Generalist Adversarial Policies in Black-Box Settings

by

Kyle D. Domico

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2024

Date of thesis submission: 08/16/2024

The thesis is approved by the following members of the Thesis Committee:

Patrick D. McDaniel
Tsun-Ming Shih Professor of Computer Sciences, School of Computer, Data, & Information Sciences
Thesis Advisor

Josiah Hanna
Assistant Professor, School of Computer, Data, & Information Sciences

Rahul Chatterjee
Assistant Professor, School of Computer, Data, & Information Sciences

© Copyright by Kyle D. Domico 2024
All Rights Reserved

Abstract

Reinforcement Learning has gained widespread popularity for its impact on real-world decision-making systems. Particularly, its ability to capture information from making decisions in certain states of an environment to apply to other states make it a worldly learner. Traditional evasion attacks in Adversarial Machine Learning are information-inefficient: each adversarial example is crafted entirely independently from one another, even when the resultant perturbations are highly similar. In this thesis, we show how to model evasion attacks as a reinforcement learning problem. The agent uses perturbation steps as actions, feature vectors as states, and improvements in adversarial objectives as rewards. We study an RL attack under this framework by instantiating an agent with a Proximal Policy Optimization algorithm and perform the attack on a Convolutional Neural Network trained on the MNIST dataset. First, we perform a single-sample test and find that of 100 clean images per source class, the agent fools the victim model on $> 99\%$ of the images within the range of 0.1% and 8.2% ℓ_0-distortion, competitive with state-of-the-art ℓ_0-attacks. Next, we study the generalizability of the RL attack by testing if the agent leverages the information from crafting old adversarial examples to craft new adversarial examples with less information from the victim model. The RL attack, indeed, accelerates the generation of adversarial examples within the first 20K victim model queries, suggesting a generalization. Last, we perform a query analysis of our attack against a state-of-the-art black-box attack,

SquareAttack. The RL attack rapidly learns how to produce adversarial examples quickly, therein reducing the total number of queries required to produce hundreds of adversarial examples compared to the SquareAttack which demonstrates no acceleration. This paper serves as an existence proof and frontier for studying evasion attacks in Adversarial Machine Learning through Reinforcement Learning.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank my parents, brother, sisters, in-laws, and niece for their unconditional love and support. To my extended family for keeping in touch throughout this journey. To my friends for their kindness and support.

I would also like to thank my adviser, Dr. Patrick McDaniel, for his support during my years as an undergraduate at The Pennsylvania State University and as a graduate student at The University of Wisconsin-Madison. To my labmates Ryan, Eric, Quinn, Yohan, Blaine, Rachel, Kunyang, Jean-Charles, Owen, and Jesse for their good advice and support.

Funding Acknowledgement: This material is based upon work supported by, or in part by, the National Science Foundation under Grant No. CNS-2343611. Any opinions, findings, and conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the National Science Foundation.

This work is dedicated to my parents, John and Tammy Domico.

1 INTRODUCTION

Reinforcement Learning (RL) stands as a prominent field in Artificial Intelligence (AI). It provides a framework for agents to learn optimal decision-making strategies through interaction with their environment. Based on the concept of learning from feedback, RL models aim to maximize cumulative rewards by iteratively exploring actions in states of the environment and their consequences. This iterative process reflects the way humans learn through trial and error, making RL useful in handling complex, dynamic environments where explicit instruction or labeled data may be scarce or costly. RL algorithms have achieved remarkable success in a myriad of domains including robotics (Ibarz et al. (2021)), finance (Liu et al. (2022)), and recommender systems (Afsar et al. (2022)).

Adversarial Machine Learning (AML) poses a great threat to the robustness and reliability of modern Machine Learning (ML) systems. These systems remain vulnerable to *adversarial examples*: imperceptible perturbations to input data that lead to model misclassifications. Prior works have shown that even without access to the victim model parameters (i.e., a "black-box" threat model) the adversary can still effectively fool the model by sending inputs and receiving output predictions (i.e., querying) (Papernot et al. (2017)) (Andriushchenko et al. (2020)) (Chen et al. (2017)). However, AML algorithms fail to leverage information from crafting one adversarial example to another (i.e. learn to craft adversarial examples). Further, an adversary that can effectively learn to craft adversarial examples will be query efficient in future crafting. This suggests a potential to use techniques like RL that, unlike AML algorithms, leverage information from past experiences to improve future experiences.

In this paper, we cast the evasion attack in AML as a RL problem. Specifically, we frame it as a Markov Decision Process (MDP) which captures the underlying semantics of all evasion attacks: victim model inputs (feature

vectors) as states, perturbation steps as actions, and a reward function based on both the introduced distortion and the model confidence of the label. By using different input samples as start states in the model, this allows the RL algorithm to learn a policy that models a query-efficient *generalist* adversary: an adversary that exploits the information learned from crafting past adversarial examples to inform future adversarial examples. This attack goes against current approaches, which perform adversarial example crafting independent to each sample.

In this paper, we build a framework to study the efficacy of a black-box RL-based attack operating in an evasion attack MDP. Using a dataset of clean inputs $\mathcal{D}_{\texttt{Clean}}$, the adversary samples an input to determine a starting state. We use the Proximal Policy Optimization (PPO) algorithm (Schulman et al. (2017)) to learn a policy that maps the feature vectors (states) to corresponding pixels to perturb (actions). Once perturbed, the adversary queries the victim model and receives the output logits. The reward is then determined as improvements in the optimization used in the Carlini & Wagner (CW) (Carlini and Wagner (2017)) attack from the previous state and its output logits to the new state and its output logits. The CW optimization follows a minimization, thus the agent is positively rewarded if the value of the objective function in the new state is less than the previous state, and vice versa for negative reward. The adversary then continues to operate in this RL-loop until a misclassification is achieved (i.e., terminal state), or the maximum number of steps (i.e., time horizon) is met. If it is indeed a misclassification, the resulting state is an adversarial example if the distortion applied from the original start state is within a specific distortion budget. In this case, the original start state is removed from $\mathcal{D}_{\texttt{Clean}}$, as the adversary has successfully crafted an adversarial example for the corresponding clean sample. Under this process, we measure how many queries to the victim model it takes to produce adversarial examples from the inputs in $\mathcal{D}_{\texttt{Clean}}$

To evaluate the efficacy of our approach, we use a Multi-Layer Perceptron policy network to model the adversary and a Convolutional Neural Network (CNN) victim model trained on the MNIST dataset. To measure the distortion of adversarial examples from the original sample, we use the ℓ_0-norm difference used in state-of-the-art image attacks (Papernot et al. (2016)). First, we perform a single-sample attack where $\mathcal{D}_{\texttt{Clean}}$ only contains exactly one clean sample. We find that after 100K queries to the victim model, the victim model makes a misclassification $> 99\%$ of the time within an ℓ_0-norm budget of 0.1%-8.2%.

Next, we study the generalizability of the attack: does the agent leverage information from crafting past adversarial examples to crafting future adversarial examples? To this end, we study the change in the number of adversarial examples produced from the samples in $\mathcal{D}_{\texttt{Clean}}$ with respect to queries to the victim model. In a 1-Class attack where $\mathcal{D}_{\texttt{Clean}}$ consists of all images belonging to a single class, we observe that the number of adversarial examples crafted accelerates with respect to queries. More generally, in an n-Class attack where $\mathcal{D}_{\texttt{Clean}}$ consists of all images belonging to n classes, the same acceleration is observed within the first 50K queries. This suggests that the crafting of adversarial examples early on helps reduce the number of queries required to craft other adversarial examples. Lastly, we compare our results in generating adversarial examples from a dataset with respect to queries against a state-of-the-art black-box attack SquareAttack (Andriushchenko et al. (2020)). Unlike our attack, the SquareAttack demonstrates a linear relationship between adversarial examples generated with respect to queries to the victim model; exposing the fact that this method does not generalize. These results suggest that there exists a new class of black-box attacks with RL under our MDP framework of an evasion attack.

