Expert Systems With Applications 289 (2025) 128297

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Enhancing trading strategies by combining incremental reinforcement learning and self-supervised prediction

Chujin Zhou [a], Yuling Huang [b], Yuting Kong [c], Xiaoping Lu [a,*]

[a] School of Computer Science and Engineering, Macau University of Science and Technology, Macao, China
[b] School of Computer Science and Software, Zhaoqing University, Zhaoqing Guangdong, China
[c] School of Computers, Guangdong University of Science and Technology, Dongguan Guangdong, China

ARTICLE INFO

Keywords:
Deep reinforcement learning
Self-supervised learning
Incremental learning
Algorithmic trading

ABSTRACT

Incremental learning provides critical adaptability in dynamic environments, enabling models to continuously adjust with new data to improve prediction accuracy. This adaptability is especially valuable in volatile financial markets, where incremental learning can help model better capture the emerging patterns of time series data. On the other hand, self-supervised learning gained significant attention during the past few years due to its ability to exploit abundant unlabeled data to uncover complex structures and temporal dependencies, improving model generalization and pattern detection, which can provide a more comprehensive understanding of data structures. In this case, it gradually become a powerful tool in the field of time series analysis, especially in financial domain. In parallel, deep reinforcement learning (DRL) has shown great potential in decision-making tasks, particularly in financial strategy optimization. However, most DRL approaches in finance rely solely on raw market data, limiting the model's ability to extract key insights and often ignoring future trends essential for profitable trading. To address the aforementioned challenge appeared in DRL domain, Incremental Forecast Fusion Deep Reinforcement Learning (IFF-DRL), an innovative framework that combines incremental learning and self-supervised learning with DRL for financial trading is proposed, for accurately and continually optimizing and improving trading strategies. In this paper, AutoConNet, as a self-supervised learning method, is incorporated to forecast future OHLCV (Open, High, Low, Close, and Volume) data based on observed values. During testing, incremental learning dynamically refines the predictive model to stay aligned with market trends. By utilizing incremental learning algorithms, the self-supervised time series prediction network, AutoConNet, achieved an average reduction of 5.93 % in MSE error across six datasets during the testing phase. The predicted OHLCV is combined with actual observed OHLCV to generate "weekly data", forming the state space termed "daily & weekly data" in reinforcement learning. Experiments conducted on six datasets show that the proposed IFF-DRL significantly improves trading performance, delivering an annualized return of 103.19 % on the HSI index. By combining incremental learning with reinforcement learning, IFF-DRL enables trading agents to adapt more effectively in fast-paced markets, capturing profit opportunities and offering a more responsive, future-oriented approach in financial decision-making. Code of this research can be found in https://github.com/AndyZCJ/IFF-DRL-Inference.

1. Introduction

Accurately predicting and making timely decisions in dynamic, high-stakes environments like financial markets has long been a central challenge in artificial intelligence. While recent advancements in financial time series modeling have achieved some success in capturing market patterns (Wang et al., 2024; Weng, Ahmed, & Megahed, 2017; Weng, Lu, Wang, Megahed, & Martinez, 2018), these models often fail to respond effectively to rapid and abrupt market changes, leading to sub-optimal trading outcomes. A key limitation of traditional models lies in their offline training on historical data, which prevents them from adapting in real time as new data becomes available. This constraint has

Abbreviations: RL, Reinforcement Learning; DRL, Reinforcement Learning; DRL, Deep Reinforcement Learning; DQN, Deep Q-Network; DDQN, Double Deep Q-Network; PPO, Proximal Policy Optimization; A2C, Advantage Actor-Critic; CR, Cumulative Return; AR, Annualized Return; SR, Sharpe Ratio; MDD, Maximum Drawdown; B&H, Buy and Hold; S&H, Sell and Hold; MR, Mean Reversion with Moving Averages; TF, Trend Following with Moving Averages.

* Corresponding author.
E-mail addresses: 3220002751@student.must.edu.mo (C. Zhou), huangyuling@zqu.edu.cn (Y. Huang), kongyuting@gdust.edu.cn (Y. Kong), xplu@must.edu.mo (X. Lu).

https://doi.org/10.1016/j.eswa.2025.128297
Received 25 November 2024; Received in revised form 14 January 2025; Accepted 22 May 2025
Available online 3 June 2025

motivated the exploration of incremental learning as a promising approach for dynamic data prediction (Li & Dai, 2019; Li, Dai, & Ye, 2019; Melgar-García, Gutiérrez-Avilés, Rubio-Escudero, & Troncoso, 2023). In financial markets, incremental learning allows continuous model updates as new data arrives, enabling models to quickly adapt to evolving data patterns and improve prediction accuracy without retraining from scratch.

Reinforcement Learning (RL) has gained considerable attention due to its capacity to optimize decision-making in uncertain, dynamic environments, including gaming (Kaiser et al., 2019; Lample & Chaplot, 2017; Zhou, Subagdja, Tan, & Ong, 2021b), fine-tuning of large language models (Achiam et al., 2023; Zhao et al., 2023), and notably, algorithmic trading (Huang, Wan, Zhang, & Lu, 2024a; Huang, Zhou, Cui, & Lu, 2024c). The dynamic and unpredictable nature of financial markets makes them particularly suitable for RL applications, as RL agents can iteratively learn and improve trading strategies based on continuous feedback from the market. However, despite the development of robust algorithms and frameworks in applying RL methods within the financial domain, notable shortcomings persist, particularly in data handling. Specifically, a significant portion of existing work relies solely on raw market price data, such as daily Open, High, Low, and Close prices, as well as trading Volume (OHLCV). Some studies even use only the closing price as the model input (Chen & Gao, 2019; Li, Ni, & Chang, 2020). This reliance on static or offline data limits RL models' ability to capture rapid market shifts, which are often essential to seizing profitable trading opportunities. Although recent efforts have attempted to increase data diversity by incorporating additional indicators, such as technical signals or social media sentiment (Friedman & Fontaine, 2018; Yang, Liu, Zhong, & Walid, 2020), this approach introduces latency, as these indicators are generated after the fact and fail to capture immediate market changes.

Moreover, most RL models rely primarily on historical price movements, often neglecting the potential impact of external factors such as political events, wars, or international relations. These factors can lead to sudden and new market shifts, making trading decisions based solely on historical data risky, as it may result in missed opportunities or substantial losses. Consequently, there is a pressing need for models that not only learn from historical data but also adapt to evolving market conditions and external influences.

To address these challenges, this paper introduces a hybrid reinforcement learning framework that combines incremental learning with Self-Supervised Learning (SSL) for enhanced financial decision-making. Incremental learning enables the model to continuously update as new data arrives, adapting to evolving market trends without retraining from scratch, which is crucial in financial markets where rapid adaptation is essential. By integrating incremental learning, the model can maintain performance over time, effectively handling non-stationary data distributions and mitigating the effects of concept drift (Li & Dai, 2019; Li et al., 2019; Melgar-García et al., 2023). Self-supervised learning leverages large amounts of unlabeled data abundant in financial markets. SSL creates auxiliary tasks that uncover complex structures and temporal dependencies within the data without requiring manual labels, enhancing the model's ability to generalize and detect subtle shifts in patterns. Additionally, SSL is effective in producing robust predictions, even in the presence of market noise and volatility.

In our framework, we integrate a self-supervised time-series prediction network with the classical incremental learning algorithm Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) within a deep reinforcement learning architecture. The self-supervised network predicts stock price movements over the next N days, enabling the RL agent to anticipate future price changes. To capture both daily fluctuations and macro-level patterns, the predicted stock prices are combined with observed prices to generate weekly data. This weekly data is then concatenated with daily data, forming what we term "daily & weekly data." This combined dataset provides the RL agent with a nuanced view of market trends across different time frames, enhancing its ability to make timely, data-driven trading decisions.

In summary, by integrating incremental learning and self-supervised learning into the reinforcement learning framework, the proposed approach addresses the limitations of existing financial RL models. Incremental learning allows the model to swiftly adapt to new data, essential for dynamic environments like financial markets. Self-supervised learning leverages the vast amounts of unlabeled data to predict future market movements, enabling the RL agent to anticipate potential future changes rather than relying solely on historical data. Together, these approaches enhance the RL agent's ability to make more informed and profitable trading decisions in volatile and unpredictable markets.

The main contributions of this paper is listed as follows:

- Proposing the Incremental Forecast Fusion Deep Reinforcement Learning (IFF-DRL) in single-asset trading, which combining the concept of incremental learning and self-supervised learning to improve trading performance.
- Developing a new types of data, namely daily & weekly data, that are different from traditional OHLCV data, which can help reinforcement learning agent see a more macro-level stock price patterns.
- Incorporating self-supervised time-series prediction network AutoConNet to predict future financial price data, which largely improve the accuracy of predicting price trend and the ability of capturing the pattern.
- Using incremental learning algorithm—online EWC during testing phase, which greatly improve the prediction accuracy of AutoConNet, and thus enhancing the trading performance of reinforcement learning agent.
- The proposed framework was evaluated using three reinforcement learning algorithms: DDQN, A2C, and PPO. Experiments conducted on six widely-used datasets, encompassing indices such as DJI, FCHI, SP500, KS11, and N225, demonstrate that the framework consistently outperforms various traditional and Deep Reinforcement Learning (DRL)-based methods. These results underscore the framework's potential to optimize stock trading strategies and enhance the reliability of algorithmic trading by effectively combining reinforcement learning with forecasts of future stock price movements.

The structure of this paper is as follows. Section 2 reviews relevant prior studies on the topic. In Section 3, the proposed method is described in detail. Section 4 presents and analyzes the experimental results. The advantages and limitations of the current research are discussed in Section 5. Finally, Section 6 summarizes the key findings and outlines directions for future work.

2. Related work

Financial time series modeling, as an important application area of time series analysis, has already been deeply explored and applied in the financial sector.

2.1. Incremental learning

Incremental learning involves continuously learning new knowledge from new samples while retaining the already learned knowledge. By adjusting model parameters to integrate new and old data, it simulates the continuous learning mode of humans without the need for a complete retraining, and has shown distinct advantages in the field of time series analysis. Incremental learning is particularly suitable for handling large-scale, high-dimensional, and dynamically changing time series data. It can effectively reduce the time cost required for model updates and significantly alleviate the problem of catastrophic forgetting. The core of incremental learning lies in maintaining the model's memory of old knowledge while learning new knowledge when new data is introduced, similar to the "stability-plasticity dilemma" in the nervous

system (Grossberg, 2012), which seeks a balance between stability and plasticity.

Prior to the rise of deep learning technologies, there had already been research on incremental learning. However, these studies were mostly confined to handling incremental learning between old and new tasks, and were primarily based on traditional machine learning models, making it difficult to cope with the challenges of long-term, large-scale data in modern data streams. With the rapid development of deep learning, particularly Deep Neural Networks (DNNs), their powerful representational capabilities have made deep learning-based incremental learning a potent tool for addressing these issues. The emergence of Vision Transformers (ViT) and pretrained models has turned deep learning-based incremental learning into a research hotspot, providing new solutions for handling complex visual tasks and data streams. Additionally, Ehret and colleagues demonstrated the effectiveness and potential of incremental learning in processing time series data by evaluating Recurrent Neural Networks (RNNs) on time series datasets (Ehret et al., 2020). Furthermore, the incremental approach's capability to significantly alleviate catastrophic forgetting in time series data has been validated in tasks such as Stroke-MNIST (Gulcehre, Chandar, & Bengio, 2017) and AudioSet (Gemmeke et al., 2017).

Currently, the application of incremental learning in time series analysis exemplifies its advantages in handling large-scale dynamic data, quickly adapting to new data, and alleviating catastrophic forgetting. With the continuous advancement of deep learning technologies, particularly the rise of self-supervised models, deep learning-based incremental learning will showcase its immense potential and value in even more domains.

2.2. Self-Supervised learning

Self-supervised learning (SSL) is a method that leverages unlabeled data by designing auxiliary tasks to mine data features for training models. It has gradually become a powerful tool in the field of time series analysis. By utilizing unlabeled time series data to train models, SSL significantly reduces the dependency on costly labeled data. Its core advantage lies in its ability to capture the intrinsic structure and patterns of time series data, thereby helping models better understand and predict future trends. For example, SSL models based on Transformers, such as Pyraformer (Liu et al., 2021) and Informer (Zhou et al., 2021a), have successfully applied SSL to time series forecasting tasks by introducing innovative methods like sparse attention mechanisms and time-aware decomposition strategies.

