A Proximal Policy Optimization Approach to Detect Spoofing in Algorithmic Trading

Iulia-Diana Groza
Faculty of Mathematics and Computer Science
Babes-Bolyai University
Cluj-Napoca, Romania
iulia.groza@stud.ubbcluj.ro

Lucia-Diana Miholca
Faculty of Mathematics and Computer Science
Babes-Bolyai University
Cluj-Napoca, Romania
diana.miholca@ubbcluj.ro

***Abstract*—Among the most pressing issues in algorithmic trading is the manipulative practice of spoofing, where traders deceive other market participants by placing and then quickly canceling large orders. These tactics can lead to artificial price movements and compromise the trust in market fairness. This paper explores the development of a Proximal Policy Optimization agent to detect spoofing in real-time, on historical Level 3 Limit Order Book data from the LUNA flash crash in May 2022. We develop an anomaly detection system to effectively label legitimate spoofing attempts. Ultimately, we simulate the market environment to serve as the playground for our Proximal Policy Optimization Policy Network. The core contributions of this research include the integration of Proximal Policy Optimization in market surveillance, the use of Level 3 Level Order Book data in a machine learning solution to harness temporal data, and the feature engineering of rolling statistics for price and size movements. Our experimental results demonstrate the feasibility of deep reinforcement learning in advancing market integrity.**

***Index Terms*—proximal policy optimization, algorithmic trading, spoofing, deep reinforcement learning, anomaly detection**

I. Introduction

With the rise of algorithmic trading during the last decades, financial markets have been drastically transformed: significant efficiencies appeared, but the major caveat that algorithmic trading introduced is its effect on market integrity [1]. Among the most pressing issues is the manipulative practice of spoofing, where traders deceive other market participants by placing and then quickly canceling large orders. These tactics can lead to artificial price movements and compromise the trust in market fairness [2].

This paper explores the application of Proximal Policy Optimization (PPO), a deep reinforcement learning technique, to detect spoofing in algorithmic trading. We aim to develop a real-time detection system, integrated in a web application.

Our main motivation for this study is given by the lack of existing tools for market manipulation detection on the web, that could help traders to gain a better understanding of the current market situation to make informed decisions promptly before participating in the market. To the best of our knowledge, there is no web tool that would allow online investors to analyze the status of market integrity in real time.

The core of this study involves the design and implementation of a PPO-based spoofing-detection model. We utilize historical Level 3 Limit Order Book (LOB) data from the LUNA flash crash of May 2022, offering a realistic and challenging testbed for our approach. The data and statistical reports on order cancellation during that event have been procured from the work of Li, Polukarov, and Ventre (2023) [3]. Since the LOB data is not labeled, we develop an anomaly detection system that allows us to effectively label legit spoofing attempts.

This model is integrated into a web application, *spoof.io*, which provides a practical tool for monitoring trading activities and identifying spoofing in LUNA. Working as a proof-of-concept tool, the application will simulate the last two hours of the LUNA flash crash from May 2022, which represent our test data.

Ultimately, this work aims to contribute to the field of market surveillance in several ways. To the best of our knowledge, there is no work in the financial markets field that employs PPO in a market manipulation detector. Additionally, this would be the first approach that utilizes Level 3 LOB data in a machine learning solution to harness temporal data, and the second in the literature of market surveillance, after [3] on statistical physics tools. Finally, our contribution brought during the anomaly detection phase is given by the feature engineering of rolling statistics on order price and size, such as mean, standard deviation, and variance, that provide additional information on market volatility during the computation of the anomaly score.

II. Related Work

Supervised learning models, particularly those employing deep learning architectures like Gated Recurrent Units (GRU) and Feedforward Neural Networks (FNN), have shown high accuracy in detecting known patterns of market manipulation. The GRU-based model developed by Tuccella, Nadler, and Serban (2021) achieved an accuracy of 0.75 [4], which is commendable for real-time detection. However, these models require extensive labeled datasets for training, which can be a significant limitation in markets where manipulation tactics evolve rapidly.

On the other hand, the FNN model discussed by Leangarun, Tangamchit, and Thajchayapong (2016) demonstrated the impact of data granularity by showing improved results with Level 2 data over Level 1 data, highlighting the importance

of detailed order book information (especially on cancellation orders and their depth) in enhancing accuracy of manipulation detection [5].

