Portfolio Management of Cryptocurrencies using Reinforcement Learning

Qinwen Wang
qinwenw@mit.edu

Rachel Duan
rduan@mit.edu

Yen Hann Yoo
yyoo@mit.edu

1 Introduction

In the complex landscape of financial markets, the ratio of noise to signal is often low, making it challenging for traders and investors to discern the best strategies for optimizing their portfolios. With the ever-changing dynamics of market conditions, even the advantage of hindsight does not always reveal the most effective trading policies. In this context, Reinforcement Learning emerges as a powerful tool to learn optimal trading policies without requiring an explicit model to predict price movements.

In this project, we explore the use of Reinforcement Learning to discover hidden patterns for trading cryptocurrencies and devise trading policies. We designed customized state space to allow the agent to consider different aspect of the market and stock before trading; different action space to let agent only consider long or both long and short. By refining the state and action space, the goal is to enhance the agent's ability to adjust to the market fluctuation, not only to mitigate the effects of overfitting, but also to increase the generalizability of the RL model to different market scenarios. Then the agent is trained using A2C, PPO, and DDPG algorithms after parameter tuning. The performance is measured by portfolio return, and is compare against two baselines strategies based on mean reversion and momentum.

2 Related Works

In recent years, Reinforcement Learning has been increasingly applied to the field of financial portfolio management. Especially for cryptocurrencies, being decentralized, digital, and secure, have become attractive investment assets. However, their high volatility also makes investment risky. RL's ability to handle dynamic and uncertain environments makes it a promising tool for cryptocurrency portfolio management.

Jiang and Liang (2017) [1] proposed an RL-based portfolio management strategy specifically for cryptocurrencies. They leveraged Deep Deterministic Policy Gradient (DDPG), a type of actor-critic RL algorithm, to learn and make investment decisions. This project follows some of the state and action space design in the paper written by Yang et. all (2020) [3], where they incorporate market and stock information in the state space and use a vector of actions over different stocks to represent selling, buying, or holding for each token. They also incorporated transaction costs and other stock trading constraints, which made their approach more realistic.

Despite the advancements, the highly unpredictable nature of the cryptocurrency market makes it challenging for RL algorithms to learn stable policies. In this project, we focus on the refinement of state space and action space to achieve more robust and resilient investment strategies.

6.8200 Computational Sensorimotor Learning Final Project (Spring 2023).

3 Problem Formulation

3.1 Data

The data [1] contains 16 Cryptocurrencies in total, and consists of High price, Low price, Close price, Volume, Open time, Close time, and Ticker symbol for each stock. The data is recorded at 5-minute intervals and spans January 1st, 2022 12am to March 21, 2023 10am. We use the first 70% for training (), the following 15% for validation (selecting the best agent for testing), and the final 15% for testing. The primary feature considered for our agent is the close price alongside 21 other technical indicators relating to periodic returns, moving averages, volatility, and more, all of which are outlined in the following section.

3.2 Formulation

- State $s_t \in \mathbf{R}^{D \times N}$, where D denotes the number of tokens we consider, and N denotes the number of features for each token we choose to include.
- Action $\boldsymbol{a}$: The action space $a_t \in \mathbf{R}^D$ is a vector of floats over D cryptocurrencies.
- Weight $\boldsymbol{w}$: The weight represent the percentage of our portfolio value assigned to each token, which is calculated from Action $\boldsymbol{a}$ transformation. A positive weight denotes a long position (buying a security, in expectation that the asset will rise in value) and a negative weight denotes short (selling a security that you do not own, in expectation that the asset will decrease in value, allowing you to buy it back in the future at a lower price for profit). The change in weights from one timestep to the other $w_t - w_{t-1}$ represents the change in portfolio composition.
- Reward $r(s_t, a, s_{t+1})$: The direct reward of taking action a at state s_t and arriving at the new state s_{t+1}.
- Policy $\pi(s)$: The trading strategy at state s, which outputs the D-dimensional action vector of continuous values.

