mTrader: A Multi-Scale Signal Optimization Deep Reinforcement Learning Framework for Financial Trading

Zhennan Chen, Zhicheng Zhang, Pengfei Li, Lingyue Wei, Shibo Feng, Fan Lin*
School of Informatics, Xiamen University, Xiamen, China
{znchen, zczhang, lpfei, 31520211154091, fengshibo1024}@stu.xmu.edu.cn, *iamafan@xmu.edu.cn

***Abstract*—It is universally acknowledged that the financial trading is a thorny issue in time-series scenarios. On the one hand, due to the great randomness and instability in financial markets, the existing machine learning methods are inadequate for modeling high-frequency financial data. On the other hand, it remains a challenge to identify the validity of transaction actions to avoid high fees. To address these issues, we propose a novel trading framework, namely mTrader, to offer suitable trading strategies automatically. We creatively design a multi-scale signal matrix to describe the temporal trends of markets. On this basis, Vector Quantized Variational AutoEncoder (VQ-VAE) was introduced to capture discrete latent variables. In addition, an offline Action Optimizer (AO) based on Proximal Policy Optimization (PPO) could help filter out sub-optimal trading action. Extensive experiments have shown that our model achieves state-of-the-art performance on many popular stock markets.**

***Keywords*—Multi-Scale, VQ-VAE, Proximal Policy Optimization (PPO), Financial Trading**

I. Introduction

Financial trading aims to obtain optimal returns while avoiding market fluctuations. During the past few decades, traders have proposed many trading strategies [4], [14], but poor usability resulted in these traditional strategies only helping traders to trade at a fixed time or in a fixed market. Due to the huge demand for funds, new intelligent trading algorithms have been proposed to help traders adapt to different market changes.

Processing financial transaction data is considered to be one of the most challenging tasks due to the evolutionary and nonlinear nature of financial markets. As the global financial market gradually becomes large and complex, the number of statistical models [2] and sophisticated models [21] has risen during the recent years. However, traditional models show insufficient processing ability while facing with large input states. Thus methods based on Deep Reinforcement Learning (DRL) have become the preferred choice. For instance, deep Q-network (DQN) [12] is proposed to learn policies from high-dimensional inputs, in order to suit the situation approaching real-world complexity. Jiang et al. [5] use the model-free Deep Deterministic Policy Gradient (DDPG) to deal the portfolio management problem with a deep learning solution. Schulman et al. [16] present Proximal Policy Optimization (PPO) to hedge a portfolio of derivatives against market frictions. Despite numerous attempts to apply DRL in financial markets, unpredictable factors can still affect prediction results, leading to highly non-stationary time series that make it challenging to observe real markets with extreme volatility. Additionally, the input states often contain a large amount of noise that interferes with the selection of subsequent actions.

To address these challenges, we propose a novel deep reinforcement learning framework, mTrader, which models the financial trading process as a Partially Observable Markov Decision Process (POMDP) [6]. In our framework, we design a multi-scale signal optimization approach to describe features of different markets using a sparse signal matrix, which helps to capture the temporal trends of markets. We then employ a Vector Quantized Variational AutoEncoder (VQ-VAE) [19] to observe current market states precisely. Moreover, we introduce an offline Action Optimizer (AO) based on Proximal Policy Optimization (PPO) to filter out sub-optimal trading actions, enabling the framework to maximize returns even in the presence of market fluctuations. Our proposed framework achieves state-of-the-art performance on various popular stock markets. Our main contributions are summarized as follows:

- We proposed mTrader, a novel multi-scale signal optimization framework, which can automatically make suitable decisions during financial transaction process.
- To extract meaningful features, we use a multi-scale sparse signal matrix for features expression and use VQ-VAE for temporal modeling of high frequency financial data. To our knowledge, this is the first effective application of VQ-VAE in this field. We introduce an offline Action Optimizer (AO) in order to identify the validity and rationality of certain transaction behaviors.
- We perform extensive experiments to demonstrate the advantages of our framework over other algorithms and its versatility in different markets.