The Adaptive Hidden Markov Model with Anomaly States (AHMMAS) introduced by Cao, Li, Coleman, Belatreche, and McGinnity (2016) [6] represents a significant advancement in unsupervised detection methods. It outperformed traditional models by effectively recognizing anomalous trading patterns without prior labeling of L2 snapshots [6]. Although this model offers powerful detection capabilities and adapts to new data through its sliding window mechanism, its complexity and computational demands could pose challenges for real-time market surveillance.

The works of Lillo & Farmer (2005) [7], and Yura, Takayasu, Sornette, and Takayasu (2014) [8] provide foundational insights into how liquidity fluctuations and price movements can be analysed using statistical physics. Their methods are particularly adept at capturing complex interactions within the order book that traditional models might overlook. This demonstrates the complex interplay between market orders, limit orders, and cancellations, a remark that is essential in our study conducted on spoofing.

Li, Polukarov, and Ventre's approach [3] successfully applied these principles to detect spoofing & layering during the LUNA cryptocurrency flash crash. Their model, which considers orders as particles whose interactions can be quantified and analysed, showcased superior performance in identifying manipulative behaviors that were not detected by more traditional Z-score-based anomaly detection methods. Their work proved that the use of more granular data, of Level 3, can be both successful and computationally efficient at the same time.

When comparing all methods, it's evident that while supervised and unsupervised learning strategies provide accurate tools for detecting known and evolving manipulative patterns, statistical physics approaches bring a novel and profound analytical depth. An ideal solution would combine the simplicity and computational power of supervised learning in real-time detection, the absence of a requirement for labeled data in unsupervised learning strategies, and the accuracy given by state-of-the-art econophysics methods. We consider that a PPO-based approach will effectively harness most of these advantages.

III. Data

A. Collection and Preprocessing

Spoofing highly occurs in events marked by market instability, such as flash crashes. Therefore, when collecting our data, there are several important aspects that we took into consideration. Analyzing Level 3 (the most granular level) LOB data as part of a PPO-based detector is extremely important, as it marks our contribution, and temporal data is effectively leveraged. Additionally, procuring data from recent flash crashes is essential - there are higher chances of more subtle and diverse spoofing methods to occur in more recent data. Finally, since we employ unsupervised methods and do not work with labeled data, to ensure the correctness of our model evaluation, we must make use of historical data for which statistical reports on spoofing attempts were already performed, verified, and are publicly available.

All these considerations lead to our decision of working on the wLUNA/USD data that was utilised by [3]. It was initially fetched from Coinbase via a WebSocket feed, and it consists of Level 3 LOB historical data on wLUNA/USD from the LUNA flash crash from May 2022 (11/05/2022, 16:00-20:00).

Orders placed at a price that matches or is less favourable than an existing order (typically the best bid or ask) are executed immediately, resulting in a $match$ record. Orders that are not executed, or only partially executed, become limit orders in the LOB, creating an $open$ record. These limit orders will remain in the LOB until they are cancelled, generating a $canceled$ record. The minimum price precision is 0.01 USD, and the smallest trading volume unit (order size) is 0.001 LUNA. The timestamps are recorded with a precision of one microsecond, and are marked when an order activity (or event) occurs, making these event-driven timestamps discrete and non-consecutive [3].

The raw data consists of six JSON files: three for the full channel data, respectively three for the ticker data. The event is split across three temporal windows (16:00 - 17:00, 18:00 - 19:00, respectively 19:00 - 20:00), each of the three files corresponding to one temporal window.

TABLE I
SUMMARY OF RAW DATASET JSON FILES AND THEIR RECORD COUNTS

#	File Name	Number of Records
1	FullChannel_GDAX_20220511_17hr.json	768,272
2	FullChannel_GDAX_20220511_19hr.json	456,282
3	FullChannel_GDAX_20220511_20hr.json	550,124
	Total Full Channel Records	**1,774,678**
4	Ticker_GDAX_20220511_17hr.json	37,761
5	Ticker_GDAX_20220511_19hr.json	27,864
6	Ticker_GDAX_20220511_20hr.json	37,180
	Total Ticker Records	**102,805**

The first step in the preprocessing pipeline is to handle the temporal aspects of the data. We extract the hour of the day from each timestamp, as trading behaviours often exhibit diurnal patterns. By including the hour of the day as a feature, we allow the model to learn and leverage these temporal patterns, which can be indicative of spoofing activities that may occur at specific times. Formally, if t is the timestamp, the extracted hour h can be represented as:

$$h = hour(t) \tag{1}$$

Next, we address the issue of missing values, which is a common challenge in financial datasets. For numeric features, we employ median imputation to fill in missing values. The median is chosen for its ability to eliminate outliers, ensuring that the central tendency of the data is preserved. This step prevents the disruption of the model's learning process due to incomplete data.

