the adversarial goal is to be classified as any other class than the ground truth one. It is a strictly less difficult task than producing a targeted adversarial example, as the results also confirm.

Hyperparameters: For the MNIST experiment, aside from limiting the episode length to 100 and using Adam as optimizer for the network updates, we set the hyperparameters to the following values: discount factor $\gamma = 0.9$, critic learning rate $\eta_c = 0.0005$, actor learning rate $\eta_a = 0.1\eta_c$, target network learning rate $\tau = 0.1$.

B. Results

We present the summary results on the MNIST test set, that includes 10000 instances in total. We report on the success rate, the average number of queries required, and the average runtime per episode. Query and runtime figures are computed only over successful episodes. Results for the targeted attack are shown in Table I, and for the untargeted attack are shown in Table II.

To have an indirect initial comparison to how the rest of the recent literature is performing on MNIST, we also include their results in Table I. Note that the C&W attack is white-box, so there are no query restrictions, and that GenAttack does not evaluate on untargeted attacks. At this point we should

mention that according to our testing methodology, when an episode ends unsuccessfully ($t > 100$), we move to the next sample. As is exhibited by the results, this is a form of trade-off between success rate and number of queries required. If we were set on optimizing success rate, we could revisit each unsuccessful sample by running new clean-slate episodes on them.

TABLE I
SUCCESS RATE (SR), MEAN QUERIES (MQ), MEAN RUNTIME PER EPISODE (MRE) AND PERTURBATION NORM L_∞

MNIST Targeted				
	SR	MQ	MRE	L_∞
C&W	100%	-	0.006hr	0.3
ZOO	98%	2,118,222	0.013hr	0.3
GenAttack	100%	996	0.002hr	0.3
AutoAttacker	6%	19	0.005hr	0.3
AutoAttacker	44%	11	0.003hr	0.4

TABLE II
SUCCESS RATE (SR), MEAN QUERIES (MQ), MEAN RUNTIME PER EPISODE (MRE) AND PERTURBATION NORM L_∞

MNIST Untargeted				
	SR	MQ	MRE	L_∞
AutoAttacker	73.4%	55	0.012hr	0.4

C. Discussion

We acknowledge that these are preliminary and work-in-progress results, and they should be interpreted as the result of a proof of concept rather than an exhaustive evaluation over a range of diverse models and a comparative study to state-of-the-art approaches. We consider it being of value to report on the obstacles that the proposed agent and framework in general have faced during the learning process.

The actor and critic networks are trained in tandem with querying the black-box model and recording $[s, a, r, s']$ tuples in the replay buffer. To be more specific, after each query, the minibatch is sampled from the replay buffer and a single batch update is made to both actor and critic networks. This might lead to a premature overfitting. So, if the noise deployed is not sufficiently diverse to generate equally diverse query results, we face an interesting conundrum: the variance between action vectors will be low, leading to the compounding effect of the critic stagnating on similar Q-Values and its gradient getting uninformative, and subsequently the actor generating similar actions. It is a vicious circle where the actor will require diverse states as input to produce diverse actions, that will not come if the exploration plateaus in a local minimum. While our proposed framework has shown decent robustness to typical hyperparameter ranges, we have discovered that it is sensitive to higher values of the discount factor γ in combination with the r_5 penalty in the reward function. In this scenario Q-Values might spiral out of control and the model will not converge.

The MNIST classification models have a fairly low dimensional input, that flattened is comprised of 784 features. This translates to a dimensionality of 784 continuous actions. Continuous control environments in the literature [30] rarely go above a dimensionality of 20 for the action space. We posit that the task to learn in our environment while with much higher dimensionality than the above environments, it has a significantly lower sequential/combinatorial complexity and thus simpler MDP graph. This observation explains why the approach can be successful with a relatively low amount of data points, but also why it can be dominated by noise and never find a path towards the designated goal.