II. Related Work

1) Feature Extraction: It is crucial to extract high-level features from financial data for accurate prediction of stock prices. Tsantekidis et al. [18] described a deep learning method

*Corresponding author
DOI reference number: 10.18293/SEKE2023-168

based on convolutional neural networks (CNN) to predict the price movement of stocks. Liu et al. [9] used LSTM recurrent neural network to extract feature values, established a corresponding stock trading prediction model by analyzing stock data. All in all, it turns out that CNN and LSTM are undoubtedly the most popular feature extraction methods. Since financial data is the sequence at regular intervals of time and LSTM has superior performance in processing series data, using LSTM for feature extraction is likely to be a more suitable choice for financial data analysis and stock price prediction.

2) Deep Reinforcement Learning: Owing to the outstanding ability of DRL to solve complex continuous decision-making problems, applications in financial market transactions have emerged one after another. Some researchers have used critic-only approach, usually DQN [11] and its variants, to solve financial transaction tasks. Other researchers have focused on actor-only approaches, where Moody et al. [13] creatively proposed an adaptive algorithm using recurrent reinforcement learning (RRL) to optimize risk for better return on investment. On the other hand, some researchers have used actor-critic approach. For example, Yang et al. [20] utilized three algorithms to obtain an ensemble policy that can robustly adapt to different market situations for better learning stock trading strategy. To sum up, in terms of the characteristics of financial transaction tasks, the actor-critic approaches [15] can learn and adapt to complex environments better, and is more suitable for financial transaction tasks.

III. Problem Formulation

In this section, we first clarify the full definition of POMDP then formally introduce the financial trading problem in DRL.

A. POMDP

The POMDP is a realistic generalization of a Markov Decision Process (MDP) for the financial trading problem. An MDP is a 5-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_T, \mathcal{R}, \gamma)$. $\mathcal{S}$ is a finite set of states. $\mathcal{A}$ is a finite set of actions. $\mathcal{P}_T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ denotes the state transition function. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, where $\mathbb{R}$ is a continuous set of possible rewards. $\gamma \in [0, 1)$ is the discount factor. The goal of this process is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected discounted cumulative reward $\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1}\mathbb{R}_t\right]$. The action value function $\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1}\mathbb{R}_t \mid \pi\right]$ is used to evaluate policy π.

In our financial trading framework, $\mathcal{O} = \{\mathbf{Z}, \mathcal{R}\}$ is a set of observations that contains our constructed feature encoding $\mathbf{Z}$ and reward $\mathcal{R}$. $\mathcal{B} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$ is the observation transition function. $\mathcal{O}$ and $\mathcal{B}$ are the remaining components of POMDP.

At each timestep t, the agent takes action $a_t \in \mathcal{A}$ according to the current environment $s_t \in \mathcal{S}$. Then get s_{t+1} through the state transition function $\mathcal{P}_T\left(s_{t+1} \mid s_t, a_t\right)$. At the same time as the environment s_{t+1} is obtained, the agent will get $o_{t+1} \in \mathcal{O}$ through the observation transition function $\mathcal{B}\left(o_{t+1} \mid s_{t+1}, a_t\right)$.

B. State

Previously, researchers mostly used the primitive properties of the raw data to represent financial market signals. In this paper, we perform state augmentation using technical indicators recognized by financial scientists as helpful in observing the market, and further process it into a multi-scale sparse signal matrix at time t. After encoding, the state is got and input into the DRL model.

C. Action

Like real market transactions, our action space is discrete. The action of period t is $a_t := \{a_b, a_h, a_s\}$, where a_b means to buy, a_s means to sell, and a_h means no action to buy or sell. The amount of each action is fixed.

