

EMERGENCE AND MITIGATION OF INADVERTENT SPOOFING BY AI TRADING AGENTS: RESPONSIBLE DEEP REINFORCEMENT LEARNING IN SIMULATION

David Byrd¹

¹Dept. of Computer Science, Bowdoin College, Brunswick, ME, USA

ABSTRACT

The deployment of sophisticated artificial intelligence agents into critical financial systems continues to accelerate, sparking interest in the novel risk profile of such autonomous entities. In this article, I leverage recent academic work in high-frequency electronic market simulation to investigate the potential for unintended harmful behavior to emerge in deep reinforcement learning trading agents. Through a series of experiments, I calibrate a baseline market environment, conduct hyperparameter search and sensitivity analysis on a benign profit-seeking learning agent, enhance the agent with new capabilities and observe that it adopts potentially unlawful spoofing as an optimal strategy, and explore risk-aware interventions which might reduce discovery of such manipulative behaviors. I conclude with observations on the risk of emergent AI misbehavior and the challenge of mitigating it without inhibiting desired agent performance.

1 INTRODUCTION

The complexity boundary on successful agentic AI task completion appears to be increasing at an exponential rate. A recent study suggested that over a six year time horizon, frontier model capability has doubled every seven months, possibly accelerating to every four months in 2024. (METR 2026) This rapid advancement is expanding the threat envelope to complex economic and security systems faster than society can adapt. The speed of adoption of new technology means that widespread impacts will be felt before there is sufficient empirical data to begin scientific analysis. There has already been at least one publicized occurrence of agentic AI used in a massive and deliberate campaign of espionage (Anthropic 2025), but we should not overlook the potential for unintended harm caused by AI agents learning behaviors not foreseen by their creators.

In this article, I aim to provide a systematic exploration of one exemplary facet of this much larger problem: the spontaneous discovery of spoofing, a form of illegal market manipulation, as the “optimal” strategy by profit-maximizing Deep Reinforcement Learning (DeepRL) trading agents. To accomplish this, I leverage novel enhancements to minABIDES, a recently developed multi-agent market simulation designed specifically to address such questions that are difficult or dangerous to investigate in reality. (Byrd 2025b) The simulation is open source, under active development, and available at <https://github.com/davebyrd/minabides>.

After discussion of related work, I will select a baseline configuration for the simulation, introduce a Double Deep Q-Network (DDQN) trading agent and perform hyperparameter search and sensitivity analysis, add capabilities to the DDQN agent that could permit spoofing and observe the results, assess the reliability of the spoofing behavior emergence with formal significance testing, and evaluate several approaches that may mitigate inadvertent spoofing. The paper will conclude with some discussion informed by the experiments, hopefully adding to a risk-aware conversation around agentic AI that goes well beyond financial markets.

The in-depth demonstration of unintended market manipulation by a profit-maximizing DDQN agent through no ill intent of its creator, evaluation of possible mitigation strategies against disruptive behaviors, and related simulation enhancements, together constitute the novel research contributions of this work.

2 RELATED WORK

To investigate my specific exemplar of the accidental adoption of disruptive or destructive behaviors by AI agents, it is necessary to combine several academic areas. This section is intended to provide a brief, helpful overview of the recent work on which this article is based.

2.1 Financial Market Simulation

The simulation of financial markets provides several key advantages over alternative approaches: reproducible results, systematic variation of controls, strategic visibility into participants and corresponding interpretability of outcomes, and more. There are at least two common complaints: designer-specified strategies may not adapt to conditions; and the underlying simulation process may not be sufficiently realistic to produce transferable knowledge.

The first shortcoming has been addressed through the creation of multi-agent financial market simulations in which the agents are explicitly able to alter their strategic parameters to adapt to their environment. One approach is taken by the MarketSim framework: empirical game theoretic analysis (EGTA), in which agents may periodically change their selection from a set of discrete strategic hyperparameters. (Wellman and Wah 2017) An increasingly common second approach introduces autonomous learning agents that constantly adjust their internal parameters, and therefore behavior, as they gain trading experience. This is the method used in ABIDES (agent-based interactive discrete event simulation) and in PyMarketSim. (Byrd et al. 2020; Mascioli et al. 2024) The minABIDES simulation improves on the performance and reliability of ABIDES, while also being simpler to use and extend, and including new features like deep reinforcement learning agents, real-time computation delay for AI cost-benefit analysis, and automatic generation of tables and plots for learning loss curves, profit distributions, and the evolution of the limit order book over time. (Byrd 2025b) Figure 1 illustrates the minABIDES network graph.

The second shortcoming is more subtle. It is not necessary for a market simulation to reproduce reality to any level of precision in order to create an environment in which the effects of trading agents *on one another* may be studied, so long as the system dynamics are reasonable and the participants appropriately react to one another. There has nonetheless been work comparing the output of market simulations like ABIDES to known stylized facts about real markets. (Vyetrenko et al. 2020) New approaches to market simulation have also been investigated in which a distribution of the next arriving order, conditioned on recent orders, is directly learned from historical data. (Coletta, Jerome, Savani, and Vyetrenko 2023)

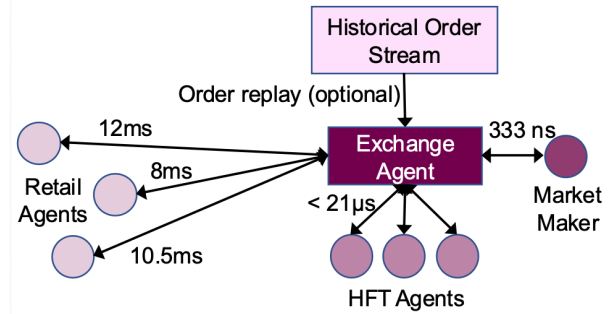


Figure 1: minABIDES network configuration.