To more deeply explore the complex structure of time series, MICN (Wang et al., 2023) and CoST (Woo, Liu, Sahoo, Kumar, & Hoi, 2022) decompose time series into trend, seasonality, and other components, and perform SSL learning and prediction for these components separately. This approach, by decomposing the different components of the time series, allows the model to focus more on the feature learning of each component, thereby improving the accuracy of predictions. Additionally, CDformer (Liu, Zhuang, Gao, & Qin, 2024) utilizes local linear scaling approximation (LLSA) encoding mutations, decomposes the time series into trend and seasonality, and achieves high-precision predictions by adding the predictions from MLP and wavelet attention mechanisms. It has shown excellent performance on datasets, demonstrating its potential in financial time series forecasting and trading.

AutoCorrelation-based Contrastive Loss Network (AutoConNet) is a network model designed for long-term forecasting using autocorrelation (Park, Gwak, Choo, & Choi, 2024). In real-world scenarios, time series often exhibit different long-term and short-term variations. Long-term variations are challenging for models to predict because they cannot be captured within a typical window. AutoConNet addresses this by sampling from long time series to obtain discrete window data. It then calculates the global autocorrelation coefficient for each pair of windows using an autocorrelation function. The model is designed with a loss function that ensures the similarity between window representations aligns with the global autocorrelation measured in the data space. This approach helps the model learn long-term representations and enhances its ability to predict long-term changes.

By combining contrastive learning and other SSL techniques, the model's discriminative ability and adaptability to new data can be further enhanced (Jing et al., 2024). This strategy not only improves the model's predictive performance but also provides new insights and methods for time series analysis. As SSL has already made significant progress in the field of time series analysis and finance, further exploration of the integration of SSL with other techniques and optimization strategies in different application scenarios is warranted to drive continuous development in time series analysis and the financial domain.

2.3. Deep reinforcement learning in algorithmic trading

Algorithmic trading is a method of executing buy and sell orders automatically using computer programs and mathematical models. This type of trading determines the timing, price, and quantity of orders based on predefined trading strategies and algorithms, thereby avoiding human interference and improving efficiency and accuracy. The rise of algorithmic trading has dramatically transformed the landscape of financial markets, with its efficiency and speed enabling large-scale trading (Gupta, Singh, Kumar et al., 2022). Machine learning, as a core tool in this field, optimizes market efficiency and aids in generating excess returns by deeply mining complex relationships and patterns within historical data.

In recent years, with the continuous advancement of machine learning technology and the massive increase in financial data, the design of algorithmic trading strategies has become more complex and precise. For example, in financial markets, the application of deep learning in trading algorithms has significantly improved the accuracy of stock return predictions. Complex neural networks structures capture market patterns and trends that are difficult to identify using traditional methods, and they process large amounts of data to extract key features (Wang & Yan, 2021). Compared to traditional methods, deep learning demonstrates higher predictive stability and accuracy (Vijayarani, Suganya, & Jeevitha, 2020; Wu et al., 2020). At the same time, reinforcement learning dynamically adjusts strategies to maximize returns, creating more flexible and adaptive trading strategies that effectively respond to market fluctuations, capture trading opportunities, and reduce risk (Ghania, Awaisa, & Muzammula, 2019). Deep reinforcement learning, by combining the feature extraction capabilities of deep learning with the strategy optimization capabilities of reinforcement learning, better adapts to market complexities and nonlinear relationships, enhancing the stability and profitability of trading (Yang et al., 2020).

Recent research has revealed significant advancements and integration trends of Deep Reinforcement Learning (DRL) in solving more generalized algorithmic trading problems. For instance, the introduction of Probabilistic Knowledge Transfer (PKT) to optimize the distillation method of DRL agents has successfully improved the training process of these agents, significantly enhancing training stability (Moustakidis, Passalis, & Tefas, 2024). Additionally, an innovative algorithm based on the Deep Q-Network (DQN) algorithm utilizes a simplified reward function with delayed feedback mechanisms to promote model reinforcement learning, simplifying the decision-making process and providing a more flexible and adaptive reward mechanism. This, in turn, drives improvements in returns management and risk control (de Azevedo Takara, Santos, Mariani, & dos Santos Coelho, 2024). In terms of model integration, a feature extractor based on GraphSAGE has been integrated into the PPO agents' architecture. This integration strategy significantly boosts the model's ability to recognize key features of stock market data, thereby enhancing its performance in portfolio optimization tasks. Its integration advantage has been effectively validated across different market scenarios (Sun, Wei, & Yang, 2024). Furthermore, the DRL framework has innovatively incorporated a bidirectional Long Short-Term Memory (BiLSTM) network combined with

an attention mechanism. This innovative fusion further enhances the model's ability to identify key features in stock market data (Huang et al., 2024a). Additionally, LSTM and CNN have been integrated into the deep reinforcement learning framework as multi-agents, effectively overcoming traditional issues of value function overestimation and overfitting. Compared to single-agent reinforcement learning algorithms, this strategy shows better performance and more stable returns (Cheng & Sun, 2024). The multi-agent approach built on a Double Deep Q-Network, employing TimesNet and multi-scale convolutional neural networks, achieves effective risk balancing through innovative integration and obtains considerable returns in complex stock market trading (Huang et al., 2024c). These research advancements collectively highlight the immense potential and vast prospects of DRL technology in the field of financial trading strategy optimization.

These studies not only demonstrate the potential of DRL in algorithmic trading, but also reveal its advantages over traditional methods, including the ability to adapt to market changes, avoid emotional biases, and accelerate trade execution. With the continuous development of DRL, the research prospects in the field of algorithmic trading are vast, and there is hope that it will further drive innovation and transformation in financial markets in the future.

3. Method

In algorithmic trading, the process of selecting and executing trading strategies can be effectively modeled as a Markov Decision Process (MDP), making it a natural fit for reinforcement learning (RL). Within this framework, financial reinforcement learning agents strive to identify the optimal trading strategy by exploring and evaluating the consequences of different actions in a dynamic trading environment. However, to thoroughly investigate the application of RL in complex financial markets and to ensure the robustness and reliability of the results, it is essential to establish several key assumptions:

1. The market operates under dynamic efficiency, integrating both the Efficient Markets Hypothesis (EMH) and the Adaptive Markets Hypothesis (AMH). EMH assumes that market prices reflect all available information, while AMH acknowledges that market efficiency evolves over time due to changes in investor behavior, market conditions, and external shocks. This dynamic view allows the proposed model to adapt to variations in market efficiency, enabling it to respond to unexpected events and shifting market dynamics.
2. Considering the complexities of stock market liquidity and the relatively small scale of assets being analyzed, particularly from the perspective of individual investors, this paper assumes that the impact of individual buy or sell orders on market prices is negligible. As a result, a single trade is presumed not to have a significant influence on price movements.
3. There is no slippage in the execution of orders, meaning the market assets possess sufficient liquidity to ensure that trades are executed at the quoted prices without any deviation.

3.1. Overview of the proposed method

In this section, we present an overview of the proposed method. As illustrated in Fig. 1, the framework combines reinforcement learning with a self-supervised time series prediction network. The process begins by introducing a pre-trained self-supervised network to forecast OHLCV (Open, High, Low, Close, Volume) values for the next n days, using past OHLCV data as input. These predictions, along with the actual observed data, form the state input to the reinforcement learning model. The state representation is designed with two components. The first, s_t^d , includes the observed OHLCV data at time t. The second, s_t^w, aggregates both observed and predicted data into weekly OHLCV summaries. Then, the s_t^d and s_t^w are concatenated to form the daily & weekly data called s_t^f , which is served as the input state s of the reinforcement learning algorithm. By combining these s_t^d and s_t^w, the model benefits from both fine-grained daily patterns and broader weekly trends, enhancing decision-making capabilities. In the testing phase, we employ the online Elastic Weight Consolidation (EWC) method to enable incremental learning within the prediction network. This approach helps the network retain previously learned knowledge while adapting to new data, ensuring consistent prediction accuracy. By maintaining the self-supervised network's learning capability, the overall framework demonstrates improved robustness and adaptability in dynamic environments.

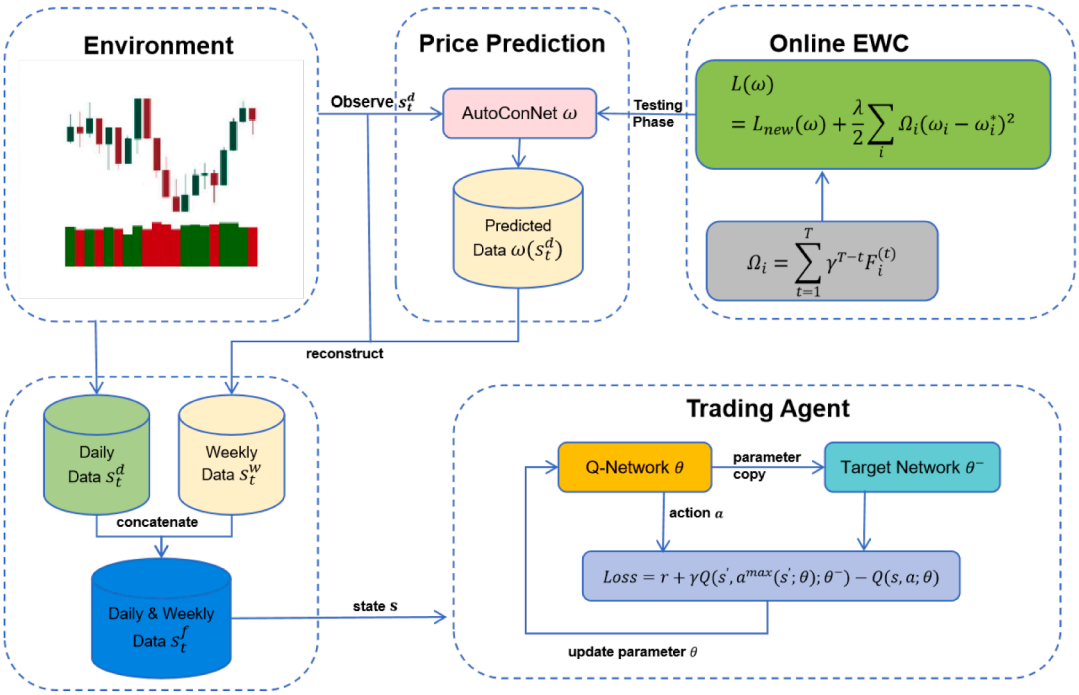


Fig. 1. The structure of the proposed method.

3.2. Problem formulation

The state provides a representation of the environment at time t, offering the agent the necessary information to make informed decisions. As mentioned earlier, the training data is organized into daily and weekly datasets. The daily data consists of observed OHLCV (Open, High, Low, Close, Volume) values, while the weekly data is generated by combining predicted prices with observed ones. In this context, the state can be categorized into daily state and weekly state. The daily state, denoted as s_t^d , is a collection of features including the opening price, high price, low price, closing price, and trading volume (OHLCV) over the past n days. The daily state can be denoted as:

$$s_t^d = \{O_n^d, H_n^d, L_n^d, C_n^d, V_n^d\}_t. \tag{1}$$

Similarly, weekly state can be represented as:

$$s_t^w = \{O_n^w, H_n^w, L_n^w, C_n^w, V_n^w\}_t, \tag{2}$$

where $O_n^w, H_n^w, L_n^w, C_n^w, V_n^w$ are the stock market prices for the OHLCV data in the weekly period of observed days and predicted days. The complete state at time t is the concatenation of daily state and weekly state, therefore, the complete state s_t^f can be denoted as:

$$s_t^f = \{s_t^d, s_t^w\}. \tag{3}$$

3.2.1. Action

At each time step t, given a state s_t^f , the agent selects and performs an action a_t guided by the RL policy. In this study, which focus on algorithmic trading, the agent has only three actions, namely buy, hold, and sell. The computation of actions can be represented as:

$$a_t = \begin{cases} -1, & \text{if } \pi(a_t|s_t^f) = \text{Sell}; \\ 0, & \text{if } \pi(a_t|s_t^f) = \text{Hold}; \\ 1, & \text{if } \pi(a_t|s_t^f) = \text{Buy}, \end{cases} \tag{4}$$

where $\pi(a_t|s_t^f)$ is RL policy that computed by the reinforcement learning agent at time step t.

