UNSURE - A Machine Learning Approach to Cryptocurrency Trading

Vasileios Kochliaridis[1*], Anastasia Papadopoulou[1] and Ioannis Vlachavas[1]

[1*]School of Informatics, Aristotle University of Thessaloniki, Thessaloniki, 54124, Greece.

*Corresponding author(s). E-mail(s): vkochlia@csd.auth.gr;

Abstract

Although cryptocurrency trading can be highly profitable, it carries significant risks due to extreme price fluctuations and high degree of market noise. To increase profits and minimize risks, traders typically use various forecasting methods, such as technical analysis and Machine Learning (ML), but developing effective trading strategies in noisy markets still remains a challenging task. Recently, Deep Reinforcement Learning (DRL) agents have achieved high performance on challenging tasks, including algorithmic trading, however it requires significant amount of time and high-quality data to train effectively. Additionally, DRL agents lack explainability, making them a less popular option for traders. The purpose of this paper is to address these challenges by proposing a reliable trading framework. Our framework, named UNSURE, generates high-quality features from candlestick data using technical analysis along with a novel parameterization method, and then exploits high price fluctuations by combining three ML components: A) Unsupervised component, which further improves feature quality by clustering market data; B) DRL component, which is responsible for training agents that open Buy or Short positions; C) Supervised component, which estimates price fluctuations in order to open and close positions efficiently, while reducing trading uncertainty. We demonstrate the effectiveness of this approach on nine cryptocurrency markets using several risk-adjusted performance metrics.

Keywords: Deep Reinforcement Learning, Machine Learning, Cryptocurrency Trading, Time-Series Clustering, Unsupervised Learning, Supervised Learning

1 Introduction

Cryptocurrency is a high-risk-high-reward trading activity, offering the potential of high profits in a short time, but with high risk, due to high price fluctuations in markets. To enhance profit potential, traders often rely on forecasting methods, such as Technical analysis and Machine Learning (ML) methods (Fang et al, 2022).

Technical analysis provides mathematical formulas, named technical indicators, which are applied on historical market data in order to identify patterns and make trading decisions. Technical indicators have become quite popular trading tools, due to their simplicity in calculation; most of these indicators require a single parameter, which is the selection of a window that determines how they analyze historical data. This simplicity provides a straightforward strategy for many traders, which makes technical indicators a very attractive strategy tool. However, they often generate false trend signals (Lin, 2012), so traders often combine multiple technical indicators to develop profitable trading strategies. Nevertheless, it still remains unknown which set of technical indicators are effective in each specific trading period, as well as what window should be selected for each indicator.

On the other hand, ML methods attempt to both identify and generalize patterns from historical market data, in order to predict upcoming market trends. ML models can be utilized to forecast upcoming trends and price volatility with satisfying accuracy, enabling traders to develop trading strategies. Many works have also combined ML models with technical analysis, which improved their overall performance (Huang et al, 2019).

Some well-studied ML models that achieve satisfying performance on time-series forecasting tasks, including price and trend estimation tasks, are Deep Learning models, such as Convolutional Neural Networks (CNNs) and Long-Short-Term-Memory Networks (LSTMs) (Nazareth and Reddy, 2023), (Pierros and Vlahavas, 2022). Bai et al. (Bai et al, 2018) has also designed a novel Deep Learning architecture, named Temporal Convolutional Networks (TCNs), which outperformed LSTMs and CNNs in several time-series forecasting tasks, including price forecasting (Zhang et al, 2022). Finally, Transformer-based architectures have also been recently used in Time-Series forecasting tasks (Son et al, 2022), which require more training time and computational resources, but have the potential to outperform several Neural Network architectures in several forecasting tasks.

Despite the benefits of Deep Learning methods, financial markets are complex dynamic systems which are influenced by high randomness, market noise and unexpected trend shifts. As a result, even with the use of advanced forecasting models, developing efficient trading strategies based on price and volatility estimation only remains a challenging task (Hirchoua et al, 2021). Moreover, these methods require proper feature selection, which can be a quite difficult task in stock and cryptocurrency trading, due to the large volume of possible technical indicators with various window sizes that can be used as potential features.

One specific sub-field of ML that has also recently gained quite attention for its potential in solving complicated tasks, including automated trading, is Deep Reinforcement Learning (DRL) (Fang et al, 2022). DRL agents can be utilized to adjust trading volume, as well as identify optimal entry and exit periods in order to increase profits.

Although DRL-based agents have demonstrated impressive performance in trading environments, they require large amounts of high-quality data and enough time to be fully scaled. Additionally, such methods generate strategies as black-boxes, making it very hard to interpet their actions. This is a major drawback of DRL-based trading agents, as (a) it can result in poorly developed strategies (b) include a lot of trading uncertainty and risk during the trading periods. All these limitations have raised concerns about the accountability of DRL methods in algorithmic trading (Heuillet et al, 2021).

Distributed Deep Reinforcement Learning (DDRL) frameworks have been developed to address some of the drawbacks associated with traditional DRL. DDRL employs multiple agents that collect samples, communicate with each other and learn in parallel, which usually results in faster scaling, improved training and better policy generalization (Lazaridis et al, 2020). As we later demonstrate in this paper, this is a key-component for learning robust trading strategies, especially in markets with high amount of noise and frequent price volatility. Despite the benefits, there is limited literature about the use of DDRL approaches in trading-related tasks.

The purpose of this paper is to propose an end-to-end trading framework named UNSURE (UNsupervised-SUpervised-REinforcement Learning), which combines technical analysis with ML techniques from three broad categories, in order to address the aforementioned issues of ML and DRL-based trading systems, while achieving robustness and high performance in cryptocurrency trading . First, UNSURE applies technical analysis on candlestick data[1], combined with a novel parameterization method, in order to construct high-quality features, while also reducing the required feature selection and training time. Then, UNSURE exploits high volatility trading periods to make profit by employing three ML components as described below:

- Unsupervised Component: Applies clustering on market data, in order to combines feature from markets with correlated price returns, increasing the amount of training data.
- Supervised Component: Trains a TCN model to estimate the maximum price volatility within a short time horizon, enabling the framework to close positions at a target price using limit orders.
- Reinforcement Learning Component: Utilizes DDRL agents that learn to open Buy and Short[2] positions in a cryptocurrency market environment, with the use of market orders.

Finally, a novel safety mechanism, named *VPM*, is also proposed that is employed during the trading period. VPM combines the predictions generated by the Supervised Component, in order to guide the agent into generating actions exclusively during periods with high profit potential. VPM mechanism allows UNSURE to further reduce the trading risks, while increasing the overall reliability and explainability of the generated decisions.

The remainder of this paper is organized into the following sections. Section 2 presents the related work of this research. Section 3 presents all the necessary background that is required for this work. Section 4 covers thoroughly the methodology

[1]Candlestick data describe the price movements using Open, High, Low and Close price and Volume.
[2]Shorting a position involves borrowing an amount of shares and selling it, with the expectation that its price will decrease.

that was followed in this work, which describes the parameterization method, the three ML components in greater detail and the safety mechanisms. Section 5 presents the experimental setup, the metrics and the results of the experiments. Finally, Section 6 discusses the insights of this work and Section 7 concludes this work and proposes future extensions.

2 Related Work

Jiang and Liang (2017) developed a trading framework for cryptocurrency markets, by combining several Deep Learning architectures, including CNNs and LSTMs with Reinforcement Learning algorithms. More specifically, they employed a relatively simple DRL algorithm, named Deep Deterministic Policy Gradient (DDPG) Lazaridis et al (2020) in order to generate budget-allocation actions for multiple cryptocurrencies. Additionally, the authors integrated LSTM layers into the policy network, followed by CNNs and a fully connected architecture that generates the actions, in order to capture more complicated patterns that appeared in the markets, and thus generate better actions. Even though their approach outperformed several popular baselines strategies in many cryptoccurency markets, they completely disregarded the use of technical analysis features and more advanced DRL algorithms.

Pendharkar and Cusatis (2018) constructed a two-asset management methodology, in order to trade financial indices, such as S&P 500 and AGG, by using several Reinforcement Learning agents, with the $td(\lambda)$-based agent having the best performance. Although their approach achieved very promising results and managed to beat several other baseline strategies, such as single asset trading, it was only evaluated on stock market indices, which present low volatility and risk, in comparison to cryptocurrency assets.

Bu and Cho (2018) attempted to train agents in Bitcoin and seven altcoin markets by combining a Long-term Recurrent Convoluitonal Network (LRCN) architecture with Deep Q-Networks (DQN). Their methodology yieled higher profits, in comparison to previous trading strategies, but with significantly higher trading risk. This drawback could be very problematic in cryptocurrency markets, due to high volatility of the assets, increasing the potential of high losses. One possible solution to this problem that the authors completely disregarded is to group assets based on their volatility, which effectively reduces trading risk and improves the trading strategy of the agents, as we later present in our work.

Huang et al (2019) investigated price changes predictability of Bitcoin using technical analysis. More specifically, they constructed a tree-based model for price-change estimation, which was trained on large set of technical indicators. Their study proved that it is possible to estimate Bitcoin price changes with fairly good accuracy by combining ML algorithms with a large set of technical indicators, even though not all indicators had a significant impact in the model's accuracy. This drawback might be attributed to several factors: firstly, the authors did not select appropriate window sizes for each indicator, which could be crucial given the large number of indicators involved. Secondly, there's a possibility that some of the selected indicators had a minimal impact on market dynamics.

In the study conducted by Chandar (2022), they introduced an innovative approach to predict trend shifts for a trading system by employing Gramian Angular Fields (GAF), a method which is used for time-series to image encoding, combined with a CNN. To further increase the performance of their model, the authors included various well-established technical indicators along with the candlesticks, such as the Exponential Moving Average (EMA), Moving Average Convergence-Divergence (MACD), and the Relative Strength Index (RSI). Although their approach demonstrated promising results, particularly in terms of F1-Score and overall accuracy, there are notable limitations in the study that should have been taken into consideration. One such limitation is the lack of backtesting, which is crucial for evaluating a model's strategy in historical data. Another thing to consider is that the authors did not compare their approach with a more advanced model, such as TCN, which is a CNN-based architecture, specifically designed for time-series data.

On the other hand, Zhang et al (2022) utilized a TCN model to forecast stock volatility and Value-at-Risk, which are crucial metrics in risk management and portfolio investment applications. In their work, they compared TCN with several volatility forecasting models, including LSTM and GRU architectures, in which TCN outperformed other Deep Learning architectures in terms of both Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Although their approach achieved low estimation error, they did not emphasize in volatility overestimation errors, which can potentially lead to the development of catastrophic trading strategies as explained later on Section 4.

Recently, Schnaubelt (2022) trained DRL agents using PPO algorithm Lazaridis et al (2020) in order to optimize execution at cryptocurrency exchanges by learning to place limit orders in an efficient way. To achieve this, they designed a virtual limit order exchange and a hand-crafted reward function. At each state, the agent chooses a future timestep on which the limit order is placed and the agent is rewarded based on Volume-Weighted Average Execution Price. Their experiments yielded that the agents achieved higher returns when placing limit orders rather than opening market orders. However, their methodology was only tested on two cryptocurrency markets and their agent was operating on timesteps with fixed intervals, while the market was continuously open.

Kochliaridis et al (2023) also utilized a PPO agent to train Integrated TraderNet, which is a trading agent that executes the Round-Trip strategy in an effective manner. The Round-Trip Strategy involves the placement of pairs of opposing orders in sequence (e.g., buy-sell or sell-buy) with the goal of generating profits from extreme price fluctuations in short time. However, the agent was initially found to overestimate market uptrends or downtrends, mainly due to high degree of market noise. To address this issue, the authors implemented the two safety mechanism that examine the agent's action and prevent uncertain actions from being executed. Although this work has shown that trading rules can be adjusted to DRL agents to improve their reliability in trading tasks, it completely disregards several mechanisms, such as volatility estimation and price correlation between several cryptocurrency markets, which could further increase trading.