Our contributions are the following:

- We model evasion attacks in AML as an RL problem through an

MDP.

- We show that an RL attack produces an adversarial example within 100K queries on $>$ 99% of clean test samples.
- We show the generalizability of an RL attack by demonstrating its acceleration of adversarial examples crafted within the first 50K queries to the victim model.

1.1 Thesis Statement

Modeling evasion attacks as a reinforcement learning problem enables adversaries to train agents that generalize crafting adversarial examples.

2.1 Adversarial Machine Learning (AML)

Here, we discuss threat models and state-of-the-art attack algorithms in AML.

Threat Models

The common objective of evasion attacks is often defined as follows: minimize the amount of perturbation applied to a sample while inducing model misclassification. The general optimization, as seen in (Sheatsley et al. (2023)), can be formally written as:

$$\begin{aligned} &\arg\min_{\delta} \|\delta\|_p \\ &\text{such that } f(\mathbf{x}+\delta) \neq y \\ &\qquad \mathbf{x}+\delta \in \mathcal{B}_\phi(\mathbf{x}) \end{aligned} \tag{2.1}$$

where δ is the perturbation, $\mathbf{x}$ the original input sample, f the target model, y the sample label, and $\mathcal{B}_\phi(\mathbf{x})$ the norm-ball centered at $\mathbf{x}$ with a radius of ϕ (i.e., a budget).

The approach adversaries take in solving the optimization largely depends on access to the victim model f. When access to the target model is restricted to its outputs, the attack setting is *black-box*. Similarly, the *white-box* setting is when the adversary is assumed to have full access to the victim models parameters and outputs. Prior works have shown that adversaries in black-box settings, despite limited model access, are still very effective in achieving their goal (Papernot et al. (2017))(Andriushchenko et al. (2020))(Chen et al. (2017)). In this paper, we consider a black-box threat model: the adversary can query the victim model and receive the output class probabilities. This is because the only information the RL

agent needs is the output logits of the victim model. This information is used to determine a the reward at each step of the algorithm.

Carlini & Wagner Attack (CW)

The CW attack (Carlini and Wagner (2017)) is an optimization-based attack algorithm which minimizes a sum of perturbation norm and victim model confidence on the label. Formally, the optimization is written:

$$\begin{aligned} &\min_{\delta} \|\delta\|_p + c \cdot f(x+\delta) \\ &\text{such that } x+\delta \in [0,1]^n \end{aligned} \tag{2.2}$$

where δ is the perturbation applied to the input sample x, f the loss function that expresses the models confidence in the class corresponding to the input label, and c the coefficient that weights the value of f. The function f in the paper is defined:

$$f(x) = \max\{\max_{i \neq y}[Z(x)]_i - [Z(x)]_y, \kappa\} \tag{2.3}$$

where y is the label of the sample x, $Z(x)$ the output logits of the victim model from input x, and κ the transferability parameter. This function is used to value the difference between the logit of the label and the highest logit that is not the label. Thus, a misclassification is wold make the value of f non-positive.

A c value too high will cause the algorithm to find solutions that have high distortion, and a value too low will cause the algorithm to find solutions with high confidence in the label (i.e., no misclassification). The CW attack uses a binary search method to find the best value of c that maximizes misclassification percentage weighted by the ℓ_p-distortion of δ. This optimization is useful for an RL attack because it can be used as a sense of reward if the objective improves at each step.

SquareAttack

SquareAttack (Andriushchenko et al. (2020)) is a state-of-the-art black-box attack designed to craft adversarial examples by perturbing input images with square patches. SquareAttack relies on random search within a constrained region of the image, making it particularly effective against defense mechanisms that rely on gradient masking: a technique used to distort the gradient information given to the attacker. Specifically, the algorithm crafts square perturbation patches δ, query's the victim model $f(\mathbf{x} + \delta)$, and keeps the perturbation if the loss function decreases from the previous iteration, otherwise discard and repeats.

SquareAttack is competitive amongst the most query-efficient black-box attack methods. That is, it requires very few interactions with the victim model to craft adversarial examples. The algorithm balances exploration (trying different perturbations via random search) and exploitation (refining promising perturbations) to minimize the total number of queries. However, the algorithm does not have trainable parameters and thus not actually performing any learning across samples. The foundation of RL relies on exploration and exploitation, however, the policies that generate actions contain trainable parameters. Thus, making an RL attack a more generalist adversary compared to the SquareAttack.

2.2 Reinforcement Learning

In this work, we study the efficacy of RL in environments that aim to fool machine learning models. RL consists of agents and environments: the agent plays actions in the environment and receives rewards and next states from the environment. The goal is to learn a policy that generates actions that maximize the cumulative reward over time. The following sections will lay out the necessary RL concepts used in this paper.

Markov Decision Processes (MDPs)

MDPs are powerful mathematical frameworks used to model decision-making processes under uncertainty. They are formally defined as the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the set of states representing configurations of the system, $\mathcal{A}$ the set of actions that can be played, P the state transition probability function mapping states and actions to the next state, R the reward function, and $\gamma \in (0, 1]$ the discount factor representing the importance of future rewards. Another important notation of MDPs are timesteps t and terminal timesteps T. At each timestep t, the state of the environment is represented as s_t and the action taken at this state is denoted a_t. The resulting next state and reward are denoted s_{t+1} and r_t, respectively. This loop continues until a terminal state s_T is reached in which no more actions can be played and the process ends. This process is known as an episode, or trajectory τ, of the environment.

The agent's goal is to learn a policy $\pi(a|s)$ that maps states to action probabilities that maximizes the sum of rewards across all timesteps t. Formally:

$$\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right] \tag{2.4}$$

where the optimization is to select the policy π that maximizes the cumulative sum of discounted rewards R. Most RL algorithms use a state-value function V and a state-action-value function Q to facilitate learning based on how valuable a state or state-action pair is in generating future rewards under the current policy π. They can be formally written as:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s \right] \\ Q^{\pi}(s, a) &= \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a \right] \end{aligned} \tag{2.5}$$

where V quantifies expected cumulative discounted rewards starting from state s and following policy π. Likewise, Q quantifying expected cumulative discounted rewards starting from state s, taking an action a at state s, and continuing to follow policy π.

A common hyperparameter used in training RL algorithms is the time horizon H. This refers to the maximum number of timesteps of an episode. A finite time horizon leads to episodic tasks, while an infinite time horizon typically relates to continuing tasks.

A key feature of MDPs is their ability to capture both the immediate rewards associated with actions, and the long-term consequences of decisions through the concept of cumulative rewards. By optimizing cumulative reward over time, agents can learn strategies that balance short-term gains with long-term objectives. Thus, making MDPs invaluable tools for designing optimal decision-making policies in uncertain environments.

Proximal Policy Optimization (PPO)

PPO (Schulman et al. (2017)) stands as a prominent algorithm in RL, particularly in addressing continuous control tasks. The core idea behind PPO revolves around constraining policy updates to a local region with little compute. This mitigates the potential for large policy changes that may lead to destabilization during training, as well as the compute latency involved with computing KL-Divergences as seen other constrained policy update algorithms like TRPO (Schulman et al. (2015)). Formally, the objective can be expressed as maximizing the clipped surrogate objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{2.6}$$

where θ the policy parameters, r_t the ratio of the new policy actions to the old policy actions, $\hat{A}_t$ the advantage estimate at time step t, and ϵ a hyperparameter controlling the degree of clipping. The advantage

function A is defined by the value functions V and Q. Formally:

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s) \tag{2.7}$$

where the function A quantifies the advantage of taking action a at state s rather than following the policy π at state s, hence the name advantage function. The PPO algorithm computes these advantage estimates from a rollout of rewards by:

$$\hat{A}_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V^{\pi}(s_T) - V^{\pi}(s_t) \tag{2.8}$$

where the sum of rewards from timestep t to $T - t$ and the value of the last state s_T represents the value of the function Q^{π}. The PPO algorithm computes this advantage to weight the value of the policy ratio $r_t(\theta)$: a high advantage will drive larger updates to the new policy parameters, and vice-versa for low advantages.