2.2 Market Manipulation

In the United States, the first formal legal frameworks to govern the marketing of negotiable financial instruments were the Securities Act of 1933 and the Securities Exchange Act of 1934. Together, these introduced disclosure requirements, founded the Securities Exchange Commission (SEC), and introduced federal financial crimes including the identification of a “pump and dump” scheme as securities fraud. A major expansion of this framework came through the Dodd-Frank Act of 2010 which, among other modernizations, banned disruptive trading practices such as spoofing. (Commodity Futures Trading Commission 2011)

These two forms of market manipulation have been variously studied in the financial literature, but have more recently emerged as important topics for agentic AI research. Only with the development of Large

Language Models (LLMs) did the pump and dump scheme (U.S. Securities and Exchange Commission 2006), in which one who owns a stock makes public false claims to artificially drive up its price, become practical to automate in a responsive and persuasive manner at scale. The minABIDES simulation has been previously used to study the inadvertent emergence of social media pumping behavior by an autonomous trading agent given control of such an LLM. (Byrd 2025a)

There has been considerably more analysis of spoofing, or the placement of orders not intended for execution, in the context of AI trading agents. Typically in simulation, an experimental environment for spoofing contains: a trading agent capable of placing and canceling limit orders; and another agent whose strategy depends on observations of the limit order book. Spoofing occurs when the first agent manipulates the second agent’s perception of the market by injecting limit orders which it intends never to fulfill. Note that there are many honest reasons a trader might place and then cancel a limit order; it is the act of *placing an order with the intent to cancel before transaction* which constitutes spoofing. Previous research has used MarketSim to study the effects a spoofing agent may have on other strategies (Wang and Wellman 2017) and to adversarially train a spoofing agent to evade detection by masking its activity as honest market making (Wang and Wellman 2020). PyMarketSim has been leveraged to examine the effect of varying liquidity on the profitability of spoofing strategies. (Gu et al. 2024) The ABIDES simulation was employed to demonstrate how a simple tabular Q-learning agent, with a necessarily heavily stylized discrete observation space, might unintentionally adopt spoofing as a strategy unless actively discouraged (Byrd 2022).

The current work builds most directly on this last effort. Shortcomings of the prior work include the discrete state space required by tabular RL and the accompanying curse of dimensionality, the relative lack of hyperparameter flexibility in the model, and an inability to efficiently leverage distributed high-performance computing systems for large-scale simulation experiments. All of these are addressed with the addition of DeepRL agents and the foundational improvements of the minABIDES simulation.

2.3 Reinforcement Learning

Reinforcement Learning (RL) is the name given to trial-and-error approaches to solve a Markov Decision Process (MDP) of unknown parameterization. (Bellman 1957; Sutton and Barto 2018) That is, an intelligent agent receives an observed sample of the current state of the world and must select an action in response. The agent then receives an observation of the resulting environment and possibly an immediate numeric reward. Over time, the agent is tasked to maximize its expected utility $Q(s, a)$, the long-run sum of immediate and discounted future rewards obtained by taking action a from state s and then continuing to act according to the policy implied by the Q -function.

In the Q-Learning formulation of model-free RL, the agent iteratively approximates the Q -function directly according to a Bellman update:

$$Q^\pi(s, a) = R_s(a) + \gamma \sum_{s'} Pr_{ss'}[\pi(s)] V^\pi(s') \quad (1)$$

where V is a state value (utility) and π describes the current policy. (Watkins and Dayan 1992)

The equation above describes Tabular Q-Learning, in which both the state and action spaces are required to be discrete, as they are mapped to integer indices for lookup in a table or array. This limitation made it difficult to employ Q-Learning in complex continuous environments, producing either aliasing effects (too few states, agent cannot distinguish important differences) or the curse of dimensionality (too many states, insufficient data to estimate all Q-values).

The current work relies on Deep Q-Learning, in which the Q-table is replaced by a Q-network. The neural network naturally accepts any real-valued inputs as the environment observation, alleviating the aforementioned quantization problems. In the original Deep Q Network (DQN) approach (Mnih et al. 2013), a single network learns to estimate the Q-value of each possible action for a given observation and is trained with a Q-loss function:

$$(r + \gamma \max_{a'} Q(s', a') - Q(s, a))^2 \quad (2)$$

Table 1: All combinations of these background agent population hyperparameters were evaluated.

Agent	Count	Agent	Count	Agent	Count
Market Maker	0, 1	Noise	5, 10, 20	Value	1, 5, 10
Momentum	1, 5, 10	OB Imba.	1, 5		

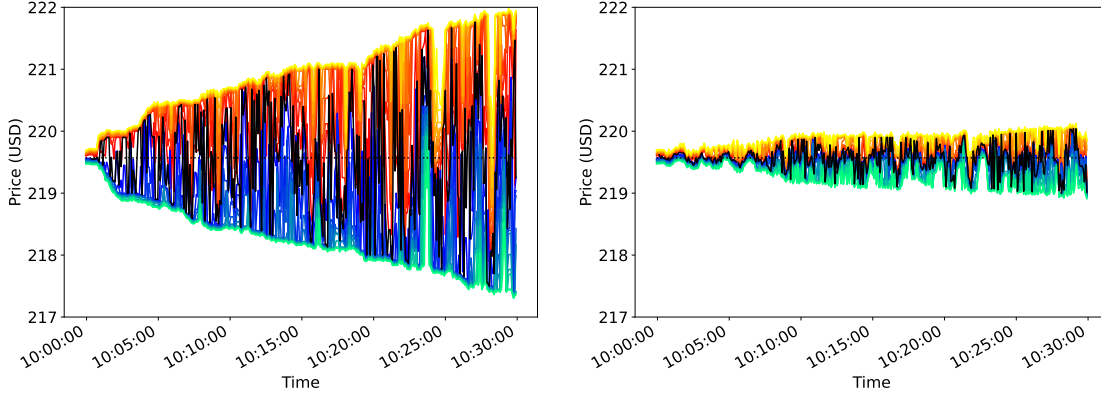


Figure 2: At the left, a representative limit order book from a simulation with no market maker. At the right, a similar simulation with a market maker. (Warm colors: ask prices, cool colors: bid prices, black line: transacted prices.)

