ZeltaLabs
untrade

Zelta untrade Crypto Trading Challenge

Report

By Team 99

December 18, 2023

Abstract

Crypto Trading is a skill which requires a deep understanding of the financial knowledge along with good analytical skills, quick at decision making and knowing when to enter and exit market by keeping emotions aside. An experienced trader finds the useful information by calculating and analysing the various technical indicators present and also sometimes the specific news which can effect the movement of a particular commodity. Ones decision to buy, sell or hold has to be as accurate as much as possible. AlgoTrading is an emerging domain where we try to build some algorithmic strategy to take decisions by processing various technical indicators related to a commodity. A good algorithmic strategy should be able to generate good profit while minimising the risk as much as possible. In our work we try to develop a CryptoBot based on Reinforcement Learning for algorithmic trading system for cryptocurrencies (specifically BTCUSDT), with the goal of maximizing the total portfolio value. RL is an autonomous system, self-teaching system that essentially learns by trial and error. In our experiments, the trained agent was able to take proper decisions and keep the portfolio in the profit.

Contents

1 Introduction

The algorithmic trading is one important domain where we try to automate market trading by using the functionality of an algorithm. It requires multiple factors to be considered while making a decision in the market namely *when to buy, sell or hold, how much quantity.* Deep Reinforcement learning is one such technique where we can train the algorithms to approximate the relation between historical data and the market price. The DRL agent can improve itself by exploring decison taking through the historical data and then perform efficiently on current market. The Deep Reinforcement Learning is a complex dynamical system which leverages the functionality of neural networks to approximate the algorithm for trading. The algorithm learned is intended to maximize the cost of the total portfolio or, in other words, the profit. While training the agent for crypto trading, it takes the the market data (OHLCV) of certain fixed time interval such as 5m (15 minutes), 30m (30 minutes), 12h (12 hours), 1d (1 day), etc which can be visualised in form of candle charts. This data is further used to calculate other techincal indicators like RSI, MACD etc. and further processed(normalisation, dealing with missing values, removing highly correlated features). Here we formalise the crypto trading as MDP (Markov decision framework). The implementation details are elaborated in further sections.

2 Data Acquisition

In the realm of cryptocurrency, particularly for BTC/USDT market, the importance of high-quality data is paramount. The dynamic nature of cryptocurrency markets requires robust datasets for training the models effectively. In this context, the acquisition of data for BTC/USDT market involves unique challenges and opportunities. Understanding the market trends, price movements, volumes in trading and relevant external factors is crucial for developing accurate and reliable predictive models.

The data collected spans a four-year time period, aligned with the specified problem statement. The dataset encompasses records from 1st January 2018 to 31st December 2022. Within this timeframe, the dataset includes the Open, High, Low, Close, and Volume (OHLCV) data for trades occurring at different time intervals, such as 3 minutes, 5 minutes, 15 minutes, 30 minutes, 1 hour.

3 Data Preprocessing

Data preprocessing is an integral step in refining datasets for effective analysis and modeling. The market data acquired from online sources are sometime ambiguous/garbage values or contains some missing values. The data sample used during training must be processed efficiently to avoid any discrepancies otherwise can result in unwanted/unclear output. During this step, data is checked for duplicates, cleaned for purity, and potential patterns are found to produce results that are acceptable. All presumptions are validated in this stage, and a sort of hypothesis is created and tested using conventional statistical models. This step-by-step procedure is known as statistical analysis, and it is essential to the predictive analysis method.

In our setup we removed unnecessary columns and replaced nan numbers with zero. We applied data cleaning steps like renaming, dropping columns, and analyzing univariate, bi-variate, and multi-variate processes. The specific steps and techniques for data cleaning and correction may vary for each dataset, but the overarching goal is to detect and remove errors and anomalies, enhancing the value of information for analytics and decision-making. We added the required technical indicators to our dataset and checked for correlation between them to drop the highly correlated features. Other steps involves essential operations such as splitting data to create distinct subsets for training and testing, ensuring model generalization. Additionally, data transformation methods like smoothing, which eliminates noise values, and discretization, reducing data granularity over intervals, contribute to dataset optimization. Another critical technique is normalization, scaling data over a specified period to enhance consistency and comparability, particularly useful in financial analyses.