The trading actions executed by the agent may differ from the trading signals generated by the policy network. For instance, if the account is already holding a short position ($\text{POS}_t = -1$), a sell signal simply indicates maintaining the existing short position. In this case, the actual action taken is "hold" rather than "sell." This adjustment arises from trading rules that prohibit additional buying while a long position is held, as well as further short selling while in a short position. Table 1 provides a detailed explanation of how actual trading operations are determined based on the combination of the current account position and the generated trading signals.

3.2.2. Reward function

The reward function plays a critical role in the reinforcement learning framework, serving as a measure of the immediate feedback or reward an agent receives based on its actions in a specific state. In the context of algorithmic trading, this function typically reflects the profitability or performance of the trading strategy. Building on the approach proposed by Huang, Zhou, Cui, and Lu (2024b), this paper utilizes a novel return-based reward function that balances short-term and long-term returns over a flexible time horizon. By considering the maximum returns from various positions and incorporating an adaptable time window, the function effectively captures trends from both short-term and long-term perspectives, ultimately aiming to maximize overall returns. Notably, the flexibility of this reward function allows for fine-tuning between short-term and long-term objectives by adjusting the time horizon parameter m, making it adaptable to the unique characteristics of different stocks. Additionally, the function is designed to emphasize information relevant to the agent's positions. The mathematical formulation of the proposed reward function is provided in Eqs. (5) to (7).

$$R_t = \begin{cases} \text{POS}_t * \text{maxRatio}, & \text{maxRatio} > 0 \text{ or } \text{maxRatio} + \text{minRatio} > 0, \\ \text{POS}_t * \text{minRatio}, & \text{minRatio} < 0 \text{ or } \text{maxRatio} + \text{minRatio} < 0. \end{cases} \tag{5}$$

where

$$\text{maxRatio} = \begin{cases} \max(r_{t+1}, r_{t+2}, \cdots, r_{t+m}), & \text{if } r_{t+i} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

$$\text{minRatio} = \begin{cases} \min(r_{t+1}, r_{t+2}, \cdots, r_{t+m}), & \text{if } r_{t+i} < 0, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

where $r_{t+i} = \frac{p_{t+i} - p_t}{p_t} * 100$.

Table 1
Actual trading operations based on the signal and the current account position.

Old Position (POS_{t-1})	Signal (a_t^*)	Actual Action (a_t)	New Position (POS_t)	Description
0	0	0	0	Hold the cash.
0	1	1	1	Open a long position.
0	−1	−1	−1	Open a short position.
1	0	0	1	Hold the long position.
1	1	0	1	Hold the long position.
1	−1	−1	0	Close the long position.
−1	0	0	−1	Hold the short position.
−1	1	1	0	Close the short position.
−1	−1	0	−1	Hold the short position.

3.3. AutoConNet for time-series prediction

AutoConNet, as shown in Fig. 2, uses autocorrelation to identify long-term changes. For example, it samples three windows from different times t_1, t_2, and t_3 of an entire time series to form a batch. These form three possible positive pairs(i.e., due to three anchors), and for each positive pair, a global autocorrelation is calculated. The lag is the distance between the two windows that make up the pair. The calculated positive pairs are then compared for autocorrelation with the remaining pairs, and those with lower autocorrelation than the anchored positive pair are considered negative pairs. AutoConNet obtains the autocorrelation function $R_{ss}(h)$ from the real discrete process $\{S_t\}$ using the following equation:

$$R_{ss}(h) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} S_t S_{t-h}, \tag{8}$$

where $S = (s_1, \ldots, s_T)$ is the entire time series, T denotes the length of the observed time series, $s_t \in \mathbb{R}^c$ is observation with c dimensional, and the lag h is the distance between two windows.

By implementing a mini-batch comprised of widely separated windows through global autocorrelation, where the time distance can span the entire sequence length and exceed the window length, the model is endowed with the capability to handle long-term dependencies by establishing relations between global windows. Set a fixed-length W window on S as $D = \{W_t\}_{t=1}^{M}$. The global index sequence of W_t is denoted as $\mathcal{T}_t = \{t + i\}_{i=0}^{W-1}$. Based on the global autocorrelation, the relationship between two windows is defined as follows. At two different times t_1 and t_2, for any two windows W_{t_1} and W_{t_2} each has W observations, with $\mathcal{T}_{t_1} = \{t_1 + i\}_{i=0}^{W-1}$ and $\mathcal{T}_{t_2} = \{t_2 + j\}_{j=0}^{W-1}$ serving as the global index in chronological order. The time distance matrix between each two observation windows is $D \in \mathbb{R}^{W\times W}$, and the time distance between elements is represented by $D_{i,j} = |(t_2 + j) - (t_1 + i)|$. The time distance between two windows at the same phase ($i.e., i = j$) has the same value $|t_1 - t_2|$. The relationship between two windows is defined using the global autocorrelation function $R_{ss}(|t_1 - t_2|)$ as follows:

$$r(\mathcal{T}_{t_1}, \mathcal{T}_{t_2}) = |R_{ss}(|t_1 - t_2|)|. \tag{9}$$

AutoConNet employs a custom loss function to assess the similarity between all pairs of window representations, following the global autocorrelation measured in the data space. It utilizes global autocorrelation R_{ss} to determine the relationships between windows. For the

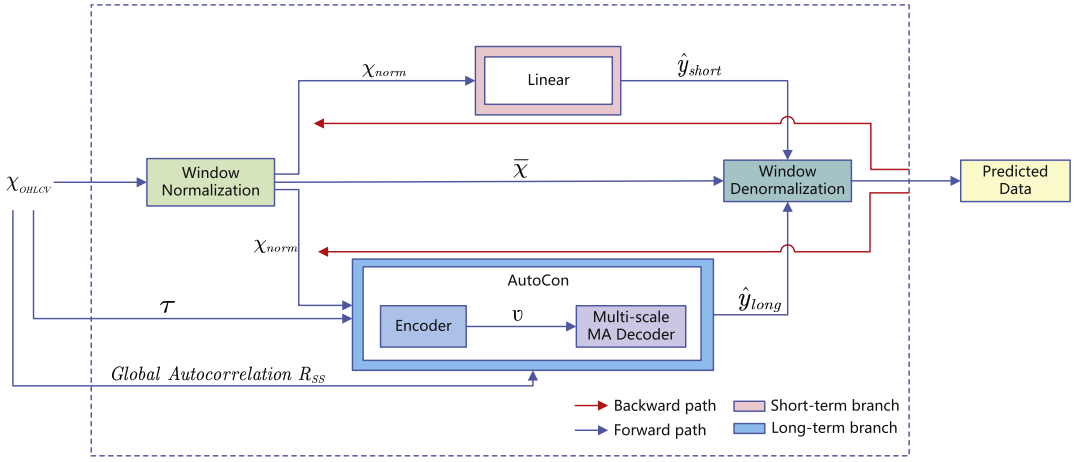


Fig. 2. The structure of AutoConNet.

Autocorrelation-based Contrastive Loss ($\mathcal{L}_{AutoCon}$), the input $X \in \mathbb{R}^{N\times I}$, which contains N windows, is fed into an encoder to obtain the representation $v \in \mathbb{R}^{N\times I\times d}$ through $v = Encode(\mathcal{X}, \mathcal{T})$. For ease of explanation, c is set to 1. Here, using i as the window index, and $r^{(i,j)} = r(\mathcal{T}^{(i)}, \mathcal{T}^{(j)})$ represents the global correlation between two windows, the formula for the loss $\mathcal{L}_{AutoCon}$ is calculated based on the representation $\{v^{(i)}\}_{i=1}^{N}$ with the corresponding time sequence $\{\mathcal{T}^{(i)}\}_{i=1}^{N}$ as follows:

$$\mathcal{L}_{AutoCon} = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{N-1} \times \sum_{j=1,j\neq i}^{N} r^{(i,j)} \log \frac{\exp(Sim(v^{(i)}, v^{(j)})/\tau)}{\sum_{k=1}^{N} \mathbb{1}_{[k\neq i, r^{(i,k)}\leqslant r^{(i,j)}]} \exp(Sim(v^{(i)}, v^{(k)})/\tau)}, \quad (10)$$

where $Sim(\cdot,\cdot)$ measures the similarity between two representations, There are a total of $N \times (N-1)$ possible pairs indexed by (i, j). For each pair (serving as an anchor pair), any pair whose global autocorrelation $r^{(i,k)}$ is less than $r^{(i,j)}$ is considered a negative pair, designating the anchor pair itself as a relatively positive pair. Additionally, $r(i, j)$ is introduced as a weight to differentiate the positive pairs with varying degrees of correlation, minimizing $\mathcal{L}_{AutoCon}$.

AutoConNet is designed with three main features to adhere to short-term temporal locality and the global nature of long-term predictions.

Firstly, use window normalization and denormalization to mitigate the issue of distribution shift brought about by the non-stationarity of real-world time series (Kim et al., 2021). The formula is:

$$\mathcal{X}_{norm} = \mathcal{X} - \overline{\mathcal{X}}, \qquad \mathcal{Y}_{pred} = (\mathcal{Y}_{short} + \mathcal{Y}_{long}) + \overline{\mathcal{X}}, \quad (11)$$

where $\overline{\mathcal{X}}$ is the mean of the input sequence.

Secondly, for the short-term Branch for Temporal Locality (Zeng, Chen, Zhang, & Xu, 2023), a linear layer is used for short-term prediction, with the formula being:

$$\mathcal{Y}_{short} = Linear(\mathcal{X}_{norm}). \quad (12)$$

And for the Long-term Branch for Temporal Globality, the AutoCon method's encoder-decoder architecture is used to learn long-term representations by leveraging sequential and global information, with the formula being:

$$v = Encode(\mathcal{X}_{norm}, \mathcal{T}). \quad (13)$$

Finally, the network model uses a Temporal Convolutional Network (TCN) (Bai, Kolter, & Koltun, 2018), and the decoder adopts Multi-scale Moving Average (MA) blocks (Wang et al., 2023). With different kernel sizes K, it captures multiple periods based on representation v, with the formula as follows:

$$\hat{\mathcal{Y}}_{long} = \frac{1}{n}\sum_{i=1}^{n} AvgPool(Padding(MLP(v)))_{k_i}. \quad (14)$$

The MA block at the head of the long-term branch smooths out short-term fluctuations, enabling the branch to focus more on long-term information. The loss function $\mathcal{L}$ for the redesigned architecture is formulated as follows:

$$\mathcal{L} = \mathcal{L}_{MSE} + \lambda \cdot \mathcal{L}_{AutoCon}. \quad (15)$$

Finally, Mean Squared Error (MSE) and AutoCon loss are combined with weight λ as hyperparameters to optimize the objective function $\mathcal{L}$.

AutoConNet constructs weekly OHLCV data using both predicted and observed data as input for the reinforcement learning model. This allows the model to sufficiently learn long-term representations from global information, enhancing its ability to predict long-term changes. As shown in Fig. 2, the AutoConNet framework.

3.4. Online elastic weight consolidation

Online Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) is an incremental learning algorithm that helps prevent catastrophic forgetting (French, 1999) by selectively constraining updates to the neural network's parameters based on their importance to previously learned tasks. This is particularly useful in settings where the model must adapt to new data without forgetting previously acquired knowledge.

The central idea behind EWC is to regularize the loss function to penalize large changes in important parameters. The importance of each parameter is measured using the Fisher Information Matrix, which captures how sensitive the model's predictions are to changes in these parameters. The online EWC approach builds upon the traditional EWC by consolidating knowledge continuously across tasks or streaming data without explicit task boundaries. The loss function for online EWC can be defined as:

$$\mathcal{L}(\omega) = \mathcal{L}_{\text{new}}(\omega) + \frac{\lambda}{2}\sum_{i}\Omega_i(\omega_i - \omega_i^*)^2, \quad (16)$$

where the $\mathcal{L}_{\text{new}}(\omega)$ is the standard loss function on the current task or new data, λ is a hyperparameter controlling the trade-off between plasticity (adaptation to new data) and stability (retaining old knowledge), ω_i are the current parameters of the model, ω_i^* are the parameters learned from previous data or tasks, Ω_i represents the importance of each parameter θ_i, computed based on the Fisher Information Matrix.

