Report on PPO Agent for Trading in Gym Trading Environment

Celestine Akpanoko

1.0 Introduction

The integration of Reinforcement Learning (RL) in financial trading offers a transformative approach to developing adaptive and intelligent trading strategies. This report focuses on the application of the Proximal Policy Optimization (PPO) algorithm within the Gym Trading Environment, a specialized platform for simulating stock market scenarios and training RL agents. The report covers the algorithm's description, the environment's characteristics, the experimental design for hyperparameter selection, and a discussion of the results.

1.1 The Gym Trading Environment

The gym trading environment under consideration is structured around a pandas data frame, which is a required parameter (df). The DataFrame must have a column named 'close' and utilize a DatetimeIndex as its index. To facilitate the rendering process, it is necessary to provide supplementary columns such as 'open', 'low', and 'high'. The environment triggers exploration from the outset of this data frame, signifying the initial condition of each episode. The action space in this environment is determined by a user-defined set of places, each labeled with values ranging from negative infinity to positive infinity. These positions represent the proportion of the portfolio's value that is allocated to a certain position. Positive values indicate a bullish stance, while negative numbers indicate a bearish stance.

The observation space consists of a numpy array that contains both static and dynamic information. The static features are obtained from the rows of the DataFrame columns that include characteristics at a certain timestep. The default dynamic characteristics encompass the agent's most recent position and the current actual position. Episodes in this setting end in two scenarios: firstly, when the DataFrame reaches its end, indicated by the value 'truncated' being True; and secondly, when the portfolio valuation drops to zero or below, typically due to taking margin positions greater than 1 or less than 0, resulting in the value 'done' being True. The reward system is regulated by the formula: $ln(p(t-1)/p(t))$. Where p represents the portfolio valuation at timestep t. This method calculates a logarithmic measure of the proportional change in portfolio value, providing a detailed and responsive feedback system for the agent's activities in the market.

2.0 PPO Algorithm Description

Proximal Policy Optimization (PPO) is a notable progress in the domain of reinforcement learning, specifically in its utilization for intricate contexts like financial trading. PPO is an on-policy algorithm that learns directly from the actions taken under the present policy, without relying on a separate dataset. This technique is based on the concepts established by its

predecessor, Trust Region Policy Optimization (TRPO), which included a trust region constraint to ensure policy stability. PPO enhances this notion by including a clipping mechanism to restrict policy modifications, thereby striking a harmonious equilibrium between significant policy updates and the preservation of stability. The equilibrium is vital in dynamic settings where significant policy changes might result in unstable or worse than ideal performance.

The operating mechanism of PPO is defined by its iterative process of improving policies, led by a trial-and-error approach. Starting with an initial policy, PPO interacts with the environment, carrying out activities based on the existing policy. The results of these acts, along with the corresponding environmental reactions, serve as the foundation for policy revisions targeted at optimizing anticipated benefits. The core of PPO is rooted in its inventive method of altering policies. PPO incorporates a clipping mechanism to limit the scope of policy change, thereby guaranteeing that the new policy does not depart substantially from its predecessor. This limitation is crucial for maintaining the stability of the learning process, which is essential in contexts characterized by significant levels of uncertainty and fluctuation.

The dual-structured actor-critic architecture of PPO has been effectively employed in the trading arena. The actor, who is in charge of choosing actions, and the critic, who is responsible for evaluating actions, collaborate closely to negotiate the complexities of financial markets. This synergy enables sophisticated decision-making that takes into account both the immediate effectiveness of acts and their long-term strategic importance. Through the implementation of PPO in this specific field, we have effectively utilized its ability to facilitate consistent and adaptable policy evolution, hence facilitating the creation of advanced trading methods. The strategies, continuously improved through PPO's iterative process, showcase the algorithm's capacity to adjust to and exploit the intricate patterns inherent in financial trading settings.

3.0 Design of Experiments

3.1 Hyperparameter Search

Grid search is a systematic method for tuning hyperparameters, known for its due diligence in thoroughly searching the parameter space. This method entails systematically assessing all potential combinations of hyperparameters inside a predetermined grid, with the objective of determining the best set that optimizes the performance of the model. Crucial elements of grid search while fine-tuning a Proximal Policy Optimization (PPO) method encompass:

The **learning rate (alpha)** is a parameter that determines the step sizes used throughout the optimization process. It can take on values such as 0.0001, 0.00025, and 0.0005.