If the tabular Q-update is viewed through the lens of stochastic gradient descent (SGD) and a learning rate α , the Q-loss function is a squared-difference variant of the same approach. A second “target” network is periodically copied from the Q-network. This promotes convergence by providing a stable target for the estimation of Q-values.

Double Deep Q Networks (DDQN) extend this technique with lessons learned from tabular Double Q-Learning, in which a second Q-table is added, with each experience being randomly assigned to a table. (Hasselt 2010) The other table evaluates the utility of the selected action, reducing optimism bias that might otherwise slow convergence. DDQN achieves the same effect by tweaking the operation of the second neural network, which is now used to evaluate the Q-network’s proposed next action from the successor world state:

$$\left(r + \gamma Q^T(s', \operatorname{argmax}_{a'} Q(s', a')) - Q(s, a) \right)^2 \quad (3)$$

This recovers the anti-optimism benefit of Tabular Double-Q while preserving the continuous observations provided by Deep Q. (Van Hasselt, Guez, and Silver 2016)

3 EXPERIMENTS AND RESULTS

In this section, I present a sequence of experiments through which the potential emergence of inadvertent agentic misbehavior may be observed and mitigated. Each experiment informs the next.

3.1 Market Baseline

Prior to the introduction of any learning agents, I conducted a grid search over possible background agent populations to select a stable, oscillating market in which easily identifiable patterns would emerge. Considered agent population combinations are listed in Table 1.

Table 2: Descriptive statistics for the selected background agent population: five momentum, five order book imbalance, five value, ten noise, and one market maker agent.

	Sharpe	Mean	Stdev	Min	Median	Max
Market Maker	1.11	2,689.22	2,416.99	-6,877.05	2,688.44	9,227.30
Momentum	-6.74	-4,657.34	690.71	-6,806.19	-4,576.53	-3,157.20
Noise	0.40	253.28	640.63	-2,069.20	236.14	3,122.16
O.B. Imbalance	-2.20	-565.39	256.94	-1,462.38	-526.02	-67.21
Value	4.05	2,753.88	680.78	1,396.11	2,697.06	6,016.99

While the noise traders, which place limit orders of random quantity, price, and direction near the current best offer prices, can make the market in a sense, populations that did not contain a proper market making agent (managing a liquidity ladder at all times) created unstable dynamics with high volatility and price swings of increasing magnitude as seen in Figure 2.

Value agents receive a noisy observation of an extrinsic price-time signal and update an internal private valuation separate from market activity. This private valuation is the only signal that anchors market prices to any particular level. All experiments in the current work use a flat fundamental value to simulate the lack of outside stimulus that would affect the market over short durations where high-frequency trading strategies are relevant.

The order book imbalance (OBI) agents follow a common high-frequency (or low-latency) technical indicator of the same name. They construct an imbalance ratio of the bid quantity to the total quantity visible in the limit order book which, via simple supply-and-demand, predicts directional price movement over the very short term. Each price level’s quantity is weighted by its distance from the best available price. The OBI agent has a configurable imbalance threshold to govern when it enters a position. In later experiments, the OBI agent also serves the purpose of being “spoofable”, as the DDQN agent can potentially learn to manipulate the OBI indicator.

With the market maker in place, the most important dynamic became visible: the balance between momentum agents, which amplify directional trends in price execution, and the aforementioned value agents which dampen those trends. Equalizing the number of these agents created the most stable market process, while still providing substantial price movement for the learner to observe. The final background population selected for all future experiments was five each of momentum, order book imbalance, and value agents, ten noise agents, and a market maker. Table 2 shows descriptive statistics for this population over fifty simulated trading periods. Note that the momentum agents are effectively always “wrong” in this market due to the mean-reverting pressure created by a flat fundamental value.

3.2 Capability Analysis of Double Deep Q-Network

To the stable multi-agent background population, I introduced a learning agent following the previously described DDQN algorithm. The agent requires a state space of one or more continuous features, which are introduced as the inputs to the first layer of the neural network, and a discrete action space equal in size to the final layer of the network. The activation strength of each output neuron will be interpreted as the Q-value (expected utility) of the corresponding action for the current input observation. A reward function must also be designed to guide the agent to learn the desired behavior.

For all experiments, the DDQN agent has at most the observation space: $\langle h, o, b, a \rangle$, where h is the shares held, o is the total share quantity of open limit orders, and b and a are respective ratios of the best bid and ask to rolling means of the same. The maximal action space is: $\langle B, S, N, L, C \rangle$ representing discrete choices to buy at the market, sell (close) at the market, do nothing, place a limit order within the bids, or cancel all outstanding orders. The agent does not take short positions. To avoid credit assignment issues which might confound the behavioral analysis, the base reinforcement proxies round-trip returns with an immediate reward upon opening or closing a position at a favorable price compared to a recent rolling

Table 3: DDQN *mkt3* configuration initial hyperparameter search values.