Normalization is the next step in the preprocessing pipeline. By applying MinMaxScaler, we scale the numeric features to a range between 0 and 1. Consistent scaling facilitates faster convergence and improves the overall stability of the learning process. The normalization for a feature x is given by:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{2}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of the feature x, respectively.

Categorical features are processed through one-hot encoding, transforming them into binary vectors, therefore, each category is treated as a distinct entity without imposing any ordinal relationships. For instance, order types such as $received$, $open$, $done$, and $match$ are each represented as separate binary features.

$remaining\_size\_change$ captures the dynamics of order size changes over time. By calculating the difference in remaining sizes across consecutive records grouped by order ID, we can identify patterns indicative of spoofing, such as frequent and abrupt changes in order sizes. This feature provides the model with additional insights into the aggressive placement and cancellation of orders. Formally, for an order i with size s at time t, the change in remaining size Δs_i can be represented as:

$$\Delta s_i = s_i(t) - s_i(t-1) \tag{3}$$

For the full channel data, we maintain identifiers such as $order\_id$ and $time$ to preserve traceability. The ticker data complements the order book data by providing additional context on price movements and trading volumes.

B. Feature Engineering

We mark our contribution by designing and processing features that effectively harness temporal data and financial statistics. Our objective is to fully leverage the theoretical foundations and the spoofing preconditions outlined and applied in [5], while still introducing innovative and heuristic elements into our approach.

We begin by calculating rolling statistics such as mean (μ), standard deviation (σ), and variance (σ^2) for windows on multiple sizes (5, 10, and 15, respectively) on important columns like $price$ and $size$. These statistics provide us with a detailed view of the central tendency and dispersion of the data over different time periods. This allows us to identify abnormal fluctuations in price and order sizes, which are indicative of spoofing. A sudden increase in the variance of order sizes, for example, may suggest the presence of spoof orders intended to create a false impression of market depth.

Next, we compute the order flow imbalance (OFI), which measures the discrepancy between buy and sell orders over a rolling window. This is achieved by assigning a signed size to each order (positive for buy orders and negative for sell orders) and summing these sizes over the window. The OFI directly reflects market pressure, caused by the imbalance between buy and sell orders. Mathematically, the OFI can be expressed as:

$$OFI(t) = \sum_{i=t-w}^{t} size_i \cdot side_i \tag{4}$$

where w is the window size, $size_i$ is the size of order i, and $side_i$ is +1 for buy orders and -1 for sell orders. High positive or negative OFI values indicate significant market pressure, which can signal spoofing activities.

We also add the cancellation ratio to the dataset, which is the ratio of cancelled orders to received orders. This ratio is a significant indicator of spoofing, as spoofers often place a large number of orders and cancel them quickly to create a misleading sense of market activity. The cancellation ratio (CR) is calculated as:

$$CR = \frac{reason\_canceled}{type\_received\_adjusted} \tag{5}$$

where $type\_received\_adjusted$ ensures that the denominator is never zero by replacing zero values with one. A high cancellation ratio suggests a higher likelihood of spoofing.

The market spread is calculated as the difference between the best ask and best bid prices. The market spread ($spread$) provides insight into the market's liquidity and the aggressiveness of trading activities. It is expressed as:

$$spread = best\_ask - best\_bid \tag{6}$$

A large spread often indicates uncertainty or manipulation in the market, which is commonly associated with spoofing.

Additionally, we one-hot encode the $hour\_of\_day$ feature to capture diurnal patterns in trading activity. This encoding ensures each hour is represented as a separate binary feature, allowing the model to identify time-specific spoofing behaviors.

IV. Proximal Policy Optimization Model

A. Market Simulation Environment

The market simulation environment provides a controlled setting where the PPO model can interact with historical market data, learn patterns, and improve its detection capabilities.

The environment maintains a `current_index` to track the current position in the dataset, ensuring that the agent processes data sequentially. The reset mechanism reinitializes the environment to its starting state, setting the `current_index` to a valid position within the data bounds and updating the spoofing threshold. This ensures that each training episode starts afresh, providing the model with a consistent and repeatable starting point.