Instead of using traditional DRL algorithms, Boukas et al (2021) proposed a DDRL framework for continuous intraday electricity market trading. More specifically, the authors employed the Buy-Low-Sell-High strategy [3] and trained a DDRL agent, named Ape-X DQN Lazaridis et al (2020), to learn submitting limit orders in continuous intraday markets. Ape-X DQN is the distributed version of the initial proposed DQN with Prioritized Experience Replay (PER), offering improved performance and faster scaling. The results of this paper demonstrated the effectiveness of DDRL approaches in complicated trading environments with high market noise, such as electricity trading environments. However, there is limited literature about the applications of DDRL algorithms in other trading markets as well, such as cryptocurrency markets, which is also a quite challenging task.

To conclude our literature review, the use of limit orders and the combination of technical analysis with ML methods enables the construction of profitable trading strategies. However, previous research overlooked the quality of the technical indicator features, as well as correlations between cryptocurrency markets, which can be exploited as demonstrated in this paper. Additionally, many cryptocurrency trading-related works employ PPO (e.g. Betancourt and Chen (2021), Schnaubelt (2022), Guarino et al (2022)) which is fast, stable learning algorithm and achieves high performance Lazaridis et al (2020), but it relies on a synchronous architecture for collecting experiences. Such architectures require a lot of training steps to be properly trained. On the other hand, DDRL approaches can achieve much higher experience sampling, due to the asynchronous collection of samples, which can be quite beneficial in trading environments, especially with large volumes of data, and therefore should be further explored in trading tasks. Finally, in contrast to previous trading approaches, UNSURE also employs a supervised learning model alongside with a safety mechanism that tackles overestimation errors and uncertainty, instead of relying solely on the decision of a single agent.

3 Background

This section provides an overview of the key concepts and methodologies that that are employed in this study, which include Distributed Deep Reinforcement Learning (DDRL) algorithms, a well-studied DDRL that was utilized in the Reinforcement Learning component and the TCN model architecture, which is utilized in the Supervised component.

3.1 Distributed Deep Reinforcement Learning (DDRL)

The goal of DDRL frameworks is to extend and improve performance of existing algorithms by scaling them up in shorter time (Lazaridis et al, 2020). DDRL approaches achieves this by employing multiple agents that learn in parallel and communicate with each other, which usually results in faster training, more robust learning and better policy generalization. DDRL frameworks are applicable to both the Value-Based

[3]Buy-Low-Sell-High is the most common strategy, where trader buy an asset when its price is low and sell it when the price is high

and Policy-Based categories of reinforcement learning algorithms. Two very popular algorithms in this framework are Ape-X DQN and Importance Weighted Actor Learner Architecture (IMPALA). The basic architecture of a DDRL framework can be visualized by Figure 1.

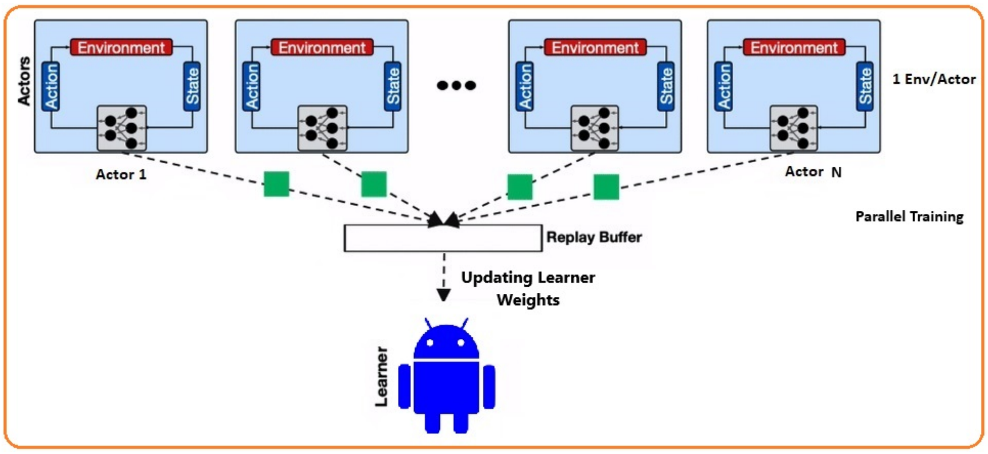

Fig. 1: Visualization of Distributed DRL frameworks with multiple workers

3.2 Importance-Weighted Actor-Critic (IMPALA)

IMPALA (Lazaridis et al, 2020) is an Off-Policy Actor-Critic learning algorithm, which include two policies; The Actor policy and the Critic policy. The Actor policy is responsible for selecting actions from the current state of the environment, while the Critic policy is used to learn the value function, which estimates the expected cumulative return. IMPALA makes use of multiple actors and a centralized learner, which is the Critic component. The actors collect experiences from the environment and send them to the learner, which uses them to update the policy and value functions for each actor separately.

IMPALA does not rely on actors to compute updates themselves, but lets the learner apply updates instead, while the actors keep interacting with the environment with their current policy. Because this technique develops a policy-lag, since the actor's policy is several updates behind the learner's policy, they authors introduce a policy-correction method, named V-Trace. By using V-Trace, the authors correct the state-value function differences caused by the lag, which enables IMPALA architecture to train in parallel and achieve high performance on many environments.

3.3 Temporal Convolutional Network (TCN)

Up until recently, sequence modeling has been mainly associated with RNN architectures such as LSTM and GRU. (Bai et al, 2018) proposed a new architecture named Temporal Convolutional Network. TCN achieves higher performance than RNNs by extending CNNs, which avoids the vanishing gradient problem and provide better memory efficiency. A basic TCN unit consists of dilated convolutions, which apply the kernel to a larger receptive field, where some input values are skipped depending on the dilation rate. Another technique that is employed in TCN units is Causal convolution, which takes into account the order of the input data. This means that for a given input, the output of the convolution at any given time step is only dependent on the input values up to that point in time.

The authors also suggest a few additions to the basic TCN architecture in order to improve its performance. First, they added a residual block, in order to address the vanishing gradient problem that can occur during the training of a Neural Network. The residual block of a TCN layer consists of a number of dilated convolutional layers, followed by a non-linear activation function. The output of convolutional layers is be added to the input of the residual block in order to produce the input for the next block. Second, they added several regularization methods between the hidden layers of each block, such as Weight Normalization or Dropout, as presented in Figure 2.

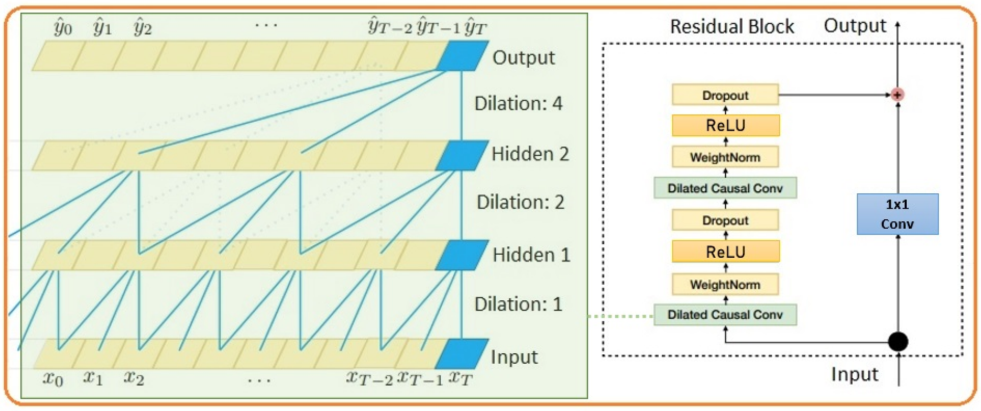

Fig. 2: Visualization of TCN architecture. **Left**: The Dilated Convolution operation of TCN layer. **Right**: Diagram of Residual Blocks combined with Dilated Convolutions. The output of Dilated Convolutions pass through a weight normalization layer, as well as a dropout layer, to reduce over-fitting.

4 Methodology

In this section we describe in detail the technical indicator parameterization method and the three ML components that are proposed in this paper, namely the Unsupervised component, the DRL component, and the TCN component, all of which are integrated into a single trading framework. Finally, we introduce the *VPM* safety mechanism, which reduces trading risks, improves reliability and the overall performance.

4.1 Technical Indicator Parameterization

Technical indicators are computed using sliding windows of past N candlestick bars. Although there are typical values of window sizes that can be used to compute indicator values, these values do not always work well on every market. To the best of our knowledge, there has been limited investigation into choosing proper window for different markets. The most common approach for selecting technical indicators involves employing several fast-to-train regression models (Peng et al (2021), Naik and Mohan (2019)) and choosing the combination of indicators and their respective window sizes that achieve the highest price estimation performance. However, training many regression models for multiple markets and multiple technical indicators with several possible window sizes can be extremely time-consuming. To address this issue, we suggest a parameterization method that enables the fast selection of proper window sizes for each market, resulting in higher-quality features.

To determine parameters of the technical indicators, multiple versions of each indicator are computed by using varying sliding window sizes. Because each version uses different sizes, it captures different patterns, and thus proposes different trading strategies. Then, each version is evaluated using backtesting, i.e. a high-speed method for assesing trading strategies by applying Buy and Sell actions in past historical data of a particular market. Then, the resulting *Sortino Ratio* of each strategy is calculated, which is a well-used and effective risk-adjusted metric that measures the performance of a strategy, as further described in Section 5. Higher Sortino Ratio values indicate higher profitability of the indicator strategy with lower risk, so the window size with the highest Sortino Ratio is selected to generate the indicator values for that particular market. Our parameterization method is further described in Algorithm 1

Algorithm 1 Parameterization Method

```
1:  **Input:** Close prices $c_t$ for $N$ markets, Technical indicator formula $I(n, c_t)$ and
    possible window sizes $N = \{3, 4, 5, ...\}$
2:  **Output:** Selected window size $n$ for indicator $I$
3:  procedure BACKTESTING($prices, actions, T$)
4:      $profitPerStep \leftarrow fill(0, T)$
5:      for $t = 1$ to T - 1 do
6:          $profitPerStep \leftarrow profitPerStep + execute(action[t], prices[t])$
7:      end for
8:      $SOR \leftarrow calculateSortino(profitPerStep)$
9:      return $SOR$
10: end procedure
11: $bestSortino \leftarrow -100000, bestWindowSize \leftarrow 0$
12: for $i = 1$ to length(N) do
13:     $n \leftarrow N[i]$
14:     $actions \leftarrow computeIndicatorStrategy(prices, I, n)$
15:     $SOR \leftarrow Backtesting(prices, actions, length(prices))$
16:     if$SOR > bestSortino$then$bestSortino \leftarrow SOR, bestWindowSize \leftarrow n$
17: end for
```

4.2 Unsupervised Component

Machine learning models, including DRL agents, often require large amounts of training data, in order to learn effective trading strategies. To increase data availability and feature quality, this component employs a clustering method that concatenates features from similar markets, including market data and technical indicators, into a single dataset. This allows the agent to recognize patterns in a market that could possibly affect the prices of the other markets in the same cluster. To group similar markets, a clustering pipeline is constructed as follows:

First, Logarithmic Return transformation, which is defined by Equation 1, is applied on Close prices C_t of each market, which converts the prices into logarithmic percentage returns L_t, and places them on the same scale, so price similarities can become comparable.

$$\hat{L_{t+1}} = \ln \frac{C_{t+1}}{C_t} \tag{1}$$

Second, *Dynamic Time Warping* (DTW) Müller (2007) is used to compute similarities between the sequences of logarithmic returns. DTW addresses any different time ranges and correlation lags between sequences by mapping each point in one sequence to one or more points with the other, which creates an optimal alignment between the two sequences that can be used to calculate a distance measure Müller (2007).

