Open-ended Deep Reinforcement Learning for a Bitcoin Trading Bot

Adrian Altermatt
adrian.altermatt@hotmail.com

***Abstract*— Cryptocurrencies have a high volatility but also provide a high potential of profit. This paper discusses the use of deep reinforcement learning in bitcoin trading aiming to optimize profit. Various parameter configurations and sampling modes were explored, focusing on hybrid sampling, which combines random and recent experiences from the replay memory. The results demonstrated that hybrid sampling consistently outperformed recent sampling and, under optimal parameters (batch size 256, training every 64 steps), showed significant promise across both validation sets. However, performance declines were noted over longer periods, likely due to catastrophic forgetting. These findings suggest that while the hybrid sampling approach can enhance a trading bot's performance, particularly in the short term, its effectiveness diminishes over time.**

Keywords—Cryptocurrency, Bitcoin, Open-Ended Learning, Deep Reinforcement Learning, Automated Trading Systems

I. Introduction

Cryptocurrency has been a topic of interest for quite some time now. One of its key features is decentralization, which means that it's not controlled by any central authority, making it different from traditional currencies,. Bitcoin being the oldest cryptocurrency has the biggest market share [1].

The concept of reinforcement learning (RL) and artificial intelligence in general can be feasible approach for trading cryptocurrencies more efficiently. RL involves learning through trial and error, a process that helps in refining strategies and making better decisions over time. Thus, RL is a promising approach for Bitcoin trading due to its ability to learn optimal strategies. Through constant updates with real-time market data a bot using RL can continually adapt to the market. Special models can be retrained in batches or in an incremental learning way while running to improve the trading.

This paper investigates the potential of improving trading efficiency, optimizing decision making and potentially generating profits in the cryptocurrency market by using a particular RL trading approach.

II. Description of the Optimization Problem

Deep reinforcement learning belongs to Artificial Intelligence, it learns from experience through trial and error and is often used in neural network approaches. It has been applied for trading cryptocurrencies in various settings.

A. Problem Model

The Bitcoin market is unregulated, and the price of one Bitcoin is pure speculative as cryptocurrencies hold no intrinsic value. A trader considers different indicators for making decisions. The price over time might be the most important information to analyse trends. Partial amounts of Bitcoins can be bought and sold potentially providing an unlimited number of possible actions. The transactions are run through a broker.

To evaluate the bot's performance different strategies can be taken: Buy and hold is the simplest strategy as it only requires the price. It basically shows how the market changes over a given period of time. Another popular method is the Sharpe ratio [2]. It additionally involves the risk of the investment in its calculation.

The Bitcoin market is everchanging and trends in the price trajectory might arise. This indicates that there is no one solution for a trading bot, and a trading agent should continue to learn open-ended while deployed.

B. Optimization Methods

Deep RL can be applied with different types of algorithms for optimization: Value function methods like Deep Q-Networks (DQN) try to estimate the expected return of being in a state. Policy Search Methods directly search for an optimal policy rather than estimating the return. A combination of the two is called actor-critic method where the actor (policy) learns by using expected rewards from the critic (value function). This technique allows a trade-off between bias introduction from value function approaches and variance reduction of policy gradients [3].

Humans have an ability to continually learn and adapt to new situations. This skill is known as open-ended, incremental or continual learning. Implementing this idea of constant adjustment is a rapidly growing area of modern machine learning [4].

C. Implementation and Comparison of Different Optimization Methods

A bot for Bitcoin trading takes different actions based on the information received. While [5], [6], [7] only permitted the full amount of either money or Bitcoin to be traded, [8] had a continuous action space allowing partial trades.

Deep RL with proximal policy optimization (PPO) is tried in [6] without beating the buy and hold strategy.

[8] proposes a solution with two different models, one predicting the future price, the other taking actions using deep reinforcement learning. It concludes that a long short-term memory (LSTM) model combined with the RL algorithm PPO outperforms the other models. [5] compares different algorithms and proposes a combination of PPO, Dueling Double Deep Q-Network (DD-DQN) and Advantage Actor Critic (A2C). The proposed method slightly outperforms the DD-DQN and PPO on their own.