D. Reward

The reward of period t is the Sharpe ratio. Sharpe ratio (Sr), a measure that considers both return and risk, is used to adjust past or expect future performance of a portfolio for excess risk taken by traders, defined as the average of the risk-free return by its deviation:

$$\mathrm{Sr} = \frac{\mathbb{E}(R_p - R_f)}{\sigma(\mathbf{r})}, \tag{1}$$

where R_p is the rate of return, R_f is the risk-free rate, $\sigma(\mathbf{r})$ is the standard deviation of returns. To accurately measure trading action, Sr is calculated over the past n sliding windows.

E. Assumptions

In our model, there are two general assumptions: 1) Taking action will not have an impact on the market. Markets for financial transactions are resilient, and the impact of agency investment on the market will recover before the next step. 2) Immediate execution of the market. We assume that the agent's behavior can be implemented directly, that is, there is no delay in each investment.

IV. Proposed Method

The overview architecture of mTrader is shown in Figure 1. In this section, we will introduce our model, mTrader, which is designed to solve decision-making problems in financial trading.

A. Multi-Scale Signal Modeling

1) Multi-Scale Signal Metrics: The special mode of the sparse signal matrix makes it have strong feature expression ability in financial data. According to the results of past research [1], [8], we have selected 32 indicators that can reflect market trends best, such as RSI, MACD, KDJ etc. In order to eliminate irrelevant or strongly correlated features among the 32 technical indicators while expanding the data expression ability to the greatest extent, we use Least Absolute Shrinkage and Selection Operator Algorithm (LASSO) [17] for further feature selection to ensure that each indicator can contribute to the final result. In this operator, features with negative or zero coefficient values can be removed from the feature subset, and

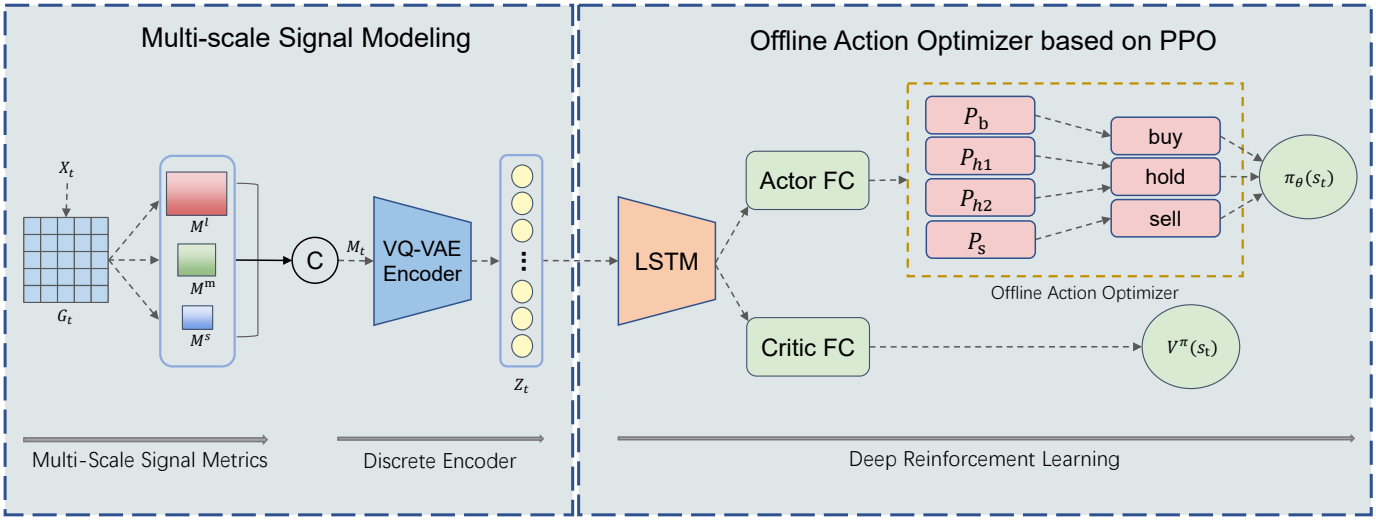


Fig. 1: The overview of mTrader. X_t is the original data. G_t is the processed sparse signal matrix. M^l, M^m, M^s are signals of three scales. ⓒ is concatenation. M_t is encoded by VQ-VAE to generate discrete encoding Z_t. The right part is the DRL algorithm. P_s, P_b, P_{h1} and P_{h2} set boundaries to optimize actions. $\pi_\theta(s_t)$ is the generated policy and $v^\pi(s_t)$ is the value function.

