Adversarial Attacks on Deep Algorithmic Trading Policies

Nancirose Piazza, *[1], Yaser Faghan, †[2], Vahid Behzadan, ‡[1], and Ali Fathi §[3]

[1]Secure and Assured Intelligent Learning (SAIL) Lab
University of New Haven, USA
[2]Instituto Superior de Economia e Gestão and CEMAPRE
Universidade de Lisboa, Portugal
[3]Enterprise Model Risk Management Group, Royal Bank of Canada (RBC)

Abstract

Deep Reinforcement Learning (DRL) has become an appealing solution to algorithmic trading such as high frequency trading of stocks and cyptocurrencies. However, DRL policies are shown to be susceptible to adversarial attacks. It follows that algorithmic trading DRL agents may also be compromised by such adversarial techniques, leading to policy manipulation. In this paper, we develop a threat model for deep trading policies, and propose two active attack techniques for manipulating the performance of such policies at test-time. Additionally, we explore the exploitation of a passive attack based on adversarial policy imitation. Furthermore, we demonstrate the effectiveness of the proposed attacks against benchmark and real-world DQN trading agents.

1 Introduction

The pursuit of intelligent agents for automated financial trading is a challenge that has captured the interest of researchers and analysts for decades according to Cartea et al. [2015]. The process of trading is well depicted as an online decision making problem involving two critical steps of *summarizing the market condition* and *execution of optimal actions*. For many years, algorithmic trading suffered from various problems ranging from difficulties in representations of the complex market conditions to real-time approaches to optimal decision-making in the trading environment. With recent advances in Machine Learning (ML), particularly in deep learning and Deep Reinforcement Learning (DRL), such challenges are dramatically alleviated via numerous novel proposals and architectures that enable end-to-end approaches to algorithmic trading (Pricope [2021]). In this context, end-to-end refers to the direct mapping of high-dimensional raw market and environment observations to optimal decisions in real-time. As data-driven agents, many such algorithms rely on sources of data that are either externally or collectively maintained, examples of which include market indicators (Cartea et al. [2015]) and social indicators (e.g., sentiment analysis from Twitter feeds by Kaur [2017]).

While the growing interest in adoption of DRL techniques for algorithmic trading is justified by their impressive success in other domains, the threat of adversarial manipulation in such

*Secure and Assured Intelligent Learning (SAIL) Lab, University of New Haven, New Haven, USA. `npiaz1@newhaven.edu`

†Instituto Superior de Economia e Gestão and CEMAPRE, Universidade de Lisboa, Portugal. `yaser.kord@yahoo.com`

‡Secure and Assured Intelligence Learning (SAIL) Lab, University of New Haven, New Haven, USA. `vbehzadan@newhaven.edu`

§Enterprise Model Risk Management Group, Royal Bank of Canada (RBC). `ali.fathi@rbc.com`

systems is yet to be explored. Recent developments in the domain of adversarial machine learning have brought attention to the security challenges in regards to the vulnerability of machine learning models to adversarial attacks, a paper by Papernot et al. [2018]. Instances of such attacks include adversarial examples like Fast Gradient Sign Method by Goodfellow et al. [2014], which are strategically induced perturbations in the input vectors that are not easily detectable by human observers.

Adversarial attacks can impact all deep learning and classical machine learning models, including DRL agents, investigated by Behzadan and Munir [2018]. Recent work by Behzadan Behzadan and Munir [2017a, 2018], Behzadan [2019] establish that DRL algorithms are vulnerable to adversarial actions at both training and inference phases of their deployment. This discovery is further verified in settings such as video games (Huang et al. [2017]), robotics (Clark et al. [2018]), autonomous navigation (Behzadan and Munir [2019]), and cybersecurity (Han et al. [2018]). Yet, the extent, severity, and the dynamics of such vulnerabilities in DRL trading agents are yet to be addressed.

Adversarial perturbations of DRL trading policies are also significant form the financial Model Risk Managment (MRM) point of view (Reserve [2011], of the Superintendent of Financial Institutions [OSFI], Morini [2011]) since the existence of such vulnerabilities can be traced back to the algorithmic underpinnings of these systems. However, principal differences between traditional financial models and algorithmic trading systems pose additional challenges for quantifying the resulting model risk. For instance, the number of model components involved in an algorithmic trading system can be large and hence, fusion of otherwise individually negligible residual model risk may result in significant system errors. Furthermore, There exist the adaptive nature of DRL based algorithms where the model components are re-calibrated (e.g., through retraining) based on a low latency schedule. It should also be noted that unlike other areas of quantitative modelling in finance (such as asset pricing or credit risk), benchmarking of model components in algorithmic systems is difficult due to competition considerations, as there may be restrictions for conducting open box validation of proprietary models within a firm.

In this paper, we investigate test-time adversarial attacks against DRL trading agents. The main contributions are:

- We present a threat model for DRL trading policies, identifying susceptible attack surfaces and practical attack vectors at test-time.
- We establish the vulnerability of current DRL trading policies to adversarial manipulation for active test-time attacks.
- We explore Imitation Learning for adversarial purposes after acquisition of expert demonstrations, both perfect and imperfect, from a passive test-time attack for policy imitation.
- We investigate the transferability of our perturbation attacks from the imitated agents to the target agent.
- We demonstrate the efficacy of the proposed attack vectors in manipulating DRL trading agents.

The remainder of the paper is as followed: Section 2 presents an overview of reinforcement learning and a review of the security issues in electronic trading platforms. Section 3 proposes a DRL threat model for trading DRL agents, outlining various attack surfaces and vectors that can be exploited by an adversary. Section 4 provides the details of our experimental setup for investigating the proposed attack mechanisms, the results of which are presented in Section 5 and 6. The paper concludes in Section 7 with a summary of our findings, as well as discussions on future directions of research on the security of deep trading policies.