3.3 Environments

We created two Gym environments that takes the historical cryptocurrency data alongside several technical indicators as inputs and simulates the trading process using this data. These two environments are outlined in subsequent subsections below. The agent makes trading decisions based on this historical data and interacts with the environment. The goal of the agent is to learn a trading strategy that maximizes the total rewards.

3.3.1 Environment 1

The first environment is long only. A cash asset was included here such that a sell would be represented by greater reallocation to the cash asset. Therefore, there were $D = 17$ tokens used in the first environment.

- State space $s_t \in \mathbf{R}^{D \times N}$: For each cryptocurrency token, the state space includes its current close price p_t and the periodic returns on the close prices over the last hour (twelve 5-minute intervals). None of the engineered features described in 3.1 were used in this initial environment.
- Action $\boldsymbol{a}$: The action space $a_t \in [0, 1]^D$ is a vector of floats over D cryptocurrencies.
- Weights: The environment applies softmax normalization on the actions to get weights $w_t \in \mathbb{R}^D$, which will add up to 1. In other words, this environment only allows long since the weights have to be non-negative.

$$w_t = \frac{\exp(a_t)}{\sum_{i=1}^{D} \exp(a_t^i)}$$

[1] Data Source: Binance API. `https://binance-docs.github.io/apidocs/spot/en/#kline-candlestick-data`

- Rewards: $r(\cdot)$ is defined as the new portfolio value when action a_t is taken at state s_t and arriving at new state s_{t+1}.

$$r(s_t, w_t, s_{t+1}) = b_{t+1} = b_t \cdot \left(\frac{p_{t+1}}{p_t}\right)^T \cdot w_t$$

where b_t denotes the portfolio value at timestep t.

3.3.2 Environment 2

The second environment allows for both long and short positions. As the strategy is inherently market neural, we assume the agent should fully invest all cash at hand, and the cash asset token was no longer required.

- State space $s_t \in \mathbf{R}^{D \times N}$. For each cryptocurrency token, the state space includes its current close price p_t alongside 21 engineered features described in 3.1. These included: EMA_{12}, $MACD$, RSI_{14}, RSI_{14} relative to the market, periodic returns over periods [1, 2, 6, 12, 24], periodic returns (relative to market mean per token) over periods [1, 2, 6, 12, 24], rolling volatility for periods [12, 24], and trading volumes for periods [1, 2, 6, 12, 24].
- Action $\boldsymbol{a}$: The action space $a_t \in [-1, 1]^D$ is a vector of floats over D cryptocurrencies. Though action is normalized afterwards and weights are applied to calculate rewards, directly allowing negative values in the action space may give the distribution more clarity and interpretability.
- Weights: To transform the outputted actions into the weights, the environment first centers actions such that they have a mean of 0, with positive values representing long positions and negative values representing short positions. These are then normalized by dividing them by their absolute sum. That is, for token i at time t,

$$w_{i,t} = \frac{a_{i,t} - \bar{a}_t}{\sum_{i=1}^{16} |a_{i,t} - \bar{a}_t|}$$

  This ensures that the sum of the absolute weights of all positions (both long and short) is equal to 0. This means, we will always put half our portfolio value in long positions and the other half in short positions, and thus the portfolio will always be market neural and have the ability to generate profit in rising and falling markets. At each time period, the token i with the highest $a_{i,t}$ will be bought the most, and the token i' with the lowest $a_{i',t}$ will be shorted the most, though the exact proportion of long and short will depend on the entire action vector.
- Rewards: In our previous approach, the agent was unable to distinguish incremental profit and loss at each timestep. Therefore in this environment we redefine reward $r(\cdot)$ as the percentage change of the portfolio value when action a_t is taken at state s_t and arriving at new state s_{t+1}.

$$r(s_t, w_t, s_{t+1}) = \left(\frac{p_{t+1} - p_t}{p_t}\right)^T \cdot w_{t+1}$$

3.4 Feature Engineering

Engineering features to enable the agent better navigate the cryptocurrency market was imperative. The technical indicators included in the state space are as follows:

- *Close Price*

  The close price of each token at the end of each 5 minute interval, normalized across the different tokens by dividing the price of each token each period by the closing price of the first period (January 1st, 2022 12am).
- *Exponential Moving Average*, EMA_{12}

  A moving average that gives more weight to recent price data compared to older price data. In cryptocurrency trading, EMA_{12} can be used to identify trends in the price of a

particular cryptocurrency, with a bullish trend indicated by a rising EMA_{12} and a bearish trend indicated by a falling EMA_{12}.

$$EMA_{12}(t) = P_t \times \frac{2}{12+1} + EMA_{12}(t-1) \times (1 - \frac{2}{12+1})$$

- *Moving Average Convergence Divergence*, MACD

  A momentum indicator measuring the relationship between two moving averages of different periods. The MACD is calculated by subtracting the 26-period Exponential Moving Average (EMA) from the 12-period EMA. In cryptocurrency trading, the MACD can be used to identify trend reversals, as well as to generate buy and sell signals based on crossovers of the MACD line and the signal line.

$$MACD = EMA_{12} - EMA_{26}$$

- *Relative Strength Index*, $\text{RSI}^{[14]}$

  A momentum oscillator that measures the speed and change of price movements. In cryptocurrency trading, the RSI_{14} can be used to identify overbought and oversold conditions, with an RSI above 0.7 indicating overbought conditions and an RSI below 0.3 indicating oversold conditions.

$$\text{RSI}_t^{[14]} = 1 - \frac{1}{1 + \frac{\text{Average Gain}_t^{[14]}}{\text{Average Loss}_t^{[14]}}}$$

  where $\text{Average Gain}_0^{[14]}$: Average price change for an asset over the last 14 periods when the price increased and $\text{Average Loss}_0^{[14]}$: Average price change for an asset over the last 14 periods when the price decreased. Then subsequently,

$$\text{Average Gain}_t^{[14]} = \text{Average Gain}_{t-1} \times 13 + \max[r_t, 0]$$
$$\text{Average Loss}_t^{[14]} = \text{Average Loss}_{t-1} \times 13 + \max[-r_t, 0]$$

- *Periodic Returns*

  A measure of the percentage change in the close prices of a particular cryptocurrency over a specific period of time. In cryptocurrency trading, periodic returns can be used to calculate the volatility of a particular cryptocurrency, as well as to compare the performance of different cryptocurrencies. Note that n denotes the number of periods over which to calculate the periodic returns. For example, if one wishes to calculate the periodic returns over 2 periods, then n = 2.

$$r_t = \frac{p_t - p_{t-n}}{p_{t-n}}$$

- *Rolling Volatility*

  A measure of the amount of price fluctuation of a cryptocurrency over a given period of time. In cryptocurrency trading, rolling volatility is often used as an indicator of risk, with higher volatility indicating a greater potential for large price swings. To calculate rolling volatility, the standard deviation of the cryptocurrency prices over a specified window of time is computed. In this problem, we used a period of 12 and 24.

$$\text{Volatility}_t = \sqrt{\frac{1}{n-1} \sum_{i=t-n+1}^{t} (p_i - \bar{p})^2}$$

- *Volume*

  A measure of the total number of units of a particular cryptocurrency that have been traded during a specific period of time. In cryptocurrency trading, the trading volume can be used to assess the liquidity of a particular cryptocurrency, as well as to identify trends in the price of a particular cryptocurrency based on changes in trading volume. We normalized the volume by dividing each datapoint by the average of the first 26 periods.