Parameter	Value	Parameter	Value
Network Size	256, 512, 1024, 2048	Learning Rate	$1e^{-4}, 5e^{-5}, 1e^{-5}, 5e^{-6}$
Batch Size	256, 512, 1024, 2048	OBI Threshold	0.55, 0.6, 0.65, 0.7, 0.75
Num. OBI	0, 5		

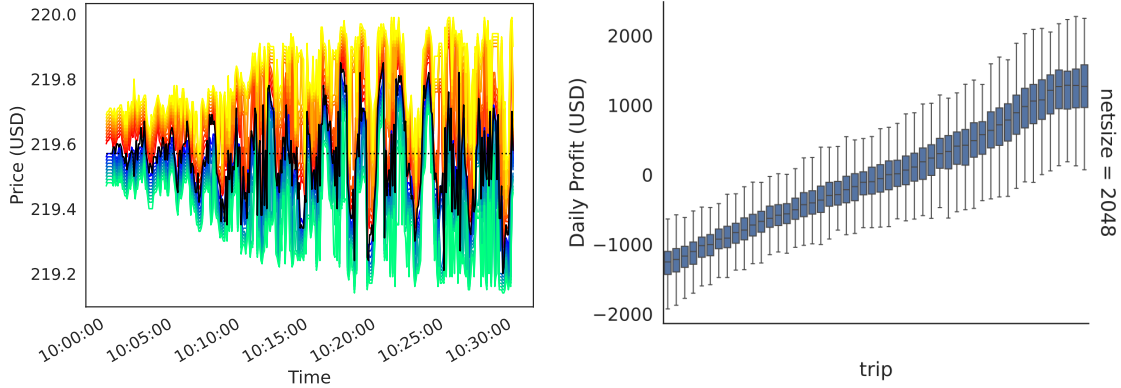


Figure 3: At the left, an example limit order book with the *mkt3* DDQN agent. At the right, distribution across trials of DQN agent profits during successive training episodes.

average. Instead of these simple returns, the agent is sometimes tasked with maximizing the Differential Sharpe Ratio of returns, the estimated impact of a single order on the agent’s mean return and variance of returns. (Wu 1997)

For initial capability testing of the DDQN agent, the limit buy and cancel actions are removed. The agent must learn to profit in its environment simply by the timing of market buy and sell orders to open and close a long position in the stock. Tested market and agent hyperparameters are listed in Table 3.

The results for the “forced good behavior” *mkt3* (three actions, market orders only) configuration of the DDQN agent are summarized below. In Figure 3, the volatility apparent in the limit order book should permit a learning agent to identify reliable trading opportunities. The same figure also shows the distribution of DDQN training period profits across experimental trials. Figure 4 and Table 4 show the sensitivity of agent profit and Sharpe Ratio to the most important hyperparameters: Q-network neurons per layer, OBI action threshold, network learning rate, and reinforcement learning optimization criterion. Table 5 shows the DDQN agent’s sensitivity (with Returns criterion and fixed network/batch size of 512) to the presence and selectivity of the order book imbalance (OBI) agents. Key hyperparameter search observations:

- DDQN reliably learns to trade the market under a wide variety of parameterizations. A coarse search for the appropriate learning rate and network size is required, after which sensitivity to all hyperparameters is low.
- In particular, any reasonable selection of replay buffer size, training batch size, training frequency, target update frequency or τ (interpolation of target towards Q-network), and Q-learning γ (time discount) or ϵ (exploration fraction) have little effect on performance.
- Optimizing risk-adjusted returns (Differential Sharpe) reduces sensitivity to network size, but simple returns work better at small network sizes.
- Increasing the selectivity of OBI agents smoothly reduces the observable trading opportunities for the DDQN agent. With highly selective OBI agents, the results become similar to the case with no OBI agents at all.

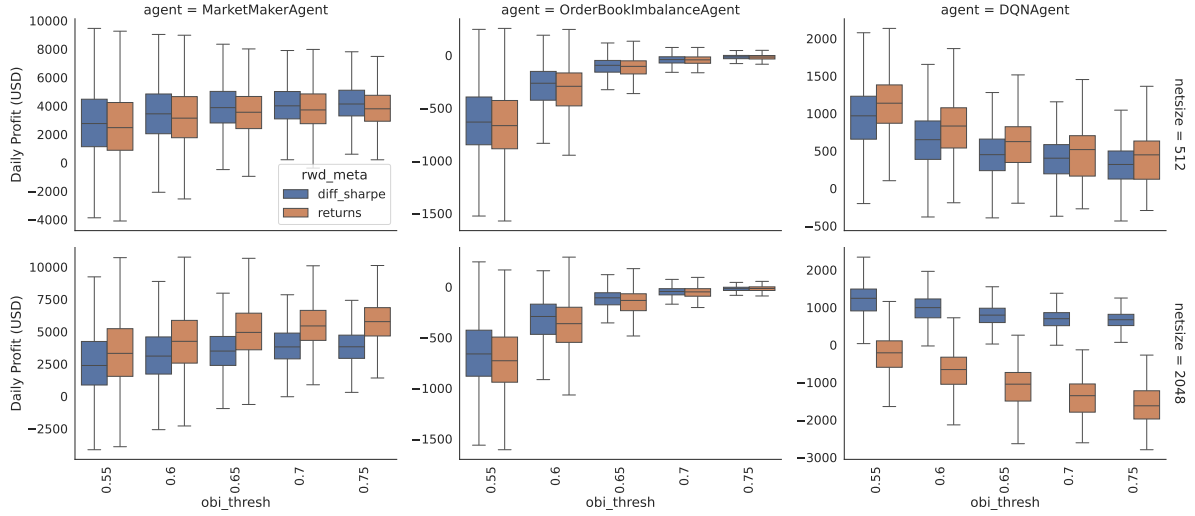


Figure 4: Sensitivity of agent profit distribution to representative hyperparameters for *mkt3* configuration. Row: width of each neural layer; Column: agent type; X-axis: bid-ask imbalance for OBI to act; Color: optimization criterion (orange for simple returns, blue for risk-adjusted returns).