2 Background

2.1 Reinforcement Learning, Value Iteration & Deep Q-Learning

Reinforcement learning (RL) is concerned with agents that interact with an environment and exploit their experiences to optimize a sequential decision-making policy. RL can be formally modeled as learning to control a Markov Decision Process (MDP) $M = (S, A, R, P)$, where S is the set of reachable states in the process, A is the set of available actions, R is the mapping of transitions to the immediate reward, and P represents the transition probabilities (i.e., state dynamics), which are initially unknown to RL agents. At any given time-step t, the agent is at a state $s_t \in S$, chooses an $a_t \in A$, transitions from s_t to a state s_{t+1} according to the transition probability $P(s_{t+1}|s_t, a_t)$ and receives a reward $r_{t+1} = R(s_t, a_t, s_{t+1})$. The solution to an MDP problem is a policy $\pi(s)$ that is a mapping from states to actions. The goal of RL is to learn a policy that maximizes the expected discounted return $E[R_t]$, where $R_t = \sum_{t=0}^{N} \gamma^k r_t$; with r_t denoting the instantaneous reward received at time t, and γ is a discount factor $\gamma \in [0, 1]$. The value of a state s_t is defined as the expected discounted return from s_t following a policy π, that is, $V^\pi(s_t) = E[R_t|s_t, \pi]$. The state-action value (Q-value) $Q^\pi(s_t, a_t) = E[R_t|s_t, a_t, \pi]$ is the value of state s_t after applying action a_t and following a policy π thereafter.

The solution approaches to RL include value iteration algorithms that optimize a value function (e.g., $V(.)$ or $Q(.,.)$) to extract the optimal policy from it. As an instance of value iteration algorithms, *Q-Learning* aims to maximize for the action-value function Q through the iterative formulation of Eq. (1):

$$Q(s, a) = R(s, a) + \gamma max_{a'}(Q(s', a')) \tag{1}$$

Where s' is the state that emerges as a result of action a, and a' is a possible action in state s'. The optimal Q value given a policy π is defined as: $Q^*(s, a) = max_\pi Q^\pi(s, a)$, and the optimal policy is given by $\pi^*(s) = \arg\max_a Q(s, a)$.

The Q-learning method estimates the optimal action policies by using the Bellman formulation to iteratively reduce the *TD-Error* given by $Q_{i+1}(s, a) - \mathbf{E}[r + \gamma \max_a Q_i]$ for the iterative update of a value iteration technique. Practical implementation of Q-learning is commonly based on function approximation of the parametrized Q-function $Q(s, a; \theta) \approx Q^*(s, a)$. A common technique for approximating the parametrized non-linear Q-function is via neural network models whose weights correspond to the parameter vector θ. Such neural networks, commonly referred to as Q-networks, are trained such that at every iteration i, the following loss function is minimized:

$$L_i(\theta_i) = \mathbf{E}_{s,a\sim\rho(.)}[(y_i - Q(s, a, ; \theta_i))^2] \tag{2}$$

where $y_i = \mathbf{E}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})|s, a]$, and $\rho(s, a)$ is a probability distribution over states s and actions a. This optimization problem is typically solved using computationally efficient techniques such as Stochastic Gradient Descent (SGD).

A Deep Q-Network (DQN) by Mnih et al. [2015] is a training algorithm and implementation of Q-value estimation by a neural network function approximator. Techniques such as *experience replay* and a target network are used in an DQN to stabilize the training process and maintain the *i.i.d.* (Independent and Identically Distributed) property of the data. Mnih et al. [2015] demonstrate the application of this new Q-network technique to end-to-end learning of Q-values in playing Atari games based on observations of pixel values in the game environment.

2.2 State of Security in Algorithmic Trading

In recent years, electronic trading platforms have made access to global capital markets easier for the public, resulting in a lower barrier to entry and influx of traffic across these platforms.

The growing interest in such trading platforms and technologies is however accompanied by the increasing risks of cyber attacks. While the literature on the cybersecurity issues of current trading platforms is scarce, few industry-sponsored studies report concerning issues in deployed trading platforms. One such study on the exposure of security flaws in trading technologies by Hernandez [2018] evaluates various popular desktop, mobile and web trading service platforms against a standard list of security checks, and reports that these trading technologies are in general far more susceptible to cyber attacks than previously-reviewed personal banking applications from 2013 and 2015. The security checks consisted of features such as 2-Factor Authentication (2FA), encrypted communications, privacy mode, anti-reverse engineering, and hard-coded secrets. This study reports that 64% of the reviewed trading applications rely on unencrypted communication channels for authentication and trading data. Also, the author finds that many trading applications utilize poor session management and SSL certificate validation, thereby enabling Man-in-The-Middle (MITM) attacks. Furthermore, this report points out the wide-scale susceptibility of such platforms to remote Denial of Service (DoS) attacks, which may render the applications useless. Building on the findings of this study, our paper investigates attacks that leverage the aforementioned vulnerabilities to manipulate deep algorithmic trading policies.

3 Threat Model of DRL Trading Agents

Adversarial attacks against DRL policies aim to compromise one or more aspects of the Confidentiality, Integrity, and Availability (CIA) triad in the targeted agents Behzadan and Munir [2018]. More specifically, the *Confidentiality* of a DRL agent refers to the need for confidentiality of an agent's parameters, such as the policy or reward function. The *Integrity* of a DRL agent relies on the policy behaving as intended by the user. *Availability* refers to the agent's capability to execute its task when needed. At a high-level, the threat landscape of DRL agents can be captured in terms of the *Attack Surface* and *Attack Model* of the agent by Behzadan [2019], as outlined below.

3.1 Attack Surface and Vectors

Adversarial attacks may target all components of a DRL agent, including the environment, agent's observation channel, reward channel, actuators, and training components (e.g., experience storage and selection), as identified by Behzadan Behzadan [2019].