3.5 Learning Algorithms

For our problem, given that we have a continuous action space, we utilised the following algorithms (as implemented in the Stable Baselines3 library [2]) to train our agent due in part to their ability to handle continuous action spaces:

- *Advantage-Actor Critic (A2C)*
  The actor learns a policy to select the best action, while the critic estimates the value function to guide the actor's learning. The actor-critic architecture allows for more stable learning, as the critic helps to reduce the variance in policy updates.
- *Proximal Policy Optimization (PPO)*
  PPO, which combines TRPO's sample efficiency and the simplicity of vanilla policy gradient methods, is particularly suited for cryptocurrency trading. Its policy constraint mechanism limits drastic policy updates, preventing large losses due to erratic changes. Given its success across various domains coupled with its inherent stability and efficiency, PPO shows potential for effective cryptocurrency portfolio management, balancing risk and returns.
- *Deep Deterministic Policy Gradients (DDPG)*
  Using neural networks to predict market movements is commonplace in finance, and DDPG utilizes such neural networks to learn Q-value function, which is analogous to predicting the price movement in the next time period similar to supervised learning approaches. It would be interesting to see how utilizing a neural network to learn the market movement conditional upon states would compare to model-less methods.

In cryptocurrency markets, small trades typically have negligible impact on market states - states are not significantly altered by our actions. This necessitates consideration of future action-dependent rewards. The selection of an optimal discount factor (γ) is a key challenge, aiming to balance immediate and future rewards. High γ values risk overfitting to the training set, as the agent may learn sequences of actions that don't generalize well to unseen datasets. Conversely, low γ values can induce myopic behaviour, focusing too much on immediate rewards and failing to optimize for long-term gains. To strike the right balance, we experimented with different γ values (0.99, 0.5, and 0.1) in our learning algorithms.

3.6 Training

For each agent, we trained for 30 episodes, where each episode is 10000 timesteps (50000 minutes / approx. 35 days). For each episode picked a random point of time from the training set (Jan 1 2022 to Nov 8 2022) to begin with, as to avoid biasing the training by starting the agent off in a particular kind of market.

The agents trained on these 3 algorithms will then be compared based on its performance on the validation set (Nov 8 2022 to Jan 13 2023). We measured the performance by each agent's Sharpe ratio on the validation set, calculated as follows:

$$\text{Sharpe}_{agent} = \sqrt{365} \cdot \frac{\text{Mean daily return}_{agent}}{\text{Standard deviation of daily return}_{agent}}$$

The agent that exhibits the highest Sharpe ratio will be selected as the final candidate for evaluation on the test set, spanning the period from January 13, 2023, to March 21, 2023.

3.7 Baseline

Two baseline methods are used for comparison - mean reversion strategy and momentum strategy. Both baselines aim to maximize the Sharpe ratio.

In both our baseline strategies we use moving average price as signals, where the moving average price for m periods at time t for token i is defined as

$$MA_{i,t}^{[m]} = \frac{1}{m} \sum_{\tau=0}^{m} p_{i,t-\tau}$$

Our mean reversion strategy operates on the principle that asset prices and returns will eventually revert to their mean or average, thus it involves buying under performing assets (those below their historical mean) and selling over performing ones (those above their historical mean). The particular strategy we utilize uses moving average price difference $\text{Signal}_{i,t} = MA^{[25]}_{i,t} - MA^{[10]_{i,t}}$ as signal, where $MA^{[10]}$ represents short term trend and $MA^{[25]}$ represents long term mean trend, and if short term trend dips below long term mean, we will buy that token in expectation of mean reversion, and vice versa. Specifically, the amount of token i we long / short in time t for the mean reversion strategy is

$$w^{MR}_{i,t} = \frac{\text{Signal}_{i,t} - \overline{\text{Signal}}_t}{\sum_{i=1}^{16} |\text{Signal}_{i,t} - \overline{\text{Signal}}_t|}$$