Finally, the *K-Means* algorithm is used to cluster the sequences, based on their distances, as presented in Algorithm 2. The number of clusters n was set to $n = 3$ for all markets, after experimenting with different values for $k \in \{2, 3, 4\}$ for the total nine markets that were used in this work, as shown in figure 3.

Algorithm 2 Clustering Cryptocurrency Markets

```
1: Input: Close prices $c_t$ for $N$ markets, $k$ clusters
2: Output: Cluster of each market
3: procedure ConvertToLogReturns($prices$)
4:     for $t = 1$ to length($prices$) - 1 do
5:         $logReturns[t] \leftarrow \ln\left(\frac{prices[t+1]}{prices[t]}\right)$
6:     end for
7:     return $logReturns$
8: end procedure
9: $logReturns \leftarrow$ ConvertToLogReturns($prices$)
10: for $i = 1$ to N-1 do
11:     for $j = i$ to N do
12:         $distances[i] \leftarrow$ DynamicTimeWarping($logReturns[i], logReturns[j]$)
13:     end for
14: end for
15: $clusters \leftarrow$ KMeansClustering($distances, k$)
```

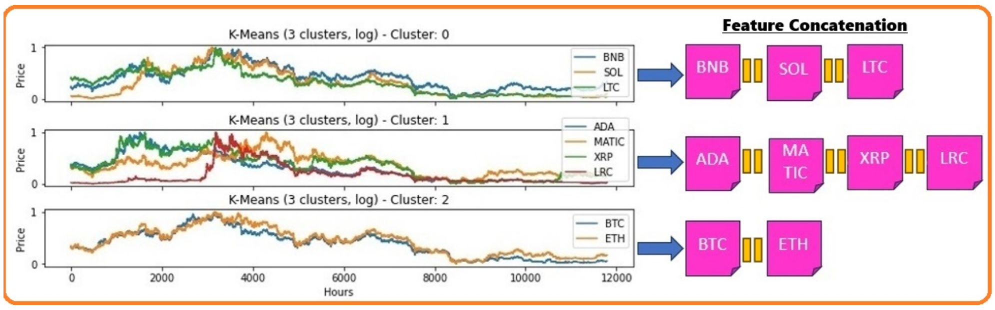

Fig. 3: K-Means Clustering with $n = 3$ clusters on 9 markets. The features from each market of the same cluster are concatenated into a single dataset, resulting in three large datasets.

4.3 DRL Component

The goal of the DRL component is to train a DRL agent that maximizes the cumulative returns by opening Buy and Short positions in an hourly frequency. The above problem can be expressed as a Markov Decision Process (MDP), which is defined as a tuple (S, A, P_a, R_a), where S, A, R_a denote the state space, action space and reward function of the problem respectively, while P_a represents the probability transition function. The objective of the agent is to learn an optimal policy π_θ, which chooses the optimal action a_t in a given state s_t.

Action Space. The action space A is composed of 3 actions: $\{a_0 : BUY, a_1 : SELL, a_2 : HOLD\}$. In each state s_t, the agent can choose to buy or short a limited amount of tokens, denoted as v or holds its position by selecting actions a_0, a_1 or a_2 respectively, with v being a user-defined setting. To simulate real-world trading conditions, the agent starts with a finite initial budget, denoted as b.

State Space. The state space S is composed of market and account observations. The market observations include candlestick data and technical indicator values. In order for DRL agent to capture both short and long-term market trends, timeframes are constructed from market observations by using a sliding window of size T, which is defined by the user. The state space is then defined as a collection of these timeframes, with each timeframe representing the current state of the agent. Additionally, to allow the agent to further understand the market dynamics, the share amount and account balance are also integrated as account observations, which changes according to the agent's actions at each timestep. Thus, a state s_t can be mathematically described as $s_t = (\vec{m}_t, d_t)$ with $\vec{m}_t = (o_t, o_{t-1}, o_{t-2}, ..., o_{t-T})$ and $d_t = (Shares_t, Balance_t)$, where o_i denotes the $i - th$ market observation.

Reward function. A volatility-based reward function is used to train the agents, which was presented by Kohliaridis et al. Kochliaridis et al (2023) and is defined by Equation 2. In this reward function, C_t denotes the Close price on timestep t, $H_{t_{max}}, L_{t_{min}}$ denote the maximum High and minimum Low price within a horizon of $t+K$ timesteps respectively and f the transaction fees constant. K is also a user-defined parameter, which is used to define the predictability range of the agent.

$$r_{t+1} = \left\{ \begin{array}{ll} \ln \frac{H_{t_{max}}}{C_t} + \ln \frac{1-f}{1+f} & a_t = BUY \\ \ln \frac{C_t}{L_{t_{min}}} + \ln \frac{1-f}{1+f} & a_t = SHORT \\ 0 & a_t = HOLD \end{array} \right\} \tag{2}$$

The agent is rewarded for choosing an action a_t if price increases within the horizon of K timesteps and opens a buy position, or price decreases and chooses to open a Short position. On the other hand, it penalizes the agent for incorrect estimations and wrong actions, unless it chooses to hold. Additionally, the proposed reward function, places a greater importance on price volatility, with higher rewards being given during more volatile market trends. Given the initial budget, which is finite, the agent is encouraged to focus on higher-volatility trends, rather than engaging uncertain and low-rewarding trades.

Architecture. The architecture of policy network component consists of separate inputs for $\vec{m}_t$ and d_t, denoted as i_1 and i_2 respectively. The layer i_1 is a batch of timeframes, followed by Convolutional (CNN) layers, which extracts feature maps and flattens these maps into a single vector u. Then, the inputs i_2 are concatenated with u, which passes into a Fully-Connected Neural Network. The output of the policy network is a dense layer, which consists of 3 units, one for each action $a \in A$.

4.4 Supervised Component

In high-volatility markets, it is quite common that uptrends or downtrends occur between two consecutive timesteps. Because DRL agents operate on discrete timesteps, this could lead to missed profit opportunities and increases the risk of losses. To address these issues, an estimator model is trained to predict the maximum High-Close and Close-Low price margin within a time horizon of K timesteps, the same as that of the DRL component. Then, during the exploitation phase, the DRL agent opens a Buy or Short position at timestep t, while the estimator predicts at which price the position should be closed. Finally, the Supervised component places a limit order on the predicted price to automatically close the agent's position.

We selected a TCN architecture as the estimator model for various reasons. First of all, TCNs are fast and easy to train, due to efficient memory usage compared to LSTMs and Transformers, requiring less computational resources to be properly trained. Additionally, TCN provide more stable training for large amounts of data in a short time, whereas Transformers are well-known to require a lot of training time to achieve satisfying accuracy. Finally, TCNs have shown to outperform LSTMs in various forecasting tasks, making it a preferable option for our framework.

Inputs. To train the TCN model, the market observations $\vec{m}_t$ are used as inputs, with the targets being the maximum logarithmic percentage increase and maximum logarithmic percentage decrease of current price c_t within a time horizon, as shown in Equation 3.

$$\begin{array}{l} Y_H = \ln \frac{H_{t_{max}}}{C_t} \\ Y_L = \ln \frac{C_t}{L_{t_{min}}} \end{array} \tag{3}$$

Then, the estimated prices to close the positions can be computed by Equation 4.

$$Y_{limit}(t) = \begin{matrix} C_t * e^{Y_H} & , a_t = a_0 \\ C_t * e^{-Y_L} & , a_t = a_1 \end{matrix} \quad (4)$$

Loss Function. The most common loss functions for Deep Learning models that are used in regression tasks, including price estimation, are MSE and MAE, which treat overestimation and underestimation errors equally. However, in trading tasks, overestimating the target price (i.e., predicting a higher or lower price than the actual price) would result in the limit orders not being executed. To increase the probability of limit orders being executed, this paper makes use of Pinball Loss (a.k.a. Quantile Loss) Somers and Whittaker (2007), which gives higher loss penalties to overestimation errors as described by Equation 5.

$$Pinball\ Loss = \max\left\{\tau * (y - \hat{y}), (\tau - 1.0) * (y - \hat{y})\right\} \quad (5)$$

Pinball loss uses τ parameter to adjust the importance of overestimation and underestimation error, where τ is a value in the interval $(0.0, 1.0)$. Values less than 0.5 penalize more the overestimated predictions, while higher values assign higher importance on the underestimation error. To increase the probability of the limit orders being executed, we select a value of τ less than 0.5

Architecture. The architecture of TCN component consists of the input layer, which is a batch of timeframes, followed by a single TCN layer with residual blocks. Then, the extracted features of TCN layers are flattened into a vector, which passes into a Fully-Connected Neural Network. Finally, the network includes two outputs, one for High and one for Low volatility prediction (defined as $Y_H and Y_L$ respectively). The network architecture is visually presented in Figure 4.

4.5 Virtual Profit Margin

To further reduce trading risks and provide UNSURE with explainability of the generated actions, we introduce a novel safety mechanism named Virtual Profit Margin (VPM), which is deployed during the exploitation period and includes two main modules. The first module is enabled once the estimated profit margin, defined as the difference between the current close price from the estimated future price, including the fees (Equation 6), becomes negative. This indicates that the agent might has been tricked by market noise and attempts to open a position in a low-volatility and high-risk trading period. So, the proposed action a_t is avoided if $VPM_t \leq 0.0$, reducing the potential risk in opening a position in timestep t.

$$VPM_t = \begin{matrix} Y_{limit}(t) - C_t - f * (Y_{limit}(t) + C_t) & a_0 \\ C_t - Y_{limit}(t) - f * (Y_{limit}(t) + C_t) & a_1 \end{matrix} \quad (6)$$

The second module replaces f parameter in Equation 6 with a virtual fee parameter, defined as $\hat{f}$, with $\hat{f} \geq f$. The use of VPM mechanism creates a larger target profit margin between the current price and the predicted price, which forces the trading framework to engage higher price fluctuation periods than it normally would. This addition contributes to decreasing the likelihood of overestimation errors, potentially arising from the Supervised learning component, by effectively raising the profit margin threshold. Consequently, even if TCN predicts an overestimated future price, profit margin will have to be large enough, making it more difficult for the agent to engage such positions, as demonstrated in Figure 5. $\hat{f}$ parameter can be adjusted by measuring the overestimation error of TCN.

4.6 UNSURE

The proposed framework integrates the parameterization method, the aforementioned ML components and the VMP mechanism into an end-to-end trading system, which is illustrated in Figures 4 and 5. Additionally, This study has conducted a thorough evaluation of two learning algorithms for this specific task, which is PPO, and IMPALA. PPO has gained popularity in trading-related tasks, as mentioned in the Related Works Section, as it is fast, stable, easy to implement and achieves satisfying performance on trading tasks. On the other hand, IMPALA is a distributed DRL algorithm, which has not been extensively explored in trading problems. Similar to IMPALA, PPO also adopts the Actor-Critic architecture, but does not include any policy lag. By comparing these two algorithms, this study aims to investigate the potential of Distributed DRL frameworks in challenging trading environments, such as cryptocurrency trading.