By optimizing this objective function, PPO balances the trade-off between exploration and exploitation that facilitates efficient learning in complex environments.

3 PROPOSED FRAMEWORK

In this chapter, we (1) define the underlying MDP of a black-box RL attack, and (2) outline the attack procedure according to adversarial goals.

3.1 Environment Dynamics

Here, we define the state space $\mathcal{S}$, action space $\mathcal{A}$, and the terminal states $s_T \in \mathcal{S}$.

The RL-based adversary's goal is to learn a policy that maps samples to corresponding perturbations that reduce model confidence in the corresponding label. Thus, in the image domain, the state space consists of all valid images. Formally:

$$\mathcal{S} = \{\mathbf{x} | \forall i, x_i \in R \wedge x_i \in [0, 1]\} \tag{3.1}$$

where $\mathbf{x}$ is the input sample and x_i the i-th feature. For this environment, we assume that all start states s_0 are samples from a dataset that are correctly classified by the victim model, i.e., $\max_i[Z(s_0)] = y$ where y is the label.

In an RL-based attack, the dimensionality of the action space must be small enough to facilitate efficient learning, i.e. the number of actions cannot be to large. In this attack, we consider an action that selects *four* features of the state: two features that get +1 added, and two features that get -1 added. In the image domain, this would be selecting 4 pixels within the valid height and width dimensions of the image. Formally,

$$(x_\delta^i, y_\delta^i) \in \{(x, y) | 0 \leqslant x < H \text{ and } 0 \leqslant y < W\} \tag{3.2}$$

where x the pixel height coordinate, y the pixel width coordinate, $\delta \in \{+1, -1\}$ the perturbation magnitude and $i \in \{1, 2\}$ the first or second pixel

of δ perturbation magnitude. More generally,

$$\mathcal{A} = [(x_{+1}^{(1)}, y_{+1}^{(1)}), (x_{+1}^{(2)}, y_{+1}^{(2)}), (x_{-1}^{(1)}, y_{-1}^{(1)}), (x_{-1}^{(2)}, y_{-1}^{(2)})]^T \tag{3.3}$$

At each timestep t, the adversary will select an action perturbation $a_t \in \mathcal{A}$ to apply to current feature vector state $s_t \in \mathcal{S}$ and clamps the image to $[0, 1]$ for the new feature vector state $s_{t+1} \in \mathcal{S}$. Then, the adversary queries the victim model and receives output logits $Z(s_{t+1})$.

We define the terminal timestep $T > 0$ as the timestep at which the victim model makes a misclassification, i.e., $\max_i [Z(s_0)]_i \neq \max_i [Z(s_T)]_i$. In the environment, we instantiate the maximum episode length (i.e. time horizon) to $t = 100$ timesteps. With this set, this means the maximum ℓ_0 distortion value is equal to 100 timesteps $\times$ 4 actions $=$ 400 pixels perturbed, thus a $400/784 = 51\%$ distortion on an MNIST dataset with 784 pixels. Setting this time horizon cap helps the adversary not explore too far outside the distortion budget of the original image s_0 with no misclassification.

3.2 Reward Function Definition

In order to meet the adversarial goal in creating a misclassification within some distortion budget, a reward function must be designed such that the agent recieves positive rewards for taking actions that lead to mislcassifications with small distortion.

The optimization objective given in Equation 2.2 provides a numerical representation of how well the adversary is doing in reaching said desirables. In an RL-based approach, the perturbation $a_t \in \mathcal{A}$ applied to state $s_t \in \mathcal{S}$ at timestep $t < T$ must *decrease* the value of the current evaluation of the objective. Let us define the function ρ as an evaluation of this objective at a timestep t:

$$\rho(s_t) = \|s_t\|_p + f(s_t) \tag{3.4}$$

where $s_t \in \mathcal{S}$ is the state at timestep t, p distortion budget norm, and f the function used in the CW Objective (Equation 2.2) to evaluate the difference between the victim model's logit in the actual class and the next most confident logit. For this environment, we set $\kappa = 0$ in the function f, thus all misclassifications will give a value of 0 from f.

With the function ρ being an evaluation of the CW Objective at state s_t, we want the reward to reflect the improvement of this evaluation relative to the previous value $\rho(s_{t-1})$. Additionally, we want to penalize the agent for remaining stationary, and provide reward bonuses for misclassifications. Thus, the reward function R is defined:

$$R(s_t) = \begin{cases} r_- & \text{if } s_t = s_{t-1} \\ r_+ + \rho(s_{t-1}) - \rho(s_t) & \text{if } f(s_t) = 0 \\ \rho(s_{t-1}) - \rho(s_t) & \text{otherwise} \end{cases} \tag{3.5}$$

where the first case corresponds to the penalty r_- as a result of stationarity, the second case where a misclassification is achieved and a bonus r_+ is earned, and the third case where misclassification has not been reached and reward is strictly the step-wise improvement of the CW objective. We include the stationarity penalty so that the agent does not fall into the "no-move is the best move" phenomena common in RL (Saisubramanian et al. (2022)). The misclassification bonus is given to the agent to realize that this state is the ultimate goal: creating a misclassification.

3.3 Adversarial Goal and State Resetting

Traditional black-box attacks on machine learning models optimize per-sample; suggesting a potential to use techniques like RL that leverage

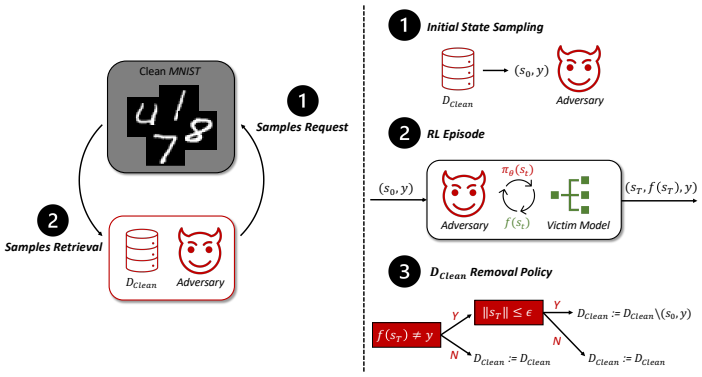

Figure 3.1: **RL Attack Overview:** (Left) The adversary collects clean samples from the dataset to insert into $\mathcal{D}_{\mathtt{Clean}}$. (Right) The adversary samples a point from $\mathcal{D}_{\mathtt{Clean}}$ at the beginning of each episode. Once an episode terminates, the adversary determines if the sample will remain in $\mathcal{D}_{\mathtt{Clean}}$ by checking if the terminal state is an adversarial example. The process repeats for every episode until $|\mathcal{D}_{\mathtt{Clean}}| = 0$.

information from many different states of the environment. To capture the information in crafting many adversarial examples different episodes of the environment, we perform a sampling from a set of samples to determine the start state s_0 at the beginning of every episode. This set of samples we refer to as $\mathcal{D}_{\mathtt{Clean}}$, as these samples are assumed to be correctly classified by the victim model. The adversary fills $\mathcal{D}_{\mathtt{Clean}}$ with all clean samples they wish to craft adversarial examples on. This process is illustrated in Figure 3.1 (Left). Thus, depending on the adversarial goal, the adversary may choose to attack a single sample, single class, multiple classes, etc (We explore all three goals in the evaluation).