Figure 1 illustrates the components of a DRL trading agent at test-time. In the context of algorithmic trading, the observation of the environment is gathered from various sources such as market indicators, social media indicators, and exchanges– we refer to these sources as input channels. This data is prepossessed and feature engineered to create the agent's observation of the state. These states are part of the observation returned by the environment to the agent along with the reward. Through the observation channel, an adversary may intercept the observation and exchange it for a perturbed observation, otherwise called a Man-In-The-Middle (MITM) attack. An adversary may also impose a delay the observation channel through a Denial of Service (DoS) attack. It has been shown that slight perturbations of the observation state impact DRL agent performance by Ding and Dong [2020]. The reward channel is often tied to internal securities such as bank accounts or portfolios, and hence are less susceptible to external adversarial manipulation. However, any external component reachable by the agent can be compromised implicitly.

3.2 Attack Model

The capabilities of an adversary are defined by two factors: actions available to the adversary and information available about the target. This section presents a classification of attacks

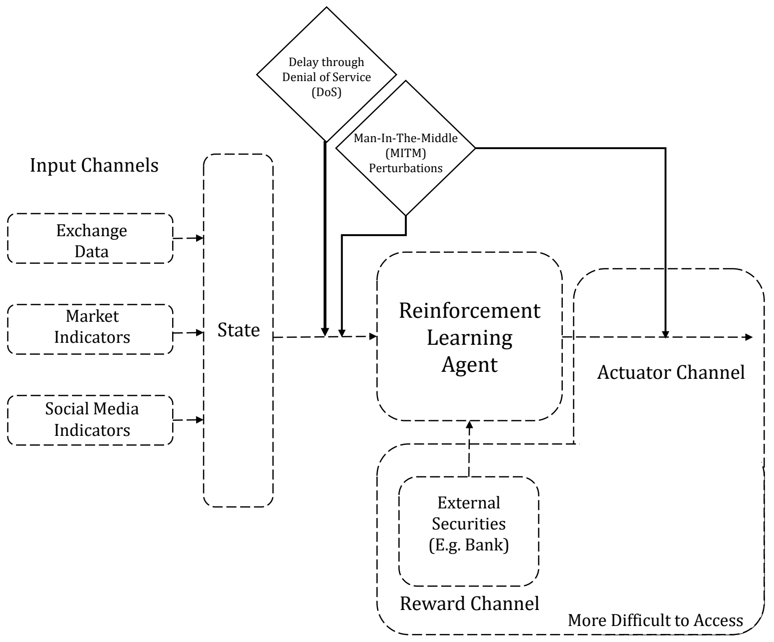

Delay through Denial of Service (DoS)
Man-In-The-Middle (MITM) Perturbations
Input Channels
Exchange Data
Market Indicators
Social Media Indicators
State
Reinforcement Learning Agent
Actuator Channel
External Securities (E.g. Bank)
Reward Channel
More Difficult to Access

Figure 1: Attack Surface and Vectors of a DRL Trading Agent at Test-Time

and adversaries at the inference phase based on the aforementioned factors. According to the available information, attacks are classified as whitebox or blackbox. Whitebox refers to when the adversary has sufficient knowledge of the target's parameter to directly craft an effective perturbation, and blackbox refers to the vice versa scenario.

Perturbations in observation affect both test-time and train-time. While this paper focuses on test-time attacks, it is noteworthy that during training, additional error is bootstrapped, potentially impacting learned policies. Work by Behzadan and Munir [2017b] show that training-time attacks under certain conditions with sufficiently high perturbation rates resulted in the agent's inability to recover performance upon test-time evaluation under non-adversarial conditions.

3.2.1 Test-Time Attacks

Test-time or inference-time attacks may be active or passive. Active attacks require adversarial intervention to manipulate the DRL policy. Instances of such attacks include adversarial examples (Goodfellow et al. [2014],Carlini and Wagner [2017],Su et al. [2019]) and delay induction in observations. Passive attacks gather information about the target agent by observing the target's behavior in various states. With sufficient observations of state-action pairs, the adversary can reconstruct the targeted policy and compromise the Confidentiality of the targeted, proprietary agents Behzadan and Hsu [2019].

Active attacks can be classified under targeted and non-targeted attacks. Successful non-targeted aim to have the policy select any action other than the one prescribed by the policy modifying (i.e., perturbing) the true observation with a perturbed observation. Targeted attacks craft perturbations such that the target selects a particular sub-optimal action a'.

In the category of passive attacks, Imitation Learning and Inverse Reinforcement Learning are avenues an adversary may exploit to either attack their target agent or steal components of the agent such as its policy. As demonstrated in work by Behzadan and Hsu [2019], adversaries can gather additional information through policy imitation, thereby enabling whitebox attacks against blackbox targets.

3.2.2 Training-Time Attacks

Training-time attacks are also referred as data poisoning attacks by impairing an agent's capability to learn optimally. In such attacks, the adversary manipulates the training data via injecting false samples, mislabeled samples, or overrepresented samples to manipulate the distribution of the training data according to Goldblum et al. [2021]. Though typically studied in supervised and unsupervised learning tasks, data poisoning can also apply to DRL as demonstrated by Behzadan and Munir [2017b].

4 Experimental Setup

We demonstrate the proposed attacks on two trading agents based on DQN policies with varying complexity, one we refer to as basic DQN which uses a simple OpenAI Gym[1] environment to emulate trading, and the other is based on an open-source framework called TensorTrade[2] which leverages a more realistic OpenAI Gym environment mimicking real-world trading settings. Our basic DQN represents less complex agents while TensorTrade's DQN will demonstrate the real-world impact of such attacks that have external components tied to the agent like a portfolio. In fact, TensorTrade is currently used and deployed for actual DRL-based trading in online cryptocurrency and stock exchanges.

There are general choices for the components of MDP M. The state space may contain a subset of four common prices such as open, high, low, and close. Technical Indicators refer to other measurements traders use to assess a stock are used in the state space. The duration of a timestep can be any interval, eg. milliseconds, minutes, hours and each interval is called a bar. The action space may include buy/sell/hold quantities, which can be continuous or discrete. Environments will implement a commission fee upon changing position (buy/sell). The reward function can be profit/loss or a more detailed metric such as the Sharpe value. Training is usually on historical data.