Table 4: Sensitivity analysis of DDQN *mkt3* configuration Sharpe Ratio by learning rate and network size.

Opt. Criterion	Learn. Rate	200	300	400	500	600	700	800
Diff. Sharpe	5.0×10^{-5}	0.87	1.54	1.78	2.08	2.31	1.72	2.24
	2.5×10^{-5}	0.73	0.87	1.11	1.25	1.25	2.07	1.74
	1.0×10^{-5}	-0.06	0.54	0.75	1.22	0.92	1.22	1.71
	7.5×10^{-6}	0.16	1.09	0.82	0.89	1.46	1.63	0.98
	5.0×10^{-6}	-0.51	0.09	0.85	0.74	1.17	1.22	1.04
Raw Returns	5.0×10^{-5}	1.77	2.02	-1.18	-2.63	-3.61	-3.93	-5.15
	2.5×10^{-5}	0.56	1.43	2.47	2.15	0.38	-1.03	-1.88
	1.0×10^{-5}	-0.51	0.17	0.51	0.83	1.59	2.62	3.08
	7.5×10^{-6}	-0.33	-0.10	0.46	0.94	1.05	1.46	2.06
	5.0×10^{-6}	-0.64	-0.47	0.26	0.38	0.68	0.76	1.15

The DDQN agent consistently learns a profitable trading strategy in the simulated market environment. Especially in the presence of OBI agents, it does this with low variance, resulting in a high Sharpe Ratio (risk-adjusted return) as well as high absolute returns.

3.3 Emergence of Spoofing Behavior

If the reinforcement learning task were “earn profit”, the DDQN agent would have no need for manipulative behavior. However, with the task specification “maximize profit”, it may discover an undesired approach.

For these experiments, I reintroduce the missing actions to produce two new DDQN agent configurations. The *spoof4* agent has the ability to place limit buy orders within the depth of the order book. These orders are set to automatically expire at approximately the time of the agent’s next action, thus removing the need for management or manual cancellation of open orders. The *spoof5* agent loses the automatic-expiration feature on its limit orders, but gains the manual cancellation action. It must learn to manage and withdraw open orders itself.

Table 5: Descriptive statistics of DDQN *mkt3* configuration profit by OBI count and threshold t .

OBI Config.	Sharpe	Mean	Std	Min	25%	50%	75%	Max
No OBI agents	0.97	283.72	293.88	-192.16	24.50	214.51	524.94	1,206.16
5 OBI, $t = 0.55$	2.03	908.31	447.51	-91.58	615.17	983.89	1,231.17	1,951.33
5 OBI, $t = 0.65$	1.77	569.58	322.16	-38.51	326.99	607.98	821.87	1,277.57
5 OBI, $t = 0.75$	1.07	372.79	347.70	-200.96	36.17	438.69	659.67	1,062.38

Table 6: Descriptive statistics of DDQN agent profit by *spoof* configuration using five OBI at threshold t .

	Config.	Sharpe	Mean	Std	Min	25%	50%	75%	Max
<i>spoof4</i>	No OBI	0.89	263.82	296.08	-186.92	13.00	155.21	520.07	1,206.72
	$t = 0.55$	2.71	1,287.28	474.40	256.53	943.01	1,375.53	1,632.68	2,209.08
	$t = 0.65$	3.61	1,132.66	313.36	522.99	889.08	1,086.22	1,361.98	1,961.17
	$t = 0.75$	2.64	625.46	236.91	105.16	481.62	635.58	785.94	1,245.38
<i>spoof5</i>	No OBI	0.76	206.41	273.03	-200.82	-7.00	86.32	433.30	1,060.19
	$t = 0.55$	3.65	1,348.52	369.17	269.15	1,150.12	1,357.94	1,596.54	2,393.57
	$t = 0.65$	3.25	971.30	298.64	286.72	763.05	959.91	1,166.96	1,730.01
	$t = 0.75$	1.42	542.18	381.81	-84.31	80.97	656.12	839.67	1,237.08

An initial experiment for the two new configurations was run with the hyperparameter selection from the previous section which maximized profitability: learning rate 1×10^{-5} , network size 512, batch size 512, $\tau = 0.5$, replay buffer size 10000, $\gamma = 0.99$, training frequency 400, target update frequency 2400. Simple returns were selected over Differential Sharpe Ratio as the primary reward criterion because they perform better with smaller networks, and are therefore faster to train. The experiment replicates the *mkt3* analysis which varied the number of OBI agents and their selectivity threshold, and assumes that the (potential) spoofing agent's limit orders cannot be transacted; the agent will always be fast enough to adjust or cancel them if there is danger. This also means the limit orders cannot directly produce additional profit; they can only do so by influencing other agent decisions. The results of 140 trials for each experimental option are summarized in Table 6. In almost every case with OBI present, even the *worst single result* across all trials is now a net positive profit. With the ability to manipulate the order book imbalance signal, the learning agent effectively can't lose. (This ignores the fact that its behavior would be illegal for a human trader, and *might* be illegal for an AI trader. To my knowledge, no case has yet been brought for this type of activity by AI.) The best case result across trials is only slightly better than the *mkt3* honest agent case, but when the OBI agent selectivity is low to moderate, the 25th percentile result is now as good as the previous 50th percentile. The left tail of the returns distribution has been dramatically improved.