At the beginning of each episode, the adversary samples (x, y) randomly from $\mathcal{D}_{\mathtt{Clean}}$ and the initial state s_0 is then set equal to x. Once a terminal state s_T is reached, a removal policy is performed on $\mathcal{D}_{\mathtt{Clean}}$ and is updated as follows:

Successful: If the resulting state s_T satisfies $\|s_T\|_p \leqslant \epsilon$ and $f(s_T) = 0$

(i.e. a misclassification within self-imposed ℓ_p-budget), we say the sample is an adversarial example. Thus, (x, y) is removed from $\mathcal{D}_{Clean}$.

Unsuccessful: If the resulting state s_T does not qualify as an adversarial example under the self-imposed ℓ_p-budget, $\mathcal{D}_{Clean}$ remains unchanged to allow the adversary to re-sample it in future episodes.

The adversary wants to craft as many adversarial examples with as few queries as possible. Thus, once an adversarial example is crafted on a specific sample, the agent will not sample it anymore due to the removal policy. This process is illustrated in Figure 3.1 (Right). Under this proposed environment, training ultimately terminates once all samples in $\mathcal{D}_{Clean}$ have been generated as adversarial examples. For our experiments, we use a ℓ_0-budget common in state-of-the-art attacks Sheatsley et al. (2023)Papernot et al. (2016)Carlini and Wagner (2017).

4 EVALUATION

With an RL-based approach to crafting adversarial examples in black-box settings, we ask the following:

1. Can RL craft adversarial examples under our proposed framework?
2. Can RL generalize crafting adversarial examples?
3. How does our RL attack compare against state-of-the-art black-box attacks?

4.1 Experimental Setup

For our experiments, the victim model architecture is a Convolutional Neural Network (CNN) trained on the MNIST Dataset with $> 99\%$ accuracy on a test dataset using an NVIDIA A100. The model is written in Pytorch (Paszke et al. (2019)) with architecture and training details (Table 4.1) similar to those used in AML literature (Carlini and Wagner

Victim Model		**PPO Policy**	
Conv. Neurons	(16,32)	π_θ Linear Neurons	(64,64,8)
Kernel Size	3	V_ϕ Linear Neurons	(64,64,1)
Stride	1	Activation	Tanh
Linear Neurons	(128,10)	Gamma	0.95
Activation	ReLU	Steps	128
Loss	CE	Batch Size	128
Optimizer	Adam	Learning Rate	5e-4
Epochs	20	K	10
Batch Size	128		
Learning Rate	1e-3		

Table 4.1: **Experiment Hyperparameters**

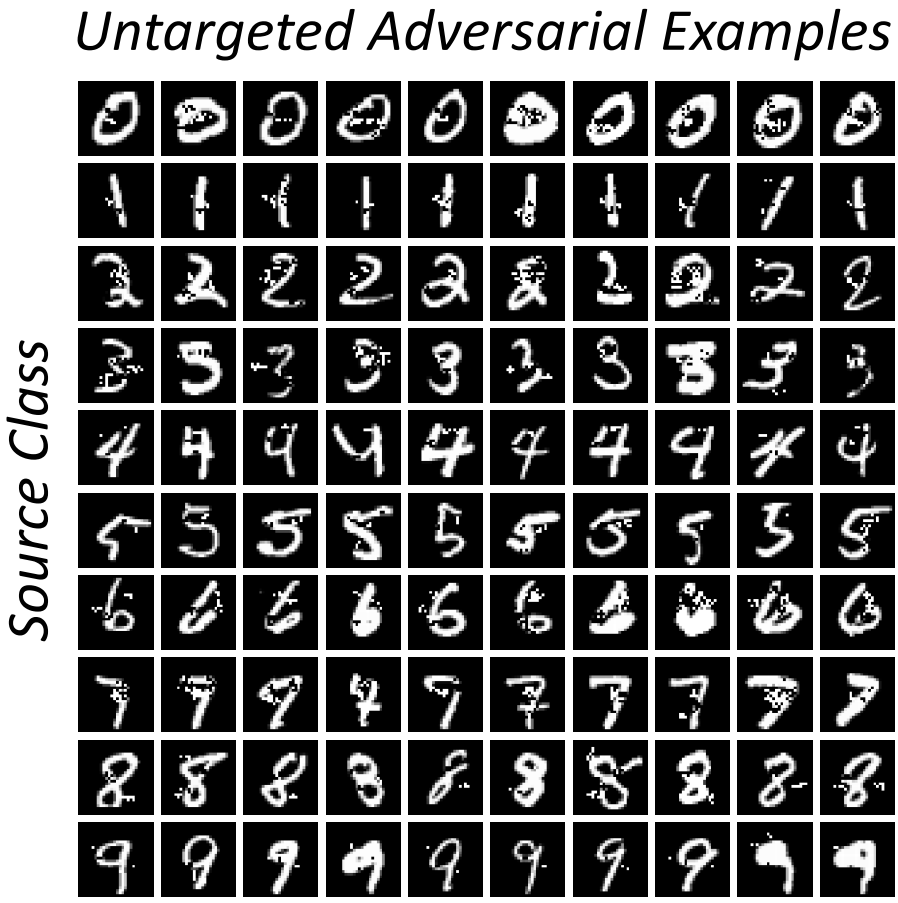

Figure 4.1: **Single-Image Attack (Examples):** The states s_T after 100K queries to the victim model. Each source class (rows) is supported by 10 different images (columns) crafted as adversarial examples. Each sample is misclassified by the victim model.

(2017))(Papernot et al. (2016))(Sheatsley et al. (2023)). The PPO policy and value networks follow a default Multi-Layer Perceptron (MLP) architecture (Table 4.1) from StableBaselines (Raffin et al. (2021)). Training the PPO agent was performed on the CPU only pool in the Center for

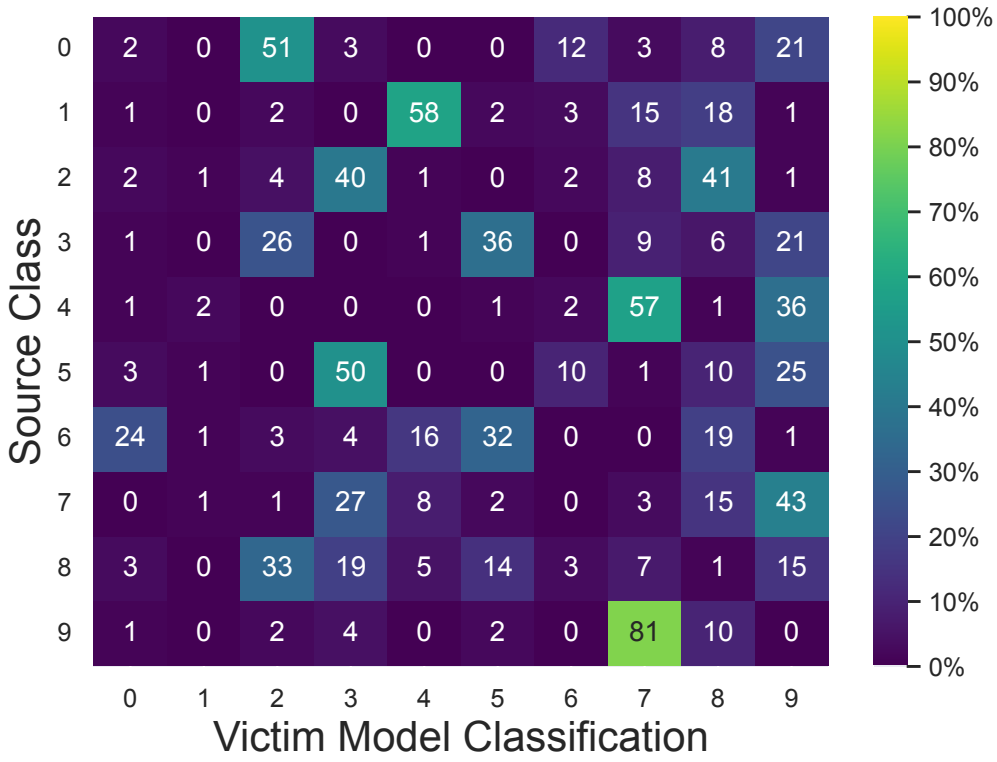

Figure 4.2: **Single-Image Attack (Classifications)**: The distribution of victim model classifications on s_T across each source class.