4.1 Basic Trading Environment

In the basic trading environment, the historical data is sourced from Yandex N.V. (YNDX)(yan) between the period of 2015-2016. The dataset is comprised of samples representing a one-minute temporal resolution, and the dynamic of the price during that minute is captured by four values: open price, high price, low price, and close price. Our agent can only hold, sell or buy a single stock. Table 1 details the specifications of the Basic Stock Environment. Table 2 contains hyperparameters of the DQN agent trained in this environment.

4.2 TensorTrade Environment

The TensorTrade environment (TT) can implement a portfolio that holds wallets of various coins or currencies. The data used for this setup is included with TT as a demonstration of training. This dataset is dated from the start of 2020, and contains the open, high, low, close and volume prices at hourly intervals. It also includes technical indicators such as the Relative Strength Indicator (RSI) and Moving Average Convergence Divergence (MACD) and $log(C_t) - log(C_{t-1})$ where C_t is the closing price at timestep t as the dataset features. Our portfolio starts with 10,000 USD and 10 BTC. We use the risk-adjusted reward scheme and manage-risk action scheme provided by TT. The risk-adjusted reward scheme uses the Sharpe Ratio which is defined by the equation below:

$$S_a = \frac{E[R_a - R_b]}{\sigma_a}$$

[1]OpenAI Gym, (2016), GitHub repository, https://github.com/openai/gym
[2]TensorTrade, (2019), GitHub repository, https://github.com/tensortrade-org/tensortrade

Table 1: Specifications of the Basic Stock Environment & TensorTrade's Environment

	Basic DQN	TensorTrade's DQN
Observation Space	– Past 10 bars consisting of: RHP, RLP, RCP – [0 or 1] bought share indicator – Profit or loss from current position RHP is Relative High Price RLP is Relative Low Price – RCP is Relative Close Price	Past 20 tuples of: – $\log(C_t)$ - $\log(C_{t-1})$ C_t is the closing price at timestep t – MACD (fast=10, slow=50, signal=5) – RSI(period=20)
Action Space	- Buy a share - Wait - Close the position (sell)	Managed Risk Scheme – Product(stop, take, trade size, [buy, sell]) (180 actions) – Wait/hold action (indexed at 0)
Reward	- No position: [100 * (SP-BP) / BP]% - C% - Position: - C% SP is Sold Price BP is Bought Price C is Commission	Risk-Adjusted Scheme Sharpe Ratio
Termination	Episode length > 250	Timestep > 250

Basic DQN

No. Timesteps	10^5
γ	0.99
Learning Rate	10^{-4}
Replay Buffer Size	10^5
First Learning Step	1000
Target Network Update Freq.	1000
Exploration	PSN
Exploration Fraction	0.1
Final Exploration Prob.	0.02
Max. Total Reward	250
Note: Parameter-Space Noise (PSN)	

TensorTrade's DQN

No. Timesteps	250
Episodes	100
Epochs	80
γ	0.9999
Learning Rate	10^{-5}
Replay Buffer Size	10^3
Target Network Update Freq.	10^3
Exploration	ϵ-greedy
Optimistic Initialization ϵ	0.9
Minimum ϵ	0.05
Decay ϵ every N steps	200

Table 2: Training Hyperparameters

where R_a is the asset return, R_b is the risk-free return, and σ_a is the standard deviation of the asset excess return. The manage-risk action scheme scales the action space depending on provided arguments such as trade size, stop and take. The default trade size is 10 which implies there will be a list of 10 trade sizes that are uniformly spaced. For instance, trade size of 3 implies 33.3%, 66.6%, and 99.9% of the balance can be traded. *Take* is a list of possible take profit percentages from an order, and *stop* is a list of possible stop loss percentages from an order. The action space is the resulting product of take, stop, trade size, and action type which is buy or sell. There is one additional action: wait/hold. In our case, we have an action space size of 181. This information as well as training hyperparameters are summarized in Table 1 and Table 2, respectively. There are other simpler reward (e.g., SimpleProfit) and action (e.g., Buy Sell Hold BSH) schemes available with TT.

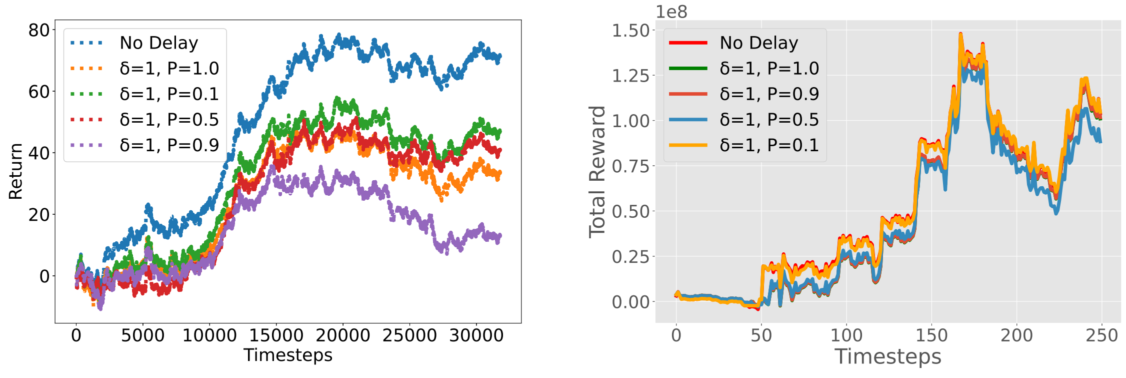
5 Active Test-Time Attacks

In this section, we investigate the impact of adversarial attacks on deep trading agents at test-time. To preserve the realism of our study, we limit the scope of our investigation to attacks that satisfy the following constraints: (1) Attacks are limited to manipulating the observation channel of the target. (2) Attacks are limited to perturbations that are not immediately detected by common human or automated anomaly detection mechanisms.