The **discount factor (gamma)** evaluates the significance of future rewards in the algorithm, with possible values of 0.98 or 0.99.

Policy Clip (policy_clip): Analyzes the limits for policy changes in PPO, offering choices such as 0.1, 0.2, 0.3.

The parameter **"n_epochs"** specifies the number of full iterations over the training dataset, with options such as 3, 4, or 5.

GAE Lambda (gae_lambda) aims to optimize the trade-off between bias and variance in advantage estimates by experimenting with several values, such as 0.92, 0.95, and 0.98.

The **batch size (batch_size)** parameter determines the number of samples that are analyzed before updating the model. It can take on values such as 32, 64, or 128.

The exhaustive nature of grid search guarantees a thorough examination of the hyperparameter space, hence increasing the probability of discovering the ideal combination. Nevertheless, this approach might be demanding in terms of computing resources, as each combination necessitates k-fold cross-validation, resulting in substantial time and resource usage. Nevertheless, grid search continues to be a powerful method, particularly when the main goal is to determine the most efficient hyperparameters without any limitations on computing resources.

3.2 Experiments

The experimental setup involved configuring the trading environment to replicate various market positions and their corresponding costs. The agent has the option to assume one of three potential positions at each decision point: -1 (engaging in short selling), 0 (staying out of the market), and +1 (taking a long position). These positions indicate the agent's market position, with short selling indicating speculation on a decline in asset values, staying out indicating a neutral view, and going long indicating anticipation of an increase in asset prices. The environment also included authentic market components, such as a trading charge of 0.01% for each instrument (i.e bitcoin) transaction and a borrowing interest rate of 0.0003% each timestep, which was applied to margin trading.

The Proximal Policy Optimization (PPO) method was utilized as the learning mechanism for the trading agent. The hyperparameters for the PPO algorithm were using grid search. The learning rate (alpha) was fixed at 0.0003 to maintain a stable learning rate and prevent any notable fluctuations or convergence problems. The discount factor (gamma) was set at 0.99, enabling the agent to accurately assess the importance of future rewards. The policy clip parameter, which is essential for ensuring the stability of the learning process, was configured at a value of 0.2. The training update consisted of 4 epochs, and the GAE lambda, a parameter used to decrease the variability of policy gradient estimations, was set to 0.95. The batch size for the learning updates was selected as 64, striking a balance between computing efficiency and learning stability.

An essential element of the experimental design was the incorporation of a halting criterion that relied on the performance of the portfolio's return. The criterion was established with a threshold

of a 95% portfolio return, assessed over the previous 50 episodes. The objective of this strategy was to strike a compromise between the requirement for enough training to achieve the best policy performance and the potential drawbacks of overfitting or excessive computing costs. The training procedure was automatically stopped when the average portfolio return for the previous 20 episodes surpassed this threshold.

The training procedure lasted for a total of 7621.37 seconds, which is approximately equivalent to 2 hours and 7 minutes. This time represents the computing need of the training process, which includes the repetitive nature of policy updates, the intricacy of the trading environment, and the extent of the state and action spaces. The experiment was designed to evaluate the effectiveness of the PPO algorithm in a simulated trading environment. It focused on creating realistic trading situations and selecting hyperparameters carefully to provide a strong and successful framework.

4.0 Discussion of Results

4.1 Training Results

The training data used in this study consisted of Binance Bitcoin data starting on January 1st, 2020. This dataset offered an all-encompassing perspective on the dynamics of the Bitcoin industry throughout a substantial timeframe. Figure 1 displays a graph that shows the normalized values of rewards, losses, and portfolio returns. This graph represents a significant statistic that is integrated into the trading environment. An important finding from this image is the consistent and upward link between rewards and portfolio performance. The trend indicates that as the model's decision-making abilities increased over time, it became capable of generating greater profits from its chosen trading moves.