A Bayesian Optimization approach has been used in [7] for parameter optimization in a Double Deep Q-Network and proved to increase the profits. It has been applied to multiple cryptocurrencies at the same time.

Considering the reward function there were different approaches. The omega ratio [5], profit reward function [8], a custom function based on the exponential moving average [6] and the Sharpe ratio [7] have been used.

An open-ended method was applied to real-time stock prediction. The method was compared to Offline-Online

learning, where the models were trained and tested in batches after every trading session. It was found that the Offline-Online model outperformed the incremental one [9].

III. Methods

A. Base Bot creation

A Bitcoin trading bot was created based on [10] and [11]. The bot is based on a DD-DQN and decided in 30-minute time steps whether to hold US-Dollars or Bitcoin. It was trained on data from Binance ranging from 01.01.2019 to 31.12.2023. For validation two time series were used.

Validation set one went from 01.01.2024 to 15.05.2024 which is the data right after the training data. In this time the bitcoin price increased by 40.69%. Validation set two ranged from 30.11.2017 to 30.11.2018 where the price decreased by 64.39%. Validation set one was weighted higher because it is the time right after the training set and thus the more realistic scenario. The second validation set was chosen as an example of a declining market.

The bot's performance was measured in percentage of portfolio value gained and compared to the market performance. To enhance the predictive capability of our model, the following features were derived from the raw Bitcoin trading data:

Feature Close: This feature represents the standardized percentage change in the closing prices. We calculated the daily percentage change in the closing prices and applied robust scaling to mitigate the impact of outliers.

$$feature\_close = robust\_scale\left(\frac{close_t - close_{t-1}}{close_{t-1}}\right)$$

Feature Open: This feature is the ratio of the opening price to the closing price, standardized using robust scaling. This captures the relative difference between the opening and closing prices.

$$feature\_open = robust\_scale\left(\frac{open}{close}\right)$$

Feature High: Like feature_open, this feature is the ratio of the highest price of the day to the closing price, standardized using robust scaling.

$$feature\_high = robust\_scale\left(\frac{high}{close}\right)$$

Feature Low: This feature is the ratio of the lowest price of the day to the closing price, also standardized using robust scaling.

$$feature\_low = robust\_scale\left(\frac{low}{close}\right)$$

Feature Volume: This feature captures the relative trading volume. We scaled the trading volume by the maximum volume observed over a rolling window of the past 7 days (24 hours each day) and applied robust scaling.

$$feature\_volume = robust\_scale\left(\frac{volume}{volume.rolling(7*24).\max()}\right)$$

A trading fee of 0.01% was applied to each transaction, deducted from the portfolio valuation whenever a position change occurred. The model used a rolling window of 15-time steps, considering this number of historical data points for predictions. The initial portfolio value was set at $1000. The reward function guided the trading agent to maximize profit while accounting for trading costs. If the trading position was 0 (not holding the asset), the reward was the negative change in the asset's closing price. If the position was 1 (holding the asset), the reward was the positive change in the asset's closing price. This reward function punished the bot for missing out on profits but rewarded it for dodging losses, along with a reward for profits. Additionally, if there was a change in position, the reward was reduced by 0.01% of the portfolio valuation to reflect the trading fee.

The agent was trained for approximately 20 hours on a Nvidia GeForce GTX 980 TI. The weights and the replay memory from the trained bot were then stored in [12]. This checkpoint was reloaded for each open-ended experiment.

B. Parameter optimization and different modes

In our experiments, we tested a range of batch sizes and training frequencies to optimize the open-ended deep reinforcement learning process. Specifically, we used batch sizes of 64, 128, and 256, combined with training frequencies of every 64, 128, and 256 steps referred to as "train every".

To investigate the impact of different experience sampling strategies on learning performance, we implemented three modes for selecting training batches from replay memory:

- A mode where the entire batch was randomly sampled from the replay memory.
- A mode where the entire batch was composed of the most recent experiences.
- A hybrid mode where half of the batch was randomly sampled, and the other half consisted of the most recent experiences.

By experimenting with these configurations, we aimed to determine the optimal balance between randomness and recency in training batch selection, and how these factors influenced the learning efficiency and stability of the model.