We implement 2 different types of attack namely untargeted delay attacks, and untargeted/targeted adversarial perturbation attacks. This study considers whitebox attacks only. However, as demonstrated in Behzadan and Hsu [2019], it is also feasible to reverse-engineer blackbox policies via imitation learning, thereby converting blackbox attacks to whitebox.

5.1 Non-Targeted Delay Attacks

We evaluate through non-targeted attacks on the observation channel through a single, most recent window history tuple of their features. The observation delay is of 1 timestep where a tuple of values seen at timestep $t-1$ will be received at timestep t. This is both practical and representative of minimal interference. Because there is no adversarial preference of when to implement the delay, this is non-targeted. Likewise, a targeted delay attack implements an intended timing; however, we did not pursue this. Results are presented in Figure 2. This type of non-targeted attack should be of concern to traders because of lack of computational expense, and adversarial predisposition because anomalies are masked by time-series locality.



(a) Basic DQN (b) TensorTrade DQN

Figure 2: Observational Delay (δ is delayed timesteps)

5.2 Non-Targeted Perturbation Attacks

To investigate the effectiveness of adversarial example attacks on DRL policies, we implemented Fast Gradient Sign (FGSM) by Goodfellow et al. [2014] and Carlini and Wagner (C&W) Carlini and Wagner [2017] adversarial sample attacks using L_2 loss for both DQNs.

In Table 4, there are failure counts and other notable counts for the basic DQN and TT's DQN. In this experiment, we perturb a single, most recent tuple of values in the observation space for all FGSM and C&W attacks. Our definition of a failure for non-target attacks is the failure to change the action or action type. Post-constraints are applied where adversarial samples must fall within a realistic distribution of the true data. There were some modifications to the C&W implementation for TT due to non-normalized data. Representative samples of a perturbed tuple of values from successful attacks are presented in Table 3. See Table 3 for the Basic DQN return performance comparison when under attack vs. not under attack. Additionally, we have provided the total reward difference and net-worth difference between the TT target agent and TT target agent under attack in Figure 5 and Figure 7, respectively. Through these results, we establish that the test-time performance of the target policy in regards to its total reward is negatively impacted by our attacks. We have also shown that the agent's net-worth is also impacted, but not necessarily reflected by total reward.

5.3 Targeted Perturbation Attacks

Targeted attacks aim to manipulate a policy into taking an adversarial action a'_t instead of action a_t at a timestep t. We have evaluated against targeted FGSM and targeted C&W attacks using L_2 loss for both DQNs with minimal Q-value actions as our selected adversarial actions. However, it is noteworthy to remember that the function approximator can regress values which

NT FGSM				
timestep	original observation	perturbed observation	a	a'
894	0.0,-0.00354677, -0.00354677	0.0000, -0.0045, -0.0025	1	0
3973	0.0, -0.00048828,-0.00048828	0.0000, -0.0006, -0.0004	1	0
9599	0.00294118,-0.0004902,0.00294118	0.0027, -0.0002, 0.0027	0	1
16323	0.00435098,0.0, 0.00290065	0.0041, 0.0000, 0.0032	2	0
23283	0.00074322,-0.00371609,0.00074322	0.0001, -0.0044, 0.0001	0	1

NT C&W				
timestep	original observation	perturbed observation	a	a'
1602	0.00203314, 0.0, 0.00203314	0.0003, 0.0000, 0.0003	0	1
4735	0.00707071, 0.0,0.00707071	0.0002, 0.0000, 0.0002	0	1
5346	0.0032695 ,-0.00140121,0.0032695	0.0002, -0.0002, 0.0002	0	1
17424	0.0010985,-0.0010985 , 0.0010985	0.0002, -0.0002, 0.0002	2	0
29779	0.00039904,-0.00079808, 0.00039904	0.0003, -0.0003, 0.0003	0	1

Table 3: Successful Non-Target (NT) FGSM & C&W Attacks against Basic DQN

	NT FGSM				NT C&W			
DQN	P	Opt	Fail	N.C.N	P	Opt	Fail	N.C.N
TT	0.1	26	25	2	0.1	17	17	0
	0.5	123	117	7	0.5	114	110	3
	1.0	242	236	7	1.0	246	240	3
Basic	0.01	286	6	-	0.01	329	163	-
	0.1	3349	176	-	0.1	3016	1751	-
	0.5	15818	3329	-	0.5	15979	9358	-
	1.0	31779	10778	-	1.0	31779	18716	-

	T FGSM					T C&W				
	P	Opt	Fail	NT	PS	P	Opt	Fail	NT	PS
TT	0.1	248	248	146	230	0.1	26	26	5	26
	0.5	123	123	65	122	0.5	131	131	25	127
	1.0	28	28	9	27	1.0	249	249	70	243
Basic	0.01	337	6	4	-	0.01	327	294	89	-
	0.1	3148	191	98	-	0.1	3135	2915	903	-
	0.5	15905	4666	1581	-	0.5	15882	15291	4837	-
	1.0	31779	16000	5334	-	1.0	31779	30779	9953	-

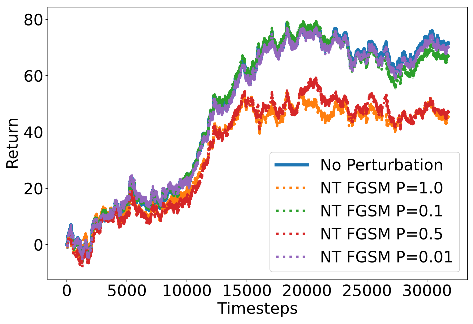
Table 4: Non-Target FGSM (NT FGSM), C&W Attacks (NT CW), Target FGSM (T FGSM), and Target CW (T CW) Opportunities (Opt), Fails, and Partial Success (PS) on TensorTrade's DQN and Basic DQN

are no longer representations of Q-values, implying min Q-value regressed adversarial actions may not be the best for the adversary.