finally the most common feature is taken as the most important feature. Its estimated value is defined as follows:

$$\hat{\boldsymbol{\beta}}_\alpha = \arg\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} \left(y_i - \sum_j x_{ij}\beta_j \right)^2 + \alpha \sum_{j=1}^{p} |\beta_j| \right), \tag{2}$$

where $\boldsymbol{\beta} \in \mathbb{R}^p$ and represents the coefficient of each feature, α is an adjustable penalty parameter, and some inactive components of $\hat{\boldsymbol{\beta}}_\alpha$ can be precisely set to 0.

The input raw data $X_t = \{X_O, X_H, X_L, X_C\} = [x_{t1}, ..., x_{tk}]$ includes the opening price indicator X_O, the highest price indicator X_H, the lowest price indicator X_L and the closing price indicator X_C. We judge each technical index after feature screening, and treat X_t as a sparse signal matrix G_t containing only -1, 0, 1. We introduce multi-scale information. For M_t of sliding window, M_t^s describes the strong correlation of transactions, M_t^m focuses on descriptiveness, and M_t^l shows trend. M_t is defined as follows:

$$M_t = \begin{bmatrix} \vdots & & \vdots & & \vdots \\ (M_t^s)_j & \cdots & (M_t^m)_j & \cdots & (M_t^l)_j \\ \vdots & & \vdots & & \vdots \end{bmatrix}. \tag{3}$$

2) VQ-VAE Encoder: VQ-VAE [19] is a self-supervised model that includes an encoder that maps observations to a sequence of discrete latent variables, and a decoder that reconstructs observations from these discrete variables. VQ-VAE is more effective than traditional encoding methods for modeling financial discrete signals because it maintains the consistency between the input and output in a discrete format, allowing it to capture the sequential nature of financial data and model complex patterns more effectively.

VQ-VAE includes three parts: encoder, vector quantization codebooks, and decoder. We define $\mathbf{E}^{\mathrm{V}} = [e_1, .., e_K] \in \mathbb{R}^{K \times D}$ as a collection of codebook vectors. To transform the multi-scale sparse signal matrix M_t into a discrete variable, the encoder first produces intermediate continuous representation $Z_e^{\mathrm{V}} \in \mathbb{R}^D$. Later on, we scan through all codebooks to find which codebook has a minimum distance between Z_e^{V} and a vector in $\mathbf{E}^{\mathrm{V}}$. After the closest codebook index $k \in \{1, .., K\}$ is found, we substitute latent variable Z_t^{V} with nearest codebook vector e_k. Then, the decoder uses codebook vector e_k to reconstruct input feature M_t. The definitions are as follows:

$$Z_e^{\mathrm{V}} = C(M_t), \tag{4}$$

$$Z_t^{\mathrm{V}} = VectorQuantization(Z_e^{\mathrm{V}}) = e_k, \tag{5}$$

$$\hat{M}_t = D(Z_t^{\mathrm{V}}). \tag{6}$$

Given input reconstruction across all timestep $\forall t \in$ [1,T], the VQ-VAE is trained using the following objective:

$$\mathcal{L} = \|M_t - \hat{M}_t\|_2^2 + \|sg[Z_e^{\mathrm{V}}] - Z_t^{\mathrm{V}}\|_2^2 + \varphi\|Z_e^{\mathrm{V}} - sg[Z_t^{\mathrm{V}}]\|_2^2, \tag{7}$$

where sg refers to a stop-gradient. The hyper-parameter φ scales the commitment loss. The first term is used to measure the reconstruction loss between the original input M_t and the reconstructed $\hat{M}_t$. The second and third term minimize the distance between intermediate representation Z_e^{V} and Z_t^{V}.