The Fisher Information Matrix F_i is used to estimate the importance of the parameters. For each parameter ω_i, its importance is defined as:

$$\Omega_i = \sum_{t=1}^{T}\gamma^{T-t}F_i^{(t)}, \quad (17)$$

where $F_i^{(t)}$ is the Fisher Information Matrix at time step t, γ is a decay factor that controls the weight of past information, T is the total number of time steps or tasks.

3.5. Incremental forecasting fusion double deep Q-Network

The proposed Incremental Forecasting Fusion Deep Reinforcement Learning (IFF-DRL) framework can incorporate various RL algorithms, including value-based methods, policy-based methods and Actor-Critic based methods. In this section, a classical value-based method called Double Deep Q-Network (DDQN) is introduced as an example to illustrate the basic logic of the proposed framework. DDQN is a special edition of DQN that designed to address the problem of overestimation. The operational dynamics of the DDQN rely on two neural networks: the main network (θ) and the target network (θ^-). For each action taken, a transition tuple $(s_t^f, a_t, r_t, s_{t+1}^f)$ is generated, where s_t^f is the fusion state generated by the predicted price and observed price. In this process, the main network (θ) selects the next action $a_{\max}(s_{t+1}^f;\theta) = \text{argmax}_{a_{t+1}} Q(s_{t+1}^f, a_{t+1};\theta)$. The immediate reward r_t is then received by the agent. It is worth to notice that, this method promotes alignment with higher rewards, encouraging more profitable and strategically advantageous trading decisions. Consequently, the system enhances the overall performance of the trading strategy. The update rule for the DDQN is given by Eq. (18):

$$Q(s_t^f, a_t;\theta) \leftarrow Q(s_t^f, a_t;\theta) + \alpha[r_t + \gamma Q(s_{t+1}^f, a_{\max}(s_{t+1}^f;\theta);\theta^-) - Q(s_t^f, a_t;\theta)], \tag{18}$$

where s_t^f represents the current state, a_t is the action taken based on this state, r_t is the reward received after executing action a_t in state s_t^f, and s_{t+1}^f is the subsequent state. Here, α denotes the learning rate, θ is the parameter of the Q-network, θ^- is the parameter of the target network, and ϕ represents the parameter of the reward network.

3.6. IFF-DDQN training

The training procedure of IFF-DDQN is illustrated in Algorithm 1 and the flowchart of IFF-DDQN is shown in Fig. 3. First, a dataset that span from January 1, 2007 to December 31, 2020 is used to train a time-series prediction network. The error between the predicted and actual values is calculated by mean square error function. The parameters of the network ω are then updated through gradient descent. Once the training of ω is finished, the time-series prediction network is transferred to the DDQN algorithm. Initially, a state s_t^d is observed and passed through ω, the predicted values then generated. Weekly state s_t^w is then constructed with predicted values and observed state s_t^d. After that, a complete state s_t^f is generated by the concatenation of s_t^d and s_t^w. By observing the state s_t^f, the DDQN agent samples an action a_t, which leads to a new state s_{t+1}^d due to the Markov Decision Process (MDP). A new complete state s_{t+1}^f is then generated by the aforementioned process. With the transition tuple $(s_t^f, a_t, r_t, s_{t+1}^f)$, DDQN agent updates its parameters by temporal difference (td) learning technique. During testing phase, EWC method is introduced to improve the prediction accuracy of the selected time-series network.

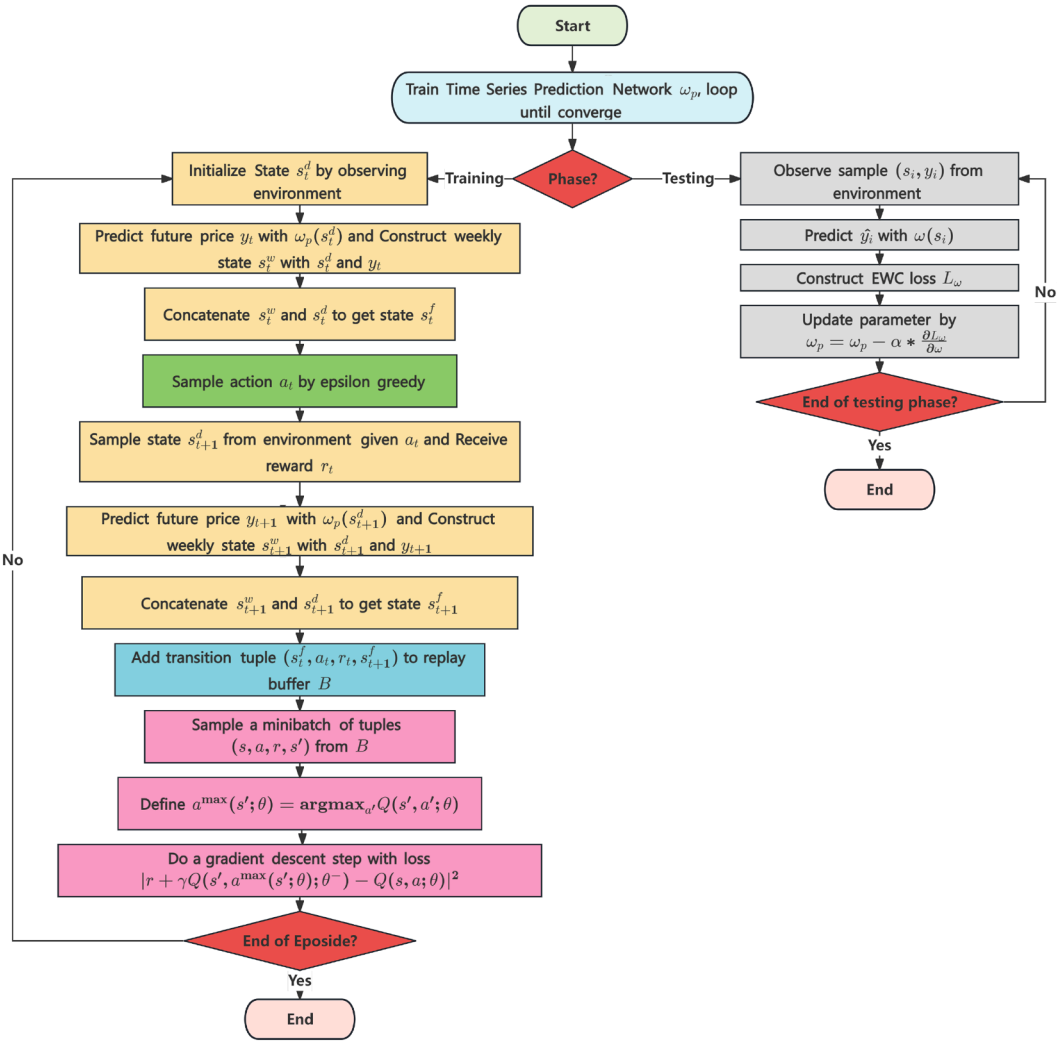


Fig. 3. The flowchart of IFF-DDQN algorithm.

Algorithm 1: IFF-DDQN algorithm.

Input: $\mathcal{B}$ - empty replay buffer; θ - initial network parameters, θ^- - copy of θ; N_r - replay buffer maximum size; N_b - trainng batch size; N^- - target network replacement frequency, ω_p - time-series prediction network parameter, α - learning rate ;

Initialize time-series prediction network parameter with random initial weights $\omega_p \leftarrow \omega_0$;
while *the training of* ω_p *is not converged* **do**
  Select sample (s_i, y_i) from dataset D;
  Predict $\hat{y}_i$ with $\omega(s_i)$;
  Calculate loss $L_\omega = \text{MeanSquareErrorLoss}(y_i, \hat{y}_i)$;
  Calculate gradient $\delta_\omega = \frac{\partial L_\omega}{\partial \omega}$;
  $\omega_p = \omega_p - \alpha * \delta_\omega$;
end
if *training phase* **then**
  for *episode* $e = \{1, 2, \cdots, M\}$ **do**
    while *state* s_t^d *is not terminal* **do**
      Initialize state s_t^d;
      Predict future price y_t with $\omega(s_t^d)$;
      Construct weekly state s_t^w with s_t^d and y_t;
      Concatenate s_t^w and s_t^d to get the complete state s_t^f;
      Sample action a_t by epsilon greedy;
      Sample state s_{t+1}^d from environment $\mathcal{E}$ given a_t and Receive reward r_t;
      Predict future price y_{t+1} with $\omega(s_{t+1}^d)$;
      Construct weekly state s_{t+1}^w with s_{t+1}^d and y_{t+1};
      Concatenate s_{t+1}^w and s_{t+1}^d to get the complete state s_{t+1}^f;
      Add transition tuple $(s_t^f, a_t, r_t, s_{t+1}^f)$ to $\mathcal{B}$, replacing the first tuple if $|\mathcal{B}| > N_r$;
      Sample a minibatch of N_b tuples $(s, a, r, s') \sim \text{Unif}(\mathcal{B})$;
      Calculate target values, one for each of the N_b tuples;
      Define $a^{\max}(s'; \theta) = \text{argmax}_{a'} Q(s', a'; \theta)$;
      Set:
      $$y = \begin{cases} r, & \text{if } s' \text{ is terminal;} \\ r + \gamma Q(s', a^{\max}(s'; \theta); \theta^-), & \text{otherwise.} \end{cases}$$
      Do a gradient descent step with loss $|y - Q(s, a; \theta)|^2$;
      Replace target parameters $\theta^- \leftarrow \theta$ every N^- steps;
    end
  end
end
if *testing phase* **then**
  Observe sample (s_i, y_i) from environment;
  Predict $\hat{y}_i$ with $\omega(s_i)$;
  Construct EWC loss L_ω by euqation (5);
  Update parameter by $\omega_p = \omega_p - \alpha * \frac{\partial L_\omega}{\partial \omega}$;
end
Output: The optimal strategy π;

4. Experiments

4.1. Dataset

To assess the robustness of the proposed model, six major stock indices from different regions are selected for evaluation: CAC40 (FCHI) from France, representing the 40 largest companies on the Paris Stock Exchange; KOSPI (KS11) from Korea, covering all companies listed on the Korea Stock Exchange; Nikkei 225 (N225) from Japan, consisting of 225 large companies listed on the Tokyo Stock Exchange; Hang Seng (HSI) from Hong Kong, representing 50 major companies listed on the Hong Kong Stock Exchange; S&P 500 (SP500) from the United States, covering 500 large U.S. companies; and Dow Jones Industrial Average (DJI) from the U.S., made up of 30 major industrial companies. The price curve of selected datasets are shown in Fig. 4. These indices offer a diverse view of global market performance. The dataset spans from January 1, 2007, to December 31, 2023, offering a comprehensive time frame for effective training and testing. The data up to December 31, 2020, is utilized for model training, while the period from January 1, 2021, to December 31, 2023, is reserved for testing. Each index in the dataset includes five key features: opening price, lowest price, highest price, closing price, and trading volume.

4.2. Evaluation metrics

4.2.1. Metrics for trading performance

To provide an objective and comprehensive assessment of the proposed method, this paper utilizes four key investment performance metrics.

- Cumulative Return (CR): This metric reflects the total return generated on investments over the evaluation period (Luenberger, 2009).
- Annualized Return (AR): Representing the average return achieved annually, AR provides a normalized measure of performance over time (Luenberger, 2009).
- Sharpe Ratio (SR): This risk-adjusted performance metric evaluates the excess return per unit of volatility, comparing the strategy's returns to a risk-free benchmark (Sharpe, 1994).
- Maximum Drawdown (MDD): This metric quantifies the largest peak-to-trough decline in portfolio value over a given period, highlighting potential downside risk (Magdon-Ismail & Atiya, 2004).