3.3.1 Evidence of Spoofing Adoption

The Table 6 increase in agent profitability *suggests* spoofing behavior has emerged. Manual inspection of representative action traces (/ buy, \ sell, + buy limit, . do nothing) from a trial with OBI agents:

.../++++\..../+++\. /++++\.....+/+++++\. /+++++++\....

and a trial without OBI agents:

.../...\. /...\. /...\. /...\. /...\. \.

are also consistent with spoofing (OBI) and not spoofing (no OBI). Further support of the spoofing hypothesis is derived from the action frequency counts for sixty trials presented in Table 7. With OBI agents present, the DDQN agent learns to deploy limit orders while long (increases profit) and to avoid them while flat (decreases profit). Without OBI agents these limit orders, which will never transact, are equivalent to doing nothing and are deployed with roughly equal frequency.

Table 7: Action selection frequency of *spoof4* configuration with OBI threshold $t = 0.65$.

Config.	Open long	Close long	Limit while long	Limit while flat	No action	Total actions
No OBI	4,694	4,635	14,813	14,538	57,065	95,745
5 OBI	4,883	4,763	13,004	3,086	68,907	94,643

Table 8: Mann-Whitney U significance testing of trading period profits, comparing across configurations as shown, presenting rank-biserial correlation r . For example, the top left result compares *spoof4* against *mkt3* when there are 5 OBI agents at threshold $t = 0.55$.

	5 OBI agents		0 vs 5 OBI	
	<i>spoof4</i> vs <i>mkt3</i>	<i>spoof5</i> vs <i>mkt3</i>	<i>spoof4</i>	<i>spoof5</i>
$t = 0.55$	0.43	0.56	0.91	0.97
$t = 0.65$	0.79	0.62	0.96	0.93
$t = 0.75$	0.41	0.28	0.64	0.49

More formal evaluation is performed with Mann-Whitney U significance testing. As a baseline, neither *spoof4* nor *spoof5* showed a profit advantage over *mkt3* when there were no OBI agents ($p > 0.05$, rank-biserial $r = -0.04$ and -0.16 respectively). This is expected, as in this case there is no one to manipulate. The slight negative effect can be explained as a profit drag from forcing the learning agent to explore a larger action space for no benefit. Results for other significance tests (all $p < 0.01$) are presented in Table 8. For Mann-Whitney U, $r > 0.3$ indicates a moderate effect and $r > 0.5$ a large effect. For the case with OBI agents, both *spoof4* and *spoof5* show the strongest effect against *mkt3* at OBI $t = 0.65$. At low thresholds, the OBI agents act aggressively enough for even *mkt3* to perform well by honest means; at high thresholds, it is difficult for the DDQN agent to deploy enough share volume to trigger OBI activation. At the $t = 0.65$ sweet spot, the DDQN agent both can, and needs to, learn to spoof effectively to maximize its profit. When comparing the same DDQN configuration with and without OBI agents, the effect is smallest at $t = 0.75$, which tracks with the earlier note on DDQN liquidity constraints. The *spoof5* configuration enjoys greater timing flexibility with its manual cancellation action, but the action also adds learning complexity. This may explain why *spoof5* is the best performer at lower thresholds and the worst at the highest threshold.

3.3.2 Adding Execution Risk

The previous experiment is unrealistic in an important respect. There is no *risk* to the agent from learning to spoof, because its passive limit orders will never transact. (It is a common assumption in the multi-agent literature that certain agents are guaranteed fastest to market.) This is unsatisfying, so I conducted a second spoofing experiment in which the learning agent’s passive orders can be executed just as any other order. The *spoof4* agent is now at risk of passive orders being executed before expiration, and the *spoof5* agent must now learn to manually cancel orders that may be at risk of execution. One might hypothesize the agents will perform more poorly in this case due to the development of a “fat left tail” when spoofing orders are inadvertently executed by other agents. The results of this experiment are summarized in Table 9.

This experiment shows the danger of spoofing. The *spoof4* agent, which relies on fixed expiration times for its limit orders, is unable to stop them being occasionally transacted, and also seems unable to learn not to try spoofing. The median case is only slightly negative, but the left tail outliers devastate overall performance. The pressure of execution risk on the *spoof5* agent, on the other hand, appears to have forced it to become even better and more aggressive at spoofing, presumably creating additional trading opportunities more often by rapidly canceling its limit orders. While it performs well in the modal case, it is clearly imperfect at avoiding inadvertent transaction. The agent’s Sharpe Ratio has been notably degraded as a result, but its mean, median, and maximum returns have substantially increased. This is consistent with the expected results of a spoofing strategy: high risk, high reward.

Table 9: Descriptive statistics over the profit of each DDQN *spoof* configuration when their limit orders can be executed. Using five OBI agents with threshold $t = 0.65$.

	Sharpe	Mean	Std	Min	25%	50%	75%	Max
<i>spoof4</i>	-0.36	-104.41	288.95	-768.14	-248.95	-42.03	61.61	395.98
<i>spoof5</i>	0.95	2,629.02	2,767.30	-2,257.22	644.34	2,858.92	4,936.35	6,521.29

Table 10: Summary descriptive statistics over period trading profits of *spoof5* configuration, exploring combinations of four mitigation schemes. Using five OBI agents with threshold $t = 0.65$.