B. Action Optimizer based PPO

The offline Action Optimizer (AO) will further analyze the actions obtained from PPO, and discard invalid actions, thereby avoiding losses or high fees.

1) Proximal Policy Optimization (PPO): In our trading framework, we use the the PPO algorithm [16], which is an actor-critic method with well-proven performance. To optimize the policy, PPO learns the policy through iteratively sampled data interacting with the environment and optimizes the surrogate objective function using stochastic gradient ascent. It alternates between the sampled data and optimizing the surrogate objective function defined as follows,

$$L^{\mathrm{C}}(\theta)=\hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t,\mathrm{clip}\left(r_t(\theta),1-\epsilon,1+\epsilon\right)\hat{A}_t\right)\right], \tag{8}$$

$$L^{\mathrm{KL}}(\theta)=\hat{\mathbb{E}}_t[(r_t(\theta)\hat{A}_t]-\beta KL\left(\pi_{\mathrm{old}}\|\pi_\theta\right), \tag{9}$$

where θ is the policy parameter. The updated β is used for the next policy update. $[1-\epsilon,1+\epsilon]$ is the probability ratio of the clipping region.

$r_t(\theta)$ is the probability ratio,

$$r_t(\theta)=\frac{\pi_\theta\left(a_t,s_t\right)}{\pi_{\theta_{old}}\left(a_t,s_t\right)}, \tag{10}$$

where π is the policy. PPO uses importance sampling to prevent the new policy going far away from the old policy.

$\hat{A}_t$ is the advantage estimator,

$$\hat{A}_t=\delta_t+(\gamma\lambda)\delta_{t+1}+\cdots+(\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{11}$$

where T is the total timesteps in an episode, γ is the discount factor, and λ is a hyperparameter to control the bias-variance trade-off. $\delta_t=r_t+\gamma V\left(s_{t+1}\right)-V\left(s_t\right)$.

In addition, we introduce LSTM [3] into PPO, hoping that the reinforcement learning algorithm can clarify the relationship between high-frequency data. Compared with modeling in the original data, this method has stronger expressive ability and weaker noise features, which is particularly critical for the subsequent selection of actions in reinforcement learning.

2) Offline Action Optimizer (AO): In order to keep agents' actions within bounds at all times and trade in the most efficient way, we propose an efficient offline Action Optimizer (AO) to optimize policies with boundaries. Inspired by methods for solving reinforcement learning via convex constraints [10], our AO allows for effective policy optimization using reinforcement learning by constraining the space of possible actions.

We define the AO as the boundaries, P_s, P_b are the upper and lower boundaries, which are related to buying and selling, respectively. P_{h1}, P_{h2} are the limits used to control the hold action,

$$P_s(i+1)=\sum_{i=1}^{n}\frac{p_c(i+1)-p_c(i)}{p_c(i)}(1+m_1), \tag{12}$$

$$P_b(i+1)=\sum_{i=1}^{n}\frac{p_c(i+1)-p_c(i)}{p_c(i)}(1-m_1), \tag{13}$$

$$P_{h1}(i+1)=\sum_{i=1}^{n}\frac{p_c(i+1)-p_c(i)}{p_c(i)}(1+m_2), \tag{14}$$

TABLE I: Price data ranges for three datasets.

Dataset	Training Period	Testing Period
Bitcoin	2018-05-01 to 2018-08-01	2018-08-01 to 2018-09-01
Tesla	2021-05-01 to 2021-08-01	2021-08-01 to 2021-09-01
CATL	2021-02-01 to 2021-06-01	2021-06-01 to 2021-07-01

$$P_{h2}(i+1)=\sum_{i=1}^{n}\frac{p_c(i+1)-p_c(i)}{p_c(i)}(1-m_2), \tag{15}$$

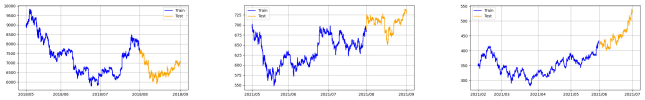
where m_1 and m_2 are margin parameters within window size n. $p_c(i)$ is the closing price on day i. Once the action given by the model exceeds the bounds set by AO, it means that action needs to be optimized. If the P_s boundary is crossed upwards, the trading action needs to be optimized to sell. In the same way, if the P_b boundary is crossed down, the trade action needs to be optimized to buy. The middle boundary is set to prevent frequent invalid transactions. If the action given by the model cannot break through the P_{h1}, P_{h2} boundary, we hope to continue to hold it, otherwise the cost of handling fees will be huge.