4 Reinforcement Learning

Reinforcement learning (RL) can be seen as the formalization of an optimal policy which is able to decide the best action given the market data at particular time and ensuring the maximization of the expected cumulative profit of an agent. The agent(here CryptoBot) interacts with the environment by executing actions and receiving observations and rewards. At each time step t, which ranges over a set of discrete time intervals, the agent selects an action a from a set of legal actions A at state $s_t \in S$, where S is the set of possible states. Action selection is based on a policy, π. The policy is a description of the behavior of the agent and tells the agent which actions should be selected for each possible state. As a result of each action, the agent receives a scalar reward $r_t \in \mathbb{R}$ and observes the next state $s_{t+1} \in S$. The transition probability of each possible next state s_{t+1} is defined as $P(s_{t+1}|s_t, a_t)$, with $s_{t+1}, s_t \in S$, and $a_t \in A$. Similarly, the reward probability of each possible reward r_t is defined as $P(r_t|s_t, a_t)$, where $s_t \in S$ and $a_t \in A$. Hence, the expected scalar reward, r_t, received by executing action a in the current state s is calculated based on $\mathbb{E}_P(r_t|s_t, a_t)(r_t|s_t = s, a_t = a)$. This framework can be seen as a finite Markov Decision Process (MDP). The aim of the learning agent is to learn an optimal policy π^*, which defines the probability of selecting action a in state s, so that the sum of the discounted rewards over time is maximized. The expected discounted return R_t at time t is defined as follows:

$$R_t = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right], \tag{1}$$

where $\mathbb{E}[.]$ is the expectation with respect to the reward distribution and $0 < \gamma < 1$ is called the discount factor. At this point, a Q-value function, $Q^\pi(s,a)$, can be defined as follows:

$$Q^\pi(s,a) = \mathbb{E}_\pi[r_t|s_t = s, a_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} r_{t+k}|s_t = s, a_t = a\right], \tag{2}$$

The Q-value, $A^\pi(s,a)$, for an agent is the expected return achievable by starting from state $s \in S$ and performing action $a \in A$ following policy π. Equation 2 satisfies a recursive property,

so that an iterative update procedure can be used for the estimation of the Q-value function:

$$Q_{i+1}^{\pi}(s,a) = \mathbb{E}_{\pi}[r_t + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a]$$
$$= \mathbb{E}_{\pi}\left[r_t + \gamma Q_i^{\pi}(s_{t+1} = s', a_{t+1} = a') | s_t = s, a_t = a\right], \quad \forall s, s' \in S \text{ and } a, a' \in A.$$

Reinforcement learning agent aims at finding the policy which achieves the greatest outcome. Hence, it must learn an optimal policy π^* with the expected value greater than or equal to all other policies, and leading to an optimal Q-value $Q^*(s,a)$. In particular, the iterative update procedure for estimating the optimal Q-value function can be defined as :

$$Q_{i+1}^{i}(s,a) = \mathbb{E}_{\pi}[r_t + \gamma \max Q^i(s',a') | s,a].$$

5 Strategy Design

5.1 Environment

The crypto trading environment serves as a fundamental framework for the evaluation and development of algorithmic trading strategies in cryptocurrency markets. The environment is meticulously designed to capture the intricacies of real-world trading scenarios, facilitating the training and testing of algorithms. The `DataProcessor` class orchestrates a comprehensive preprocessing pipeline, transforming raw market data into essential components such as asset prices, encapsulating pertinent technical indicators, and reflecting market volatility. Two distinct environments, namely the training and testing environments, are instantiated. These environments are characterized by parameters such as `lookback` (historical window considered, crucial for tailoring algorithmic trading strategies by influencing the trade-off between responsiveness to short-term market dynamics and identification of long-term trends), starting capital for trading, and transaction costs. The γ parameter introduces the temporal discount factor, shaping the agent's decision-making process. This environment not only provides a simulated platform for algorithmic trading strategies but also ensures the reproducibility and consistency of experiments by storing relevant data, such as asset prices, for subsequent analysis. Overall, this crypto trading environment serves as a crucial tool for advancing the understanding of algorithmic trading dynamics in the dynamic landscape of cryptocurrency markets.

- **State** $s \in S$: A state represents an agent's perception of a market environment, which may include balance, shares, OHLCV values, technical indicators, social data, etc.

- **Action** $a \in A$: An action is taken from the allowed action set at a state. Actions may vary for different trading tasks, e.g., for stock trading, the actions are the number of shares to buy/sell for each stock, while for portfolio allocation, the actions are the allocation weights of the capital.

- **Reward** $r(s, a, s_0)$: Reward serves as an incentive mechanism for an agent to refine its policy. Several common reward functions are available: 1) Change of portfolio value, where $r(s, a, s_0) = v_0 - v$, with v_0 and v representing portfolio values at states s_0 and

s, respectively; 2) Portfolio log return, expressed as $r(s, a, s_0) = \log\left(\frac{v_0}{v}\right)$; and 3) [Insert description of the third reward function]. These functions play a critical role in shaping the learning process of the agent, guiding it towards the discovery of optimal policies within the dynamic context of the trading environment.