The step processes the agent's actions, advancing the market simulation by one step and returning the new state, reward, and other relevant metrics. The reward structure is designed to reinforce correct detections of spoofing and penalize incorrect actions. This is mathematically represented as:

$$reward = \begin{cases} 1 & \text{if } (action = 1 \text{ and } is\_spoofing) \\ & \text{or } (action = 0 \text{ and } \neg is\_spoofing) \\ -1 & \text{otherwise} \end{cases} \tag{7}$$

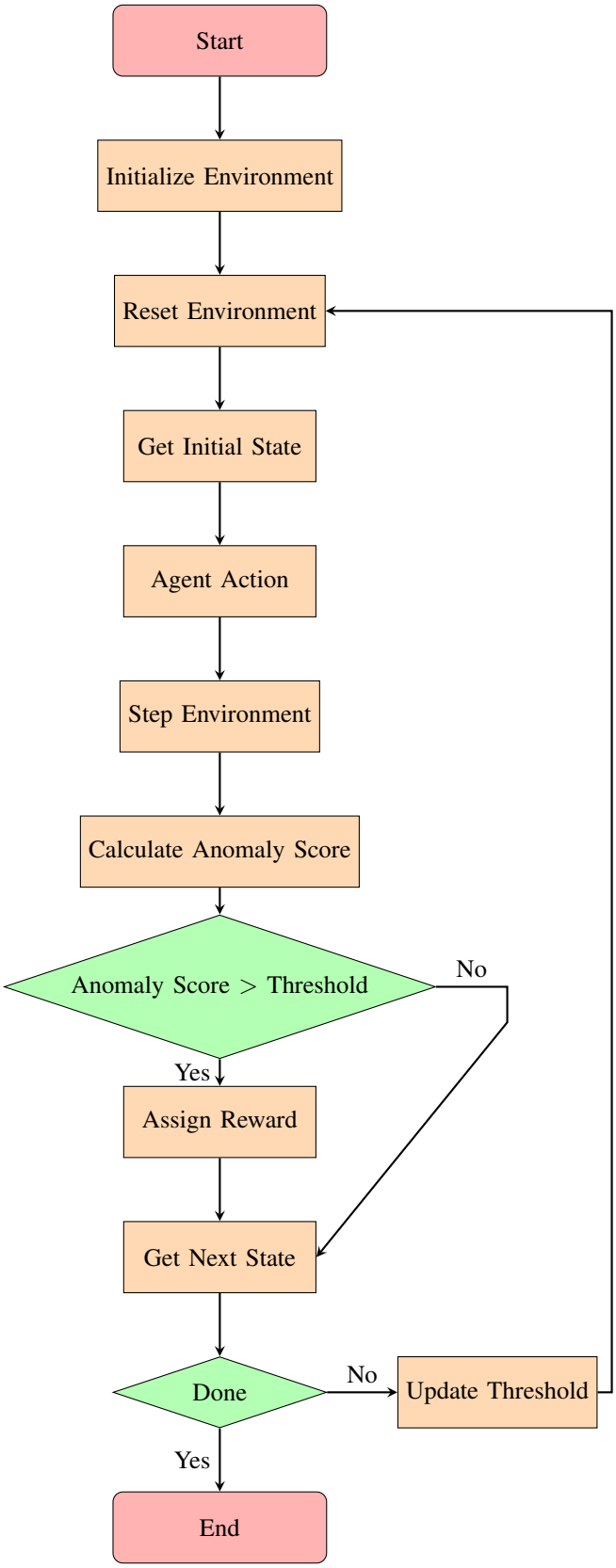

Fig. 1. Market Simulation Environment Pipeline

where $is\_spoofing$ is determined by comparing the anomaly score to the spoofing threshold.

The state getter generates the current state by concatenating historical features from both full channel and ticker datasets. This approach ensures that the state includes a comprehensive view of market conditions over a specified history window.

Finally, since our dataset is not labeled, we need a method to assess the correctness of the agent's decisions and correlate it with the reward system. For this, we perform anomaly detection on orders. The anomaly score of each order is calculated based on predefined feature weights and market data features at the given index. The score is computed as:

$$anomaly\_score = \sum_i w_i \cdot \log(1 + |feature_i|) \tag{8}$$

To adapt to changing market conditions, we implement an adaptive spoofing threshold, dynamically updated based on recent anomaly scores. This ensures that the model remains responsive to shifts in market behavior. The threshold is updated to the 75% of the recent anomaly scores, represented as:

$$spoofing\_threshold = \text{percentile}(\{anomaly\_score_i\}_{i=n-50}^{n}, 75) \tag{9}$$

where n is the current index in the dataset.

B. Policy Network

The PPO Policy Network is implemented using PyTorch and features a straightforward FNN architecture. The network consists of an input layer, two hidden layers with 256 neurons each, and an output layer. Each hidden layer employs ReLU activation functions to introduce non-linearity, allowing the network to capture complex patterns within the trading data. Additionally, the network uses a softmax activation function in the output layer to produce action probabilities.