C. Multiple runs of best configuration

After choosing a best set of parameters and mode, it was run for multiple times, as there still could be randomness involved. The best set of parameters were run for 21 times to find out how they perform on average.
The mode chosen was Hybrid sampling with the parameters batch size: 256 and train every: 64.

D. Comparison of Median Performance with the Market

The median performance of the 21 runs was plotted on a timeline alongside the Bitcoin price and the baseline bot without additional training. This comparison provides a clear visualization of the trained bot's performance in relation to

actual market movements and the untrained bot's performance.

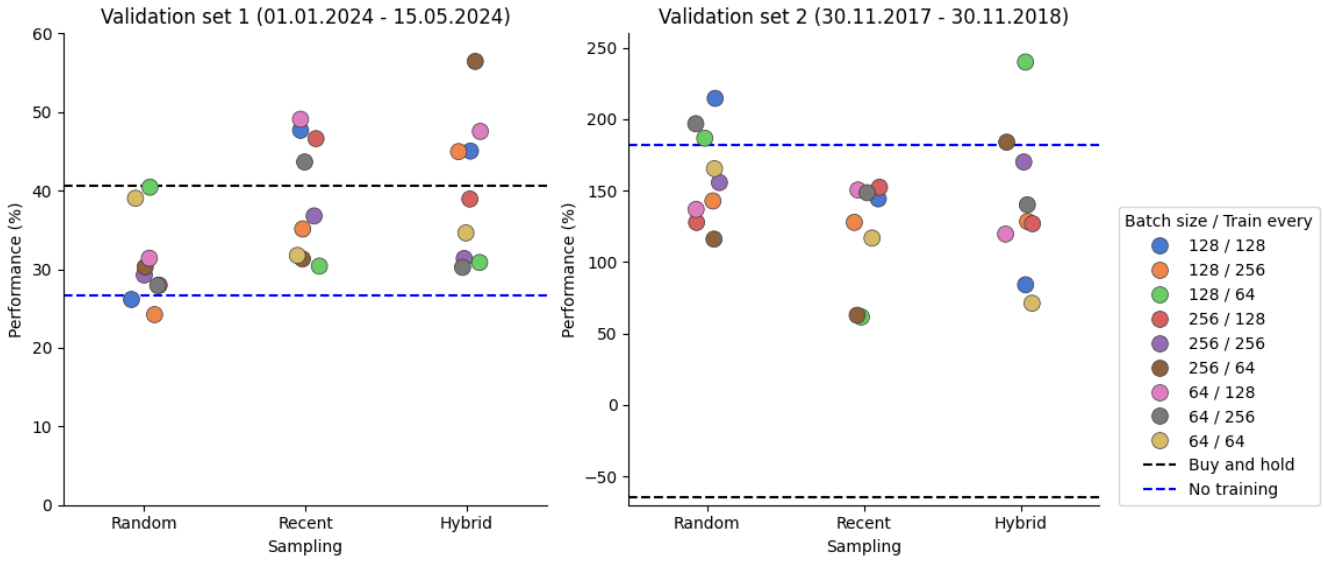

Figure 1 The three sampling modes are on the X-axis with the return in percentage as performance on the Y-axis. The blue dotted line shows how well the bot did without open-ended training. The black dotted line represents the bitcoin market return. The data points are coloured after the parameter combination of batch size and training interval.

IV. Results

A. Base Bot without open-ended training

In the first validation set, the market return was 40.69%, while the trading bot achieved a return of 26.59%. In the second validation set, the market return was -64.39%, and the trading bot achieved a return of 181.98%.

B. Parameter optimization and different modes

The results were visualized in Figure 1. For validation set one any kind of open-ended training improved the performance of the bot. The recent and hybrid sampling method were able to beat the market return with the right parameters.

For validation set two most training configurations decreased the return compared to the bot without training. The market was still beaten by a large margin. Hybrid sampling was again able to beat the market with two parameter sets.

The hybrid sampling with the parameters batch size equals to 256 and training every 64 steps was among the top performers for both validation sets.

C. Multiple runs of best configuration

The results of the 21 runs of each validation set were shown in Figure 2. For validation set one the bot without open-ended training was beaten every time. The median return (41.46%) slightly surpasses the market return (40.69%).