V. Experiments

In this section, we back-test mTrader in three completely different markets, define evaluation metrics, compare with other baselines, and evaluate its performance.

A. Experimental Setup

1) Datasets and Baselines: We selected the most representative five-minute datasets from three different markets: Bitcoin, Tesla, and Contemporary Amperex Technology Co. Limited (CATL) to verify the generalization and robustness of our model, as shown in Figure 2. The training and test datasets' period division is illustrated in Table I. The three datasets represent the cryptocurrency market, and it should be noted that their overall trends are completely different, which can better test the model's capability. We compared our proposed mTrader with three baselines, DQN [12], DDPG [5], and PPO [16].



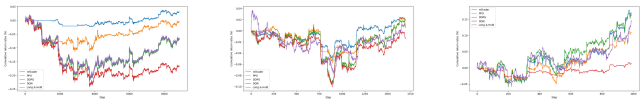
(a) Bitcoin. (b) Tesla. (c) CATL.

Fig. 2: Closing price variation of Bitcoin, Tesla and CATL.

2) Metrics: Four commonly used metrics are used in our experiments, including Annualized Sharpe Ratio (ASr), Annualized Volatility (AVol), Maximum Drawdown (MDD), Total Return Rate (Tr). The ASr is the average return earned in excess of the risk-free rate per unit of volatility or total risk. The AVol is a statistical measure of returns. In general, greater volatility means greater uncertainty. MDD denotes the biggest loss from a peak to a trough. Tr is the most important objective of financial trading to make profits. Therefore, the final accumulative value reflects the performance of models.

B. Experiment Results

The performances of all the evaluated methods in the three testing datasets are summarized in Table II. Figure 3 shows the cumulative returns across three datasets.



(a) Bitcoin. (b) Tesla. (c) CATL.

Fig. 3: Cumulative return rates on Bitcoin, Tesla and CATL.

From Table II, our results show that mTrader outperforms all other models on all metrics for the Bitcoin and Tesla datasets, achieving state-of-the-art results on the ASr and Tr metrics. The DQN model performs best on AVol and MDD, but its performance on Tr is much lower than the other models, so we guess that it may be trapped in a local optimal solution. Furthermore, we observed that the PPO algorithm outperforms other DRL algorithms, indicating that mTrader based on the PPO algorithm can generate higher excess returns, while other algorithms result in losses.

In the Bitcoin dataset, the overall trend was characterized by a rapid decline. As shown in Figure 3(a), mTrader not only avoided the peak of the decline, but also grasped the profit points in a short period of time when the whole market was in a downturn. This result demonstrates that the AO plays a crucial role in ensuring the correctness of the output action. In the Tesla dataset, mTrader's ASr value far exceeds DQN, a method that has been proven to be extremely prominent in the financial field in other papers. Figure 3(b) shows that our method still avoided the most violent moment of fluctuation in the middle and obtained a profit more quickly than when it rose rapidly. In the CATL dataset illustrated in Figure 3(c), the overall fluctuation of our model was relatively stable, and the maximum return was reached at the last moment. The existence of AO allows the model to avoid market declines while limiting its performance in the rapid upward phase.

The above experiments can prove that our proposed model is capable of producing high returns while also being resistant to market downturns in various financial markets.