- **Value Function** : It is critical component guiding the agent's decisions. It quantifies the expected cumulative reward starting from a specific state s and following a policy. In financial markets, a well-crafted value function is crucial for the agent to learn optimal trading strategies, balancing short-term gains and long-term objectives.

Algorithms	Input	Output	Type	State-action spaces support	Finance use cases support	Features and Improvements	Advantages
DQN	States	Q-value	Value based	Discrete only	Single stock trading	Target network, experience replay	Simple and easy to use
Double DQN	States	Q-value	Value based	Discrete only	Single stock trading	Use two identical neural network models to learn	Reduce overestimations
Dueling DQN	States	Q-value	Value based	Discrete only	Single stock trading	Add a specialized dueling Q head	Better differentiate actions, improves the learning
DDPG	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Being deep Q-learning for continuous action spaces	Better at handling high-dimensional continuous action spaces
A2C	State action pair	Q-value	Actor-critic based	Discrete and continuous	All use cases	Advantage function, parallel gradients updating	Stable, cost-effective, faster and works better with large batch sizes
PPO	State action pair	Q-value	Actor-critic based	Discrete and continuous	All use cases	Clipped surrogate objective function	Improve stability, less variance, simply to implement
SAC	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Entropy regularization, exploration-exploitation trade-off	Improve stability
TD3	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Clipped double Q-Learning, delayed policy update, target policy smoothing.	Improve DDPG performance
MADDPG	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Handle multi-agent RL problem	Improve stability and performance

5.2 Advantages of using RL

Reinforcement Learning (RL) holds significant promise in financial markets due to its adaptability in dynamic environments, capacity to capture non-linear relationships, and optimization capabilities for trading strategies. RL enables real-time decision-making, continuous learning, and customization to specific financial objectives, such as portfolio management and risk optimization. Its ability to explore unknown strategies and reduce human bias makes it a valuable tool in algorithmic trading and portfolio optimization. However, challenges such as robust model evaluation and ethical considerations must be carefully addressed for successful implementation in financial research and applications.

Our analysis of various types of trading algorithms, showing their shortcomings and strengths, is presented in Table 1.

5.3 Agents Implemented

Primarily two types of agents have been used to obtain the optimal outcome, single-algorithm models and multiple algorithm models. All the models have been trained on trading data corresponding to 1 day interval, from 01/01/2018 to 31/12/2022. - $\gamma = 0.95$ - commission = 0.1% - initial cash = 1,000,000 USD - initial btc = 0 - total timesteps = 100,000 - size of 1 time step = 2048

Following indicators have been used while computing the results:

Table 1: Comparison of RL, ARIMA, DL, and Conventional ML Models in Algorithmic Trading

Criteria	**Reinforcement Learning (RL)**	**ARIMA**	**Deep Learning (DL)**	**Conventional ML**
Adaptability to Dynamic Environments	Well-suited for non-stationary environments; adapts to changes over time	Assumes stationarity; may not perform well in highly dynamic markets	Adapts to changing conditions; requires substantial data and computational resources	Adapts with proper feature engineering and retraining
Handling Complex Nonlinearities	Capable of handling complex nonlinear relationships in data	Better suited for linear relationships; struggles with nonlinear patterns	Capable of capturing complex nonlinearities with proper architecture	Can capture nonlinear relationships with appropriate model selection
Model Interpretability	Interpretability can be a challenge, especially in deep RL models	Relatively simple and offers better interpretability	Interpretability can be limited, especially in deep learning models	Varies; interpretability depends on the chosen model
Data Requirements	Often requires a significant amount of data and longer training periods	Can work well with smaller datasets; quicker to train	Benefits from larger datasets; training may be computationally intensive	Data requirements can vary based on the chosen algorithm
Exploration vs. Exploitation	Naturally handles exploration-exploitation trade-off	May require additional mechanisms to balance exploration and exploitation	May require specific exploration strategies	Depends on the chosen ML algorithm
Risk Management	Can incorporate risk management strategies directly into the learning process	Risk management may need to be explicitly integrated into the strategy	Can incorporate risk management with proper design	Requires explicit consideration of risk management in the strategy

- **ADX (Average Directional Index):** The ADX is employed to quantify the robustness of a prevailing trend within a financial instrument, facilitating the discernment of the intensity of price movements within the observed market.
- **ADXR (Average Directional Movement Rating):** Serving as a smoothed variant of the ADX, the ADXR provides a protracted analytical horizon, affording a nuanced assessment of trend strength through the incorporation of a moving average.
- **Aroon:** The Aroon indicator is utilized to delineate the temporal distance since the last occurrence of the highest and lowest prices, thereby contributing to the identification of potential instances of trend reversal or continuity.
- **ATR (Average True Range):** Functioning as a metric of market volatility, the ATR aids

traders in establishing judicious stop-loss levels and evaluating the prospective amplitude of price fluctuations.