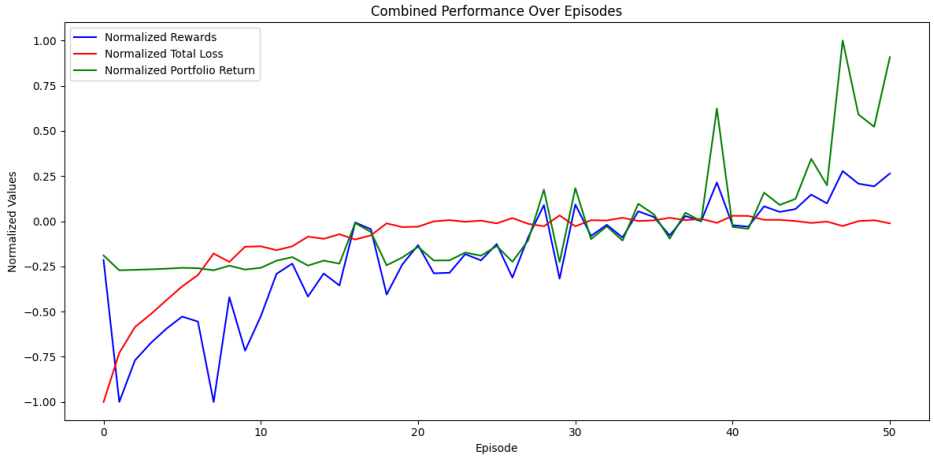
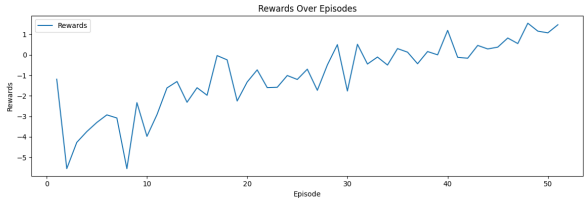


Figure 1: Normalized Rewards, Losses, and Portfolio Returns

Moreover, the aggregation of losses, which became increasingly apparent when the portfolio returns started to rise dramatically, demonstrates an enhancement in the model's capacity to mitigate unprofitable actions or decrease risk. Figures 2 and 3 provide visual representations of

the rewards and portfolio returns over episodes, respectively, thereby further illustrating these ideas. The results are important for training trading algorithms because they show that reinforcement learning models can adapt and learn effective trading strategies in real, complicated financial markets.



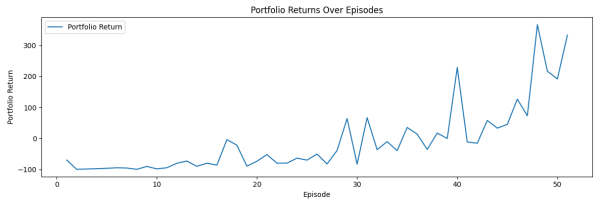


Figure 2: Reward over episodes

Figure 3: Portfolio Returns over episodes

4.2 Inference Results

In the inference phase, the model's performance was evaluated by studying its behavior in a simulated trading environment. This environment was represented using candlestick charts and simple moving averages (SMA). Candlestick charts are widely used in financial research to provide valuable information about price fluctuations, while Simple Moving Averages (SMAs) assist in spotting patterns by eliminating noise from price data.