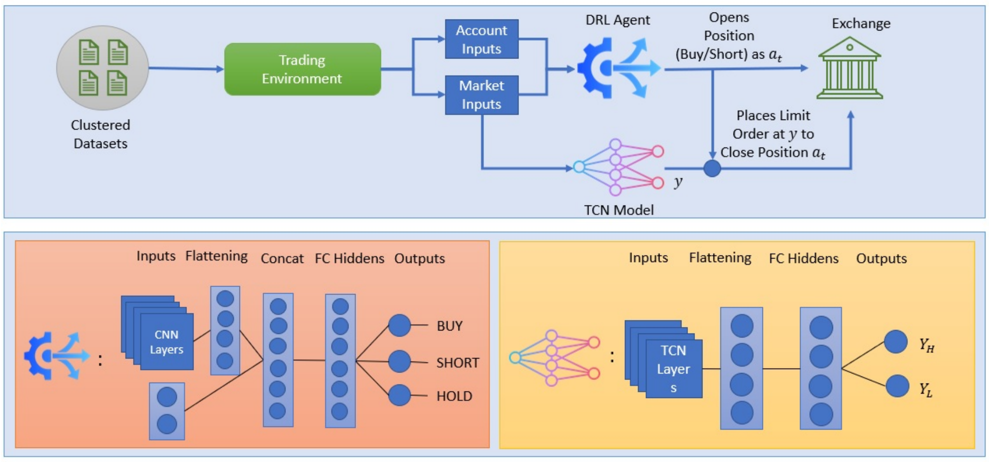

Fig. 4: Coordination between UNSURE's components. **Top**: Unified UNSURE system. **Bottom Left**: DRL component. **Bottom Right**: TCN component.

5 Experiments & Results

In this section, we analyze the historical market datasets that are used to conduct the experimental procedures. A range of evaluation metrics are additionally described to assess the performance of the framework. Also, this section provides details about the hyper-parameter tuning procedure. Finally, the section concludes by presenting and discussing the experimental results and concludes with a sensitivity analysis of the framework's parameters.

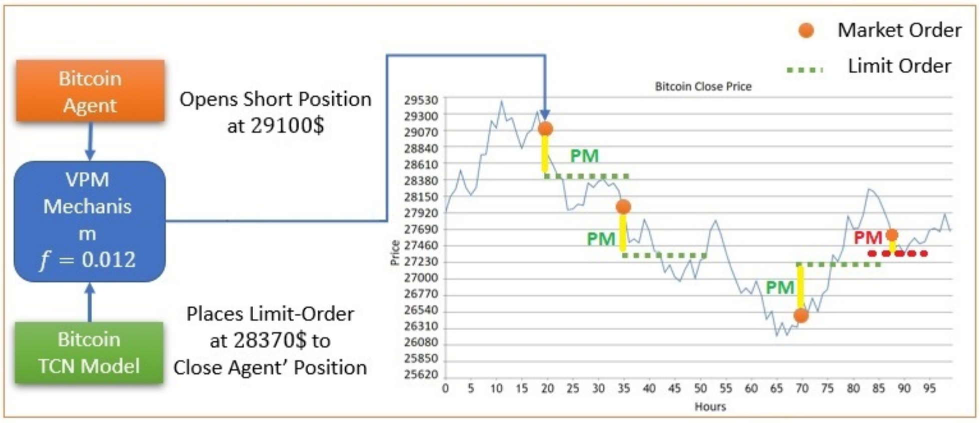
Market Order
Limit Order
Bitcoin Agent
Opens Short Position at 29100$
VPM Mechanism $f = 0.012$
Bitcoin TCN Model
Places Limit-Order at 28370$ to Close Agent' Position
Bitcoin Close Price
PM
Price
Hours

Fig. 5: A trading case scenario using VPM. Green *PM* symbol illustrates that Profit Margin between current price and predicted volatility is high, while the red one indicates that it is not. Once VPM mechanism is triggered, it prevents the agent from opening positions.

5.1 Datasets

Each datasets include hourly candlesticks consisting of Open-High-Low-Close-Volume (OHLCV) data for Bitcoin (BTC), Ethereum (ETH), Cardano (ADA), Litecoin (LTC), Ripple (XRP), Polygon (MATIC), Binance Token (BNB), Solana (SOL), and Loopring (LRC) markets. The datasets are downloaded from Binance platform [4], because they provide high quality OHLCV data. Then, the candlesticks are converted to logarithmic returns, because they represent price changes over time, which is a more informative indicator of the current market state than absolute prices. Finally, technical analysis is applied to each dataset, followed by the parameterization method described in Section 4.

The technical indicators that are used in the experiments include DEMA, MACD, VWAP, RSI, STOCH, CCI, ADX, B-BANDS, AROON, ADL, OBV, with their definitions being provided in Appendix A. While there are hundreds of technical indicators available, each one of which capture different market aspects, our selection of technical indicators was guided according to their popularity on AlphaVantage platform [5], which is a well-known financial platform used by researchers and investors to retrieve historical data. Aside popularity, we also aim to include indicators that capture a variety of aspects. For instance, DEMA, MACD, CCI, ADX and AROON indicators reflect the overall market trends, while VWAP, ADL and OBV capture the market volume. Finally, we include RSI and STOCH to assess the momentum of the markets, as well as Bollinger Bands (B-BANDS) indicator to capture the volatility. Finally, to avoid features biases in the training process all features are scaled in range $[-1.0, 0.0]$ for the negative values and $[0.0, 1.0]$ for the positive ones, using the Max-Absolute scaling method, described by Equation 7.

[4]https://www.binance.com/en
[5]https://www.alphavantage.com

$$x_{scaled} = \frac{x}{\max(|x|)} \tag{7}$$

5.2 Experimental Setup

To train DRL agents, a training and an evaluation environment was are constructed for each market. The training environment consists of 50000 past historical observations, while the evaluation environments include the 5000 most recently hourly observations, equivalent to (10%) of the total observations for each dataset, which is a trading period of approximately seven months of hourly trading. We have also tried small evaluation portion, such as %5 of the total dataset observations, but we did not found significant differences in the final results of the experiments, so we kept the evaluation size to 10%.

Moreover, to avoid data overfitting, the evaluation samples were used only during the evaluation period and were not available during the training period. Also, we ensured that the evaluation periods includes sudden market shifts, where there is a lot of profit, as well as loss potential, such as the one presented by figure 6.

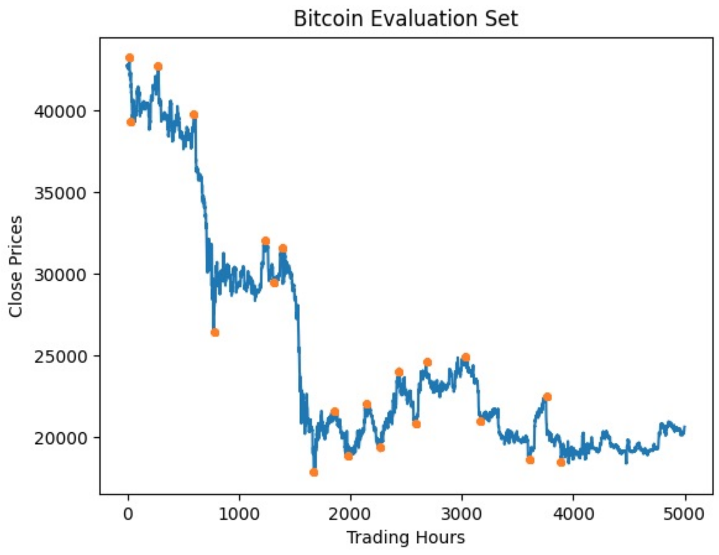

Fig. 6: Visualization of Bitcoin evaluation dataset. Orange circles indicate sudden market shifts followed by high-volatility trading periods.

To allow the agent to capture real-time market dynamics, we add several options to the simulation environments. First, commission fees are set to $f = 1.0\%$ to each market, which aligns with the typical transaction charges of Binance. Second, we add the ability to place limit-orders in order to close positions automatically at a specified price, just like in Binance platform. Finally, the agent starts with a finite budget of 100,000 US dollars and is allowed to buy or short only limited volume of assets per transaction. We provide both the Binance market simulator and the code for reproducing the experiments, which can be downloaded from the Google Drive platform [6].

5.3 Metrics

DRL Agents. To evaluate the performance of DRL agents, several metrics have been used, including: *(i) Cumulative Returns (CR), (iv) Sharpe Ratio (SHR), (v) Sortino Ratio (SOR) and (vi) Average Drawdown (ADD)* for DRL agents.

CR is defined as the sum of all returns, as described by Equation 8.

$$CR = \sum_{t=0}^{T} r_{t+1} \tag{8}$$

where r_{t+1} is defined by Equation 2. The higher the Cumulative Returns, the higher the profits generated by the agent. Although it is a straightforward indicator of profits, CR does not measure the overall trading risk. Traders usually consider risk-adjusted metrics, such as SHR, SOR and MDD.

SHR is defined by Equation 9.

$$SHR = \frac{E\left[R_t\right] - r_f}{\sigma\left[R_t\right]} \tag{9}$$

where r_f indicates the risk-free interest rate. In order to evaluate the performance of the agent, the risk-free rate was set to zero. SHR is a widely used metric, however, it considers both upside and downside portfolio wealth Mamoghli and Daboussi (2009). On the other hand, SOR (Equation 10), which is a similar metric to SHR, emphasizes more on the downside returns by dividing the average returns with the standard deviation of the negative returns. Although both SHR and SOR are risk-adjusted metrics, SOR can be a more relevant metric for investors who are particularly concerned about losses rather than steadily increasing gains. A trading strategy achieves satisfying performance if SHR and SOR are above 1.0, while values above 2.0 indicate that the strategy achieves strong performance. Mamoghli and Daboussi (2009).

$$SOR = \frac{E\left[R(t)\right] - r_f}{\sigma\left[NR(t)\right]} \tag{10}$$

where $NR(t)$ are all negative returns ($R(t) < 0$).

Finally, ADD measures the average downside loss percentage of a portfolio wealth, which is a good indicator of trading risk, and is defined by Equation 11.

$$ADD = \frac{\sum_{i=1}^{N} DD_i}{N} \tag{11}$$

where N is the number of drawdown periods and DD_i = the i-th drawdown period, measured as a percentage decline from the previous peak of the portfolio's wealth.

[6]Code: https://drive.google.com/drive/folders/1aW7Miv0cf2jCPe8D0W42pCerKYp1AcNh?usp=sharing

TCN. To evaluate the performance of TCN model, *Mean Absolute Error (MAE)*, *Mean Overestimation Error* and *Mean Overestimation Percentage Error (MOPE)* have been used, which are described by Equations 12, 13 and 14 respectively, with y_i being the target values and $\hat{y}_i$ the predicted ones.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{12}$$

$$MOE = \frac{1}{N}\sum_{i=1}^{N} \left\{ \begin{array}{ll} \hat{y}_i - y_i & \hat{y}_i \geq y_i \\ 0 & \hat{y}_i < y_i \end{array} \right\} \tag{13}$$

$$MOPE = \frac{1}{N}\sum_{i=1}^{N} \left\{ \begin{array}{ll} 1 & \hat{y}_i - y_i \geq 0 \\ 0 & \hat{y}_i < y_i \end{array} \right\} \tag{14}$$

UNSURE. To evaluate the trading performance of UNSURE, SHR, SOR, and ADD metrics were utilized, as well as *Profit and Loss* (PNL) metric, which represents the ratio of final account wealth to initial wealth.

5.4 Hyper-Parameter Tuning

The hyperparameters of PPO, IMPALA and TCN which are presented in this work have a large combinatorial space, making an exhaustive search infeasible, so only limited search was conducted. The tuning process began with the default values of each algorithm, and the most sensitive hyperparameters were manually fine-tuned. The optimal values were determined by evaluating each set based on the Cumulative Returns (CR) for the DRL agents and validation MAE error for the TCN models. The selection of hyperparameters are presented in Tables 1, 2 and 3 for PPO, IMPALA and TCN respectively.