In validation set two the open-ended version outperformed the no further training version most of the times. The median return was 196.86% whereas the bot without training reached 181.98%. The market was beaten comfortably on every iteration.

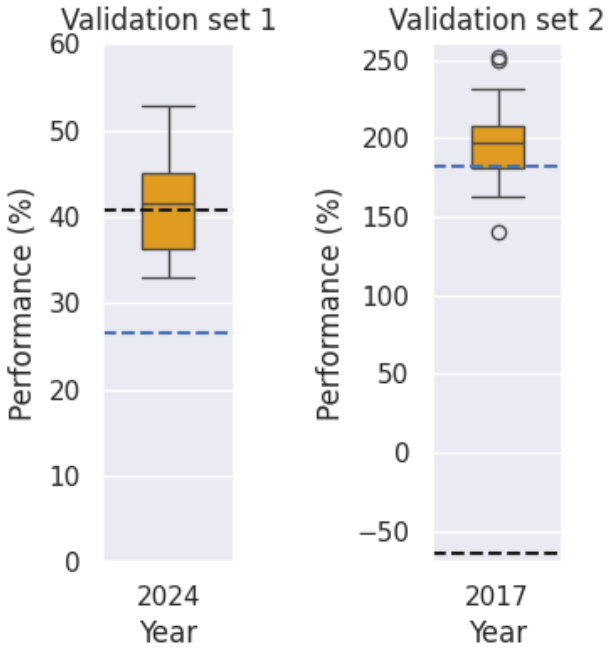

Figure 2 shows the performance of the bot with the parameters batch size: 256 and train every: 64. The boxplot contains 21 runs for each validation set. The blue dotted line shows how well the bot did without open-ended training. The black dotted line represents the bitcoin market return.

D. Comparison of median performance with the market

The performance over time was plotted in **Error! Reference source not found.**. For both validation sets an improvement was visible when the bot continued training in an open-ended fashion. The gap between the two agents closed again after approximately seven months. Validation set one was comprised of only five and a half months of data.

The market in validation set one showed a slight decline in January and February whereas the bot was able to return a profit.

B. *Parameter optimization and different modes*

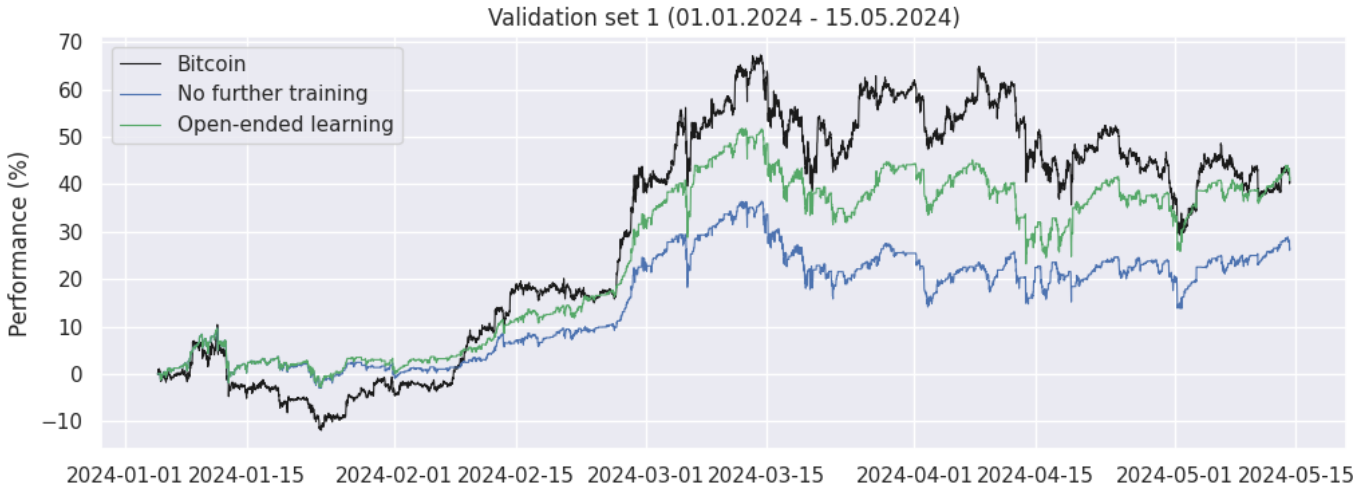

Figure 4 This graph shows the performance of the Bitcoin market (black line), the "No further training" strategy (blue line), and the "Open-ended learning" strategy (green line). The Bitcoin market achieved a cumulative return of around 40%. The "No further training" strategy underperformed with a maximum return of approximately 26.59%. The "Open-ended learning" strategy slightly outperformed Bitcoin by the end, achieving a return about 1% higher than Bitcoin's.