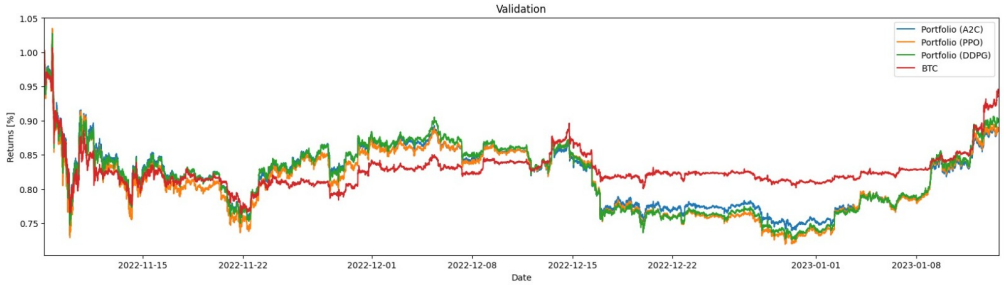
The momentum strategy is based on the observation that price trends, whether increasing or decreasing, tend to persist over time. This strategy involves buying assets that have recently risen in price and selling those that have recently fallen compared to the market. For our baseline, if short term trend dips below long term mean, we will short that token in expectation of continuing momentum. We will use the opposite of the mean reversion strategy, in particular:

$$w^{MM}_{i,t} = -w^{MM}_{i,t} = \frac{\overline{\text{Signal}}_t - \text{Signal}_{i,t}}{\sum_{i=1}^{16} |\overline{\text{Signal}}_t - \text{Signal}_{i,t}|}$$

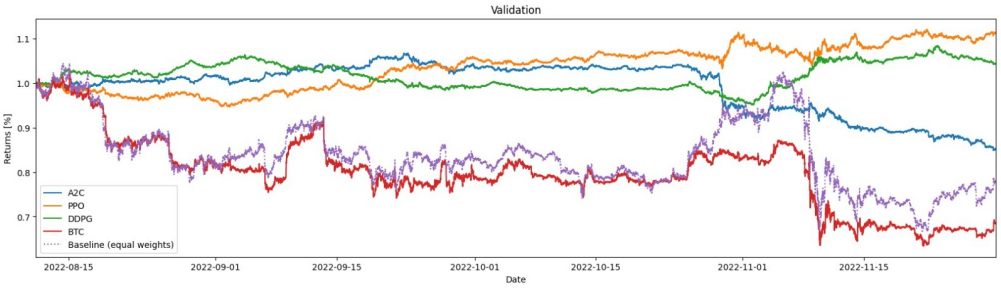
We will evaluate our agent against these two traditional trading strategies through comparing annualized Sharpe ratios on the test set.

4 Results

4.1 Comparison of Environments 1 and 2



(a) Environment 1



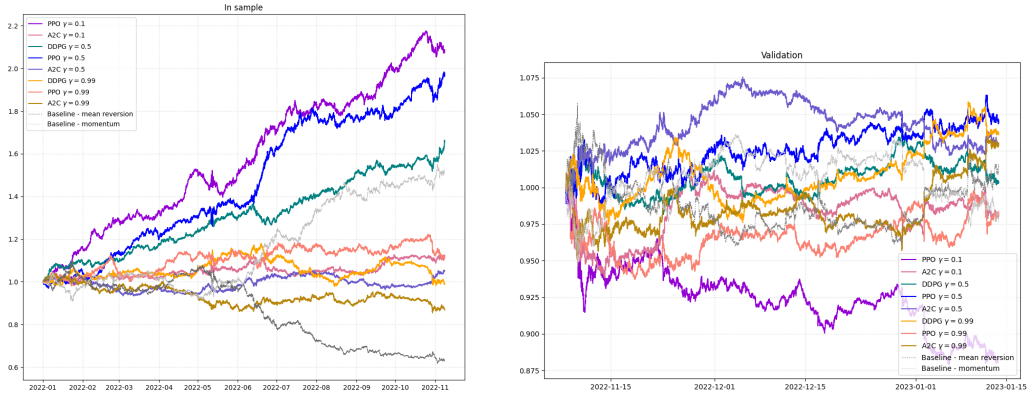
(b) Environment 2

Figure 1: Out-of sample performance comparison in different environments

According to Figure 1, the agents trained in Environment 2 outperforms that of Environment 1 from pure comparison of A2C, PPO, and DDPG between environments - the blue, orange, and green curves respectively.