In addition, there are several other hyperparameters as outlined in Section 4 that can affect the performance of the framework. These include Transaction Volume (v), Timeframe size (T), Horizon K, Virtual Fees ($\hat{f}$), the window size (N) for each technical indicator and τ parameter of the TCN model. To beging with, the transaction volume was set to $v = 1.0$ asset per transaction. The Timeframe size was set to $T = 20$ because it allowed both the DRL agents and TCN models to train quickly while achieving satisfying predictive results and the Horizon K was set to 12 timesteps (12 hours ahead), after experimenting with various values in range 8 to 18. We selected a virtual fee of $\hat{f} = 1.4\%$ from a set of options ranging from 1.0 to 1.5, because it achieved the best results. Also, the τ parameter was set to 0.3, as we further describe later in this Section. Finally, we provide the selected window sizes for each technical indicator that resulted after the technical indicator parameterization method in Table 4.

5.5 DRL Evaluation

In this evaluation part, we only compare the performance of PPO and IMPALA in every market and analyze their potential to open profitable positions during high volatility periods. Our primary metric for evaluating their performance is *CR*. Higher CR values indicates better exploitation of high price fluctuation by the agent. In

Table 1: PPO Hyperparameters

Parameter	Value
Epsilon Clipping (ϵ)	0.2
Entropy Regularization (h)	0.01
Generalized Advantage Estimation (GAE)	$True$
Mini-Batch size	128
Epochs	5
Discount Factor (γ)	0.99
Optimizer	$Adam(lr = 0.001)$
Actor Network Convolutional Layers	2 x $(32, 3)$ + ReLU
Actor Network Hidden Dense Layers	2 x 256 + ReLU
Critic Network Convolutional Layers	2 x $(32, 3)$ + ReLU
Critic Network Hidden Dense Layers	2 x 256 + ReLU

Table 2: IMPALA Hyperparameters

Parameter	Value
Actors	16
Entropy Regularization (h)	0.001
Mini-Batch size	512
Epochs	1
Discount Factor (γ)	0.99
Optimizer	$Adam(lr = 0.001)$
Actor Network Convolutional Layers	2 x $(32, 3)$ + ReLU
Actor Network Hidden Dense Layers	2 x 256 + ReLU
Critic Network Convolutional Layers	2 x $(32, 3)$ + ReLU
Critic Network Hidden Dense Layers	2 x 256 + ReLU

Table 3: TCN Hyperparameters

Parameter	Value
Convolutional layers	$(64, 5)$ + ReLU
Dilation Rates	$[1, 2, 4, 8]$
Dense layers	$[128, 32]$ + ReLU
Residual Blocks	True
Weight Normalization	$True$
Dropout Rate	0.0
Batch Size	64
Epochs	200
Optimizer	$Adam$
τ	0.3

Table 4: Technical Indicator parameterization using Sortino Ratio Metric

Market	DEMA	MACD	VWAP	RSI	STOCH	CCI	ADX	BBANDS	AROON
ADA	40	15, 35	40	14	20	20	14	3	14
BNB	40	33, 72	40	25	8	3	25	25	40
BTC	15	33, 72	40	5	3	3	25	3	25
ETH	15	33, 72	40	5	3	40	14	15	40
LRC	40	26, 57	25	15	15	25	14	15	20
LTC	5	33, 72	40	5	40	3	40	40	5
MATIC	20	15, 35	20	10	20	10	10	8	40
SOL	25	33, 72	40	10	40	3	8	3	40
XRP	25	33, 72	20	40	20	20	20	3	3

addition to CR, the rest of the presented metrics consider the overall risk-adjusted performance of the agent's strategy.

We used four different methods for the evaluation part:

a - **No Parameterization - No Clustering**
b - **No Parameterization - Clustering + Feature Concatenation**
c - **Parameterization - No Clustering**
d - **Parameterization - Clustering + Feature Concatenation**

The performance of both PPO and IMPALA using methods (a) and (b) are presented in Table 5, while methods (c) and (d) are presented in Table 6. To evaluate the performance of each agent, we used *CR, SHR, SOR and ADD* metrics, which are highlighted with different colors inside the table. Due to low computational resources, both algorithms are trained for 200 iterations in each experiment, with each iteration being 512 steps.

It can be observed by the tables that IMPALA demonstrates superior performance compared to PPO in almost every market. More specifically, for the same number of training iterations, IMPALA achieves higher cumulative returns in the evaluation environment in comparison with PPO. Additionally, the trading performance of IMPALA is superior, as it can be observed by risk-adjusted metrics. Finally, the performance of IMPALA in comparison with PPO can be illustrated by Figure 7

Furthermore, by comparing the metrics in Tables 5 and 6, it can be seen that the parametarization method employed in this study improved the overall performance for both learning algorithms, especially in terms of cumulative returns. Additionally, when the parameterization method is combined with the Unsupervised component, which clusters the datasets and concetenates their features, then the performance of both PPO and IMPALA is increased.

Table 5: Comparison of the Performance of PPO with IMPALA on 9 Cryptocurrency markets, without employing the parameterization method. "*Y*" columns indicate that clustering was employed, while "*N*" indicate it wasn't. The boldings highlight the method that achieves the highest performance for each metric. Each metric is displayed with a color: CR - Blue, SHR - Green, SOR - Red, ADD: Orange

Algorithm	**PPO**								**IMPALA**							
Metric	**CR**		**SHR**		**SOR**		**ADD**		**CR**		**SHR**		**SOR**		**ADD**	
Clustering (Y/N)	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**
ADA	1.770	1.638	1.914	1.922	3.894	3.606	0.052	0.059	**1.915**	1.854	1.969	**2.094**	**4.388**	4.068	0.034	**0.029**
BNB	0.441	0.537	1.107	**1.426**	1.728	1.588	0.107	**0.081**	**0.590**	0.552	**1.358**	1.269	**1.741**	1.561	0.161	0.070
BTC	1.116	1.039	1.174	**1.258**	1.815	1.550	0.095	0.066	**1.211**	1.15	**1.340**	1.214	**1.856**	**1.825**	**0.042**	0.048
ETH	3.572	3.567	2.376	**2.399**	**6.010**	5.908	**0.022**	0.042	**3.698**	3.597	**2.423**	2.359	5.813	5.203	0.024	**0.022**
LRC	0.922	0.960	1.525	1.500	4.411	4.086	0.045	0.091	**1.262**	1.062	**2.056**	1.819	**6.380**	**5.191**	**0.033**	0.044
LTC	2.164	**2.236**	**2.068**	2.036	3.937	4.140	0.042	0.053	**2.239**	2.232	2.064	2.002	4.116	**4.223**	**0.033**	0.040
MATIC	1.012	1.026	1.733	1.677	6.495	5.449	**0.046**	0.056	**1.114**	1.102	1.749	**1.891**	5.472	**6.827**	0.161	0.047
SOL	1.283	1.183	2.047	1.921	8.596	8.585	0.042	0.073	**1.338**	1.295	**2.231**	2.022	**9.535**	9.440	**0.026**	0.043
XRP	2.742	2.661	2.204	2.156	6.169	6.213	0.044	0.050	**2.883**	2.870	**2.326**	2.288	6.774	**7.095**	0.038	**0.036**

5.6 TCN Evaluation

The key metric for evaluating the performance of TCN models in our work is *MOPE*, because lower MOPE scores indicate higher probability that limit orders will be executed at the estimated prices. Another important metric to be considered is *MOE*,

Table 6: Comparison of the Performance of PPO vs IMPALA on 9 Cryptocurrency markets, including the parameterization method

Algorithm	**PPO**								**IMPALA**							
Metric	**CR**		**SHR**		**SOR**		**ADD**		**CR**		**SHR**		**SOR**		**ADD**	
Clustering (Y/N)	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**	**Y**	**N**
ADA	1.734	1.688	2.018	1.985	4.288	3.661	0.033	**0.029**	**2.042**	1.930	**2.195**	2.169	**4.801**	**4.721**	**0.029**	0.033
BNB	**0.570**	0.501	**1.307**	1.224	1.554	1.565	0.043	0.062	0.530	**0.620**	1.147	**1.405**	**1.577**	**2.113**	0.051	**0.042**
BTC	0.970	1.050	1.286	1.276	1.507	1.560	**0.034**	0.044	**1.370**	1.213	**1.502**	1.389	**2.261**	**1.897**	0.047	0.036
ETH	3.169	3.206	2.135	2.197	4.761	4.795	0.031	**0.024**	**3.720**	3.631	2.361	**2.404**	**5.611**	**5.989**	0.027	0.025
LRC	1.001	0.988	1.558	**2.214**	3.797	**4.370**	0.044	0.039	**1.064**	1.054	**3.975**	1.836	**3.975**	4.111	0.040	**0.034**
LTC	2.257	2.105	2.056	1.928	4.266	3.935	0.035	0.027	2.334	**2.408**	2.183	2.126	**4.567**	**4.643**	**0.025**	0.026
MATIC	1.051	1.039	1.807	1.773	5.245	5.849	0.025	0.184	**1.187**	1.061	**1.860**	1.836	**5.335**	**6.005**	0.027	**0.014**
SOL	1.244	1.191	1.912	1.849	**8.186**	7.246	0.027	0.053	**1.410**	1.324	**2.051**	2.039	7.681	**8.269**	**0.023**	0.024
XRP	2.881	2.722	**2.219**	2.217	**7.236**	6.236	0.026	0.032	**2.922**	2.831	**2.323**	2.209	7.046	**6.521**	0.028	**0.027**

because it highlights the overestimation error of predicted volatility. Lower *MOE* values indicate that the predicted volatility is slightly above the actual volatility, which can be tackled by increasing virtual fee parameter during the trading period. Finally, *MAE* metric was also included, because it is a standard metric used for regression models.

In this study, each TCN model was trained for 100 epochs, with the clustering method employed, and we compared the performance of TCN models with and without the parameterization method, as shown in Table 7. While the parameterization method appears to decrease the estimation error of the models most of the times, it did not always provide a major improvement. For instance, in the LRC and LTC markets, the MOPE error was higher when using the parameterization method. Nonetheless, the parameterization method enhanced the overall accuracy of the models, and should be considered as an addition during the dataset construction pipeline.

Table 7: Final Validation metrics of TCN Model using the clustering method and with/without parameterization method.

-	**Parameterization**			**No Parameterization**		
Market	**MAE**	**MOE**	**MOPE**	**MAE**	**MOE**	**MOPE**
ADA	**0.0198**	0.0092	**0.3145**	0.0221	**0.0091**	0.3164
BNB	**0.0180**	0.0083	**0.3052**	**0.0180**	**0.0072**	0.3098
BTC	**0.0198**	**0.0088**	**0.2956**	0.0206	0.0094	0.3011
ETH	**0.0286**	**0.0151**	**0.3241**	0.0313	**0.0151**	0.3283
LRC	**0.0185**	0.0097	0.2785	0.0238	**0.0092**	**0.2762**
LTC	0.0243	**0.0117**	0.2943	**0.0221**	0.0126	**0.2876**
MATIC	**0.0253**	**0.0149**	**0.3133**	0.0282	0.0161	0.3171
SOL	0.0321	**0.1723**	**0.2653**	**0.0312**	0.0183	0.2701
XRP	0.0322	0.0184	**0.2798**	**0.0306**	0.0192	0.2821

5.7 UNSURE Evaluation

In the final evaluation part, we measure the profitability and effectiveness of the integrated UNSURE framework and VPM, as presented in Table 8. It can be noticed that even without VMP mechanism, UNSURE achieves satisfying SHR and SOR metric values, with exceptional performance in BTC, ETH, LRC, and XRP markets. Although the PNL returns are less than 5% in some markets, UNSURE achieves trading risk less than 10% in all market except ETH, where trading risk is 12.3%, which is quite satisfying, and thus it can be a preferred option for investors who prioritize on low trading risks.