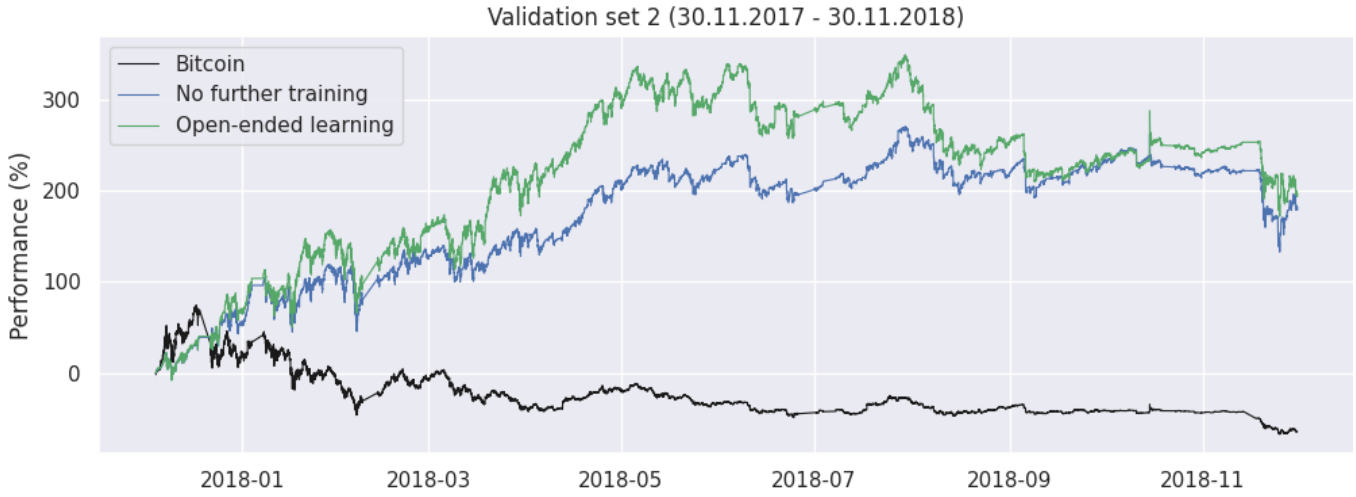

Figure 3 This graph shows the performance of the Bitcoin market (black line), the "No further training" strategy (blue line), and the "Open-ended learning" strategy (green line). During this period, the Bitcoin market experienced a significant decline, with a return of approximately -64.39%. The "No further training" strategy achieved a return significantly higher than the market at around 181.98%. The "Open-ended learning" strategy demonstrated even higher performance, reaching a return of approximately 196.86%, generating substantial profits despite the overall market decline.

V. Discusson

A. *Base Bot*

While the base bot was able to beat the market in validation set two by a staggering 246.36% the validation set one could not be beaten. A different bot or the same bot with different training might have achieved better results, but this was not the focus of this research. The bot's performance could potentially be improved by incorporating additional trading indicators.

The results indicate that hybrid sampling consistently outperforms recent sampling. Random sampling did not provide meaningful insights due to its inherent randomness. Although running multiple iterations and using the median performance could mitigate this randomness, this approach was computationally intensive and thus only applied to the most promising parameter sets. When using random sampling as a benchmark, it is evident that both hybrid and recent sampling perform worse in validation set two than in validation set one, suggesting that their performance likely declines over time. The decline in recent sampling

performance from validation set one to validation set two, compared to hybrid sampling, further suggests the impact of catastrophic forgetting. This issue is more pronounced in the longer validation set two, but it is also visible in validation set one where shorter training intervals, thus more training, performs worse.