C. Impact of Hyper-Parameters

To further investigate the impact of feature selection on multi-scale signal modeling, we conducted experiments on the m_1 and m_2 parameters in AO. The results of these experiments, displayed in Figure 4, indicate that our proposed method achieved the best performance on all three datasets when $m_1 = 0.15$ and $m_2 = 0.01$. In contrast, performance decreased when $m_1 = 0.10$ and $m_2 = 0.005$. In two of the three experiments with $m_1 = 0.15$, the performance was higher than when $m_1 = 0.10$, suggesting that loosening the P_s and P_d boundaries of AO can enhance model performance.

Furthermore, the experiment on m_2 demonstrated that narrowing the boundary between P_{h1} and P_{h2} can lead to better

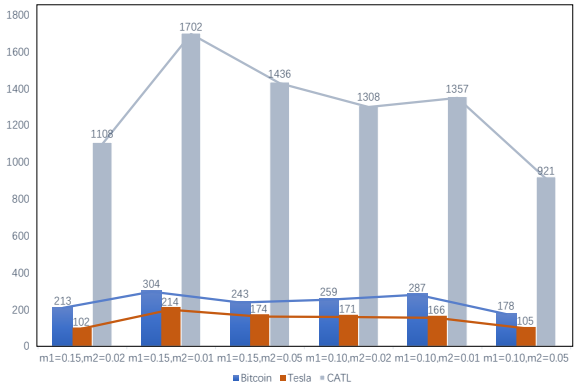


Fig. 4: Cumulative returns of different m_1 and m_2 value on Bitcoin, Tesla and CATL.

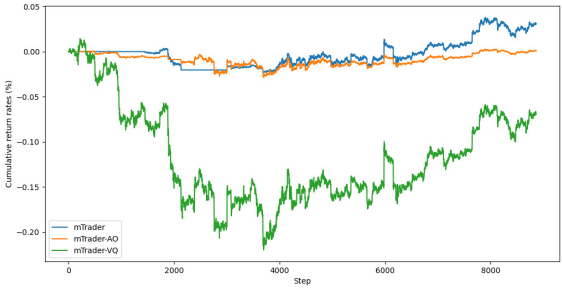


Fig. 5: Cumulative return rates on Bitcoin.

trading outcomes. As m_1 and m_2 are kept within a reasonable range, our model outperforms the trading framework without AO, indicating that AO can effectively maximize returns and has sufficient generalization capabilities.

D. Ablation Experiments

We conducted ablation experiments on Bitcoin to evaluate the impact of different components of mTrader on its performance. In particular, mTrader-VQ refers to the method with the VQ-VAE module, and mTrader-AO refers to the method with the AO module. The results of these experiments are presented in Table III, and the cumulative return rates are summarized in Figure 5. The results suggest that the AO module significantly improves the stability of the method. Additionally, the method with the VQ-VAE module correctly responds to market upswings demonstrates the successful removal of noise and extraction of valid signals. From Figure 5, by leveraging the strengths of both modules, mTrader achieved a higher cumulative return rate compared to the other two methods for most of the time.

To further demonstrate the role of the VQ-VAE encoder, we use a VAE encoder [7] that produces a continuous distribution named mTrader-VAE. As shown in Table IV, each metric of mTrader-VAE is worse than using the VQ-VAE encoder. This means that the feature of discrete latent variables exploits the properties of our designed multi-scale signal matrix.

TABLE II: Performance of comparison methods. The best results are marked in **bold**.

Models	Bitcoin				Tesla				CATL			
	ASr	AVol	MDD(%)	Tr(%)	ASr	AVol	MDD(%)	Tr(%)	ASr	AVol	MDD(%)	Tr(%)
DQN	-3.37	0.59	24.27	-16.69	-0.45	0.64	10.09	1.23	9.24	**1.04**	**3.67**	5.56
DDPG	-1.01	0.64	22.16	-6.98	1.56	0.71	9.90	0.82	11.62	1.49	5.91	16.45
PPO	0.01	0.35	11.49	-0.54	2.79	0.44	5.83	1.28	10.34	1.35	5.71	10.98
Long & Hold	-0.89	0.65	21.77	-6.41	0.94	0.63	7.9	0.6	10.47	1.50	5.83	15.01
mTrader	**2.36**	**0.16**	**2.34**	**3.04**	**3.46**	**0.41**	**5.13**	**2.14**	**14.01**	1.25	5.60	**17.02**

TABLE III: Ablation experiments on Bitcoin.