Table 8: UNSURE Trading Performance Evaluation with VPM (Left) and without (Right)

-	**VPM**				**No VPM**			
Market	**PNL (%)**	**SHR**	**SOR**	**ADD**	**PNL (%)**	**SHR**	**SOR**	**ADD**
ADA	**16.992**	**1.651**	**2.267**	**0.043**	1.021	0.411	0.593	0.047
BNB	**3.236**	**1.643**	**1.809**	**0.026**	-0.237	-0.432	-0.505	0.034
BTC	**12.087**	**1.916**	**1.963**	0.036	4.265	1.489	1.572	**0.034**
ETH	**24.554**	**2.725**	**2.896**	**0.123**	20.763	1.635	1.787	0.149
LRC	**2.315**	**2.402**	**2.596**	0.025	2.269	1.119	1.392	**0.018**
LTC	3.792	1.451	1.603	**0.041**	**3.824**	**1.513**	**1.708**	0.045
MATIC	2.263	1.697	**1.892**	**0.084**	**3.580**	**1.712**	1.782	0.093
SOL	**3.221**	**1.277**	**1.268**	**0.068**	2.342	1.171	1.215	0.083
XRP	**21.583**	**2.220**	**2.744**	**0.039**	13.136	1.798	2.204	0.078

Additionally, it can be seen that VPM mechanism improves the overall performance of UNSURE, as highlighted in Table 8 and reduces drawdowns caused by overestimation errors in most of the markets. This can be attributed to UNSURE's ability to engage fewer trades, but during periods of high volatility, as well as its ability to trade less aggressively, due to higher profit margin threshold, resulting in a more effective trading approach. Finally, the overall performance of UNSURE with the use of VPM mechanism is also illustrated in Figure 8.

5.8 Baseline Evaluation

To highlight the effectiveness of UNSURE in comparison with traditional technical indicator strategies, we conducted an evaluation comparing it to three well-studied technical indicators: a) RSI, b) CCI, c) VWAP in all 9 cryptocurrency markets. Furthermore, we compare UNSURE with 5 previous trading methodologies in Bitcoin Market, including:

1. - $TD(\lambda)$ (Pendharkar and Cusatis, 2018)
2. - CNN (Jiang and Liang, 2017)
3. - DQN (Bu and Cho, 2018)
4. - $DNA-R$ (Betancourt and Chen, 2021)
5. - $DNA-S$ (Betancourt and Chen, 2021)

To benchmark our methodology against the five aforementioned systems, we employed both PNL returns and the Sharpe Ratio as key metrics for comparison. These metrics were selected not only for their relevance in assessing both profitability and risk but also because they were utilized in the evaluations of the previous approaches as well, ensuring a fair and direct comparison.

For $TD(\lambda)$ approach, we used the most parameters that are presented and recommended in the original paper, including $\gamma = \lambda = 0.9, \eta = \alpha = 0.1$ and fine-tuned the exploration rate ϵ of the algorithm, which was selected to be $\epsilon = 0.05$, from the

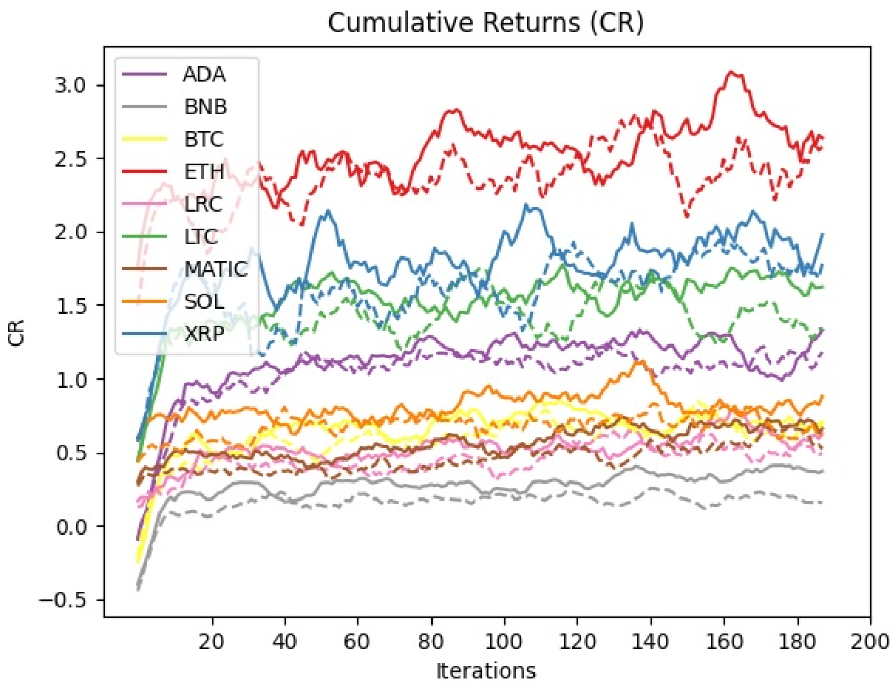

Fig. 7: Impala evaluation using the Unsupervised component and a) With technical indicator parameterization (Continuous line) b) Without parameterization (Dashed Line)

set of $\{0.01, 0.02, 0.05\}$. For CNN approach, we used a single CNN layer of 32 convolution filters of size $5x5$, which was followed by a fully connected neural network of 500 units and the output layer, similar to Jiang and Liang (2017) work. Although the authors used only 12 filters in their work, we found that 32 filters yields higher profits in our dataset. For Bu and Cho (2018) approach, we used the same parameters that were optimized in the original work, including $\delta = 0.995$ and LSTM window size of 55 inputs. Additionally, we selected a small γ value of 0.1 with high exploration rate as it suggested in their paper and trained the policy network with a batch size of 64 experiences per step, instead of 8, which achieved slightly more satisfying results. Finally, for $DNA - R$ and $DNA - S$ approaches, we set the clipping parameter of PPO to 0.2, which was fined-tuned from a set of $\{0.1, 0.2, 0.3\}$ and the number of batch size to 32. The rest of the parameters, such as $GAE, H(\tau), \lambda$ and $Adam'sparameters$ were set according to the authors' work.

The comparison results presented in Figures 9, 10, and 11, illustrate that each technical indicator employs a unique trading strategy in each market. However, these baseline strategies usually result in negative PNL returns by the end of the trading period, across various cryptocurrency markets, with some exceptions. These include

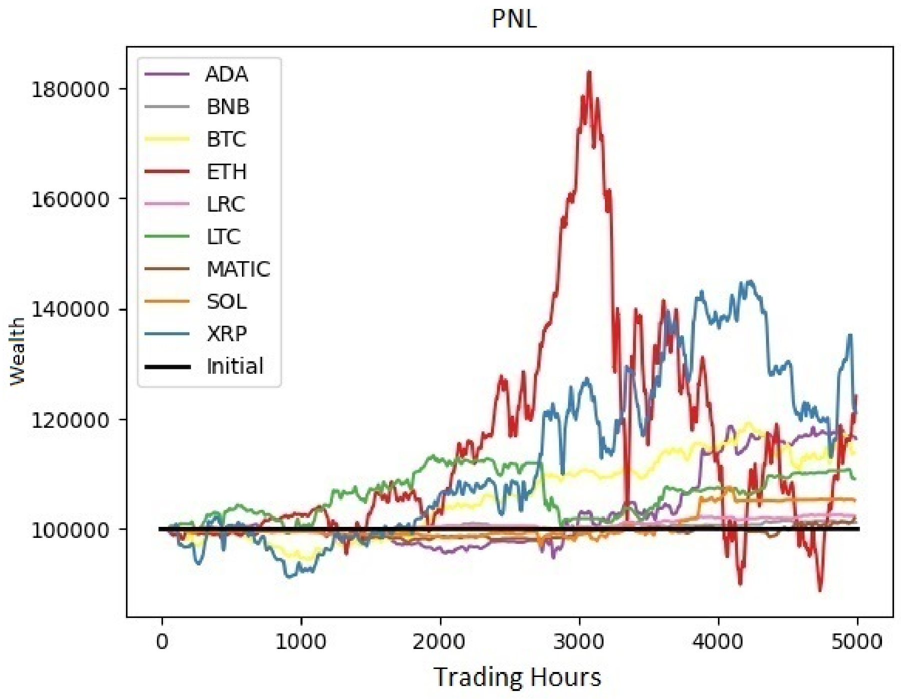

Fig. 8: Evaluation of UNSURE framework, including parameterization method, all three ML components and VPM mechanism.

the RSI strategy, which eventually yielded profits in BTC and ETH markets, as well as VWAP strategy, which also demonstrated satisfying performance in BTC market, achieving a final wealth of 180000 USD. On the other hard, UNSURE achieved lower PNL returns compared to VWAP in BTC market, but as demonstrated in Figure 8, it presented lower drawdowns and a steadily increasing PNL over the time. This is more important for investors who seek profits with lower trading risks. Finally, although UNSURE did not yield large profits in every market, it did not yield negative PNL at the end of the trading period in any market, unlike RCI, CCI and VWAP strategies.

In our final evaluation setup, our trading agents obtained significantly higher scores than those obtained by previous approaches. These results are shown in Table 9, where UNSURE scored 12.087 total returns and 1.916 SHR, outperforming $DNA-R$, which achieved 7.116 and 0.758 of total returns and SHR respectively.

5.9 Sensitivity Analysis

In addition to hyper-parameter tuning and the evaluation of UNSURE, we also conducted a sensitivity analysis to examine the effect of various hyper-parameters on our framework. These include Transaction Volume (v), Initial Budget(b), Timeframe Size (T), Horizon (K), Virtual Fees ($\hat{f}$) and Quantile τ. Our analysis was carried out using

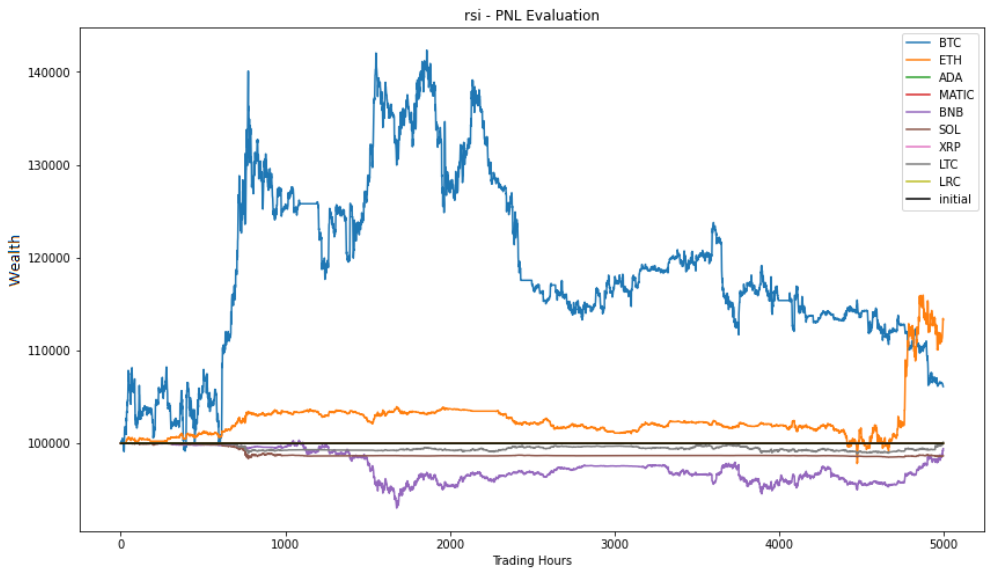

Fig. 9: Visualization of RSI Performance using PNL metric

Table 9: Baseline Evaluation in Bitcoin Market

Algorithm	**PNL(%)**	**SHR**
$TD(\lambda)$	0.329	0.404
CNN	1.012	0.527
DQN	1.151	0.661
$DNA-S$	2.798	0.609
$DNA-R$	7.116	0.758
UNSURE	**12.087**	**1.916**

the evaluation dataset from the Bitcoin market. We used the Sortino Ratio (SOR) as a metric to assess the influence of these hyper-parameters on both profitability and trading risk.