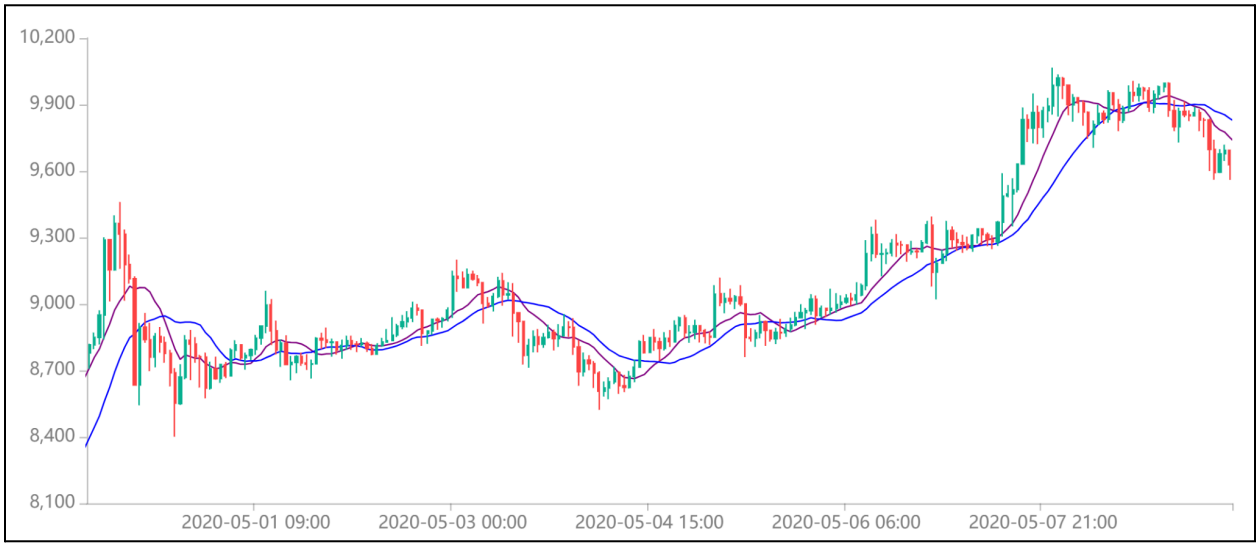


Figure 4: Rendered Trading Environment with candlestick and 2 simple moving average lines (10 and 20)

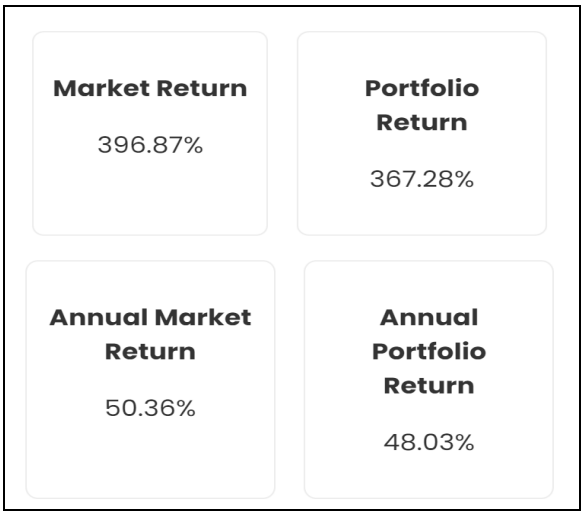


Figure 5: Environment Metrics

Figure 4 illustrates a simulated trading environment generated via inference. It includes SMA indicators and presents an example of a complete episode. This graphical depiction facilitates a clear comprehension of how the model interacts with market dynamics. Figure 5 depicts the resulting metrics of agent performance in the environment, where the focus should be on portfolio return and the annual portfolio return for a single episode to emphasize the financial success of the model.