High Throughput Computing (Center for High Throughput Computing (2006)).

4.2 Single Sample Crafting

Given the current setup of the AML environment, we ask: *can RL craft adversarial examples?* To this end, we begin by inserting a single (x, y) sample from the clean test dataset into $\mathcal{D}_{\texttt{Clean}}$ and run the attack with $\epsilon = 0$, i.e. **no** dataset removal. Thus, the policy learns to continually optimize the C&W objective for smaller perturbations with misclassifications. We

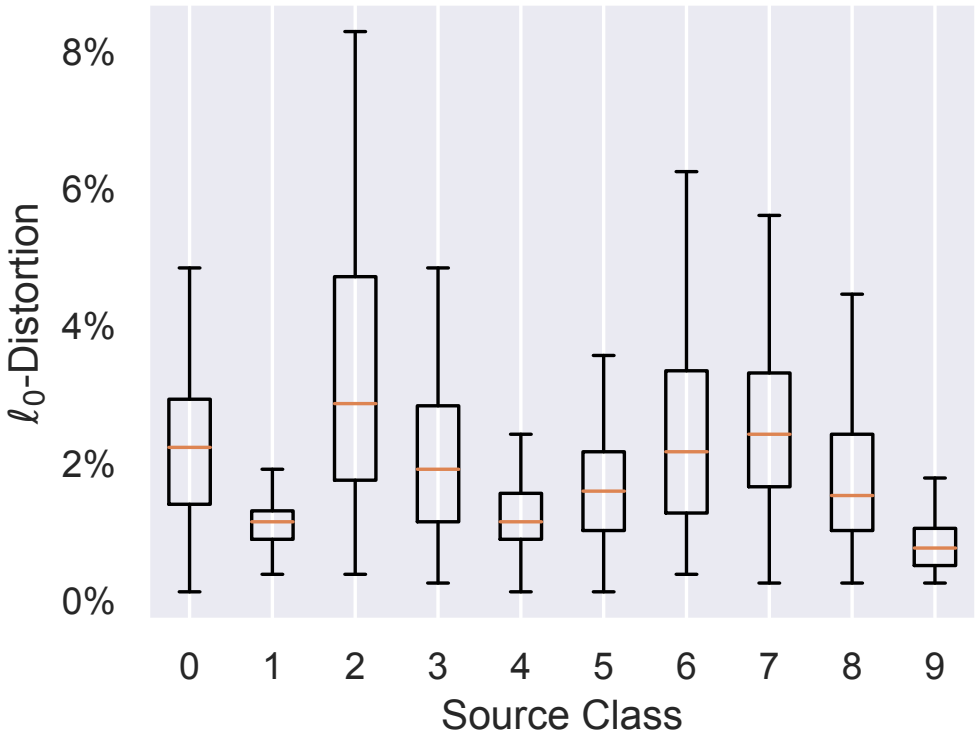

Figure 4.3: **Single-Image Attack (Distortion):** The ℓ_0-Distortion distribution of the 100 final states s_T per source class after 100K queries.

run this test 1000 times; randomly sampling 100 images per source class. At the end of 100K queries in each test, we run a final episode until termination with a fixed policy and collect the final state s_T. Of these 1000 final states, we show a sample of 10 per class in Figure 4.1.

In this untargeted attack scenario, a subsequent question arises: what is the victim model classifying these adversarial examples as? To this end, we plot a heatmap in Figure 4.2 that gives the distribution of the victim model classifications after 100K queries for the 100 clean samples per source class. We observe that the attack induces misclassification on $> 99\%$ of the images, and most notably, the vast majority of misclassified

samples are classified as 7, 8, or 9. On the contrary, very few adversarial examples are misclassified as 0, 1, and 6.

In the plot seen in Figure 4.3, we observe the mean, 25% quartile, 75% quartile, minimum, and maximum values of the ℓ_0 distortion associated with the adversarial examples crafted at the end of each run. State-of-the-art white-box attacks such as JSMA (Papernot et al. (2016)) boast ℓ_0 distortions within the range of 2% and 14%. We observe that distortion created by our RL attack on single samples falls within the range of 0.1% and 8.2%, which is competitive with that of JSMA.

Given the high success rate in crafting adversarial examples with perturbations competitive with known attacks on MNIST, we know that our formulation of an RL attack can craft adversarial examples.

4.3 Multi-Sample Generalist Crafting

Given the empirical evidence that our RL-based attack can learn to craft a single adversarial example, we ask: can RL generalize crafting adversarial examples? To test this hypothesis, we insert many samples from the clean test dataset into $\mathcal{D}_{\texttt{Clean}}$ and fix $\epsilon > 0$ to employ the dataset removal once a misclassification, or terminal state, is reached within the budget. We investigate a variety of attack settings fixated on the $\mathcal{D}_{\texttt{Clean}}$ initialization. To this end, we define the n-Class Attack: the adversary wishes to attack all clean samples of *n classes*, thus initializing $\mathcal{D}_{\texttt{Clean}}$ with all clean samples in the set of particular $\texttt{nclasses}$. In the case of MNIST with 10 classes, the total number of possible n-Class attack combinations is $\binom{10}{n}$, as there exists many choices of n classes belonging to an n-Class Attack. In our experiments, we test all possible combinations for each n-Class Attack. In the *clean* test dataset where the victim model has $> 99\%$ accuracy, the number of samples per class in MNIST is approximately 1000.

Since the RL Attack seeks to craft adversarial examples on all samples

in $\mathcal{D}_{\texttt{Clean}}$, in the fewest amount of queries q, we use the term Adversarial Yield as a measurement for the number of samples that have been crafted as adversarial examples under the ϵ budget. Formally,

$$\text{Adversarial Yield } (q) = \frac{|(\mathcal{D}_{\texttt{Clean}}\Big|_0)\setminus(\mathcal{D}_{\texttt{Clean}}\Big|_q)|}{|(\mathcal{D}_{\texttt{Clean}}\Big|_0)|} \cdot 100 \tag{4.1}$$

where we evaluate the set cardinality of $\mathcal{D}_{\texttt{Clean}}$ before training (query 0) and at the current query value (query q). This formula then becomes a measurement of the percentage of samples that have been crafted as adversarial examples.

In the 1-Class attack, we perform 10 trials per 1-Class attack combination ($\binom{10}{1} = 10$ total). In Figure 4.4, we can observe the relationship of Adversarial Yield with respect to the Query count on a per 1-Class attack combination (0-9) hue with 95% confidence intervals for a ϵ =7% threat model. The trend across all combinations is an acceleration up to a 75% - 80% Adversarial Yield, then decreasing slope to convergence at 100% Yield. The acceleration early in the training says that the agent is improving its ability to craft examples in the specific class. Further, this suggests that the agent is generalizing perturbations steps across samples in $\mathcal{D}_{\texttt{Clean}}$. A caveat with the 1-Class attack is that some combinations take much longer than others to reach 100% Adversarial Yield. Particularly, on average, the class of 2 and 7 take greater than 300K queries to do so, whereas 1, 4, and 9 take less than 50K queries.