This architecture is designed to process the input features derived from market data and produce logits for the action probabilities. The logits indicate the likelihood of each possible action, where the actions are binary: 0 (no spoofing) and 1 (spoofing).

The initialization of the network weights is performed using the Kaiming Normal method for the weights of the linear layers, and biases are initialized to zero. The network is optimized using the Adam optimizer, with the learning rate set to 1×10^{-3}.

To optimize the policy network, we compute discounted rewards. Discounted rewards take into account future rewards to emphasize immediate actions that could indicate spoofing. The formula used is:

$$R_t = r_t + \gamma R_{t+1} \tag{10}$$

where r_t is the reward at time step t and γ is the discount factor.

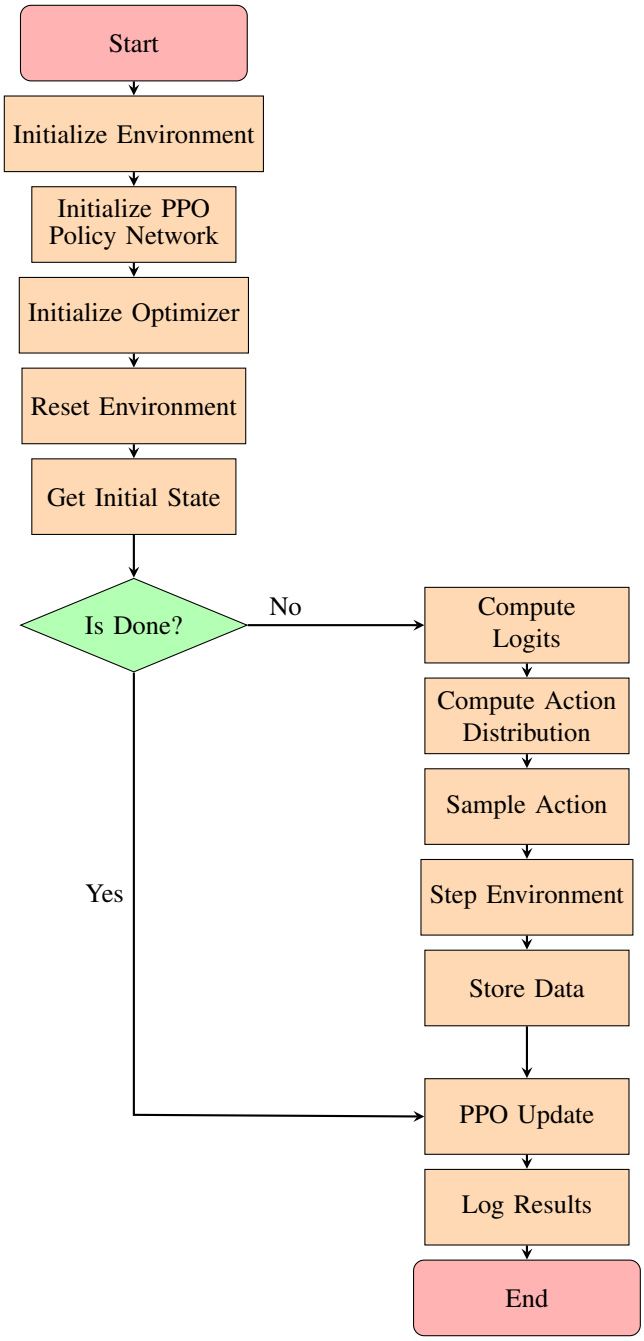

Fig. 2. PPO Policy Network Training Pipeline

The Generalized Advantage Estimation (GAE) is used to compute more stable advantage estimates. The advantage A_t at time step t is calculated using:

$$A_t = \delta_t + \gamma\lambda A_{t+1} \tag{11}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, and $V(s_t)$ represents the value function. This method smooths out the advantage estimates, making policy updates more stable and reliable; the detection of spoofing needs to be effective against the noisy and volatile nature of financial markets.

The PPO update mechanism is designed to adjust the network weights in a controlled manner. It calculates the loss function and performs gradient descent to update the policy. The loss function includes a clipped surrogate objective, a value function loss, and an entropy bonus. The clipped surrogate objective is defined as:

$$L^{\text{CLIP}} = \mathbb{E}\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \cdot A_t, \right.\right. \\ \left.\left. \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right) \cdot A_t\right)\right] \tag{12}$$

where $\pi_\theta(a_t|s_t)$ represents the new policy, and $\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the old policy.