C. Multiple runs of best configuration

Using just one combination of parameters multiple times showed that the dispersion of the hybrid mode is quite large. But it also showed that it can be consistently successful as seen for validation set one, where it beat the no training bot every time. This consistency was not observed in validation set two, possibly due to its longer time span.

D. Comparison of median performance with the market

The performance over time shows a steady rise in approximately the first three months. After that the portfolio begins to stagnate. The open-ended training bot outperforms the no training bot in the beginning. More data on how it works out over time would be interesting. Also, a look at other validation sets could show if this a trend or an anomaly. The market in validation set one shows a slight decline in January where the bot with and without training outperforms the market significantly. This leads to the conclusion that the bot does very well in declining markets.

E. Conclusion and future work

This research suggests that an open-ended learning approach can improve a bitcoin trading bot for around seven months. The hybrid sampling approach shows promise but due to the part random sampling the result can still vary. The hybrid mode's performance falls of after a certain time. Training every 64 steps with a batch size of 256 showed the best results for the hybrid mode. The rainbow agent used has shown that it is well suited for this kind of task.

Future work could include testing different hybrid sampling modes with varying percentages of random and recent samples. They should also be tested over longer times. It is likely that performance degrades over long periods, as seen in validation set two. Another potential improvement could be full retraining on random samples after the bot has been running with open-ended learning for a while. Additionally, testing the hypothesis that open-ended training enhances performance on models trained for longer durations or with different methods would be valuable.

References

[1] "Cryptocurrency Market Capitalization." Accessed: Apr. 10, 2024. [Online]. Available: https://www.slickcharts.com/currency

[2] E. Benhamou, B. Guez, and N. Paris, "Omega and Sharpe Ratio." Rochester, NY, Oct. 14, 2019. doi: 10.2139/ssrn.3469888.

[3] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[4] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards Continual Reinforcement Learning: A Review and Perspectives," *J. Artif. Intell. Res.*, vol. 75, pp. 1401–1476, Dec. 2022, doi: 10.1613/jair.1.13673.

[5] B. El Akraoui and C. Daoui, "Deep Reinforcement Learning for Bitcoin Trading," in *Business Intelligence*, M. Fakir, M. Baslam, and R. El Ayachi, Eds., in Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2022, pp. 82–93. doi: 10.1007/978-3-031-06458-6_7.

[6] D. Mahayana, E. Shan, and M. Fadhl'Abbas, "Deep Reinforcement Learning to Automate Cryptocurrency Trading," in *2022 12th International Conference on System Engineering and Technology (ICSET)*, Bandung, Indonesia: IEEE, Oct. 2022, pp. 36–41. doi: 10.1109/ICSET57543.2022.10010940.

[7] M. Tran, D. Pham-Hi, and M. Bui, "Optimizing Automated Trading Systems with Deep Reinforcement Learning," *Algorithms*, vol. 16, no. 1, Art. no. 1, Jan. 2023, doi: 10.3390/a16010023.

[8] F. Liu, Y. Li, B. Li, J. Li, and H. Xie, "Bitcoin transaction strategy construction based on deep reinforcement learning," *Appl. Soft Comput.*, vol. 113, p. 107952, Dec. 2021, doi: 10.1016/j.asoc.2021.107952.

[9] T. Singh, R. Kalra, S. Mishra, Satakshi, and M. Kumar, "An efficient real-time stock prediction exploiting incremental learning and deep learning," *Evol. Syst.*, vol. 14, no. 6, pp. 919–937, Dec. 2023, doi: 10.1007/s12530-022-09481-x.

[10] M. Hessel *et al.*, "Rainbow: Combining Improvements in Deep Reinforcement Learning." arXiv, Oct. 06, 2017. doi: 10.48550/arXiv.1710.02298.

[11] C. Perroud, "ClementPerroud/Rainbow-Agent." Apr. 18, 2024. Accessed: Jun. 05, 2024. [Online]. Available: https://github.com/ClementPerroud/Rainbow-Agent

[12] A. Altermatt, "adrianad/Open-Ended-Deep-RL-Bitcoin-Trading-Bot." Jun. 05, 2024. Accessed: Jun. 05, 2024. [Online]. Available: https://github.com/adrianad/Open-Ended-Deep-RL-Bitcoin-Trading-Bot