As the actions are consecutive, the path drawn within the N-dimensional box that constrains the maximum perturbation might not be improving the reward fast enough, i.e. it might be stagnating in a local suboptimal maximum. In that case and in order to conserve the amount of queries required, it might be beneficial to reset the episode conditional on the reward function's rate of improvement.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we presented AutoAttacker, a reinforcement learning based framework for black-box adversarial attacks. It is novel in the sense that it is able to construct adversarial perturbations conditional on sample input, in such a way that information queried from the black-box model is maintained and reused between samples. Our approach for constructing adversarial examples is staged as a game being played between the learning agent and the black-box model, and is capable of operating in high-dimensional continuous action and state environments all the way to discrete input / binary output environments with non-differentiable elements. We carried out experimental evaluations with the black-box MNIST classification model, and compared the success rate, the average number of queries required, and the average runtime per episode with the related work. The results demonstrate the practical feasibility of our reinforcement learning-based AutoAttacker approach.

There are several unexplored paths that we leave open for future investigation. Regarding this work mainly as a proof of concept, we intend to expand the evaluation on different environments, e.g. in binary input and output domains where DQN approaches are pertinent [31]. Additionally, there is an ever-evolving literature that improves the state of the art on different environments, so we intend to incorporate more algorithms [32] [24] in our framework. We also take note on the fact that prioritizing specific experiences from the replay buffer has been shown to be advantageous in many environments [33] [34]. In the context of reinforcement learning, experience replay is the mechanism that controls the all-important overarching task of statistical learning, and it might be impactful to devise a strategy of retrieving experiences adapted to the adversarial goal and domain in general.

On a final note, a main assumption made in this work is that the black-box model is stationary, that is it does not retrain or adapt or employ countermeasures between each query. Part of

the recent literature is examining the adversarial attack domain from game-theoretic and multi-agent points of view [35] [36] [37] and we are inquisitive as to how intelligent agents could be deployed from the defensive side to counter adversarial attacks, and on the disparate and complex interplays possible under a unified multi-agent adversarial environment.

ACKNOWLEDGMENT

This research is partially funded by the Research Fund KU Leuven.

REFERENCES

[1] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing Robust Adversarial Examples," 7 2017. [Online]. Available: http://arxiv.org/abs/1707.07397

[2] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical Black-Box Attacks against Machine Learning," 2 2016. [Online]. Available: http://arxiv.org/abs/1602.02697

[3] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into Transferable Adversarial Examples and Black-box Attacks," 11 2016. [Online]. Available: http://arxiv.org/abs/1611.02770

[4] W. Brendel, J. Rauber, and M. Bethge, "Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models," 12 2017. [Online]. Available: http://arxiv.org/abs/1712.04248

[5] 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," 8 2017. [Online]. Available: http://arxiv.org/abs/1708.03999 http://dx.doi.org/10.1145/3128572.3140448

[6] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box Adversarial Attacks with Limited Queries and Information," 4 2018. [Online]. Available: http://arxiv.org/abs/1804.08598

[7] M. Alzantot, Y. Sharma, S. Chakraborty, and M. Srivastava, "GenAttack: Practical Black-box Attacks with Gradient-Free Optimization," 5 2018. [Online]. Available: http://arxiv.org/abs/1805.11090

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," Tech. Rep. [Online]. Available: https://www.cs.toronto.edu/ vmnih/docs/dqn.pdf

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 9 2015. [Online]. Available: http://arxiv.org/abs/1509.02971

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," Tech. Rep. [Online]. Available: https://arxiv.org/pdf/1312.6199.pdf

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," 12 2014. [Online]. Available: http://arxiv.org/abs/1412.6572

[12] A. Kurakin, G. Brain, I. J. Goodfellow, and S. Bengio, "ADVERSARIAL MACHINE LEARNING AT SCALE." [Online]. Available: https://arxiv.org/pdf/1611.01236.pdf

[13] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," 6 2017. [Online]. Available: http://arxiv.org/abs/1706.06083

[14] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," 8 2016. [Online]. Available: http://arxiv.org/abs/1608.04644

[15] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples," Tech. Rep. [Online]. Available: https://arxiv.org/pdf/1709.04114.pdf

[16] Y. Sharma and P.-Y. Chen, "Attacking the Madry Defense Model with L_1-based Adversarial Examples," 10 2017. [Online]. Available: http://arxiv.org/abs/1710.10733

[17] J. Hayes, G. D. a. p. ArXiv:1708.05207, U. 2017, and G. Danezis, "Machine Learning as an Adversarial Service: Learning Black-Box Adversarial Examples," *arxiv.org*, 2017.