The PPO update ensures that the policy improvement is conservative and stable when dealing with the highly dynamic and stochastic environment of financial markets. The inclusion of an entropy term in the loss function encourages exploration, preventing the policy from becoming too deterministic and potentially missing out on new strategies.

To integrate the PPO Policy Network with our market simulation environment, we initialize the environment and the network. The environment provides state data, while the network processes these states to predict actions and update its policy.

V. Experimental Results

A. Hyperparameter Tuning

The hyperparameter tuning process was divided into two phases. In the first phase, we compared the performance of the PPO model with and without rolling statistics in the anomaly score calculation. This comparison aimed to evaluate the impact of our feature engineering contribution. In the second phase, we focused on optimizing PPO parameters, such as the learning rate, batch size, epochs, and spoofing threshold.

Table II lists the feature weights used during anomaly detection.

TABLE II
Feature Weights for Anomaly Score Calculation

Feature	Weight
order_flow_imbalance	0.15
cancel_to_received_ratio	0.15
price_5_std	0.05
price_10_std	0.05
price_15_std	0.05
size_5_var	0.05
size_10_var	0.05
size_15_var	0.05
spread	0.10
last_size_5_var	0.05
last_size_10_var	0.05
hour_of_day	0.15
hour_15	0.05
hour_16	0.05
hour_17	0.05
hour_18	0.05
hour_19	0.05

By removing our features consisting of rolling statistics of order price and size, we heuristically adjust the weight configuration as depicted in Table III.

TABLE III
FEATURE WEIGHTS FOR ANOMALY SCORE CALCULATION

Feature	Weight
order_flow_imbalance	0.25
cancel_to_received_ratio	0.25
spread	0.10
hour_of_day	0.15
hour_15	0.25
hour_16	0.25
hour_17	0.25
hour_18	0.25
hour_19	0.25

These weights were selected based on domain knowledge and empirical experimentation. During hyperparameter tuning, we adjusted these weights to find the optimal combination that maximizes the model's performance.

The hyperparameters for the second phase of tuning are as follows:

1) **Learning Rate:** This controls how much to change the model in response to the estimated error each time the model weights are updated.

$$\alpha \in \{1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$$

2) **Batch Size:** Batch size influences the stability and speed of the training process. Smaller batch sizes typically provide a regularizing effect and lower generalization error, while larger batch sizes can speed up the training process.

$$\text{Batch Size} \in \{32, 64, 128\}$$

3) **Number of Epochs:** The number of epochs determines how many times the entire training dataset passes through the network. More epochs can lead to better learning, but may also cause overfitting if not controlled properly.

$$\text{Epochs} \in \{10, 20, 30\}$$

4) **Spoofing Threshold:** The spoofing threshold is unique to our application, defining the anomaly score threshold above which actions are classified as spoofing. We start with an initial value for our adaptive threshold.

$$\text{Spoofing Threshold} \in \{0.7, 0.8, 0.9\}$$

We empirically selected 10 of the most promising configurations, that could potentially balance the trade-off between all the learning parameters. The hyperparameter tuning process leverages parallelization to efficiently explore the vast search space of the ten established hyperparameter configurations. Specifically, we employ the `joblib` library, which allows for parallel execution of all the ten hyperparameter combinations across multiple CPU cores.

B. Performance Analysis

The final results of our experiments are depicted in Figure 3, including the anomaly scores over time, cumulative rewards over time, training loss over epochs, and reward distribution.

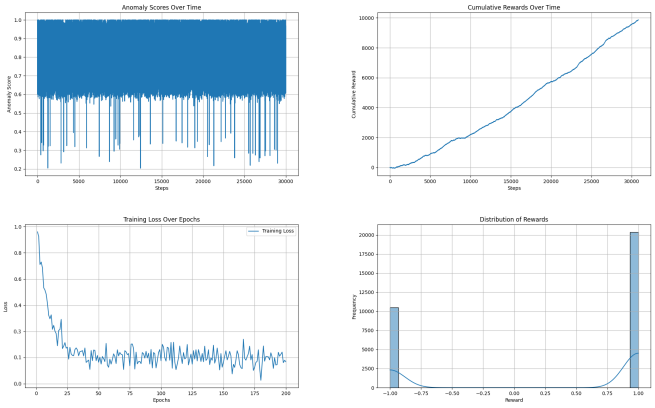

Fig. 3. Training results: Anomaly Scores Over Time (top left), Cumulative Rewards Over Time (top right), Training Loss Over Epochs (bottom left), Reward Distribution (bottom right).