In Figure 4.5, we observe the relationship of Adversarial Yield with respect to different ϵ values and 1-Class Attack combinations. The entries are averaged across the 10 trials and taken at a fixed Query count 100K. There are two apparent trends: (1) the Adversarial Yield increases as the budget value ϵ increases, and (2) each 1-Class combination responds differently as the ϵ budget increases. The trends across each attack combi-

nation are correlated with findings in Figure 4.3 and Figure 4.4: classes 2 and 7 are much more difficult to attack than classes 1, 4, and 9.

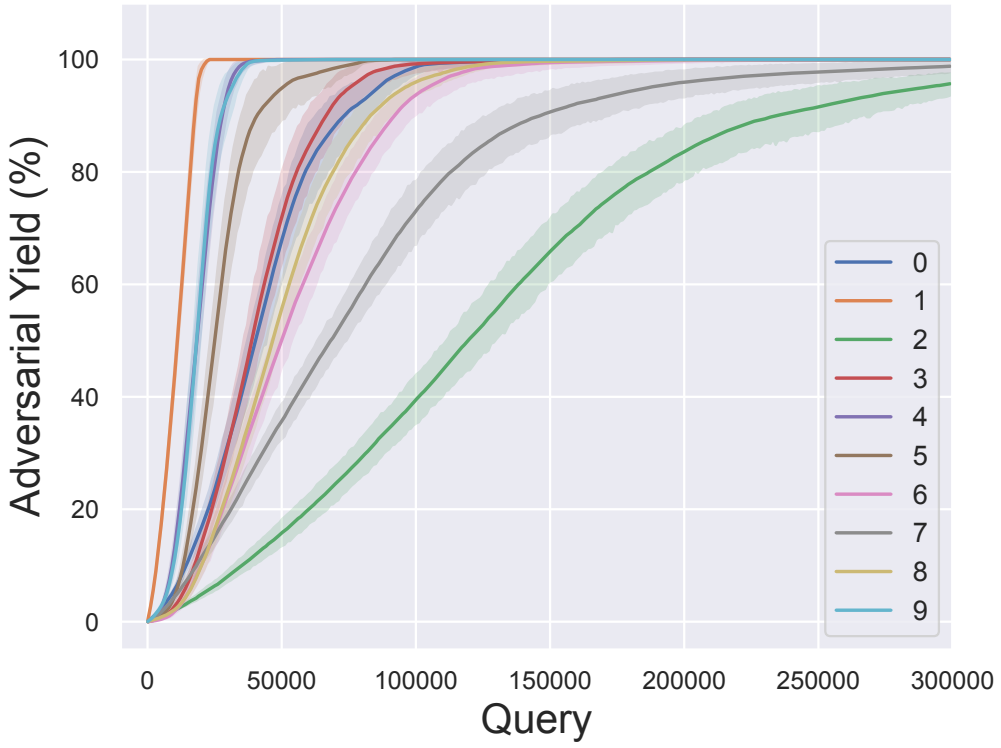

Figure 4.4: **1-Class Attack:** The Adversarial Yield (%) with respect to Query count for each 1-Class attack combination on the MNIST dataset. Each line is averaged across 10 trials of the corresponding 1-Class attack combination.

Lastly, we want to stress how generalizable an RL attack is: do the results from the 1-Class attack hold for all n-Class attack settings? To this end, we run every possible combination of n-Class attack settings for 10 trials and plot the Adversarial Yield with respect to Query count on a per n-Class attack setting averaged across combinations. The plot seen in Figure 4.6 shows the Adversarial Yield with respect to the Query count on a per n-Class attack setting hue with 95% confidence intervals

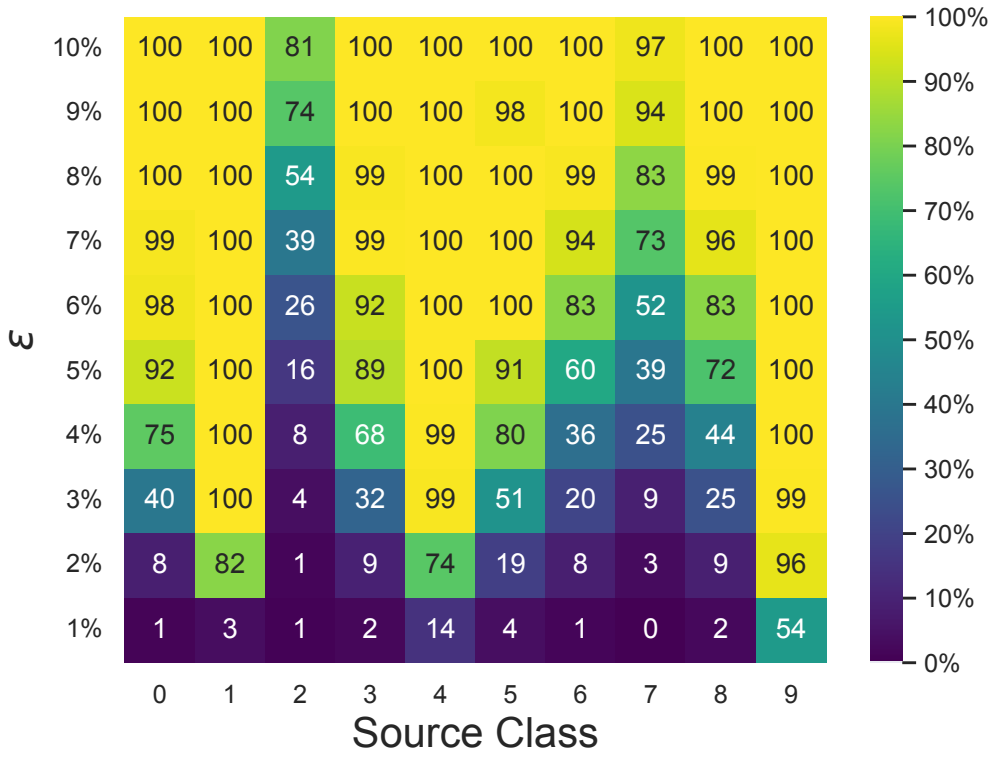

Figure 4.5: **1-Class Attack:** The Adversarial Yield (%) with respect to epsilon budget ϵ and source class combination of the 1-Class attack. The values are all taken at 100K queries.

for a ϵ =10% threat model. At first glance, we see that the slope generally decreases as n increases in the n-Class attack settings. This is due to the increased diversity in initial start states s_0 for higher values of n. However, the increasing to decreasing slope seen in Figure 4.4 is also seen in this plot across all attack settings. When the slope increases, the RL agent is crafting adversarial examples with less queries than previous examples. This is evidence that the agent is generalizing: the agent leverages the information from crafting old adversarial examples to craft others with less queries. The decreasing slope or convergence-like behavior provides

evidence that the agent learns a policy that generalizes crafting on the *majority* of samples in $\mathcal{D}_{\texttt{Clean}}$; samples that are left are considered outliers to the generalist adversarial policy.