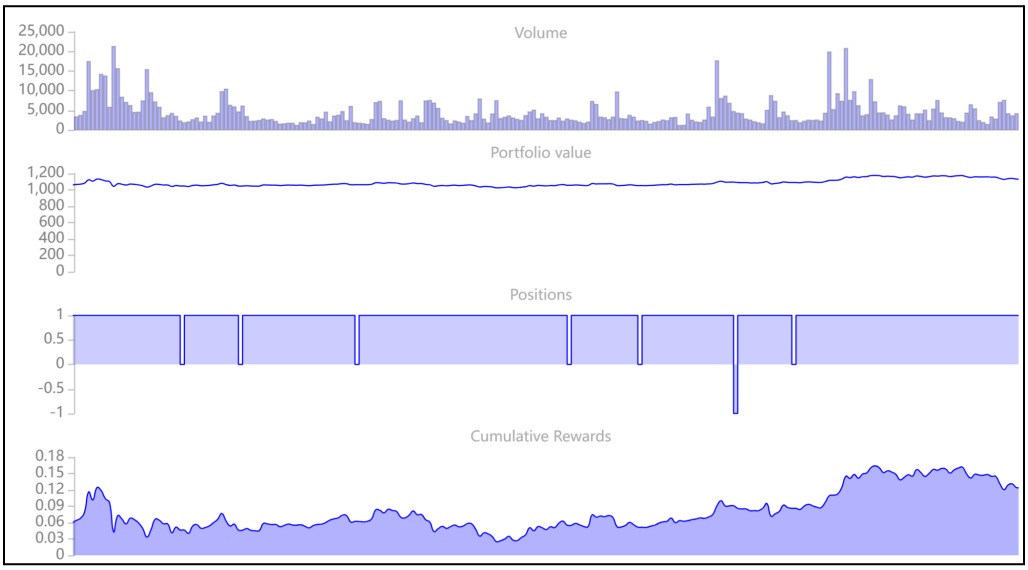


Figure 6: Volume, Portfolio Value, Positions, and Cumulative Rewards

Figure 6 provides an intricate depiction of the volume, showcasing the agent's positions over several time steps within a single episode, along with the cumulative rewards. The positions, encompassing long positions, short positions, and standing on the sidelines, primarily mirror the market fluctuations. Aligning with market trends is essential as it showcases the model's capacity to take advantage of lucrative opportunities, hence improving portfolio returns. The result is significant; it indicates that reinforcement learning models, when adequately trained and

adjusted, may successfully traverse and capitalize on financial markets, adjusting their tactics in real-time to accommodate changing market conditions. The capacity to adapt and make accurate decisions is crucial for success in automated trading systems, making this feature extremely valuable.

4.3 Adaptation to New Data

The robustness of the model was further tested by introducing it to new data, specifically Bitcoin Binance data from January 2023. This phase was essential in assessing the model's capacity to adapt and operate in the face of market situations that had not been encountered before. The findings were praiseworthy, demonstrating the model's ability to apply its learned tactics to unfamiliar data. Figures 7, 8, and 9 demonstrate the model's proficiency in navigating this unfamiliar setting. These numbers together illustrate the model's steady performance, showcasing its capacity to sustain profitability and adjust its trading tactics efficiently in a dynamic and developing market.

Figure 7: Rendered Trading Environment with New Data

Figure 7 displays the market movement of the new data; Figure 8 depicts the resulting portfolio returns, and Figure 9 offers insights into the cumulative rewards and positions adopted over time which explains the model's decision-making process in the new environment. The model's resilience and possible usefulness in real-world trading circumstances are highlighted by the consistent performance demonstrated by these statistics.

Figure 8: Environment Metrics with New Data

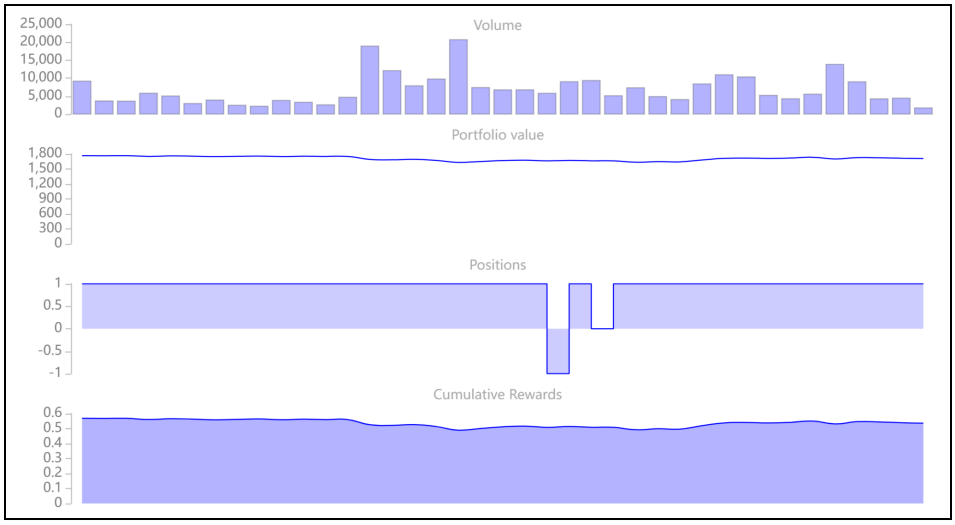


Figure 9: Volume, Portfolio Value, Positions, and Cumulative Rewards with New Data

5.0 Concluding Remarks

To summarize, the model's progression from being trained on past data to adjusting to current market conditions in 2023 displays its capacity as an influential instrument in the field of automated trading. The success of reinforcement learning techniques in financial applications is demonstrated by its capacity to acquire knowledge, adjust, and make lucrative selections under diverse market situations. The model's ability to comprehend and exploit intricate market dynamics not only demonstrates its effectiveness but also opens up avenues for further exploration and advancement in this domain. The findings have important ramifications, as they provide intriguing opportunities for the creation of advanced, AI-powered trading systems that can effectively and independently navigate the financial markets.