In a long-only setting, the primary means of generating returns relies on the appreciation of assets. However, our training dataset primarily covers the year 2022, during which the cryptocurrency market experienced significant declines. In such circumstances, a long-only strategy's optimal approach would involve holding cash during periods of market downturn and selectively investing only during periods of market upswing. However, due to the softmax transformation applied to actions, wherein the weight assigned to each token is influenced by the entire array of actions, the agent struggles to learn the precise allocation of holding cash exclusively during specific periods while allocating weights to various tokens during other periods. Consequently, it appears that the agents allocate some weight to nearly every token in almost every period. In the absence of a more suitable method for directly imposing constraints on the action vector summing up to 1, we propose that the long and short strategy provided by Environment 2, which allows profits to be made even with a partial positions in each token, may offer a more appropriate solution within the given problem framework. We will use environment 2 for all ongoing exploration.

4.2 Training and Validation Results



(a) In-sample (b) Out-of-sample

Figure 2: In-sample and out-of-sample performance for Environment 2 (long & short)

After training the agent, we evaluate its performance trading on the entire training set and validation set. Comparing across models, PPO gave the best results both in terms of absolute profit and Sharpe ratio. In finance, a Sharpe ratio of above 2 is usually considered as good. As the reward for our agent is the profit / loss of period, the agent has successfully optimized its strategy to avoid large losses and capture profits in a stable manner. For most agents the portfolio growth is relatively steady, without experiencing large downward shocks. The agent has managed to limit volatility and reduce standard deviation in rewards despite not explicitly instructed to do so.

The DDPG strategy is the most computationally intensive and still showed signs of further improvement when our 30 training episodes ended. Given more time, perhaps the DDPG agent could have learned more from training and outperformed PPO.

Validation Sharpe Ratio	A2C	PPO	DDPG
γ**=0.1**	-0.64	-3.44	
γ**=0.5**	1.80	**2.30**	0.47
γ**=0.99**	1.08	-0.51	1.68

4.3 Test Set Evaluation

The PPO strategy with a discount factor of $\gamma = 0.5$, achieved a Sharpe ratio of 1.63. While this ratio is notably lower than that observed during the training and validation periods, it still outperforms both baseline strategies. The decrease in Sharpe ratio could potentially be attributed to overfitting to the training period, with the validation period exhibiting more similarities to the training period than the testing period. To address this issue, gathering additional data and training

the model over a longer time frame encompassing more diverse market conditions could prove beneficial. Another potential factor contributing to the lower Sharpe ratio is the phenomenon known as "alpha decay," wherein markets evolve over time due to participants employing similar optimized strategies. Consequently, strategies that generated profits over a specific period may gradually lose their effectiveness. Nevertheless, the strategy's relatively stable portfolio value growth and the achievement of a Sharpe ratio of 1.63 demonstrate its success in delivering consistent profits in turbulent market environments.

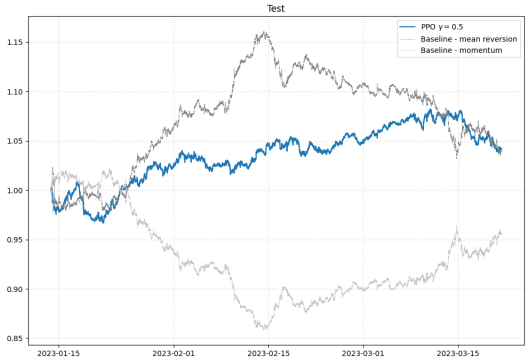


Figure 3: Performance of the best agent on test set

	Sharpe Ratio
PPO(γ=.5)	**1.63**
Mean Reversion	1.46
Momentum	-1.46

Figure 4: Sharpe ratio comparison

5 Conclusion

In conclusion, this project underscores the importance of refining the state space representation to capture the complex dynamics of the market, as evidenced by the comparisons between Environments 1 and 2. Furthermore, hyperparameter tuning demonstrated that the choice of γ significantly influences the agent's focus on immediate versus future rewards, thereby shaping its trading strategy and overall performance. Nonetheless, there are various areas to explore further including but not limited to:

- *Transaction Costs*
  Incorporating transaction costs into the trading environment can make the simulated trading environment more realistic. Costs such as brokerage fees, slippage, and spread can significantly impact the profitability of a trading strategy. Moreover, adding transaction costs may reduce the trading frequency as each transaction incurs a cost, and frequent trading can lead to these costs accumulating rapidly, which can significantly diminish the net returns of the strategy.