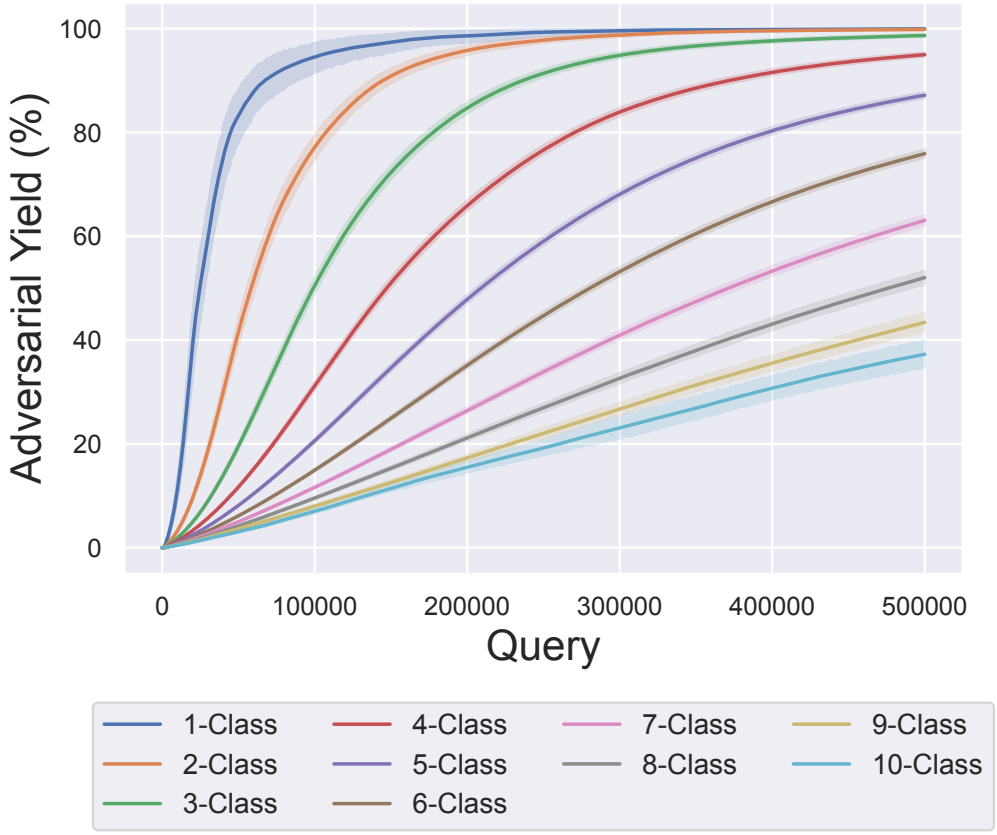

Figure 4.6: **n-Class Attack:** The Adversarial Yield (%) with respect to Query count for all n-Class attacks on the MNIST dataset. Each line is averaged across all possible combination of the corresponding n-Class attack.

4.4 RL vs. Known Attacks

A generalist adversarial attack is desirable for an adversary seeking to generate adversarial examples on *many* samples with as little interaction with the victim model as possible. However, this begs the question: how well does the RL attack, RLAttack, stack up against state-of-the-art black box attacks? To this end, we perform an Adversarial Yield vs. Query count evaluation between our RLAttack and SquareAttack (Andriushchenko et al. (2020)).

To observe the utility of SquareAttack, we run it on each of the 1-Class attack settings with an ℓ_2-budget of $\epsilon = 4.0$. To emulate a time horizon in the RLAttack, the SquareAttack is given 100 queries maximum on samples to reach a misclassification within specified budget. If an adversarial example is achieved, a removal policy like the RLAttack is employed. We run 10 trials per 1-Class attack combination using SquareAttack and plot the results with 95% confidence intervals in Figure 4.7. It is clear that classes 2 and 7 require more queries to reach higher Adversarial Yield, and conversely, fewer queries in classes 1 and 9 to reach higher Adversarial Yield. These are consistent with results that were found in our RLAttack. It is also apparent that these lines have a linear relationship with respect to queries. This is expected because SquareAttack operates on a per-sample basis and thus, fails to leveraging experience from crafting prior adversarial examples.

We are interested to see how well, on average, the RLAttack performs in the 1-Class attack setting against SquareAttack. To this end, we fix the RLAttack ℓ_0-budget ϵ =7% and the SquareAttack ℓ_2-budget $\epsilon = 4.0$. For each 1-Class attack combination per attack, we run 10 trials and average the results across all combinations per class and plot the results with 95% confidence intervals in Figure 4.8. It is clear that the RLAttack after few queries increases Adversarial Yield rapidly, as this was seen across the beginning of all 1-Class attacks in Figure 4.4. Leveraging the crafting of ad-

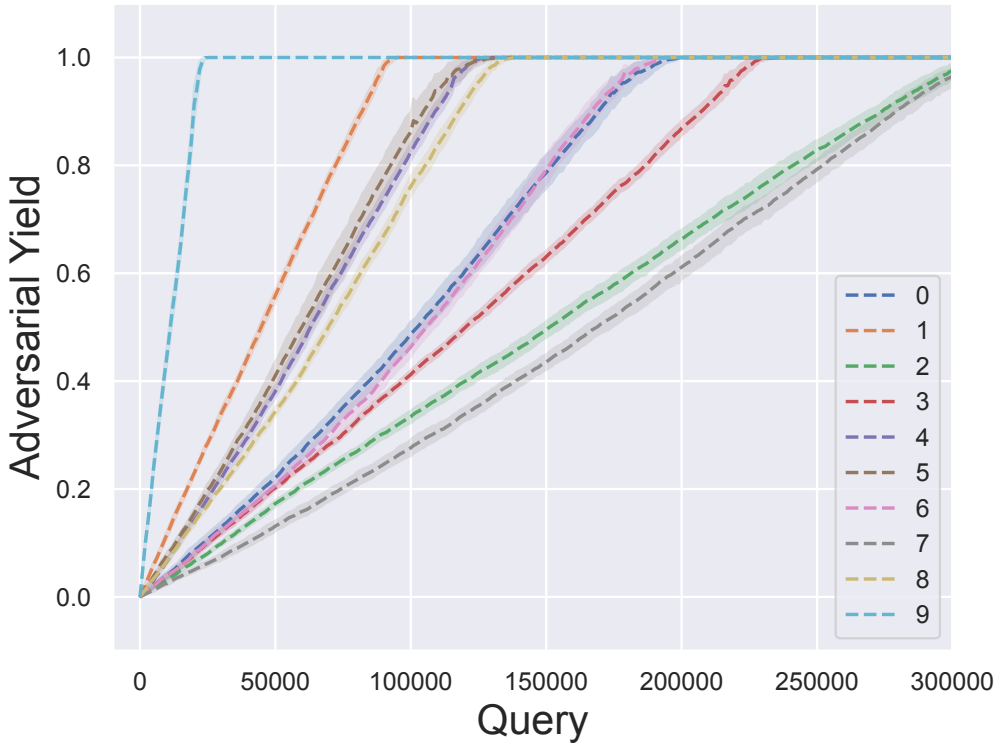

Figure 4.7: **SquareAttack 1-Class Attack:** The Adversarial Yield (%) with respect to Query count for each 1-Class combination of the MNIST dataset.

versarial examples to exploit other samples in distribution with RLAttack introduces new adversarial capabilities when considering this whole-sale version of producing adversarial examples.