By addressing different dimensions of return, risk, and volatility, these metrics offer a well-rounded evaluation of the investment strategy's performance.

4.2.2. Metrics for prediction accuracy

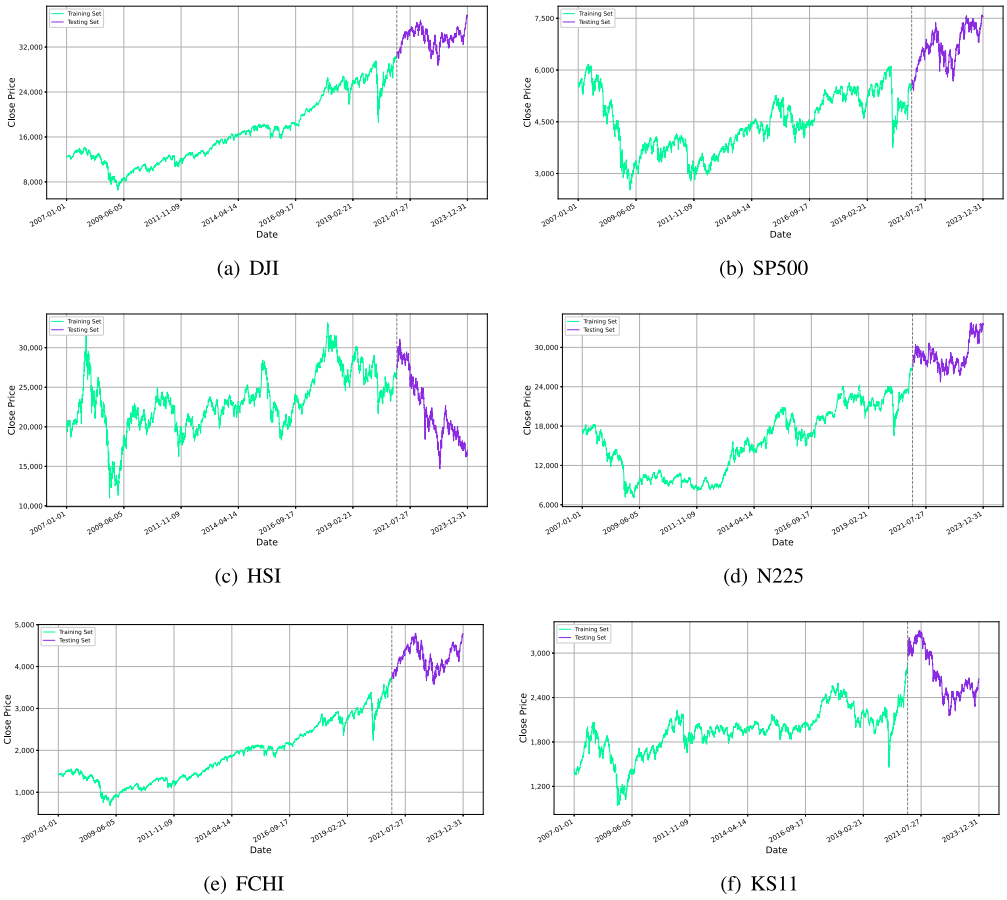
To evaluate the predictive accuracy, this study employs four widely used error metrics.

- Mean Squared Error (MSE): MSE calculates the average squared difference between predictions and actual values, assigning greater weight to larger errors due to the squaring operation (Bishop & Nasrabadi, 2006).
- Mean Absolute Error (MAE): MAE computes the average absolute difference between predicted and observed values, offering a straightforward measure of prediction error in the same units as the original data (Willmott & Matsuura, 2005).
- Root Mean Squared Error (RMSE): RMSE, derived as the square root of MSE, provides a scale-sensitive error metric that is easier to interpret while maintaining sensitivity to larger deviations (Chai & Draxler, 2014).
- Mean Absolute Percentage Error (MAPE): MAPE expresses the error as a percentage of the actual values, making it particularly useful for comparing model performance across datasets of varying magnitudes (Hodson, 2022).

These metrics collectively address different dimensions of prediction error, enabling a more comprehensive assessment of the models' performance in terms of accuracy and interpretability.

4.3. Baseline methods

To better evaluate the effectiveness of the proposed method, several classic algorithmic trading methods and some cutting-edge financial reinforcement learning algorithms are introduced as baseline methods.



(a) DJI (b) SP500

(c) HSI (d) N225

(e) FCHI (f) KS11

Fig. 4. Close price curve of six stock indices.

The classic methods include Buy and Hold (B&H), Sell and Hold (S&H), Mean Reversion with Moving Averages (MR), and Trend Following with Moving Averages (TF). The cutting-edge financial reinforcement learning algorithms include TDQN, DQN-Vanilla, and Fire (DQN-HER).

The B&H strategy involves an investor purchasing an asset and holding it for the duration of the investment period, without reacting to price fluctuations. In contrast, the S&H strategy requires the investor to short-sell an asset at the start of the period and maintain this position until the end. MR strategies are based on the assumption that asset prices will revert to their historical averages, with this study using a 10-day Simple Moving Average (SMA) to identify potential reversion points. TF strategies, commonly employed in algorithmic trading, leverage moving averages to generate trading signals. For this analysis, we use 10-day and 20-day SMAs to identify possible trading opportunities. The TDQN, introduced by Théate and Ernst (Théate & Ernst, 2021), aims to optimize trading positions using a five-layer fully connected Deep Q-Network. The DQN-Vanilla, developed by Taghian et al. (Taghian, Asadi, & Safabakhsh, 2022), utilizes a simpler two-layer fully connected network to make trading decisions based on stock OHLC data. Additionally, the Fire (DQN-HER) model, proposed by Cornalba et al. (Cornalba, Disselkamp, Scassola, & Helf, 2024), explores multi-objective deep reinforcement learning for trading in both stock and cryptocurrency markets. This model incorporates dynamic reward functions and discount factors to improve learning outcomes.

Table 2
The hyperparameters of various deep reinforcement learning methods.

Hyperparameter	IFF-A2C	IFF-PPO	IFF-DDQN
Agent network	TimesNet	TimesNet	TimesNet
Discount factor γ	0.9	0.9	0.9
Learning rate (α)	0.001	0.001	0.001
Episode	100	100	100
Replay memory	10000	10000	10000
Batch size	32	32	32
Greedy	–	–	0.9
Clip	–	0.1	–
Training Time	3.06 h	3.5 h	2.5 h

Table 3
The hyperparameters of various time-series prediction networks.

Hyperparameter	AutoConNet	PatchTST	SimMTM
Network layer	6CNN + 7MLP	21MLP	5CNN + 12MLP
Activation function	ReLu	GELU	ReLu
Batch size	32	32	32
Learning rate (α)	0.0001	0.0001	0.0001
Optimizer	Adam	Adam	Adam
Window size	20	20	20

4.4. Experimental setup

The proposed method is trained separately on six distinct stock index datasets, and after the training phase, the model's performance is evaluated using the corresponding test set for each dataset. This evaluation approach assesses the model's ability to accurately predict actions across various stock markets by training on multiple datasets. Table 2 provides the details of the hyperparameters for the three reinforcement learning methods. Additionally, the initial capital is set at $500,000, and a transaction cost of 0.3 % is applied to the total amount of each trade. In this

article, the selection of the agent network follow settings from previous work. (Huang et al., 2024c; Zhou, Huang, Cui, & Lu, 2024) Also, sliding window is selected as the data sampling method during the training and testing phases. The hardware and software used are listed as follows: the processor is 13th Gen Intel(R) Core(TM) i7-13700KF, the GPU is NVIDIA RTX 4090 with the 24GB memory. The software versions used are: Python is v3.10, Pytorch is v1.12, and cuda is v12.1. The operating system is Windows 10.

4.5. Experimental results

4.5.1. Analysis of time-series prediction network

As previously mentioned, the proposed method uses predicted stock price data for several future days by a time series forecasting network as input to the reinforcement learning agent. This provides the agent with potential future market trends as a reference when making trading decisions, leading to better trading outcomes. Therefore, to enable the reinforcement learning agent to make more informed decisions, a careful evaluation of existing cutting-edge time series forecasting networks is essential. Recent research has revealed that many widely-recognized works in time series forecasting, such as PatchTST (Nie, Nguyen, Sinthong, & Kalagnanam, 2022) and AutoConNet (Park et al., 2024), are based on self-supervised learning principles. Additionally, studies related to incremental learning have shown that networks based on self-supervised learning tend to be more robust when combined with incremental learning algorithms compared to other networks. Consequently, a comparative analysis was conducted on three outstanding networks in the time series forecasting field-AutoConNet, PatchTST, and SimMTM (Dong et al., 2024)-to select the most suitable network for the proposed method. Table 4 presents a comparison of the MSE, MAE, RMSE and MAPE of these three networks on the selected dataset and Fig. 5 shows the close price curve predicted by three networks versus the real trend. Also, the hyperparameters for training the three networks are provided in Table 3.

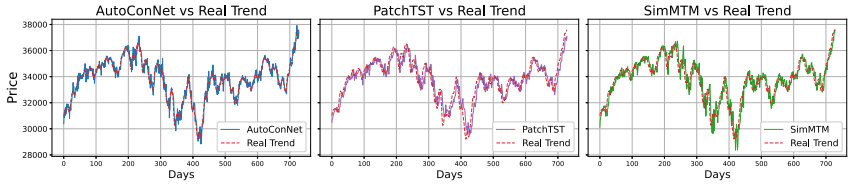
It can be seen from Table 4 and Fig. 5 that, PatchTST performed the worst compared to the other two networks. As shown in Fig. 5, AutoConNet and SimMTM demonstrate similar prediction performance, with both models successfully capturing the overall price movement trends. These models effectively predict the general direction of price changes, showcasing their strong predictive capabilities. In contrast, PatchTST performs considerably worse across all datasets, with the exception of the DJI dataset, where it shows some reasonable accuracy. In many cases, PatchTST fails to predict the price direction correctly, leading to substantial errors. The experimental results, presented in Table 4, further emphasize the performance gap between PatchTST and the other two models. PatchTST often shows Mean Squared Error (MSE) and Mean Absolute Error (MAE) values that are 5–10 times higher than those

Table 4
The prediction accuracy of self-supervised networks on six datasets.

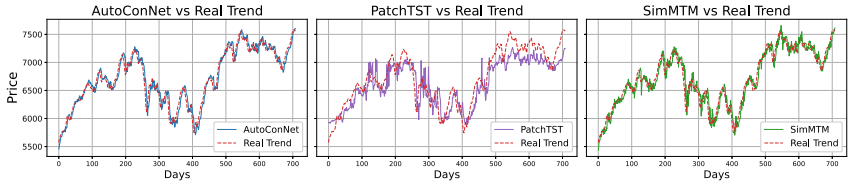
Dataset	AutoConNet				PatchTST				SimMTM			
	MSE↓	MAE↓	RMSE↓	MAPE↓	MSE↓	MAE↓	RMSE↓	MAPE↓	MSE↓	MAE↓	RMSE↓	MAPE↓
DJI	**0.005**	**0.053**	**0.067**	**0.256**	0.055	0.212	0.070	0.287	0.006	0.056	0.076	0.265
FCHI	**0.005**	**0.048**	**0.061**	**0.305**	0.011	0.084	0.097	0.390	0.005	0.049	0.068	0.318
HSI	**0.004**	**0.039**	**0.057**	**0.207**	0.015	0.083	0.100	0.300	0.004	0.041	0.061	0.213
KS11	**0.002**	**0.033**	**0.042**	**0.240**	0.022	0.121	0.143	0.412	0.002	0.035	0.047	0.261
SP500	**0.005**	**0.049**	**0.062**	**0.227**	0.041	0.157	0.181	0.596	0.005	0.051	0.069	0.239
N225	**0.006**	**0.056**	**0.072**	**0.296**	0.021	0.113	0.137	0.599	0.007	0.058	0.079	0.300

Table 5
The performance of the various methods on the six datasets.