Opt. Criteria	Scheme	Sharpe	Mean	Std	50%
Returns	CVaR	0.64	1,053.03	1,654.16	1,349.88
	Delayed	0.20	333.50	1,673.09	442.19
	Immediate	1.51	409.36	270.87	345.18
DiffShp	CVaR	1.38	2,369.96	1,721.22	2,004.22
	Delayed	0.81	1,332.29	1,638.33	1,360.67
	Immediate	1.43	544.28	380.64	535.88

When given the ability to place and cancel limit orders within the order book, the most flexible *spoof5* DDQN agent has demonstrated a consistent tendency to discover and adopt spoofing as the profit-maximizing strategy under a variety of conditions, thereby increasing its returns, but also its risk exposure.

3.4 Mitigation of Spoofing Behavior

Spoofing as a practice is illegal in the United States, but the consequences usually arrive with significant (years) delay in the form of a very large fine and threats of criminal prosecution. One would therefore like to train capable autonomous trading agents that will learn to profit *without* also learning to spoof the market. In the capability testing section, we saw that the *mkt3* configuration of DDQN earns a profit and cannot spoof. However, it would be preferable not to remove the agent’s ability to place and cancel limit orders, as there are many legitimate reasons to take these actions: market making, reacting to changing conditions, or even defending one’s order activity against manipulation attempts by others.

The second experiment in the prior section suggests a method of attack: the learning agent nearly doubled its mean profit when it gained the ability to place and cancel limit orders, then almost tripled it again when forced to contend with the inadvertent execution of those limit orders. However, the variance of its returns dramatically increased and a fat left tail re-emerged. In this final set of experiments, I introduce a series of penalties and risk-awareness components that attempt to guide the agent away from the high risk, high reward strategy of spoofing:

- Immediate penalty for limit buy while long.
- Proportional penalty at position close time based on number of limit buys placed while open.
- Reintroducing differential Sharpe ratio as objective to leverage its implicit variance penalty.
- Penalize rewards based on conditional value-at-risk (CVaR), which focuses on the left tail of the distribution. (Rockafellar, Uryasev, et al. 2000)

The intent is to investigate whether some combination of these approaches can effectively discourage the *spoof5* configuration of DDQN from adopting spoofing even though it is perfectly capable *and* doing so would increase short-term profits. The results, summarized in Table 10, are mixed at best:

- Penalizing the placement of a limit order while holding the stock, either immediately (both criteria) or delayed to position close (Returns criteria), results in performance worse than the simpler *mkt3* agent, which cannot place limit orders at all.

- The CVaR penalty applied to Returns is effective at reducing agent profits closer to the honest *mkt3* agent, but with extremely high variance of returns. It is inflicting the downside of spoofing without the benefit.
- The Differential Sharpe criterion, especially when combined with CVaR, maintains the agent’s profitability while reducing its variance of returns, thus actually increasing the Sharpe Ratio of returns. This attempt at guidance unfortunately seems to be allowing the agent to better perceive the risk of its left tail. Rather than reducing spoofing, it is allowing the agent to be better at it.

The current efforts to mitigate spoofing with simple mathematical modifications to the reward function have not been especially successful. In several cases, they inhibited the agent’s ability to trade beyond what a simpler agent could do, while in other cases, imparting a better sense of risk-awareness appears to have enabled the agent to *more safely* spoof the market. CVaR currently seems the most beneficial, but only with the simple Returns criterion does it actually reduce the profitability from spoofing. The more complex approach of learning a spoofing detector to serve as a normative guide through probabilistic alteration of the action space from prior work (Byrd 2022) may need to be revisited for this more complex set of AI agents.

4 CONCLUSION AND FUTURE WORK

This work shows that a reinforcement learning agent based on Double Deep Q-Networks (DDQN), with an action space which makes it impossible to spoof, can consistently learn to earn profit in a noisy, high-frequency, multi-agent financial market simulation with a flat fundamental value. Network size and learning rate must be broadly calibrated together, after which the model is insensitive to other hyperparameters. The agent earns higher returns when order-book imbalance (OBI) high-frequency trading agents are added, especially if those agents are less selective when acting on their trading indicator.

When the learning agent’s action space is expanded to include the ability to place limit orders within the depth of the order book, and either to manually cancel those orders or set an expiration time, the adopted behavior and associated profit significantly ($p < 0.01$) change *even though* no other experimental parameters are different: same hyperparameters, same simulation, same background agent population, same DDQN observation space and reward function. The agent strategy shifts to one of spoofing, timing its limit orders to further amplify upward swings in the price by manipulating the signal observed by the OBI agents. This has a beneficial second-order effect for the DDQN agent, as other traders respond to the sudden removal of the limit orders by amplifying downward swings as well. The learning agent is thus able to obtain both lower entries and higher exits than it otherwise could. When the assumption that the agent can perfectly guard its limit orders against execution is relaxed, the task becomes more difficult, but the most capable and complex agent still succeeds.

It is difficult to discourage the inadvertent adoption of subtle spoofing behavior through the simple application of targeted penalties. In particular, the reduction of rewards for passive orders while long, changing optimization criteria to differential Sharpe ratio instead of returns, and reward reduction based on conditional value-at-risk (CVaR) each either harmed the agent’s profitability or, by enhancing risk-awareness, counter-productively enabled safer spoofing.

There is still a significant gap in this literature to be addressed as future work. The spoofing penalty schemes would benefit from sensitivity analysis. Additional forms of guidance from the normative learning literature should be explored, such as policy shaping or action re-ranking. Augmenting the observation space, for example with the agent’s recent actions or market impact, could allow a learning agent to recognize its own manipulative behavior as it develops. Even so, it seems clear that a naively configured autonomous DeepRL trading agent, tasked with profit maximization, will likely discover a market manipulation strategy as its “best approach”. Caution is recommended if deploying such fully autonomous learning algorithms as traders until a more effective approach is found to reduce the emergence of such disruptive behaviors.