- **BOP (Balance of Power):** The BOP metric is applied to gauge the relative strength between market buyers and sellers, thereby enhancing predictive capacities concerning potential shifts in prevailing trends.
- **CCI (Commodity Channel Index):** Employed for the detection of overbought or oversold market conditions, the CCI facilitates the identification of potential inflection points, indicative of impending trend reversals.
- **Open/High/Low/Volume:** Fundamental inclusions of open, high, low, and volume data furnish elemental information essential for technical analysis and the identification of chart patterns within the financial markets.
- **MACD (Moving Average Convergence Divergence):** Comprising two moving averages, the MACD serves to delineate prevailing trend directions and potential reversal points, constituting a ubiquitous tool for momentum analysis and trend change identification within financial markets.

All the agents have been tested on 3 different time-frames:

- 01/01/2018 to 31/12/2022 - To ensure that our model does not overfit.
- 01/01/2023 to 30/06/2023 - To analyse its performance on an unknown dataset.
- 01/02/2021 to 01/04/2022 - To analyse its resilience in peak volatility.

5.3.1 Single-Algorithm Agents

In this part we have analysed performance of 5 different algorithms on BTCUSDT market. The insights obtained from this analysis further helped us in developing multi-algorithm agent.

- **Deep Deterministic Policy Gradient (DDPG)**: DDPG is used to encourage maximum investment return. DDPG combines the frameworks of both Q-learning and policy gradient, and uses neural networks as function approximators. In contrast with DQN that learns indirectly through Q-values tables and suffers the curse of dimensionality problem, DDPG learns directly from the observations through policy gradient. It is proposed to deterministically map states to actions to better fit the continuous action space environment.

  At each time step, the DDPG agent performs an action a_t at s_t, receives a reward r_t and arrives at s_{t+1}. The transitions (s_t, a_t, s_{t+1}, r_t) are stored in the replay buffer R. A batch of N transitions is drawn from R and the Q-value y_i is updated as:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta_{\mu'}, \theta_{Q'})), \quad i = 1, \ldots, N. \tag{3}$$

  The critic network is then updated by minimizing the loss function $L(\theta_Q)$, which is the expected difference between outputs of the target critic network Q' and the critic network

Q, i.e.,

$$L(\theta_Q) = \mathbb{E}_{s_t,a_t,r_t,s_{t+1}\sim\text{buffer}}\left[(y_i - Q(s_t, a_t|\theta_Q))^2\right]. \tag{4}$$

DDPG is effective at handling continuous action space, and so it is appropriate for stock trading.

- **Advantage Actor Critic (A2C)**: A2C is a typical actor-critic algorithm and we use it a component in the ensemble strategy. A2C is introduced to improve the policy gradient updates. A2C utilizes an advantage function to reduce the variance of the policy gradient. Instead of only estimates the value function, the critic network estimates the advantage function. Thus, the evaluation of an action not only depends on how good the action is, but also considers how much better it can be. So that it reduces the high variance of the policy network and makes the model more robust.

  A2C uses copies of the same agent to update gradients with different data samples. Each agent works independently to interact with the same environment. In each iteration, after all agents finish calculating their gradients, A2C uses a coordinator to pass the average gradients over all the agents to a global network. So that the global network can update the actor and the critic network. The presence of a global network increases the diversity of training data. The synchronized gradient update is more cost-effective, faster and works better with large batch sizes. A2C is a great model for stock trading because of its stability.

  The objective function for A2C is:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t)\right] \tag{5}$$

  Where $\pi_\theta(a_t|s_t)$ is the policy network, $A(s_t, a_t)$ is the Advantage function can be written as:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{6}$$

  or

$$A(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}) - V(s_t) \tag{7}$$

- **Proximal Policy Optimization (PPO)**: We explore and use PPO as a component in the ensemble method. PPO is introduced to control the policy gradient update and ensure that the new policy will not be too different from the previous one. PPO tries to simplify the objective of Trust Region Policy Optimization (TRPO) by introducing a clipping term to the objective function.