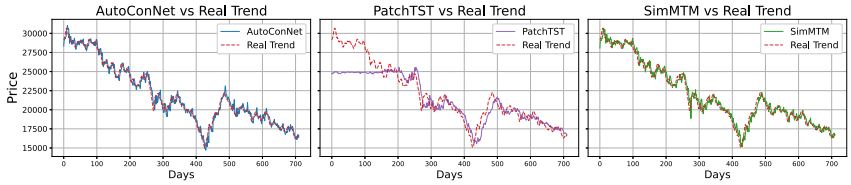
Datasets	Metrics	B&H	S&H	MR	TF	TDQN	DQN-Vanilla	DQN-HER	IFF-DDQN	IFF-A2C	IFF-PPO
DJI	CR↑	4.12 %	−4.12 %	1.53 %	−3.08 %	−43.04 %	58.42 %	76.79 %	395.09 %	435.06 %	**459.83 %**
	AR↑	10.55 %	−10.38 %	4.84 %	−7.92 %	−22.51 %	21.06 %	19.11 %	64.43 %	66.91 %	**68.38 %**
	SR↑	0.54	−0.40	0.27	−0.44	−1.26	1.28	1.51	4.50	4.75	**4.78**
	MDD↓	20.23 %	25.37 %	17.95 %	23.78 %	45.57 %	10.96 %	6.92 %	**2.53 %**	4.37 %	3.21 %
FCHI	CR↑	7.83 %	−7.83 %	0.34 %	3.91 %	40.86 %	127.84 %	34.03 %	450.37 %	**482.74 %**	482.10 %
	AR↑	18.28 %	−23.17 %	2.94 %	10.89 %	11.57 %	32.00 %	10.48 %	70.56 %	66.91 %	**68.38 %**
	SR↑	0.77	−0.51	0.12	0.48	0.75	1.95	0.69	4.34	4.43	**4.49**
	MDD↓	21.95 %	39.57 %	23.41 %	16.08 %	21.11 %	7.01 %	12.50 %	7.36 %	3.84 %	**3.43 %**
HSI	CR	−7.07 %	7.07 %	−3.47 %	−2.01 %	37.11 %	86.61 %	46.92 %	1410.73 %	1244.17 %	**1462.42 %**
	AR	−22.11 %	16.28 %	−6.32 %	−2.51 %	11.05 %	29.25 %	13.46 %	102.15 %	98.79 %	**103.19 %**
	SR	−0.65	0.84	−0.18	−0.08	0.67	1.19	0.70	**4.64**	4.48	4.59
	MDD	45.38 %	17.05 %	43.65 %	32.69 %	33.49 %	17.62 %	18.72 %	4.82 %	4.15 %	**4.83 %**
N225	CR↑	2.98 %	−3.08 %	−2.24 %	−7.30 %	−15.90 %	23.31 %	44.22 %	564.87 %	492.47 %	**696.58 %**
	AR↑	8.83 %	−7.03 %	−4.91 %	−25.15 %	−4.48 %	12.14 %	12.87 %	76.58 %	72.75 %	**82.34 %**
	SR↑	0.39	−0.28	−0.23	−0.98	−0.25	0.48	0.80	4.47	4.44	**4.75**
	MDD↓	16.61 %	24.67 %	27.58 %	38.94 %	36.56 %	19.41 %	15.95 %	4.28 %	3.85 %	**2.46 %**
SP500	CR↑	0.54 %	−0.54 %	1.46 %	−1.57 %	−37.85 %	26.22 %	90.21 %	470.91 %	553.27 %	**554.09 %**
	AR↑	4.89 %	1.90 %	7.91 %	−4.06 %	−25.03 %	18.37 %	21.24 %	69.23 %	73.39 %	**73.51 %**
	SR↑	0.17	0.05	0.30	−0.15	−1.09	0.76	1.37	4.21	**4.63**	4.46
	MDD↓	25.05 %	26.70 %	16.06 %	24.28 %	47.15 %	20.80 %	11.69 %	4.50 %	**2.73 %**	2.79 %
KS11	CR↑	−2.69 %	2.69 %	−0.36 %	4.70 %	−5.00 %	−9.04 %	18.71 %	488.77 %	479.35 %	**552.84 %**
	AR↑	−5.86 %	7.70 %	0.86 %	12.20 %	−0.46 %	−2.69 %	6.33 %	72.14 %	71.60 %	**75.46 %**
	SR↑	−0.24	0.40	0.04	0.57	−0.03	−0.11	0.35	4.74	4.75	**5.05**
	MDD↓	34.23 %	13.29 %	29.59 %	15.47 %	24.98 %	33.65 %	21.60 %	4.07 %	4.38 %	**3.10 %**



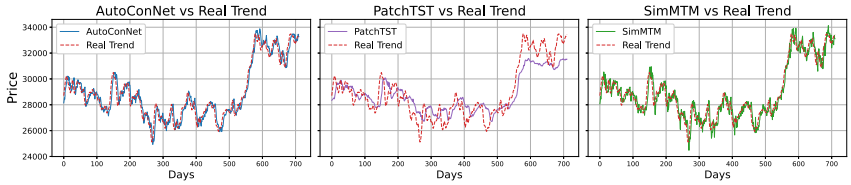
(a) DJI



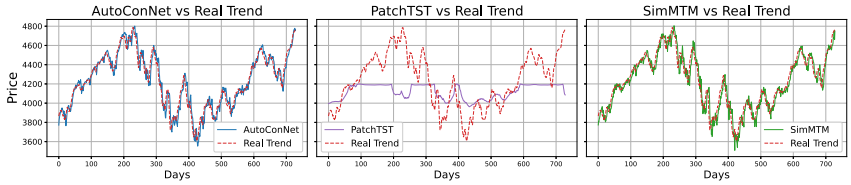
(b) FCHI



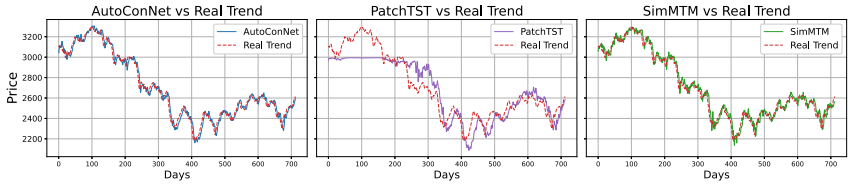
(c) HSI



(d) N225



(e) SP500



(f) KS11

Fig. 5. Prediction of AutoConNet, PatchTST and SimMTM vs Real Trend on six datasets.

observed for AutoConNet and SimMTM, indicating its overall inferior performance.

When comparing AutoConNet and SimMTM, we find that their performance metrics-including MSE, MAE, Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE)-are quite similar, with only slight variations between them. However, on a broader scale, AutoConNet consistently outperforms SimMTM across all six datasets, albeit with marginal improvements. This suggests that AutoConNet is a more robust model for the task at hand, achieving better overall prediction accuracy and stability.

Therefore, it can be concluded that AutoConNet is better suited for the objectives of this study, offering a slight yet consistent advantage over SimMTM in terms of predictive performance.

4.5.2. *Comparison with baselines*

To better assess the effectiveness of the proposed method, the test results of the IFF-DRL agent have been used for comparative analysis against the baselines mentioned earlier.

It is worth noting that although the proposed method introduces an incremental learning algorithm during the testing phase to enhance the

Table 6
The prediction accuracy of AutoConNet with Elastic Weight Consolidation.

Dataset	AutoConNet				AutoConNet (EWC)			
	MSE↓	MAE↓	RMSE↓	MAPE↓	MSE↓	MAE↓	RMSE↓	MAPE↓
DJI	0.0050	0.053	0.067	0.256	**0.0043**	**0.043**	**0.061**	**0.230**
FCHI	0.0050	0.048	0.061	0.305	**0.0047**	**0.047**	**0.060**	**0.290**
HSI	0.0041	0.039	0.057	0.207	**0.0039**	**0.038**	**0.056**	**0.197**
KS11	0.0022	0.033	0.042	0.240	**0.0021**	**0.033**	**0.042**	**0.220**
SP500	0.0048	0.049	0.062	0.227	**0.0047**	**0.046**	**0.060**	**0.218**
N225	0.0058	0.056	0.072	0.296	**0.0056**	**0.056**	**0.071**	**0.280**

pre-trained time series forecasting network, AutoConNet, with continuous learning capabilities and improved prediction accuracy, the incremental learning algorithm will not be applied in this phase. The focus here is to validate the improvement in the reinforcement learning agent's performance through the use of daily & weekly data generated from both predicted and observed prices. On the other hand, to verify whether the proposed method improves performance across different reinforcement learning algorithms, three classical algorithms-DDQN, A2C, and PPO-were integrated into the proposed framework, and called IFF-DDQN, IFF-A2C, and IFF-PPO, respectively. The detailed experimental results are shown in Table 5. As shown in Table 5, the proposed method outperformed all baseline approaches in terms of CR, SR, AR, and MDD. Across the six datasets involved in the experiments, IFF-DDQN, IFF-A2C, and IFF-PPO each achieved over 390 % CR, while the best baseline method only reached 127.84 % CR. Among the proposed methods, the one combined with PPO, IFF-PPO, delivered the best performance, achieving a 1462.42 % CR and 103.19 % AR on the HSI dataset. On the other hand, IFF-DDQN, the method combined with DDQN, had the lowest performance, with a CR of 395.09 % on the DJI dataset. However, this still significantly exceeded all baseline methods. These results indicate that the proposed method effectively enhances traditional reinforcement learning algorithms, enabling the agent to gain a clearer understanding of price data during both the training and testing phases, leading to superior trading strategies.

4.5.3. *Improving prediction accuracy with incremental learning*

As being discussed in the previous sections, an incremental learning algorithm will be introduced into the current framework once the testing phase is begin. The primary goal of incremental learning is to enhance the continue learning ability of the selected time-series prediction network, improving its prediction accuracy even during testing. In this experiment, a classic incremental learning algorithm online EWC has been used during testing phase. As a variant of EWC method, online EWC is widely used in lots of incremental learning paradigm and achieved astonishing result in recent years. Detail explanation of online EWC has shown in Section 3.4.

Table 6 showed the comparative analysis of the prediction accuracy of AutoConNet before and after incorporating online EWC method in testing phase. It is evident that the MSE, MAE, RMSE and MAPE has decreased with the help of incremental learning. Even in KS11 dataset, which AutoConNet got a extremely small MSE and MAE, online EWC can still improve its performance.

To further demonstrate the effectiveness of the proposed method, a comparative analysis between current framework with and without Incremental Learning is necessary. Due to the fact that a lower MSE and MAE demonstrate a smaller gap between prediction and ground-truth. Therefore, the proposed method should achieve a better result with the help of online EWC method. Tables 7–9 have shown the comparison result.

The bold part of three tables represent the best performance. As introduced in previous section, IFF-DDQN, IFF-A2C and IFF-PPO means the Incremental Forcasting Fusion Deep Reinforcement Learning (IFF-DRL) integrated with three traditional DRL algorithm, which involve incremental learning during testing phase. In this case, FF-DQN, FF-A2C and FF-PPO means the proposed method without incremental learning. It can be observed that with the help of online EWC algorithm, the prediction accuracy of AutoConNet has been increased, resulting an improvement of the performance of three agents. However, not all metrics

Table 7
The performance of FF-DDQN and IFF-DDQN on six datasets.

Method	Datasets	CR↑	AR↑	SR↑	MDD↓
IFF-DDQN	DJI	**395.09 %**	**64.43 %**	**4.50**	**2.53 %**
	FCHI	**450.37 %**	**70.56 %**	**4.34**	7.36 %
	HSI	**1410.73 %**	**102.15 %**	**4.64**	**4.82 %**
	N225	**564.87 %**	**76.58 %**	**4.47**	**4.28 %**
	SP500	**470.91 %**	**69.23 %**	**4.21**	4.50 %
	KS11	**488.77 %**	**72.14 %**	**4.74**	4.07 %
FF-DDQN	DJI	297.06 %	56.85 %	3.91	5.86 %
	FCHI	408.20 %	67.95 %	4.02	**2.35 %**
	HSI	1275.02 %	99.58 %	4.36	6.33 %
	N225	484.72 %	72.45 %	4.13	4.51 %
	SP500	390.90 %	64.48 %	3.73	**3.89 %**
	KS11	479.46 %	71.63 %	4.68	**3.87 %**

Table 8
The performance of FF-A2C and IFF-A2C on six datasets.