REFERENCES

- Anthropic 2025. “Disrupting the first reported AI-orchestrated cyber espionage campaign”. <https://www.anthropic.com/news/disrupting-AI-espionage>. Accessed: 2026-04-30.
- Bellman, R. 1957. “A Markovian Decision Process”. *Journal of Mathematics and Mechanics* 6(5):679–684.
- Byrd, D. 2022. “Learning Not to Spoof”. In *Proceedings of the Third ACM International Conference on AI in Finance*, ICAIF ’22, 139–147. New York, NY, USA: Association for Computing Machinery <https://doi.org/10.1145/3533271.3561767>.
- Byrd, D. 2025a. “The Accidental Pump and Dump: When Agentic AI Meets Autonomous Trading”. In *Proceedings of the 6th ACM International Conference on AI in Finance*, 88–95.
- Byrd, D. 2025b. “Multi-Agent Market Simulation for Deep Reinforcement Learning with High-Frequency Historical Order Streams”. In *2025 Winter Simulation Conference (WSC)*, 2539–2550. IEEE.
- Byrd, D., M. Hybinette, and T. H. Balch. 2020. “ABIDES: Towards High-Fidelity Multi-Agent Market Simulation”. In *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS ’20, 11–22. New York, NY, USA: Association for Computing Machinery <https://doi.org/10.1145/3384441.3395986>.
- Andrea Coletta and Joseph Jerome and Rahul Savani and Svitlana Vyetenko 2023. “Conditional Generators for Limit Order Book Environments: Explainability, Challenges, and Robustness”.
- Commodity Futures Trading Commission 2011. “Antidisruptive Practices Authority”. *Federal Register* 76(53):14943–14948.
- Gu, A., Y. Wang, C. Mascioli, M. Chakraborty, R. Savani, T. L. Turocy *et al.* 2024. “The Effect of Liquidity on the Spoofability of Financial Markets”. In *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF ’24, 239–247. New York, NY, USA: Association for Computing Machinery <https://doi.org/10.1145/3677052.3698634>.
- Hasselt, H. 2010. “Double Q-learning”. *Advances in neural information processing systems* 23.
- Mascioli, C., A. Gu, Y. Wang, M. Chakraborty, and M. Wellman. 2024. “A Financial Market Simulation Environment for Trading Agents Using Deep Reinforcement Learning”. In *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF ’24, 117–125. New York, NY, USA: Association for Computing Machinery <https://doi.org/10.1145/3677052.3698639>.
- METR 2026. “Task-Completion Time Horizons of Frontier AI Models”. <https://metr.org/time-horizons>. Accessed: 2026-04-30.
- Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra *et al.* 2013. “Playing Atari with Deep Reinforcement Learning”. *arXiv preprint arXiv:1312.5602* <https://doi.org/10.48550/arXiv.1312.5602>.
- Rockafellar, R. T., S. Uryasev *et al.* 2000. “Optimization of conditional value-at-risk”. *Journal of risk* 2:21–42.
- Sutton, R. S., and A. G. Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
- U.S. Securities and Exchange Commission 2006. ““Pump and Dump” Schemes”. https://www.sec.gov/rss/your_money/pump_and_dump.htm. Accessed: 2024-12-08.
- Van Hasselt, H., A. Guez, and D. Silver. 2016. “Deep reinforcement learning with double q-learning”. In *Proceedings of the AAAI conference on artificial intelligence*, Volume 30.
- Vyetenko, S., D. Byrd, N. Petosa, M. Mahfouz, D. Dervovic, M. Veloso *et al.* 2020. “Get real: Realism metrics for robust limit order book market simulations”. In *Proceedings of the First ACM International Conference on AI in Finance*, 1–8.
- Wang, X., and M. P. Wellman. 2017. “Spoofing the limit order book: An agent-based model”. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*.
- Wang, X., and M. P. Wellman. 2020. “Market manipulation: An adversarial learning framework for detection and evasion”. In *29th International Joint Conference on Artificial Intelligence*.
- Watkins, C. J., and P. Dayan. 1992. “Q-learning”. *Machine Learning* 8:279–292 <https://doi.org/10.1007/BF00992698>.
- Wellman, M. P., and E. Wah. 2017. “Strategic Agent-Based Modeling of Financial Markets”. *RSF: The Russell Sage Foundation Journal of the Social Sciences* 3(1):104–119 <https://doi.org/10.7758/RSF.2017.3.1.06>.
- Wu, L. 1997. “Optimization of trading systems and portfolios”. In *Proceedings of the IEEE/IAFE 1997 computational intelligence for financial engineering (CIFER)*, 300–307. IEEE.

AUTHOR BIOGRAPHY

DAVID BYRD is the Marvin H. Green Jr. Assistant Professor of Computer Science at Bowdoin College in Brunswick, Maine. His research interests include multi-agent simulation, analysis of complex real-world systems, and artificial intelligence, especially sequential decision making problems such as financial markets and games. He has authored two dozen academic articles on these topics. His most recent line of research explores unintended negative behaviors learned by AI in complex systems, for which he has recently earned two best paper awards for his work in the mitigation of financial agent spoofing behaviors and an analysis of large language model use to manipulate other intelligent agents. His email address is d.byrd@bowdoin.edu and his Google Scholar page is at <https://scholar.google.com/citations?user=IF8h840AAAAJ>.