  Let us assume the probability ratio between old and new policies is expressed as:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}. \tag{8}$$

The clipped surrogate objective function of PPO is:

$$J_{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}(s_t, a_t), \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}(s_t, a_t) \right) \right], \tag{9}$$

Where $r_t(\theta)\hat{A}(s_t, a_t)$ is the normal policy gradient objective, and $\hat{A}(s_t, a_t)$ is the estimated advantage function. The function $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ clips the ratio $r_t(\theta)$ to be within $[1-\epsilon, 1+\epsilon]$. The objective function of PPO takes the minimum of the clipped and normal objective. PPO discourages large policy changes moving outside of the clipped interval. Therefore, PPO improves the stability of the policy network training by restricting the policy update at each training step. We select PPO for stock trading because it is stable, fast, and simpler to implement and tune.

- **Soft Actor-Critic (SAC)** : Soft Actor-Critic (SAC) is an off-policy actor-critic deep reinforcement learning algorithm designed for continuous action spaces. SAC combines the benefits of entropy regularization and off-policy learning to achieve stable and efficient policy optimization.

  In SAC, the objective function includes an entropy term that encourages the policy to be stochastic. This entropy term is subtracted from the expected return, resulting in an objective that maximizes both the expected return and entropy. The objective function is given by:

$$J(\theta) = \mathbb{E}_{(s_t, a_t)\sim\mathcal{D}} \left[\alpha \log \pi_\theta(a_t|s_t) - Q_\phi(s_t, a_t) \right], \tag{10}$$

  where $\pi_\theta(a_t|s_t)$ is the policy, $Q_\phi(s_t, a_t)$ is the state-action value function, and $\mathcal{D}$ is the replay buffer.

  The entropy-regularized objective improves the exploration-exploitation trade-off by discouraging premature convergence to suboptimal deterministic policies. SAC is well-suited for tasks with high-dimensional state and action spaces, making it applicable to various real-world problems.

  SAC is known for its stability, robustness, and ability to handle continuous control tasks effectively. Its incorporation of entropy regularization distinguishes it from traditional actor-critic algorithms and contributes to its success in challenging environments.

- **Twin-Delayed Deep Deterministic policy gradient (TD3)**:

  Twin Delayed DDPG (TD3) is an off-policy actor-critic deep reinforcement learning algorithm designed for continuous action spaces. TD3 introduces several modifications to the original Deep Deterministic Policy Gradient (DDPG) algorithm to enhance its stability and sample efficiency.

  One key innovation in TD3 is the use of three distinct Q-value networks to mitigate overestimation bias. The minimum of the three Q-values is used in the Bellman update, reducing the potential for optimistic overestimation.

  Additionally, TD3 implements target policy smoothing, a technique that adds noise to the target actions during the critic update. This regularization improves the robustness of the learned policy and prevents overfitting to specific states.

The TD3 objective function is similar to DDPG but includes modifications to account for the three Q-value networks and target policy smoothing. The objective function is given by:

$$J(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[Q_{\phi_1}(s_t,a_t) - \alpha \log \pi_\theta(a_t|s_t)\right], \tag{11}$$

where $\pi_\theta(a_t|s_t)$ is the policy, $Q_{\phi_1}(s_t,a_t)$ is one of the three Q-value networks, and $\mathcal{D}$ is the replay buffer.

TD3 has demonstrated improved performance and stability compared to DDPG, particularly in challenging environments with sparse rewards and high-dimensional state and action spaces.

5.3.2 Multi-Algorithm Agent

We aim to create a robust trading strategy. Therefore, we employ an ensemble strategy to automatically select the best-performing agent among PPO, A2C, SAC, TD3 and DDPG based on the Sharpe ratio. The ensemble process is described as follows:

Step 1. We train all the aforementioned agents concurrently over the same time period.

Step 2. We validate all five agents by using a validation rolling window after the training window to pick the best-performing agent with the highest Sharpe ratio. [**sharpe_ratio_ref**]. The Sharpe ratio is calculated as:

$$Sharpe\,Ratio = \frac{\bar{r}_p - r_f}{\sigma_p}, \tag{12}$$

where $\bar{r}_p$ is the expected portfolio return, r_f is the risk-free rate, and σ_p is the portfolio standard deviation. We also adjust risk aversion by using the turbulence index in our validation stage.

Step 3. After the best agent is selected, we use it to predict and trade for the next trading window.