Method	Datasets	CR↑	AR↑	SR↑	MDD↓
IFF-A2C	DJI	**435.06 %**	**66.91 %**	**4.75**	4.37 %
	FCHI	**482.74 %**	**72.46 %**	**4.43**	**3.84 %**
	HSI	**1244.17 %**	**98.79 %**	**4.48**	**4.15 %**
	N225	**492.47 %**	**72.75 %**	**4.44**	**3.85 %**
	SP500	**553.27 %**	**73.39 %**	**4.46**	**2.79 %**
	KS11	**479.35 %**	**71.60 %**	**4.75**	4.38 %
FF-A2C	DJI	407.11 %	65.21 %	4.54	**3.42 %**
	FCHI	382.43 %	66.14 %	4.00	4.18 %
	HSI	1133.20 %	96.31 %	4.31	8.42 %
	N225	430.89 %	69.25 %	3.96	6.24 %
	SP500	365.94 %	62.82 %	3.54	8.47 %
	KS11	401.41 %	66.87 %	4.27	**3.28 %**

Table 9
The performance of FF-PPO and IFF-PPO on six datasets.

Method	Datasets	CR↑	AR↑	SR↑	MDD↓
IFF-PPO	DJI	**459.83 %**	**68.38 %**	**4.78**	**3.21 %**
	FCHI	**482.10 %**	**72.40 %**	**4.49**	**3.43 %**
	HSI	**1462.42 %**	**103.19 %**	**4.59**	**4.83 %**
	N225	**696.58 %**	**82.34 %**	**4.75**	**2.46 %**
	SP500	**554.09 %**	**73.51 %**	**4.46**	**2.79 %**
	KS11	**552.84 %**	**75.46 %**	**5.05**	**3.10 %**
FF-PPO	DJI	343.15 %	60.82 %	4.13	4.92 %
	FCHI	388.56 %	66.69 %	3.81	5.09 %
	HSI	1206.72 %	98.07 %	4.33	4.83 %
	N225	560.70 %	76.47 %	4.29	7.47 %
	SP500	413.60 %	66.00 %	3.72	5.44 %
	KS11	494.31 %	72.49 %	4.64	4.04 %

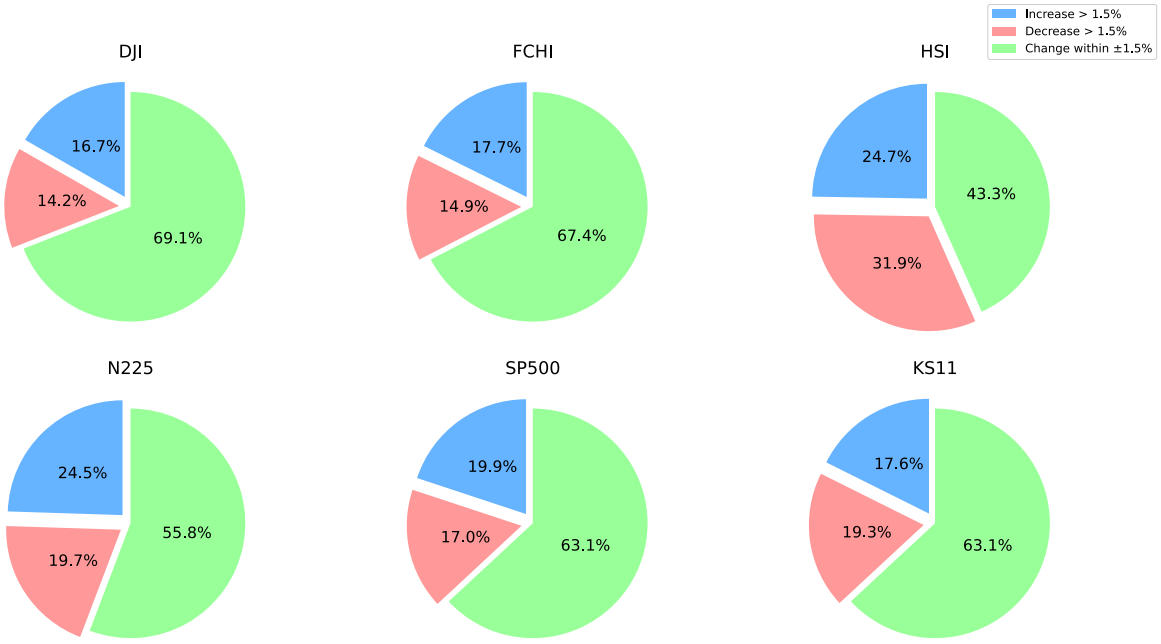


Fig. 6. Comparison of Future 3-Day Price Change Distribution Across 6 Datasets.

showed improvement. By observing the Maximum Drawdown (MDD), it can be seen that, except for IFF-PPO, the MDD of the other two algorithms decreased to some extent on certain datasets. For example, the MDD of IFF-DDQN on the FCHI dataset increased from 2.35 % to 7.36 %. The MDD of IFF-A2C on the DJI dataset increased from 3.42 % to 4.37 %. These reductions in MDD also made the improvement in CR (cumulative return) relatively less prominent. Interestingly, the proposed method showed the most significant improvement for the PPO algorithm. IFF-PPO not only achieved improvements across all metrics but also outperformed the other two algorithms in average performance. This could, to some extent, indicate the effectiveness of the PPO algorithm in single-asset trading. Future research could focus on exploring the PPO algorithm further.

However, by observing the experimental results, it can be found that the results on the HSI dataset are unexpectedly high. The CR on the other datasets mostly fluctuates between 300 % and 600 %, with only a few exceeding 600 %. In contrast, all CR values on the HSI dataset exceed 1100 %. To avoid potential issues, it is necessary to analyze the dataset. Fig. 6 shows the proportion of closing prices in the test sets of six datasets that fluctuate by more than 1.5 % over three days. As mentioned earlier, the reward function we used is the Min-Max function, which assigns the maximum price change ratio over the next three days as the reward for buying or selling. When the agent chooses to hold, the reward is calculated as a threshold minus the maximum price change ratio over those three days, with this threshold set to 1.5 % in our experiments. It is evident from the six pie charts that the HSI dataset has the highest proportion of closing prices fluctuating over 1.5 %, while the other datasets show less than half of that proportion, with HSI exceeding 50 %. Notably, the HSI dataset has 31.9 % of its declines surpassing 1.5 %. It's important to note that our experimental environment allows the agent to execute short-selling actions, and since a significant portion of HSI is in a downtrend, it is understandable why the experimental results on the HSI dataset are so high.

4.6. Further analysis of model reliability and robustness

To further validate the reliability and robustness of the proposed method, it is necessary to evaluate the model's performance in response to unexpected events through additional experiments. As described in Section 4.1, the training dataset used in this study includes price data from January 1, 2007, to December 31, 2020, while the test dataset spans January 1, 2021, to December 31, 2023. However, it is widely known that the Covid-19 pandemic began in January 2020 and lasted for nearly three years, ending around late 2022. Evidently, the training dataset used in this study includes part of the data from the pandemic period. To demonstrate the reliability of the proposed method and its robustness to unforeseen events, experiments should be conducted using a training dataset that excludes the pandemic period. The pandemic and post-pandemic periods would then serve as the test dataset, allowing us to observe the model's performance across multiple datasets.

Therefore, the IFF-DRL method was tested on aforementioned six datasets, where the training set spanned from January 1, 2007, to December 31, 2019, and the test set covered January 1, 2020, to December 31, 2023. The experimental results are recorded in Table 10. It can be observed that, despite the test set length increasing by nearly one year to a total of four years, the trading performance of IFF-DRL on DJI, FCHI, HSI, and KS11 is lower than the corresponding results shown in Tables 7–9, where the test set length was only three years for the same algorithm. Additionally, a comparison reveals that the experimental results in Table 10 exhibit relatively high Maximum Drawdown (MDD), with most exceeding 10 %. Even for SP500 and N225, which showed relatively favorable outcomes, the MDD values remain high. This clearly demonstrates the negative impact of the Covid-19 pandemic, as an unforeseen event, on trading performance.

To further observe the impact of including or excluding Covid-19 period data in the training set on the trading performance of IFF-DRL, the models trained in Table 9 will be tested on the test set containing data from January 1, 2021, to December 31, 2023. This test set is consistent with the one used in the experiments of Sections 4.5.2 and 4.5.3. The results of these tests are recorded in Tables 11–13. In these three tables, IFF-DRL (with Covid-19) refers to the experimental results where the training set includes data from the pandemic period, which have been discussed in Section 4.5.3. On the other hand, IFF-DRL (without Covid-19) represents the experimental results where the training set excludes data from the pandemic period.

From the three tables, we can observe that when the training set does not include data from the pandemic period, the trading performance of IFF-DRL significantly declines. For example, in Table 11, the cumulative return of IFF-DDQN on the DJI dataset dropped from 395.09 % to 73.72 %, while on the HSI dataset, the cumulative return decreased from

Table 10
The performance of IFF-DRL with training set from 2007.1.1 to 2019.12.31 and testing set from 2020.1.1 to 2023.12.31.

Dataset	IFF-DQN				IFF-A2C				IFF-PPO			
	CR↑	AR↑	SR↑	MDD↓	CR↑	AR↑	SR↑	MDD↓	CR↑	AR↑	SR↑	MDD↓
DJI	164.35 %	31.11 %	1.42	14.49 %	221.09 %	35.75 %	1.67	12.58 %	342.19 %	42.84 %	2.09	7.79 %
FCHI	377.52 %	47.84 %	2.33	9.94 %	484.89 %	52.46 %	2.55	17.44 %	700.24 %	59.00 %	3.02	6.06 %
HSI	870.02 %	60.19 %	3.06	15.16 %	889.57 %	60.62 %	3.02	18.15 %	1121.54 %	64.57 %	3.19	14.21 %
N225	855.43 %	59.97 %	3.58	7.21 %	981.69 %	62.32 %	3.74	12.47 %	812.98 %	59.15 %	3.44	14.53 %
SP500	950.16 %	59.7 %	3.14	5.73 %	779.45 %	56.51 %	2.9	6.29 %	541.87 %	50.54 %	2.53	6.77 %
KS11	389.29 %	45.88 %	2.52	10.62 %	436.08 %	47.85 %	2.63	10.68 %	525.07 %	50.97 %	2.93	6.83 %

Table 11
Comparison of IFF-DDQN Performance Between Training Sets With and Without Covid-19 Period Data.

Method	Datasets	CR↑	AR↑	SR↑	MDD↓
IFF-DDQN (with Covid-19)	DJI	**395.09 %**	**64.43 %**	**4.50**	**2.53 %**
	FCHI	**450.37 %**	**70.56 %**	**4.34**	**7.36 %**
	HSI	**1410.73 %**	**102.15 %**	**4.64**	**4.82 %**
	N225	**564.87 %**	**76.58 %**	**4.47**	**4.28 %**
	SP500	**470.91 %**	**69.23 %**	**4.21**	**4.50 %**
	KS11	**488.77 %**	**72.14 %**	**4.74**	**4.07 %**
IFF-DDQN (without Covid-19)	DJI	73.72 %	26.39 %	1.59	9.02 %
	FCHI	151.09 %	42.48 %	2.39	9.91 %
	HSI	555.37 %	77.07 %	3.26	15.13 %
	N225	307.67 %	60.34 %	3.26	7.18 %
	SP500	256.09 %	53.76 %	2.99	5.73 %
	KS11	141.38 %	40.68 %	2.27	10.63 %

Table 12
Comparison of IFF-A2C Performance Between Training Sets With and Without Covid-19 Period Data.