Before reaching these results, we performed hyperparameter tuning as presented in the previous section. The comparative analysis of the performance metrics between the model with rolling stats and without rolling stats is shown in Figure 4. This highlights that the inclusion of rolling stats significantly improves the anomaly detection phase and, by extension, the performance of the PPO model, as they help the model to capture more subtle patterns in the data, leading to more accurate detection of spoofing attempts.

For the second phase of the hyperparameter tuning, focusing on the PPO parameters, we selected 10 configurations based on their potential performance (methodology presented in previous section). Table IV provides a summary of these configurations along with their performance metrics, ranked by total reward.

TABLE IV
PERFORMANCE METRICS FOR SELECTED PPO CONFIGURATIONS

Total Reward	Avg Reward	Std Reward	Learning Rate	Batch Size	Epochs	Spoofing Threshold
9500	0.317	0.120	1×10^{-3}	128	30	0.8
9200	0.307	0.115	5×10^{-4}	128	30	0.8
9000	0.300	0.110	1×10^{-3}	64	20	0.8
8900	0.297	0.105	5×10^{-4}	32	20	0.8
8500	0.283	0.102	1×10^{-3}	128	30	0.7
8300	0.277	0.099	5×10^{-4}	32	20	0.9
8000	0.267	0.094	1×10^{-3}	64	20	0.7
7800	0.260	0.091	5×10^{-4}	64	20	0.7
7500	0.250	0.088	1×10^{-4}	32	10	0.8
7300	0.243	0.085	1×10^{-4}	64	10	0.9

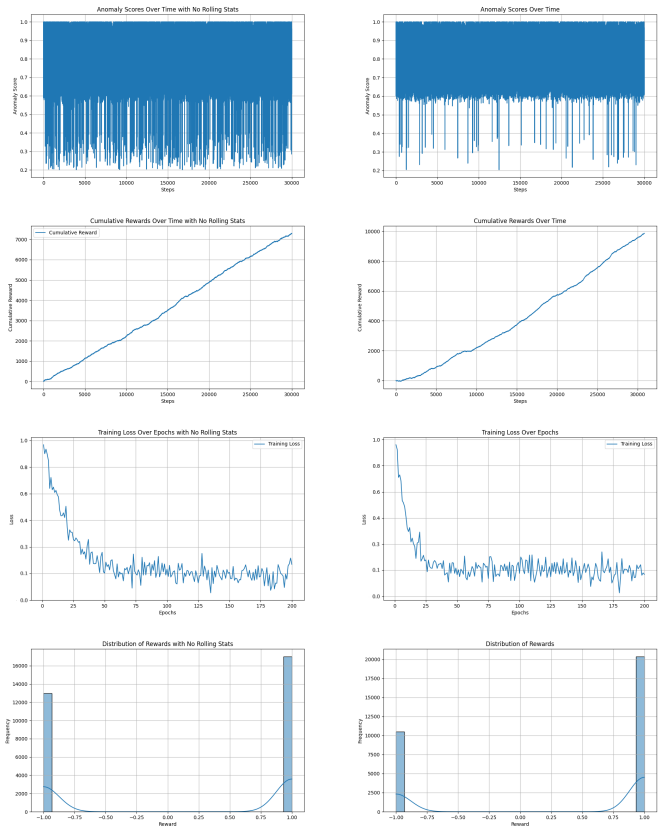

Fig. 4. Hypertuning results: Comparison of model performance without (left) and with rolling stats (right).

The table is ranked by total reward, which indicates the overall performance of each configuration. The average reward and the standard deviation of rewards provide additional insights into the model's stability and effectiveness.

The results from above prove the feasibility of employing PPO in advancing market integrity, as the clear convergence of training loss over epochs (stabilising around 0.13) indicates effective learning. Reward optimisation managed to achieve consistent results, while minimising volatility. However, it is clear that leveraging an unsupervised learning approach still represents a great challenge in accurately detecting legit spoof attempts. By analysing the anomaly over time graph, we notice that only 89% of orders were canceled, therefore potentially marked as spoofed orders.

However, in our dataset, roughly 98.6% open orders were eventually canceled. According to [3], *"a total of 248,796 order submission and cancellation records were identified, with 233,134 occurring within the active area, 8,302 within the passive area, and 7,360 outside"*. This leads to 97% of the order submissions and cancellations to occur within the active area, with only 3% within the passive area [3].