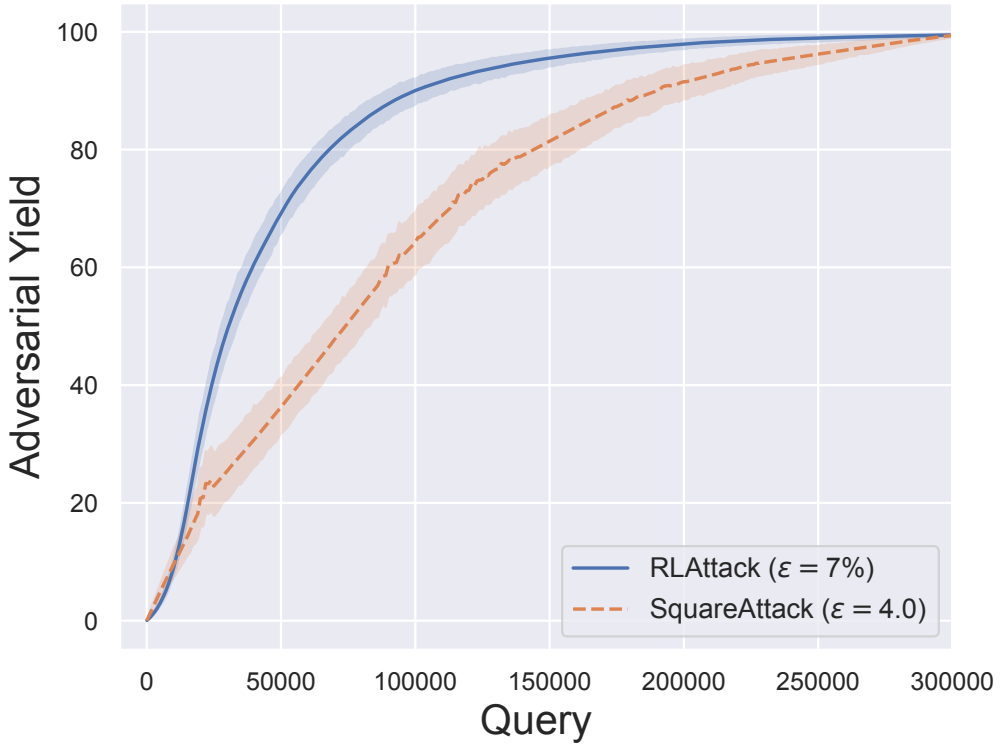

Figure 4.8: **RLAttack vs. SquareAttack in 1-Class Attacks:** The Adversarial Yield (%) with respect to Query count averaged across all 1-Class attack combinations of the MNIST dataset.

5 DISCUSSION

In this section, we discuss the limitations of this work and potential future directions.

5.1 Limitations

While the state space of the RL attack consist of grayscale images, we choose to model the PPO policy and value networks with Multi-Layer Perceptrons. Despite the more favorable CNN architecture for this image attack, we achieve compelling results that emphasize the potency of a Reinforcement Learning attack. This suggests that its effectiveness is not contingent upon the complexity of the model architecture, and further studies on this attack framework may choose to use a CNN to achieve even better results.

This thesis performs a query analysis of an ℓ_0 based RL attack against an ℓ_2 based SquareAttack. While ℓ_0 and ℓ_2 norms quantify image distortion differently, we weight the takeaways of our attack on the trends of query-efficiency more than query counts. Our method exhibits accelerated and generalizable performance improvements, with non-linear trend in query-efficiency. SquareAttack, operating per-sample, shows a linear progression, underscoring the adaptability of our approach.

In this work, we model the agents actions on MNIST as selecting 4 features of the feature vector. This approach is inspired by the steps taken in the JSMA on MNIST, where 2 pixels are selected to turn white (or black). Our perturbations measurement, thus, became the same as what is used in the JSMA (ℓ_0-norm). By manipulating the state and actions space of the environment in this work, other variants of an RL-based attack can be studied under different ℓ_p threat models.

In our approach, several limitations should be acknowledged. An RL-

based method introduces additional complexity in terms of training time and computational resources. The requirement to model the attack as an MDP can also result in less straightforward design and implementation. Furthermore, the RL model's performance is heavily dependent on the exploration strategy and reward engineering, which can introduce variability and potentially limit the robustness and generalizability of the attack. Lastly, while RL enables leveraging information from previous examples, this advantage may be offset by the overhead associated with policy learning, particularly in high-dimensional input spaces where standard black-box methods might remain more efficient.

5.2 Future Directions

This thesis primarily provides a casting of evasion attacks in adversarial machine learning to a reinforcement learning problem through a Markov Decision Process. The utility of using RL in this settings comes from generating many adversarial examples with fewer and fewer victim model interaction. An interesting study branching off this idea of "learning to craft" may be to consider no removal policy and evaluate the agents performance on a test set of samples that are excluded from set of start states. Therefore, the model continues to train on all start states from a set of samples and is evaluated based off of unseen samples, providing a new angle in studying an RL attack's generalizability.

With the development of a new attack framework with RL, the question that immediately arises: how can this be implemented as a defense mechansim? To this end, an intuitive approach may be to flip-flop the training of a machine learning model and an RL attack much like the standard adversarial training technique. However, since the RL attack learns a much more broad understanding of the decision boundary, it may be more interesting to study a defense that facilitates updates to the machine

learning model based on the value function of a sample according to an RL attack.

6 CONCLUSION

In this thesis, we cast evasion attacks as a reinforcement learning problem. We first define a Markov Decision Process that models feature vectors as states, perturbation steps as actions, and reward as improvements to minimizing sample distortion and model confidence. We use a Proximal Policy Optimization algorithm to learn a policy that crafts adversarial examples under this framework. Then, we test the effectiveness of this attack on single samples from an MNIST dataset and find that our method fools the victim model $> 99\%$ of the time with distortions competitive with the state-of-the-art. Next, we evaluate the generalizability of our attack by training the policy to craft many examples, and observe that the number of victim model queries required to produce adversarial examples decreases within the firs 20K queries. Last, we perform a query analysis of our attack against SquareAttack, a state-of-the-art black-box attack, and show that this acceleration behavior of adversarial examples generated with respect to victim model queries allows it to surpass the SquareAttack, a state-of-the-art black-box attack, in generating hundreds of adversarial examples with less queries. By framing using reinforcement learning as an evasion attack under our proposed model, this suggests a new dimension in studying model robustness and machine learning security.

REFERENCES

Afsar, M. Mehdi, Trafford Crump, and Behrouz Far. 2022. Reinforcement Learning based Recommender Systems: A Survey. *ACM Comput. Surv.* 55(7):145:1–145:38.

Andriushchenko, Maksym, Francesco Croce, Nicolas Flammarion, and Matthias Hein. 2020. Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII*, 484–501. Berlin, Heidelberg: Springer-Verlag.

Carlini, Nicholas, and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57. ISSN: 2375-1207.

Center for High Throughput Computing. 2006. Center for High Throughput Computing.

Chen, Pin-Yu, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. 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*, 15–26. AISec '17, New York, NY, USA: Association for Computing Machinery.

Ibarz, Julian, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. 2021. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research* 40(4-5):698–721.

Liu, Xiao-Yang, Ziyi Xia, Jingyang Rui, Jiechao Gao, Hongyang Yang, Ming Zhu, Christina Wang, Zhaoran Wang, and Jian Guo. 2022. FinRL-Meta: Market Environments and Benchmarks for Data-Driven Financial

Reinforcement Learning. In *Advances in Neural Information Processing Systems*, ed. S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, vol. 35, 1835–1849. Curran Associates, Inc.

Papernot, Nicolas, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 506–519. ASIA CCS '17, New York, NY, USA: Association for Computing Machinery.

Papernot, Nicolas, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 372–387.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 8026–8037. 721, Red Hook, NY, USA: Curran Associates Inc.

Raffin, Antonin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22(268):1–8.

Saisubramanian, Sandhya, Shlomo Zilberstein, and Ece Kamar. 2022. Avoiding Negative Side Effects Due to Incomplete Knowledge of AI Systems. *AI Magazine* 42(4):62–71.

Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *Proceedings of the 32nd international conference on machine learning*, ed. Francis Bach and David Blei, vol. 37 of *Proceedings of Machine Learning Research*, 1889–1897. Lille, France: PMLR.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. ArXiv:1707.06347 [cs].

Sheatsley, Ryan, Blaine Hoak, Eric Pauley, and Patrick McDaniel. 2023. The space of adversarial strategies. In *32nd usenix security symposium (usenix security 23)*, 3745–3761. Anaheim, CA: USENIX Association.