The reason behind this choice is that each trading agent is sensitive to different types of trends. One agent performs well in a bullish trend but performs poorly in a bearish trend. Another agent is more adapted to a volatile market. The higher an agent's Sharpe ratio, the better its returns have been relative to the amount of investment risk it has taken. Therefore, we pick the trading agent that can maximize the returns adjusted to the increasing risk. This significantly improves the robustness of the strategy.

6 Backtesting

In the realm of cryptocurrency trading algorithm research, backtesting serves as a fundamental methodological tool for the evaluation and refinement of trading strategies. Backtesting involves the retrospective application of predefined trading rules to historical cryptocurrency market data, offering researchers and algorithm developers a means to simulate the performance of their strategies under past market conditions. This process enables the identification of potential strengths and weaknesses in a trading algorithm, aiding in the optimization and enhancement of its predictive capabilities. Given the dynamic and volatile nature of the cryptocurrency markets, backtesting is particularly crucial in assessing the adaptability and robustness of trading

algorithms to various market scenarios. Transaction costs, slippage, and liquidity considerations specific to the cryptocurrency space are integral components of a comprehensive backtesting framework. However, it is imperative to acknowledge the limitations inherent in backtesting, such as the challenge of predicting future market dynamics based solely on historical data.

6.1 Methods

A profitable and reliable trading strategy in the cryptocurrency market is critical for hedge funds and investment banks.The cryptocurency market face many challenges like the market is highly volatile, the historical market data have a low signal-to-noise ratio and there are large fluctuations (e.g., market crash) in the cryptocurrency market. The challenges can be overcomed by the optimization of the model during backtesting by different methods like:

- Walk-Forward method which divides the data in a training-validation-testing split.
- K-fold crossvalidation method (leaving one period out) with an underlying assumption that the training and validation sets are drawn from an IID process.

Finally this types of DRL algorithms are highly sensitive to hyperparameters resulting in high variability of DRL algorithms' performance. Now due to the strong momentum effect in crypto returns the hyperparameter(external configuration settings of a model or algorithm that need to be specified) tunning comes into the picture which checks over the different hyperparameters for the model and helps us to check the best returns from the model.

6.2 Hyperparameter optimisation(Tuning)

In simple terms, hyperparameter tuning involves selecting various parameters to train a model and comparing the model's performance with different parameter sets to identify the most efficient configuration.

This technique is particularly applied to different agents such as PPO, DDPG, SAC, and TD3, which we use to create crypto trading strategies.

For this purpose, we have employed the popular framework called Optuna, which automates the tuning process. Optuna simplifies and accelerates hyperparameter optimization, allowing agents to enhance their performance through automated tuning rather than manual adjustments.

Optuna utilizes Sequential Model-Based Optimization (SMBO) with Bayesian optimization principles. This approach efficiently explores and exploits the hyperparameter space through iterative trials on which our agents operate, making Optuna a valuable tool.

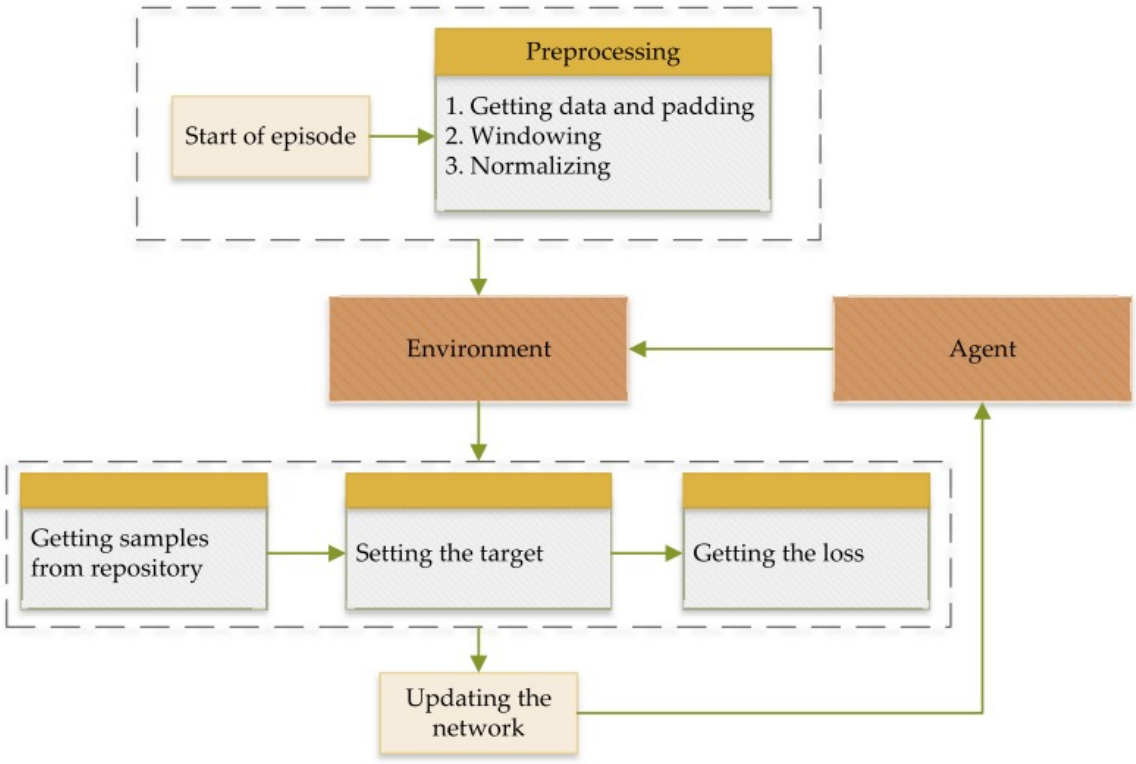
Trials are conducted with different parameter combinations, and each trial produces a result. In the end, Optuna identifies the best-performing trial that outperforms all others.