VI. Conclusion

Through our work, we managed to reach all of our goals in terms of our contribution to the field of deep reinforcement learning in market surveillance. *spoof.io*, the first available web tool for spoofing detection, has demonstrated the feasibility of employing Proximal Policy Optimization (PPO) in the detection of market manipulation, specifically spoofing, in algorithmic trading. Additionally, our contribution on feature engineering of rolling statistics of order price and size was proven to highly improve the performance of our model. And lastly, we successfully harnessed Level 3 LOB information in processing time series as part of our model development.

After coming up with a good configuration as a result of carefully hypertuning both the anomaly detection features, and the PPO parameters, our model reached a great performance. With a clear convergence of training loss (stabilising around 0.13), and accumulating a consistent total reward of 9,500, with minimal volatility, PPO is proven to have huge potential in enhancing market integrity.

A notable caveat of our work is the reliance on an anomaly detection model. One of our targets was developing a solution that can harness unlabeled data, without the obvious disadvantages of a supervised learning method. However, the amount of low-score orders, even after integrating an adaptive spoofing threshold, and performing hypertuning, denotes that there is still room for improvement. This conclusion comes from a comparison to the information on order cancellation in our dataset given by [3]. This implies that a supervised approach could potentially yield more accurate and safer results. This is particularly critical given the safety and legal implications associated with detecting market manipulation. Ensuring the accuracy and reliability of detection systems is the key to prevent false positives and the legal repercussions they might entail.

When compared to existing literature, our results show promising advancements. Previous studies have primarily focused on supervised learning approaches or unsupervised anomaly detection methods, such as those leveraging statistical physics and traditional machine learning techniques. Our PPO-based model, however, offers a deep reinforcement learning perspective, capable of dynamically adapting to market conditions in real-time and potentially uncovering more subtle forms of spoofing that other methods might miss.

In summary, our research proves the potential of PPO in spoofing detection, offering a deep reinforcement learning approach that brings innovation and complements existing methodologies. By addressing the highlighted limitations, we can continue to enhance the reliability of market integrity tools, ultimately contributing to a fairer and more transparent trading environment.

Acknowledgment

We would like to express our deepest gratitude to Haochen Li, Maria Polukarova, and Carmine Ventre for providing the dataset utilized in this research. Thanks to their exceptional results and insightful statistics as part of [3], our solution made use of the exact same L3 real market data on the LUNA/USD pair, gathered during the LUNA flash crash from May 2022 (11/05/2022, 16:00-20:00).

REFERENCES

[1] T. Hendershott and R. Riordan, "Algorithmic Trading and the Market for Liquidity," *Journal of Financial and Quantitative Analysis*, vol. 48, no. 4, pp. 1001–1024, 2013, doi: 10.1017/S002210901

[2] "Dodd-Frank Wall Street Reform and Consumer Protection Act," Pub. L. No. 111-203, 124 Stat. 1376, 2010. Available at U.S. Government Publishing Office: [https://www.govinfo.gov/content/pkg/PLAW-111publ203/pdf/PLAW-111publ203.pdf].

[3] H. Li, M. Polukarova, and C. Ventre, "Detecting Financial Market Manipulation with Statistical Physics Tools," arXiv preprint arXiv:2308.08683, 2023.

[4] J.-N. Tuccella, P. Nadler, and O. Serban, "Protecting Retail Investors from Order Book Spoofing using a GRU-based Detection Model," arXiv preprint arXiv:2110.03687, 2021.

[5] T. Leangarun, P. Tangamchit, and S. Thajchayapong, "Stock price manipulation detection using a computational neural network model," in *2016 Eighth International Conference on Advanced Computational Intelligence (ICACI)*, Chiang Mai, Thailand, 2016, pp. 337-341, doi: 10.1109/ICACI.2016.7449848.

[6] Y. Cao, Y. Li, S. Coleman, A. Belatreche, and T. M. McGinnity, "Adaptive Hidden Markov Model With Anomaly States for Price Manipulation Detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 2, pp. 318-330, Feb. 2015, doi: 10.1109/TNNLS.2014.2315042.

[7] F. Lillo and J. D. Farmer, "The key role of liquidity fluctuations in determining large price changes," *Fluctuation and Noise Letters*, vol. 5, no. 2, pp. L209-L216, 2005, doi: 10.1142/S0219477505002574.

[8] Y. Yura, H. Takayasu, D. Sornette, and M. Takayasu, "Financial Brownian Particle in the Layered Order-Book Fluid and Fluctuation-Dissipation Relations," *Phys. Rev. Lett.*, vol. 112, no. 9, p. 098703, Mar. 2014, doi: 10.1103/PhysRevLett.112.098703.