- *Multi-Agent Systems*
  Real-world financial markets are multi-agent environments. Future studies could explore multi-agent reinforcement learning frameworks where multiple agents interact with each other, influencing market dynamics and learning to adapt to other agents' strategies.

- *Alternative RL Algorithms*
  Algorithms such as Twin Delayed DDPG (TD3), Soft Actor-Critic (SAC), or distributional RL algorithms could potentially improve upon the results obtained in this study. However, they were not investigated here due to constraints pertaining to computational resources.

References

[1] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem, 2017.

[2] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[3] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. 2020.

A Appendix

A.1 Figure

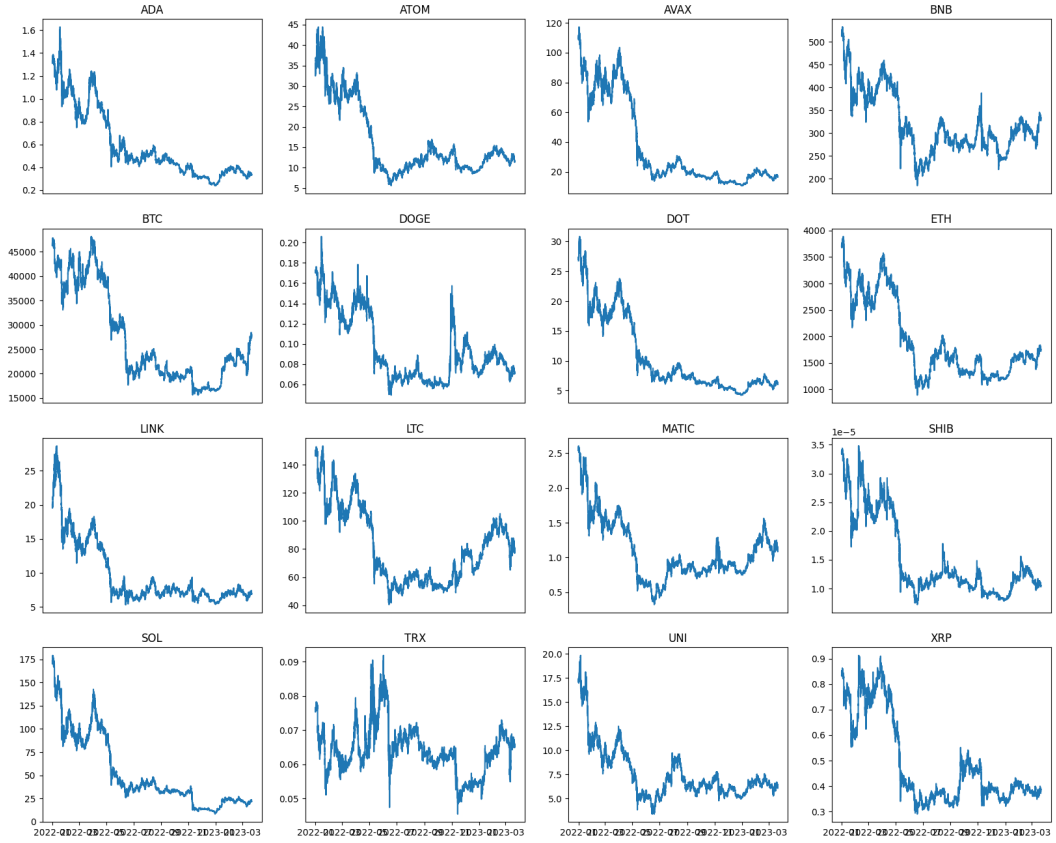


Figure 5: Trend for 16 cryptocurrency tokens