7 Risk Management

The process of risk management using RL. The meaning of risk management in the proposed system is to identify, evaluate and prioritization the system risks. As it shown, the problem

statement of the portfolio describes the RL-based trading system with specifications by considering the risks and profits of the management issues. Regarding the expressing of the problem statement in the RL architecture, the system agent provides the strategies of trading the assets in the current state of the environment of capital market. All the assets trading information's are connected to the environment. The strategy of trading provided by agent. Risk management proves especially valuable during periods of substantial market volatility, such as sudden market downturns or the occurrence of unexpected events that impact the market. We implement a typical reinforcement learning workflow to train the actor and critic. We build the multi-agent environment. We adjust reward functions to manipulate agents' relationships. We use Algorithm to find a policy that can generate the optimal trading trajectory with minimum implementation shortfall. We feed the states observed from our simulator to each agent. These agents first predict actions using the actor model and perform these actions in the environment. Then, environment returns their rewards and new states. This process continues for a given number of episodes.

To facilitate this process, we can construct a simulator that serves as the environment for our agents. The agents, equipped with actor models, receive states from the simulator, predict actions based on these states, and subsequently execute these actions in the environment. Following each action, the environment provides feedback in the form of rewards and updated states. This iterative cycle continues over a predetermined number of episodes, allowing us to iteratively refine the policy and enhance the agents' performance in generating optimal trading trajectories while minimizing implementation shortfalls.



Risk Management can be implemented in a model using some methods:

1)Position sizing: Position sizing in trading is a strategic methodology employed to ascertain the optimal volume or quantity of an asset that a trader should either buy or sell in the context of a specific trade. This critical risk management technique serves as a mechanism for traders

to effectively govern their exposure to the market and mitigate potential losses. Within the domain of crypto trading, position sizing assumes paramount significance by ensuring that traders allocate a judicious portion of their capital to each trade. This allocation is meticulously aligned with their risk tolerance levels.

To mitigate risk through position sizing, a systematic approach involves the following steps:

i)Define Risk Tolerance: Establish a predetermined maximum percentage of capital to be exposed to risk per trade. This parameter is derived from a comprehensive assessment of one's risk appetite and the overarching strategy governing the entire portfolio.

ii)Calculate Trade Risk: Quantify the potential loss in a given trade by determining the difference between entry and stop-loss levels. This calculation serves to objectively assess the potential downside of the trade under consideration.

iii)Adjust Position Size: Employ a robust position sizing formula to compute the appropriate quantity of the asset to be traded. This calculation is meticulously tailored to align with the predetermined risk tolerance, ensuring a proportional and calibrated exposure to the market.

By employing position sizing strategies in crypto trading, traders can align investments with their risk tolerance, protecting portfolios from market downturns. Determining optimal position sizes mitigates risk and enhances profitability, fostering a disciplined and balanced trading approach. This strategic implementation contributes to a resilient trading framework in the dynamic cryptocurrency market.

2)Stop-losses: In the realm of multi-agent reinforcement learning, the integration of stop-loss orders assumes significance as a risk mitigation strategy. Beyond individual agent actions, the introduction of stop-loss mechanisms within the multi-agent framework becomes instrumental in curtailing potential losses associated with the collective actions of multiple agents. By incorporating stop-loss orders into the decision-making process of each agent, the overall system gains resilience against adverse market conditions. This adaptation aligns with the overarching goal of reducing downside risks in multi-agent reinforcement learning scenarios, enhancing the stability and effectiveness of the collective decision-making process within the dynamic environment of cryptocurrency trading.