[18] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud, "Backpropagation through the Void: Optimizing control variates for black-box gradient estimation," 10 2017. [Online]. Available: http://arxiv.org/abs/1711.00123

[19] X. Ling, S. Ji, J. Zou, J. Wang, C. Wu, B. Li, and T. Wang, "DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Model," *DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Model*. [Online]. Available: https://www.computer.org/csdl/proceedings/sp/2019/6660/00/666000a381-abs.html

[20] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples," 5 2016. [Online]. Available: http://arxiv.org/abs/1605.07277

[21] W. Hu and Y. Tan, "Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN." [Online]. Available: https://arxiv.org/pdf/1702.05983.pdf

[22] A. Ilyas, L. Engstrom, and A. Madry, "Prior Convictions: Black-Box Adversarial Attacks with Bandits and Priors," 7 2018. [Online]. Available: http://arxiv.org/abs/1807.07978

[23] T. S. Sethi and M. Kantardzic, "Data Driven Exploratory Attacks on Black Box Classifiers in Adversarial Domains," 3 2017. [Online]. Available: http://arxiv.org/abs/1703.07909 http://dx.doi.org/10.1016/j.neucom.2018.02.007

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 7 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction Second edition*.

[26] Y. Zhou, M. Kantarcioglu, and B. Xi, "A survey of game theoretic approach for adversarial machine learning," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, p. e1259, 4 2018. [Online]. Available: http://doi.wiley.com/10.1002/widm.1259

[27] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks," 11 2015. [Online]. Available: http://arxiv.org/abs/1511.04508

[28] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples," 2 2018. [Online]. Available: http://arxiv.org/abs/1802.00420

[29] D. Silver, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," Tech. Rep. [Online]. Available: http://proceedings.mlr.press/v32/silver14.pdf

[30] Y. Duan, X. Chen, C. X. B. Edu, J. Schulman, P. Abbeel, and P. B. Edu, "Benchmarking Deep Reinforcement Learning for Continuous Control," Tech. Rep., 2016. [Online]. Available: https://github.com/

[31] Z. Wang, T. Schaul, M. Hessel, and M. Lanctot, "Dueling Network Architectures for Deep Reinforcement Learning Hado van Hasselt," Tech. Rep. [Online]. Available: https://www.youtube.com/playlist?list=

[32] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft Actor-Critic Algorithms and Applications," 12 2018. [Online]. Available: http://arxiv.org/abs/1812.05905

[33] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight Experience Replay," 7 2017. [Online]. Available: http://arxiv.org/abs/1707.01495

[34] T. Schaul, J. Quan, I. Antonoglou, D. Silver, and G. Deepmind, "Prioritized Experience Replay," Tech. Rep. [Online]. Available: https://arxiv.org/pdf/1511.05952.pdf

[35] J. Perolat, F. Strub, B. Piot, and O. Pietquin, "Learning Nash Equilibrium for General-Sum Markov Games from Batch Data," pp. 232–241, 4 2017. [Online]. Available: http://proceedings.mlr.press/v54/perolat17a.html

[36] J. Perolat, M. Malinowski, B. Piot, and O. Pietquin, "Playing the Game of Universal Adversarial Perturbations," 9 2018. [Online]. Available: http://arxiv.org/abs/1809.07802

[37] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Perolat, D. Silver, and T. Graepel, "A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning," pp. 4190–4203, 2017. [Online]. Available: dl.acm.org/citation.cfm?id=3295174