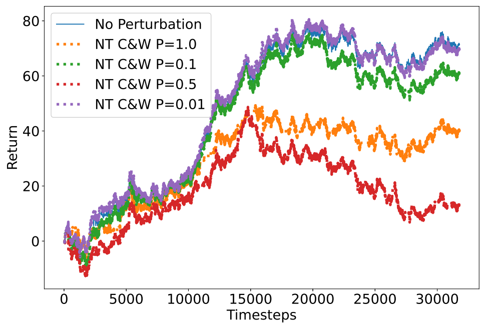
Again we leave the exact implementation details to the full paper. See Table 4 for failure counts, attempts, and partial successes (PS). We define partial success if the attack results in an action a^m where the action type of a^m is the adversarial action type for action a'_t. Our adversarial tuples are simple but should emphasize that adversarial attacks crafted under expensive parameters like low learning rate, high number of iterations, and high confidence can produce more human convincing adversarial samples. Performance under attack for Basic DQN can be found in Figure 4, TT's total reward difference can be found in Figure 6, and TT's net-worth difference in Figure 8. We thus establish the impact of targeted attacks on TT's DQN on test-time performance as well as its significant impact on the agent's net-worth.

6 Passive Test-Time Attacks and Policy Imitation

An adversary performs a passive test-time attack when they observe the target policy rollout trajectories through some active interception like a MiTM. An adversary may use other learning methods such like Imitation Learning (IL) to leverage these demonstrations training given an appropriate environment. We will make two strong assumptions to investigate an adversarial approach to IL which will require perfect adversarial information: (1) Access to an identical MDP that produced the target policy. (2) Ability to observe complete trajectories.

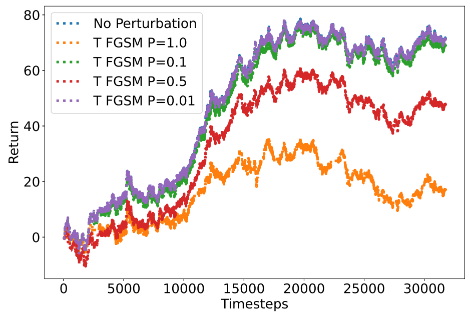


(a) Non-Targeted FGSM Attack

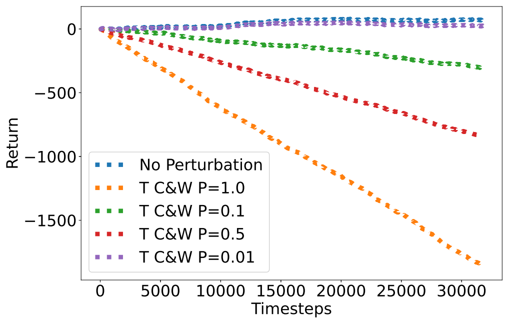


(b) Non-Targeted C&W Attack

Figure 3: Non-Targeted Attacks on the Basic DQN



(a) Targeted FGSM Attack



(b) Targeted C&W Attack

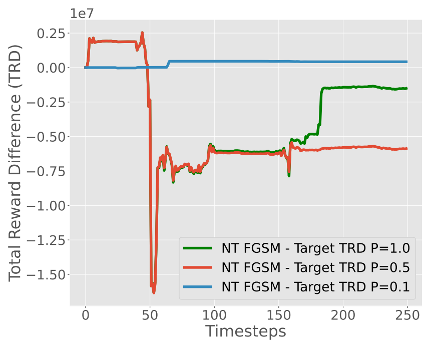
Figure 4: Targeted Attacks on Basic DQN

We use Deep Q-Learning from Demonstration (DQfD) as our IL method. There are two adversarial objectives for imitated agents: the first is policy imitation and the second is profitability relative to training cost. We define policy imitation for this paper as the training of a policy π' where its objective is to mimic a target policy's observable behavior. Policy imitation can result in additional adversarial knowledge, providing an adversary a way to perform more whitebox attacks based on attack transferability. Policy imitation can possibly lead to similar performances of the target agent. Depending on the adversary's objective, policy imitation can be feasible. The second objective refers to having an imitated policy converge sooner to an optimal policy which may imply a smaller adversary's budget than the target agent's training budget.

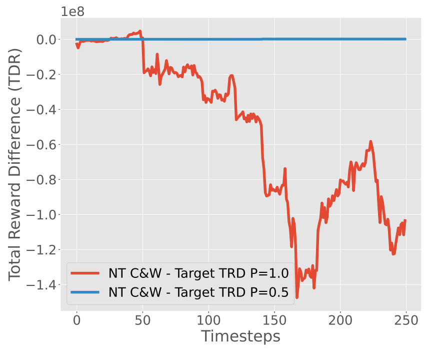
6.1 Imitation Learning & Deep Q-Learning from Demonstration

IL is a learning framework for imitating an expert policy through demonstrations. There are two objectives to IL: to imitate behavior exhibited in the demonstrations or to learn an underlying task from demonstrations. Agents that follow the first are often modeled as naive supervised learners known as behavioral clones (BC). The second objective is also referred to as Apprenticeship Learning. Learning frameworks like Inverse RL and RL are often used for this objective, but there are other methods that use supervised learning architectures.