3)Risk-Return Ratio: In this statement we explore the single agent working for these ratios but we can explore the application of Multi-Agent Reinforcement Learning in the optimization of Risk-Return Ratios within investment portfolios. Our aims is to leverage MARL techniques to dynamically adjust portfolio allocations, enhancing risk-adjusted returns. Through collaborative decision-making among multiple agents, each representing a unique asset or strategy, we seek to optimize the Risk-Return Ratio. This approach aligns with the goal of developing adaptive and intelligent investment strategies capable of navigating dynamic market conditions. Our findings contribute to the advancement of portfolio optimization methodologies, particularly in the context of employing Multiagent model for enhanced risk management. Sharpe Ratio is a measure that helps investors assess the performance of an investment or portfolio by considering the return achieved relative to the amount of risk taken.

Sharpe is calculated by subtracting the risk-free rate from the investment's return and dividing the result by the standard deviation of the investment's returns.

Sortino Ratio is a risk-adjusted performance metric that evaluates the return of an investment

Algorithm	max portfolio value	min portfolio value	net profit	sharpe ratio	sortino ratio	Daily VAR
a2c	47,42,000	4,97,000	6,36,070	0.375	0.534	-0.05
ppo	76,01,000	3,61,000	8,62,815	-0.73	-0.97	-0.05
sac	22,80,000	9,64,000	6,90,000	-0.224	-0.3	-0.005
ddpg	67,55,172	3,23,761	6,57,320	-0.73	-0.97	-0.06
td3	30,51,000	7,50,000	-2,19,000	0.09	0.13	-0.05

Figure 1: Backtesting done on timeframe 01-January-2018 to 31-December-2022

Algorithm	max portfolio value	min portfolio value	net profit	sharpe ratio	sortino ratio	Daily VAR
a2c	12,90,847	9,76,275	2,82,000	1.08	1.81	-0.03
ppo	12,29,000	10,00,000	12,29,000	1.13	1.81	-0.03
sac	10,80,000	10,00,000	68,000	1.68	2.94	-0.006
ddpg	14,27,353	9,42,432	4,27,353	1.46	2.54	-0.04
td3	13,43,181	9,61,000	3,43,181	0.92	1.48	-0.03

Figure 2: Backtesting done on timeframe 01-Jan-2023 to 30-June-2023

relative to its downside risk. The Sortino Ratio is equal to the downside deviation divided by the difference between the expected return and the risk-free rate.

8 Results and Discussion

We trained our CryptoBot on the timeframe 01-January-2018 to 31-December-2022 with time interval taken as 1day. We checked the performance of our model in various timstamps to check for it's robustness and adaptability to perform well on unseen data. In figure 1 we validated our model for training dataset and can be observed that it is not overfitted. We also observed that sac(Soft Actor Critic) algorithm performed best out of three. Further we back tested our model on two other timestamps. First on 16 months data from 1 January 2021 to 1st April 2022 which had highest volatility in bitcoin market history. We observed that our agent was able to generate positive return during this high volatile time period. The same can be verified in Figure 3.

Our last backtest was performed on the latest data with period 01-January-2023 to 30-June-2023 (6 months). We can observe the results from Figure 2 that our trained model generated interested returns and also managed the risk very well.

Algorithm	max portfolio value	min portfolio value	net profit	sharpe ratio	sortino ratio	Daily VAR
a2c	15,57,161	9,26,461	3,05,461	0.61	0.92	-0.04
ppo	19,34,000	8,53,000	3,58,000	0.57	0.86	-0.07
sac	12,86,000	6,25,101	-2,32,000	-0.114	-0.15	-0.05
ddpg	19,08,151	8,51,557	3,45,602	0.56	0.84	-0.07
td3	15,74,155	8,37,000	2,77,000	0.511	0.75	-0.06

Figure 3: Backtesting done on timeframe 01-January-2021 to 01-April-2022

9 Conclusions

An optimal automated trading strategy should aim for risk-management, maximised profit and efficient in decision making and be able to handle the volatility in the market. In our experiments we build a agent which was able to maintain a consistent positive portfolio and better risk

adjusted ratio. We tested several algorithm to find the best policy agent and used several methods to tune the hyperparameter. We also tried ensembling methods to combine the strength of each algorithm into one. In future we wish to explore various techniques to make our agent even more robust so that it can handle the market volatility efficiently.