Transaction Volume & Initial Budget. Both of these parameters can significantly influence both the profitability and the risk levels of our system. Specifically, a higher initial budget allows the agent to hold multiple positions simultaneously, potentially increasing the opportunities for profit. However, this also exposes the agent to greater risks. Similarly, increasing the transaction volume enables the agent to trade in larger asset volumes, which can also enhance profits, as shown in Table B1. Nonetheless, it is important to consider real-world market constraints during the tuning process, which are discussed in detail in Section 6, suggesting that trading in smaller volumes is often more advisable.

Timeframe Size. This parameter is utilized in both the DRL module and the Supervised module to create timeframes from past data samples, which are essential for the action generation and volatility estimation respectively. Smaller timeframe

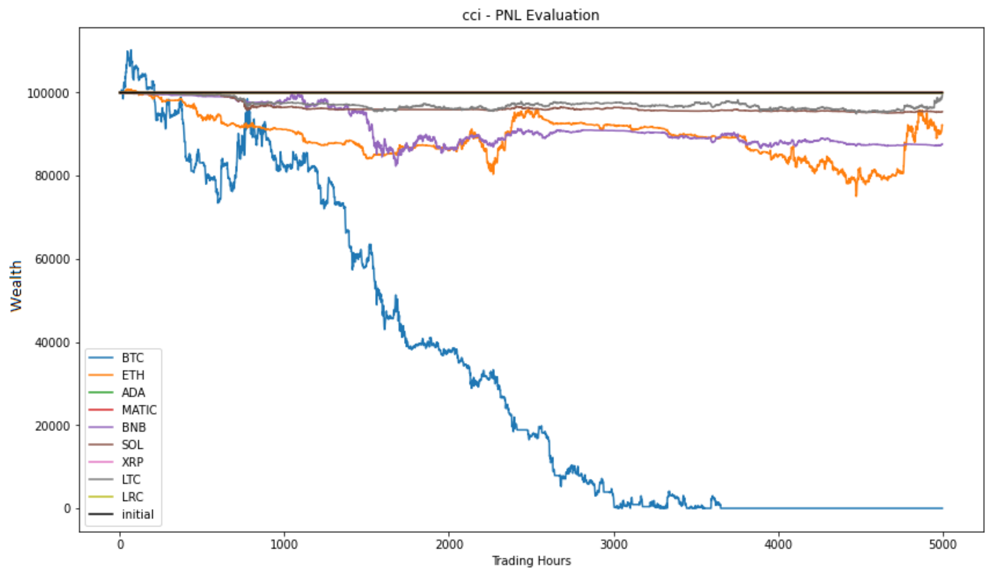

Fig. 10: Visualization of CCI Performance using PNL metric

sizes may lead to insufficient information for the agent to make informed decisions. For instance, it might be crucial for the agent to be aware of at least the past 12 hours of price activity. Similarly, the TCN model could potentially enhance its volatility estimation with longer input timeframes. However, larger timeframe sizes can introduce noisy inputs and increase the overall model complexity, leading to longer training time required for them to converge to a satisfying solution. In our experiments, we set $T = 20$, as we observed that it achieves satisfying balance between training duration and performance for both the DRL and the Supervised module, even though further tuning of this parameter could result in a higher performance trading system.

Horizon. The horizon parameter holds considerable importance as it impacts both the accuracy of volatility estimation and the trading strategy of the system. Small horizon values increase the Supervised module's performance, but it may reduce the system's overall profitability, because it limits the duration for which positions remain open, before significant price changes can occur (Figure B1). Oppositely, increasing this parameter, decreases the Supervised module's performance, as it will have to estimate the maximum volatility for a longer horizon. Although the volatility increases in longer horizons, large horizon values could also drop the trading performance of the system, as it would result to larger overestimation errors (Figure B1), which would result in the orders not being executed.

Virtual Fees. The virtual fees parameter is highly dependent on the exchange's fees per transaction. In our case, with transaction fees set at $f = 1.0\%$ per transaction, the virtual fees are constrained to a maximum of $\hat{f} \leq 1.0\%$, according to VPM rule. By increasing its value, the trading system adopts a more conservative trading approach, resulting in fewer trading actions, which aims to identify trading periods with higher potential for profitability. Consequently, while the opportunity for profit

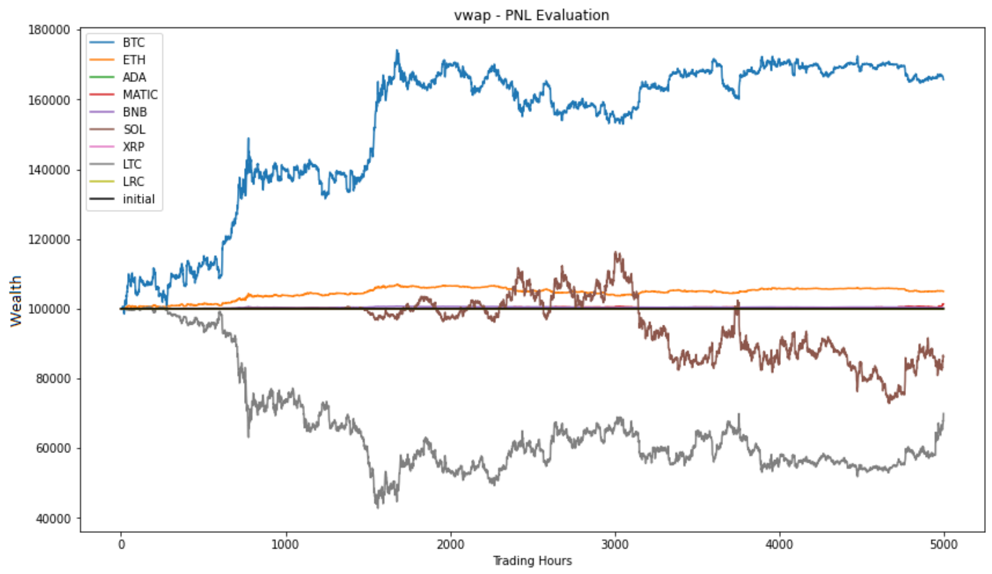

Fig. 11: Visualization of VWAP Performance using PNL metric

may decreases, the risk associated with trading also decreases. This flexibility allows the system to be adjusted to the risk tolerance of the user: more conservative traders might opt to increase the virtual fees value, whereas more aggressive traders could align the virtual fees with the actual transaction fees. Table B2 demonstrates how the total PNL and (ADD) metrics change in respect to $\hat{f}$.

Quantile. An important consideration in determining the *tau* parameter is finding the right balance between underestimation and overestimation error. If *tau* parameter is set above 0.5, the model will tend to overestimate its targets, so most of limit orders will not be executed. On the other hand, if a very small value is chosen for *tau*, TCN model may underestimate the estimated prices by a large margin, resulting in lower profit potential. In this work, *tau* was set to 0.3, because it reduced the mean overestimation percentage error of the TCN model to only $0.25\% - 0.30\%$, while maintaining satisfying underestimation error levels, as presented in Table B3.

6 Discussion

In almost our experiments and under different settings in each experiment, IMPALA managed to outperform PPO algorithm in terms of cumulative returns and trading risk. Due to its asynchronous architecture, IMPALA is able to sample greater number of experiences and learn more robust trading strategies under same number of iterations, proving that DDRL algorithms, such as IMPALA, should strongly be considered in trading tasks.

Additionally, it is evident that the integration of the technical indicator parameterization method alongside with the Unsupervised component significantly enhanced the trading performance of UNSURE in each market. This combination not only led to improved trading strategies, but also ensured efficiency during the feature selection process, requiring minimal time to calibrate and select proper window sizes of each technical indicator in each market.

Furthermore, it is possible that due to price correlations between the clustered markets, the agents could make use of several features from one market to make better trading decisions in other markets as well. For example, a DRL agent could consider specific features from Bitcoin market that indicate an upcoming trend in Ethereum market, and thus generate better transactions. Another benefit of the Unsupervised component is that by considering a range of similar markets, the agents are trained on a broader range of market trends, patterns, and price changes, enhancing their ability to learn complex trading strategies. This approach also helps to reduce overfitting, which can occur when agents are trained on a single dataset, potentially limiting their ability to generalize well on complex patterns.

Finally, it is noteworthy that each one of these methods can be integrated to other works as well. For instance, instead of using IMPALA, another DDRL algorithm could be employed, such as Apex-DQN, which outperformed IMPALA in several atari games, but with the cost of longer scaling time. As it is also highlighted in Section 2, Apex-DQN has also achieved great performance in energy-trading tasks. Additionally, TCN models of UNSURE could also be replaced by a more advanced regression models, such as Transformers, capable of achieving higher volatility estimation accuracy with the cost of more training time as well. Finally, the combination of the Unsupervised component with the technical indicator parameterization method can easily be employed to enhance the feature quality of the datasets that were used in previous works as well.

6.1 Assumptions & Weaknesses

During the implementation of our unified system, we encountered a variety of assumptions and limitations that need to be highlighted. Initially, our strategy involves a several hyper-parameters that require calibration for each market, as described in Section 5. Furthermore, in order for our system to properly work in real-world markets, we included two critical assumptions during our experimental phase.

First, we make the assumption that the market remains unaffected by our agent's trading activity. Second, we mame the assumption that there are always available assets in the market for our agent to trade. To adhere to these assumptions and to minimize our system's impact on the market's behavior, we have a) restricted the agent to transact only small volume of assets at each timestep (interval) and b) provide the agent with a finite initial budget of 100,000$. Although permitting the agent to handle a larger trading budget and increasing the allowed trading volume per timestep could potentially amplify the returned profits, our strategy aims to optimize returns while maintaining a low profile in the market to avoid significant deviations of the market behavior.

7 Conclusion & Future Work

In this paper, we have integrated three ML Components, namely the Unsupervised, the Supervised and the Reinforcement Learning component, as well as a technical indicator parameterization method, into an end-to-end trading framework, named UNSURE. Our findings demonstrate that UNSURE is a viable option for investors who emphasize on profit returns with low trading risks.

Despite the effectiveness of IMPALA algorithm, in both terms of speed and performance, DRL-based trading agents are not much reliable and can be tricked by frequent market noise. To address these issues, we have incorporated a TCN model to estimate price volatility, alongside with VPM mechanism, in order to avoid high-risk trading periods and enter and exit market positions in an effective way. Furthermore, the clustering algorithm as well as the technical indicator Parameterization method are able to construct high quality inputs and state spaces in a reasonable time, which also increased the overall performance of the trading framework.

Finally, the trading strategies formulated by each individual technical indicator exhibited sub-optimal performance, with some strategies even yielding losses in some markets. This emphasizes the significance of incorporating various machine learning techniques to develop profitable trading strategies that aim to reduce PNL drawdowns and maximize profits. By employing the presented Machine Learning techniques, we constructed an effective and robust trading system for several cryptocurrency markets.

Future work regarding UNSURE framework includes the implementation of several other portfolio-parameterization methods, which could further improve the overall trading performance. Furthermore, a feature importance methodology could be employed for the combined datasets, in order to disregard low-importance features, and thus increase the trading performance, while also reduce the required training time of the components. Moreover, the same methodology could be expanded into other several other trading markets as well, including energy, stocks and additional cryptocurrency markets. The final objective of our future work includes the use of more advanced and recent policy architectures with the use of Autoencoders and Transformers, as well as the use of time-series data to image encoding techniques. The use of more advanced models and further data engineering pipelines have the potential to improve even further the trading strategy of UNSURE.

Appendix A Technical Analysis

A.1 Exponential Moving Average - EMA

EMA is a moving average type of indicator, which is given by Equation A1. EMA is used to smooth price signals, in order to remove market noise.

$$EMA(n) = Price(n) * k + EMA(n-1) * (1-k) \tag{A1}$$

where k is the smoothing constant and is defined as $k = \frac{2}{N+1}$. N parameter defines the *"Look-Back"* sliding window and is used in many technical indicators.