DQfD Hester et al. [2017] has a few components: the pretraining phase, cost function, and prioritized demonstration sampling. We use the default parameters set by the authors. The pretraining phase is training prior to interaction with the environment. There is the prioritization of sampling expert demonstrations. The cost function is the sum of four loss functions: a Temporal Difference (TD) with 1 step double DQN loss $J_{DQ}(Q)$, n-step double DQN loss $J_n(Q)$,

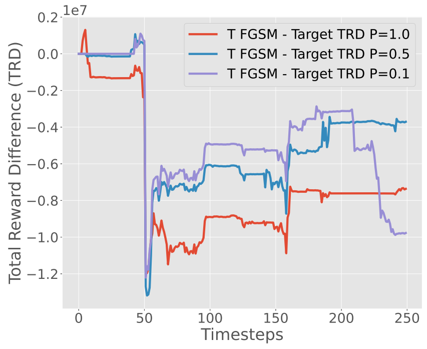


(a) Non-Targeted FGSM Attack Reward Difference

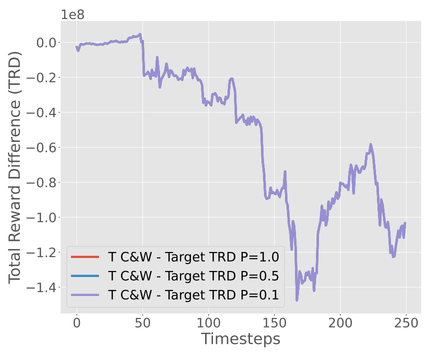


(b) Non-Targeted C&W Attack Reward Difference

Figure 5: Reward Differences between Control Total Reward and Non-Targeted Attacks on TensorTrade's DQN Total Reward

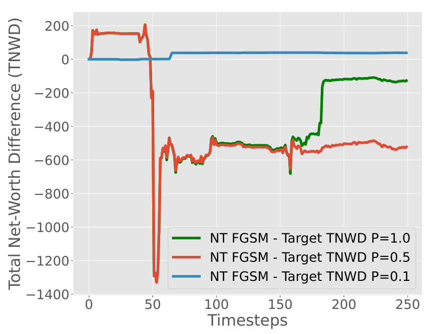


(a) Reward Difference Targeted FGSM Attack

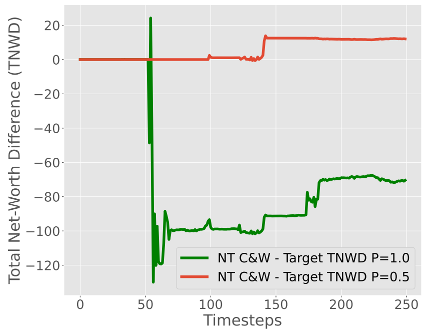


(b) Reward Difference Targeted C&W Attack

Figure 6: Reward Difference between Control Total Reward and Targeted Attacks on TensorTrade's DQN Total Reward

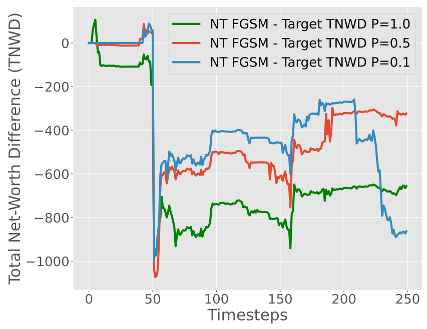


(a) Non-Targeted FGSM Attack Total Net-Worth Difference

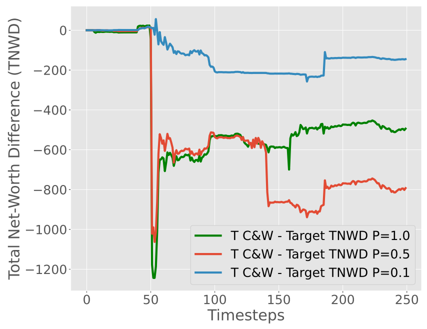


(b) Non-Targeted C&W Attack Total Net-Worth Difference

Figure 7: Net-Worth Differences between Control Total Net-Worth and Non-Targeted Attacks on TensorTrade's DQN Total Net-Worth



(a) Net-Worth Difference Targeted FGSM Attack



(b) Net-Worth Difference Targeted C&W Attack

Figure 8: Net-Worth Difference between Control Total Reward and Targeted Attacks on TensorTrade's DQN Net-Worth

a large margin supervised loss $J_E(Q)$, and L2 regularization $J_{L2}(Q)$. The proper equation is as followed where $\lambda_1, \lambda_2, \lambda_3$ are scalars Hester et al. [2017]:

$$J(Q) = J_{DQ}(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q) + \lambda_3 J_{L2}(Q)$$

6.2 Perfect Demonstrations & Imperfect Demonstrations

Perfect demonstrations refers to observing complete trajectories that inititates from a start state s_0 and terminates at a state s_T. DQfD's pretraining phase uses perfect demonstrations but we may also be interested in imperfect demonstrations.

Once again we leave exact implementation details to full paper however we have tested various amounts of demonstrations among various quantities of timesteps. We included a behaviorial clone as a form of alternative method to DQfD. We fixed the number of iterations in the pretraining phase and quantity of episodes based on a parameter search. For policy behavior evaluation during both the randomized evaluation and training evaluation, we include Table 5. As expected for the supervised learner (S.L.), it is capable of imitating the training trajectory, but not necessarily the testing trajectories. Additionally, the difference in tasks such as Q-value regression vs. maximum likelihood contribute to the expectation of policy imitation. We provide Figure 9(b) as net-worth comparison among agents.