Models	Bitcoin			
	Asr	AVol	MDD(%)	Tr(%)
mTrader-AO	0.15	**0.12**	2.87	-0.09
mTrader-VQ	-1.04	0.63	21.97	-6.94
mTrader-VAE	-1.41	0.68	25.69	-8.72
mTrader	**2.36**	0.15	**2.34**	**3.04**

VI. Conclusion

In this paper, we proposed a novel framework, mTrader, which uses multi-scale signal optimization deep reinforcement learning for financial trading. We introduced a complete feature augmentation and screening process to model minute-level data as multi-scale sparse matrices, and used VQ-VAE to generate discrete embeddings. We also incorporated the PPO and LSTM to effectively capture the time-series relationship. Our offline Action Optimizer (AO) further improved the stability of the trading framework. Our experiments on multiple markets with different properties confirmed the superiority of our proposed model.

We further aim to explore how to model macro financial indicators and the interplay between multiple stocks or cryptocurrencies, and even different markets. We also plan to test the trading framework on a real financial engine and explore the possibility of applying mTrader to other time-series tasks.

References

[1] Agrawal, J., Chourasia, V., Mittra, A.: State-of-the-art in stock prediction techniques. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering **2**(4), 1360–1366 (2013)

[2] Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time series analysis: forecasting and control. John Wiley & Sons (2015)

[3] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[4] Hong, H., Stein, J.C.: A unified theory of underreaction, momentum trading, and overreaction in asset markets. The Journal of finance **54**(6), 2143–2184 (1999)

[5] Jiang, Z., Xu, D., Liang, J.: A deep reinforcement learning framework for the financial portfolio management problem. arXiv preprint arXiv:1706.10059 (2017)

[6] Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial intelligence **101**(1-2), 99–134 (1998)

[7] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)

[8] Kouatli, I., Yunis, M.: A guide to stock-trading decision making based on popular technical indicators. In: 2021 International Conference on Decision Aid Sciences and Application (DASA). pp. 283–287. IEEE (2021)

[9] Liu, S., Liao, G., Ding, Y.: Stock transaction prediction modeling and analysis based on lstm. In: 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA). pp. 2787–2790. IEEE (2018)

[10] Miryoosefi, S., Brantley, K., Daume III, H., Dudik, M., Schapire, R.E.: Reinforcement learning with convex constraints. Advances in Neural Information Processing Systems **32** (2019)

[11] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)

[12] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)

[13] Moody, J., Saffell, M.: Learning to trade via direct reinforcement. IEEE transactions on neural Networks **12**(4), 875–889 (2001)

[14] Poterba, J.M., Summers, L.H.: Mean reversion in stock prices: Evidence and implications. Journal of financial economics **22**(1), 27–59 (1988)

[15] Pricope, T.V.: Deep reinforcement learning in quantitative algorithmic trading: A review. arXiv preprint arXiv:2106.00123 (2021)

[16] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)

[17] Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological) **58**(1), 267–288 (1996)

[18] Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., Iosifidis, A.: Forecasting stock prices from the limit order book using convolutional neural networks. In: 2017 IEEE 19th conference on business informatics (CBI). vol. 1, pp. 7–12. IEEE (2017)

[19] Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. Advances in neural information processing systems **30** (2017)

[20] Yang, H., Liu, X.Y., Zhong, S., Walid, A.: Deep reinforcement learning for automated stock trading: An ensemble strategy. In: Proceedings of the First ACM International Conference on AI in Finance. pp. 1–8 (2020)

[21] Yoo, P.D., Kim, M.H., Jan, T.: Machine learning techniques and use of event information for stock market prediction: A survey and evaluation. In: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06). vol. 2, pp. 835–841. IEEE (2005)