A.2 Double-Exponential Moving Average - DEMA

$DEMA$ is an extension of EMA, given by Equation A2, which attempts to remove price lag that is caused by EMA. The default value of N parameter is 14.

$$DEMA_N = 2EMA_N - EMA\,of\,EMA_N \tag{A2}$$

A.3 Moving Average Convergence/Divergence - MACD

$MACD$ is also a trend type of indicator, which shows the relationship between two moving averages (usually $EMAs$), one with a short period and one with a longer period. The mathematical formula of $MACD$ is provided by Equation A3.

$$MACD = EMA_{N_1} - EMA_{N_2} \tag{A3}$$

with $N_2 >> N_1$. The default values are $N_1 = 12$ and $N_2 = 26$

A.4 Aroon UP/DOWN

$Aroon$, which is described by Equations A4 and A5, identifies trend changes and estimate the strength of an upcoming trend.

$$\frac{25 - N_H}{25} * 100 \tag{A4}$$

$$\frac{25 - N_L}{25} * 100 \tag{A5}$$

with N_H and N_L being the timesteps between a new High price or Low price respectively.

A.5 Commodity Channel Index - CCI

CCI is also trend indicator, which estimates price trends, as well as the direction and strength of a trend. CCI is presented by Equation A6.

$$CCI = \frac{TP(n) - EMA_N(TP)}{MD(TP)} * 0.015 \tag{A6}$$

where, TP the typical price defined as $TP(n) = \frac{High(n)+Low(n)+Close(n)}{3}$ and $MD(TP)$ the Mean Deviation of Typical Price. Typically, $N = 20$.

A.6 Average Directional Index - ADX

ADX is a well know trend type of indicator that measures the strength of a trend and is given by Equation A7

$$ADX = MA * \frac{PDI - NDI}{PDI + NDI} * 100 \tag{A7}$$

where MA is a Moving Average and PDI, NDI are Positive Directional Indicator and Negative Directional Indicator respectively.

A.7 Stochastic Oscillator - STOCH

$STOCH$ is a momentum indicator, given by Equation A8, which is usually used in large trading ranges or to capture slow moving trends.

$$\frac{Close(n) - L_N}{H_N - L_N} \tag{A8}$$

where L_N and H_N are the minimum past value Low value and maximum past High value respectively, within a window of N prices. The default value of N is 14.

A.8 Relative Strength Index - RSI

RSI is a well-studied momentum indicator, which attempts to identify over-bought or over-sold securities. The mathematical formula of RSI is described by Equation A9

$$RSI = 100 - \frac{100}{1 + RS} \tag{A9}$$

where RS is the ratio of Average Price Increases to Average Price Drops within a window of N timesteps. The default period is 14.

A.9 On-Balance Volume - OBV

OBV is a volume type of indicator that measures the volume flow. Its formula is provided by Equation A10.

$$\begin{gathered} OBV(n+1) = OBV(n) + \\ \left\{ \begin{array}{ll} Volume(n) & Close(n+1) < Close(n) \\ 0 & Close(n+1) = Close(n) \\ -Volume(n) & Close(n+1) > Close(n) \end{array} \right\} \end{gathered} \tag{A10}$$

A.10 Bolliger Bands - BBANDS

$BBANDS$ is a volatility indicator, which is provided by Equations A11 and A12 and measures the standard deviation of the simple moving average.

$$BBAND_{UP} = Mean(TP) + 2 * Std(TP) \tag{A11}$$

$$BBAND_{DOWN} = Mean(TP) - 2 * Std(TP) \tag{A12}$$

where $Mean(TP)$ is the average typical price and $Std(TP)$ is the standard deviation of typical price within a look back window N. The default window of Bollinger-Bands is $N = 20$.

A.11 Volume-Weighted Average Price - VWAP

$VWAP$ is a volume-weighted moving average, which is given by Equation A13 and defined as the average price of an asset weighted by the total trading volume over a period of N timesteps. The default period is usually $N = 14$.

$$VWAP(n) = \frac{\sum_{n=0}^{N-1} TP(N-i) * Volume(N-i)}{\sum_{n=0}^{N-1} Volume(N-i)} \tag{A13}$$

A.12 Accumulation/Distribution Line - ADL

ADL is another volume-based indicator, which attempts to measure the underlying supply and demand (bids and asks). The mathematical formula of ADL indicator is described by Equation A14.

$$ADL(n) = ADL(n-1) + \frac{(Close - Low) - (High - Close)}{High - Low} \tag{A14}$$

Appendix B Sensitivity Analysis

Table B1: Analysis of Budget and Transaction Volume parameters in the evaluation datasaet of Bitcoin market.

Volume (assets)	**Budget (\$)**	**SOR(%)**
0.5	50,000	1.631
0.5	100,000	1.949
1.0	50,000	2.152
1.0	100,000	2.896
2.0	100,000	2.588
2.0	**200,000**	**3.616**

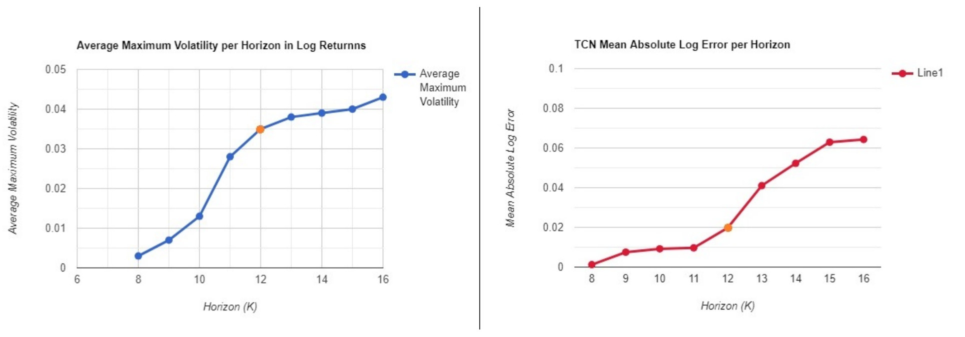

Fig. B1: Average Maximum Volatility (**left**) and the respective TCN's Mean Absolute Log Error (**right**) per Horizon in Bitcoin Market. The orange point represents the selected value of our framework.

Table B2: UNSURE's perforance in terms of PNL and ADD metrics in respect to Virtual Fees parameter in the evaluation dataset of Bitcoin market.

Fees (%)	**Virtual Fees (%)**	**PNL ($)**	**ADD**
1.0	1.0	4.265	0.034
1.0	1.5	**12.087**	0.036
1.0	2.0	6.483	**0.029**

Table B3: Analysis of TCN's performance in respect to Pinball Loss Quantile (τ) parameter in the evaluation dataset of Bitcoin market.

Pinball Quantile	**MOPE**	**MOE**
0.2	0.2253	0.0063
0.3	0.2956	0.0088
0.4	0.3949	0.0113
0.5	0.5152	0.0129

Statements and Declarations

- **Funding** The authors did not receive support from any organization for the submitted work.

- **Conflicts of interest/Competing interests** The authors declare they have no conflict of interests.
- **Financial Interest** The authors have no relevant financial or non-financial interests to disclose.
- **Ethics Approval** Not Applicable.
- **Consent to participate** Not applicable.
- **Consent for publication** Not applicable.
- **Availability of data and material** All data and materials required for this research are provided alongside with the code.
- **Code Availability** Custom code.
- **Authors' contributions** Not applicable.

References

Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:180301271 Preprint on webpage at https://arxiv.org/pdf/1803.01271.pdf

Betancourt C, Chen WH (2021) Deep reinforcement learning for portfolio management of markets with a dynamic number of assets. Expert Systems with Applications 164:114002

Boukas I, Ernst D, Théate T, et al (2021) A deep reinforcement learning framework for continuous intraday market bidding. Machine Learning 110:2335–2387

Bu SJ, Cho SB (2018) Learning optimal q-function using deep boltzmann machine for reliable trading of cryptocurrency. In: Intelligent Data Engineering and Automated Learning–IDEAL 2018: 19th International Conference, Madrid, Spain, November 21–23, 2018, Proceedings, Part I 19, Springer, pp 468–480

Chandar SK (2022) Convolutional neural network for stock trading using technical indicators. Automated Software Engineering 29:1–14

Fang F, Ventre C, Basios M, et al (2022) Cryptocurrency trading: a comprehensive survey. Financial Innovation 8(1):1–59

Guarino A, Grilli L, Santoro D, et al (2022) To learn or not to learn? evaluating autonomous, adaptive, automated traders in cryptocurrencies financial bubbles. Neural Computing and Applications 34(23):20715–20756

Heuillet A, Couthouis F, Díaz-Rodríguez N (2021) Explainability in deep reinforcement learning. Knowledge-Based Systems 214:106685

Hirchoua B, Ouhbi B, Frikh B (2021) Deep reinforcement learning based trading agents: Risk curiosity driven learning for financial rules-based policy. Expert Systems with Applications 170:114553

Huang JZ, Huang W, Ni J (2019) Predicting bitcoin returns using high-dimensional technical indicators. The Journal of Finance and Data Science 5(3):140–155

Jiang Z, Liang J (2017) Cryptocurrency portfolio management with deep reinforcement learning. In: 2017 Intelligent systems conference (IntelliSys), IEEE, pp 905–913

Kochliaridis V, Kouloumpris E, Vlahavas I (2023) Combining deep reinforcement learning with technical analysis and trend monitoring on cryptocurrency markets. Neural Computing and Applications pp 1–18

Lazaridis A, Fachantidis A, Vlahavas I (2020) Deep reinforcement learning: A state-of-the-art walkthrough. Journal of Artificial Intelligence Research 69:1421–1471

Lin TC (2012) The new investor. UCLA L Rev 60:678

Mamoghli C, Daboussi S (2009) Performance measurement of hedge funds portfolios in a downside risk framework. The Journal of Wealth Management 12(2):101

Müller M (2007) Dynamic time warping. Information retrieval for music and motion pp 69–84

Naik N, Mohan BR (2019) Optimal feature selection of technical indicator and stock prediction using machine learning technique. In: Emerging Technologies in Computer Engineering: Microservices in Big Data Analytics: Second International Conference, ICETCE 2019, Jaipur, India, February 1–2, 2019, Revised Selected Papers 2, Springer, pp 261–268

Nazareth N, Reddy YYR (2023) Financial applications of machine learning: a literature review. Expert Systems with Applications p 119640

Pendharkar PC, Cusatis P (2018) Trading financial indices with reinforcement learning agents. Expert Systems with Applications 103:1–13

Peng Y, Albuquerque PHM, Kimura H, et al (2021) Feature selection and deep neural networks for stock price direction forecasting using technical analysis indicators. Machine Learning with Applications 5:100060

Pierros I, Vlahavas I (2022) Architecture-agnostic time-step boosting: A case study in short-term load forecasting. In: Artificial Neural Networks and Machine Learning–ICANN 2022: 31st International Conference on Artificial Neural Networks, Bristol, UK, September 6–9, 2022, Proceedings, Part III, Springer, pp 556–568

Schnaubelt M (2022) Deep reinforcement learning for the optimal placement of cryptocurrency limit orders. European Journal of Operational Research 296(3):993–1006

Somers M, Whittaker J (2007) Quantile regression for modelling distributions of profit and loss. European Journal of Operational Research 183(3):1477–1487

Son Y, Vohra S, Vakkalagadda R, et al (2022) Using transformers and deep learning with stance detection to forecast cryptocurrency price movement. In: 2022 13th International Conference on Information and Communication Technology Convergence (ICTC), IEEE, pp 1–6

Zhang CX, Li J, Huang XF, et al (2022) Forecasting stock volatility and value-at-risk based on temporal convolutional networks. Expert Systems with Applications 207:117951