	Training Evaluation			Randomized Evaluation		
Agent	Exact	A. Type	Length	Exact	A. Type	Length
250	19	207	250	207	1797	2490
300	27	162	300	146	1305	2490
350	26	61	350	233	251	2490
400	27	322	400	207	1799	2490
450	32	353	450	220	1772	2490
500	31	399	400	188	1819	2490
B.C.	345	349	350	341	1431	2490

Table 5: DQfD Agents Policy Action Match or Action Type (A. Type) with Target Agent

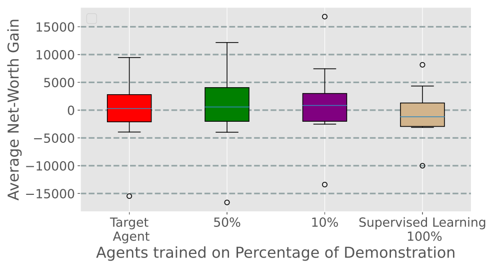
	Randomized Evaluation		
Agent	Succ. Attack	Succ. Transfer	No Attack Needed
250	7	1	10
300	166	44	253
350	15	5	17
400	2	1	10
450	10	2	11
500	88	24	75
250 (P)	422	166	427
300 (P)	195	70	245
350 (P)	169	57	216
400 (P)	938	500	702
450 (P)	230	103	291
500 (P)	228	98	260
B.C.	1545	979	544

Table 6: Successful (succ.) Transferred FGSM Non-Targeted Attacks (P is pretraining phase agents)

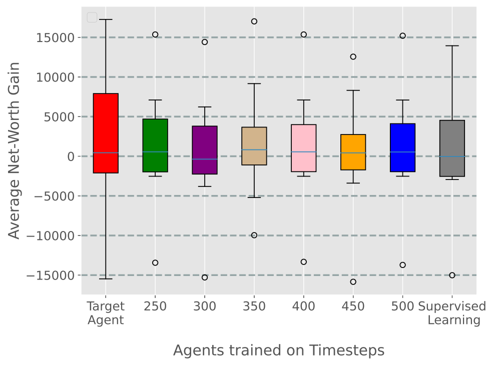
Imperfect demonstrations refer to a non-continuous set of demonstrations. One method is to choose another IL method without the perfect demonstration assumption; however, we choose to estimate the missing demonstrations. The full paper presents further work on this and its limitations; however, it is possible for competitive performance as seen with a boxplot of the average agent's net-worth in Figure 9(a).

6.3 Transferability to Target Agent

We present Table 6 which contains the count of successful transferred non-targeted FGSM attacks from our imitated agents to the target agent. We considered an attack successful at a timestep t if an imitated agent changes to any action other than optimal action a and a successful transfer if observation is shown to target agent also results in change of action. We note that all pretraining phase policies were more susceptible to our FGSM attacks and had higher successful transfer to the target agent. For the intentions of attacking the target agent, it is plausible that the pretraining phase agents and B.C. agent are sufficient.



(a) Average Net-Worth Gain over 10 Randomized Starts



(b) Average Net-Worth Gain over 10 Randomized Starts

Figure 9: Imperfect Demonstration Net-Worth Comparison (Left) & Perfect Demonstration Net-Worth Comparison (Right)

7 Conclusion

We investigated the vulnerability of DRL trading agents to adversarial attacks at inference time. We identified the attack surface and vectors of algorithmic trading policies in a novel threat model, and proposed 2 active test-time attack techniques namely: non-targeted DoS-based delay induction and targeted/non-targeted MITM-based adversarial perturbation. We demonstrate the susceptibility of a benchmark DRL trading agent and an agent based on a popular open-source framework for algorithmic trading called TensorTrade. Through perturbation of a single tuple from the history window, we show adversarial intervention can easily result in sub-optimal performance upon test-time. The results demonstrate that our target agents are sensitive to even weak attacks such as FGSM, as well as and more powerful attacks like C&W which provide human-fooling adversarial samples. Furthermore, portfolios tied to the agent may be impacted in ways that is not directly reflected in the performance metric at test-time, namely total reward. With TensorTrade's DQN, our attacks were shown to adversely affect the agent's net-worth. This finding may have significant repercussions on risk mitigation, as test-time performance through total reward may not alert human traders of the severity of impact upon external securities tied to the agent.

We looked to Imitation Learning methods for adversary usage through a passive, test-time attack. We have considered objectives for an adversary such as: policy imitation, self-gain or knowledge gain for whitebox attacks. We have shown that imitated agents can possibly perform competitively or better for equivalent or less computational expense than its target. There are limitations when using supervised learners but may be useful enough to an adversary for whitebox optimization attacking. With the use of our imitated agents, we were able to show the transfer-ability of an adversarial attack to our target agent.

The reported findings establish the need for further research on various aspects of security in DRL trading agents such as a need for metrics and measurement techniques for benchmarking the resilience and robustness of trading policies to adversarial attacks. Furthermore, our results call for further studies on mitigation and defense techniques against adversarial manipulation. These studies are likely to find current risk-aware DRL approaches of limited utility in this domain, as such techniques are typically addressing accidental (i.e., non-adversarial) noises in the dynamics of the environment. Lastly, considering the significance of R&D efforts in developing and acquiring proprietary algorithmic trading policies, there remains a critical need to study the impact of policy imitation attacks highlighted by Behzadan and Hsu [2019] targeting algorithmic trading.

References

Yandex n v: Yndx - stock price: Live quote: Historical chart. URL `https://tradingeconomics.com/yndx:rm`.

V. Behzadan. *Security of deep reinforcement learning.* PhD thesis, Kansas State University, 2019.

V. Behzadan and W. Hsu. Adversarial exploitation of policy imitation. *arXiv preprint arXiv:1906.01121*, 2019.

V. Behzadan and A. Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017a.

V. Behzadan and A. Munir. Whatever does not kill deep reinforcement learning, makes it stronger. *arXiv preprint arXiv:1712.09344*, 2017b.

V. Behzadan and A. Munir. The faults in our pi stars: Security issues and open challenges in deep reinforcement learning. *arXiv preprint arXiv:1810.10369*, 2018.

V. Behzadan and A. Munir. Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles. *IEEE Intelligent Transportation Systems Magazine*, 2019.

N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

Á. Cartea, S. Jaimungal, and J. Penalva. *Algorithmic and high-frequency trading.* Cambridge University Press, 2015.

G. Clark, M. Doran, and W. Glisson. A malicious attack on the machine learning policy of a robotic system. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 516–521. IEEE, 2018.

Z. Ding and H. Dong. Challenges of reinforcement learning. In *Deep Reinforcement Learning*, pages 249–272. Springer Singapore, 2020. doi: 10.1007/978-981-15-4095-0_7.*URL*