Method	Datasets	CR↑	AR↑	SR↑	MDD↓
IFF-A2C (with Covid-19)	DJI	**435.06 %**	**66.91 %**	**4.75**	**4.37 %**
	FCHI	**482.74 %**	**72.46 %**	**4.43**	**3.84 %**
	HSI	**1244.17 %**	**98.79 %**	**4.48**	**4.15 %**
	N225	**492.47 %**	**72.75 %**	**4.44**	**3.85 %**
	SP500	**553.27 %**	**73.39 %**	**4.46**	**2.79 %**
	KS11	**479.35 %**	**71.60 %**	**4.75**	**4.38 %**
IFF-A2C (without Covid-19)	DJI	87.05 %	29.33 %	1.88	10.51 %
	FCHI	151.84 %	42.59 %	2.41	10.77 %
	HSI	544.51 %	76.54 %	3.22	16.9 %
	N225	350.47 %	63.72 %	3.57	10.75 %
	SP500	232.59 %	51.36 %	2.87	6.28 %
	KS11	120.43 %	37.04 %	2.11	10.00 %

Table 13
Comparison of IFF-PPO Performance Between Training Sets With and Without Covid-19 Period Data.

Method	Datasets	CR↑	AR↑	SR↑	MDD↓
IFF-PPO (with Covid-19)	DJI	**459.83 %**	**68.38 %**	**4.78**	**3.21 %**
	FCHI	**482.10 %**	**72.40 %**	**4.49**	**3.43 %**
	HSI	**1462.42 %**	**103.19 %**	**4.59**	**4.83 %**
	N225	**696.58 %**	**82.34 %**	**4.75**	**2.46 %**
	SP500	**554.09 %**	**73.51 %**	**4.46**	**2.79 %**
	KS11	**552.84 %**	**75.46 %**	**5.05**	**3.10 %**
IFF-PPO (without Covid-19)	DJI	82.05 %	28.25 %	1.78	10.04 %
	FCHI	153.14 %	42.79 %	2.43	10.79 %
	HSI	550.22 %	76.81 %	3.25	19.56 %
	N225	296.92 %	59.36 %	3.26	14.36 %
	SP500	225.77 %	50.64 %	2.82	6.27 %
	KS11	118.66 %	36.76 %	2.05	10.66 %

1410.73 % to 555.37 %. It is evident that the IFF-DRL algorithm, when trained without the pandemic data, experienced a decline of more than 50 % in cumulative return across most datasets. Additionally, the increase in Maximum Drawdown (MDD) is also noteworthy. For instance, IFF-DDQN, IFF-A2C, and IFF-PPO all saw their MDD on the HSI dataset rise from less than 5 % to over 15 %. Obviously, when the training set excludes the pandemic data, the overall trading performance of IFF-DRL suffers a substantial decline, with many datasets performing less than half as well as before.

However, when comparing the experimental results from the three tables with those in Table 5, we can observe that, despite the absence of the critical pandemic data in the training set, IFF-DRL outperforms traditional methods such as Buy-and-Hold (B&H) and Mean Reversion (MR) across all datasets. When compared to baseline models such as TDQN, DQN-HER, and DQN-Vanilla, IFF-DRL also demonstrates superior performance, with better results in most datasets. For example, IFF-DDQN on DJI dataset achieves only 73.72 % CR, which is lower than the performance of DQN-HER shown in Table 5. However, the annualized return of IFF-DDQN, which yields 26.39 %, is way better than DQN-HER that only have 19.21 %. This implies that, although the unforeseen pandemic event had a negative impact on the proposed method, the approach still ensures relatively strong trading performance. The method exhibits robustness and adaptability, achieving substantial success even in the face of missing critical training data.

Based on the above experimental results, three conclusions can be drawn. First, IFF-DRL is highly reliable and robust, capable of delivering reliable trading strategies and excellent trading performance when confronted with unknown events. Second, this study incorporates a “short selling” mechanism within the trading framework, enabling participating agents to take advantage of declining stock prices by executing actions beyond merely buying or holding. By leveraging this capability, the proposed IFF-DRL framework demonstrates an enhanced ability to adapt to sudden market downturns. Even during sharp price drops, the framework can effectively navigate unfavorable conditions and achieve profitable outcomes, showcasing its robustness in volatile market environments. Third, these experiments demonstrate that IFF-DRL is not an algorithm that only performs well on data from specific time periods. The integration of the self-supervised predictive network enables it to react to unexpected events, thus avoiding potential large losses. This shows that the proposed method is adaptable and can maintain strong performance even in the presence of unforeseen disruptions.

4.7. Analysis of trading strategy

Following the analysis of the IFF-DRL’s effectiveness across various experiments, the actions taken by the agent within the datasets were further examined. Figs. 7–9 illustrate the trading actions executed by the IFF-DDQN agent across three datasets. These actions are superimposed on stock price trend charts, providing a clear visual representation of how the agent’s decisions align with price movements, making it easier to understand the rationale behind its trading strategies.

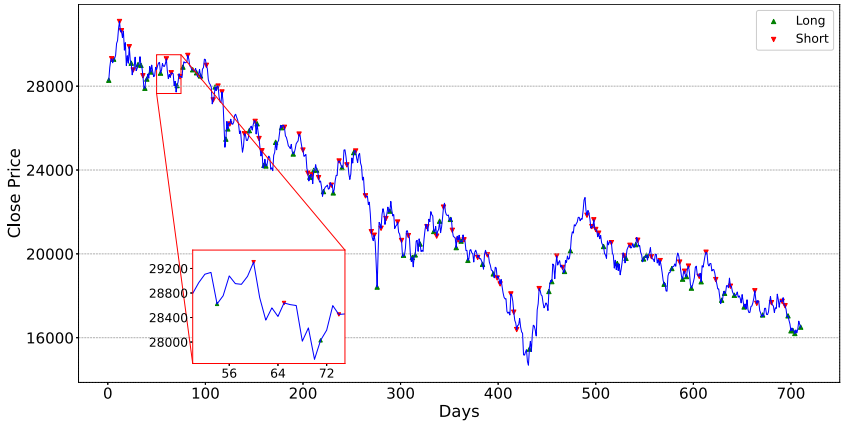


Fig. 7. Actions of IFF-DDQN Agent in HSI.

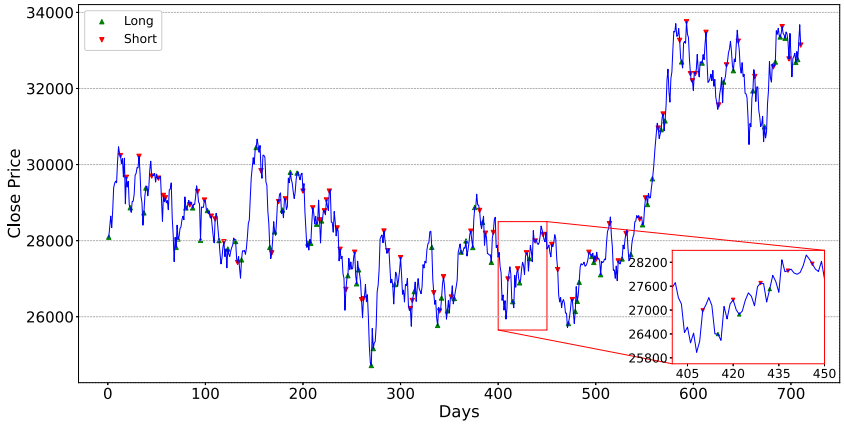


Fig. 8. Actions of IFF-DDQN Agent in N225.

Figs. 7 and 9 show the trading actions of the IFF-DDQN agent on the HSI and N225 datasets. A notable observation is that both datasets exhibit significant price volatility, which can be confirmed by Fig. 6, as these two datasets have the highest proportion of price fluctuations exceeding 1.5 % over the next three days. In response to this situation, the agent employs a high-frequency trading strategy. A close-up view of Figs. 7 and 9 clearly shows that stock prices experience substantial short-term fluctuations, and the agent often buys at the lows of these fluctuations and sells at the highs, undoubtedly resulting in profits.

When the agent encounters a situation with low price volatility, its strategy also changes accordingly. Taking the zoomed-in section of Fig. 9 as an example, the agent executed a “short” trading decision around day 325. A few days later, anticipating an upcoming price increase, the agent switched to a “long” trading decision and then refrained from trading for nearly 10 days until predicting a long-term downward trend starting around day 340, at which point it executed a “short” operation. Therefore, by observing the three figures, it can be confidently stated that the agent effectively monitors the market environment and price

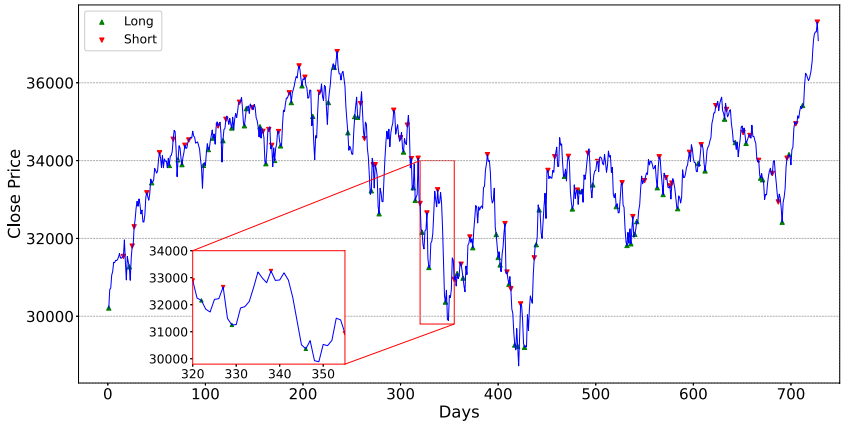


Fig. 9. Actions of IFF-DDQN Agent in DJI.

movement patterns, adjusting its trading strategy appropriately based on different conditions to achieve profits.

5. Discussion

In this comprehensive analysis of the proposed IFF-DRL, our experiment through various datasets and scenarios demonstrate a consistent outperformance of selected baseline methods, as measured by CR, AR, SR and MDD. The superiority of the proposed method, detailed in Table 5, can be attributed to its novel data generation mechanism. This unique mechanism allow a time-series prediction network to predict price data in the next N days base on the observed state, then generate the daily & weekly data that consist of observed and predicted price data. In this case, once the daily & weekly data is treated as the input state of reinforcement learning, agent have the chance to not only making decision by seeing future, but also observe the price data in a macro-level. This arrangement enhances the consistency and reliability of the learning process, particularly in the volatile environment of financial markets.

Further analysis of prediction accuracy has been shown in Section 4.5.3. The integration of online EWC algorithm increased the prediction accuracy of AutoConNet, resulting in the improvement of the overall performance of the proposed method. Since financial markets are inherently unstable and continuously evolving, the stock price patterns learned by time series prediction networks in the training set may not necessarily be effective in the test set. Therefore, the greatest significance of introducing the concept of incremental learning is to enable the network to maintain its learning capability and, to some extent, improve the accuracy of its predictions. Experiments show that with the help of the incremental learning algorithm, AutoConNet's prediction accuracy has been enhanced. Moreover, as the predicted price data more closely aligns with the real data, the agent made more optimal trading decisions during the trading process, leading to an overall performance improvement.

Furthermore, the proposed method not only improves a single deep reinforcement learning algorithm but also shows significant enhancement across several types of algorithms, including value-based, actor-critic based, and policy-based methods. By observing Tables 7–9, it can be seen that the PPO algorithm demonstrates the best performance under the proposed method.

However, this study has several limitations. First, the experiments conducted in this research were limited to only a few indices and stocks, and the potential for generalizing the model to other assets or broader financial markets was not explored. This restriction limits the scalability of the proposed methods, and future work could focus on testing these algorithms across a wider range of assets and market conditions to assess their generalizability.

Second, the study only used daily and weekly OHLCV data as input for the models, without exploring the possibility of incorporating additional financial indicators that could provide more comprehensive insights into market behavior. Including other features, such as macroeconomic variables, sentiment analysis, or technical indicators, could potentially improve the models' ability to capture the complexities of financial markets and enhance their predictive power.

Third, this research only used online EWC algorithm to improve the prediction accuracy of AutoConNet and did not explore the use of other incremental learning methods. However, it does not imply that online EWC is the only choice for incremental learning algorithms. Classical methods such as replay buffer, online gradient descent, or recursive least squares can also be applied to the proposed method. The experiments on incremental learning in this paper aim to explore whether time series prediction networks with continual learning capabilities lead to better prediction accuracy, and whether higher accuracy improves the agent's performance in trading. In future work, some cutting-edge incremental learning methods will be incorporated into the research.

6. Conclusion

This paper introduces a novel deep reinforcement learning framework called Incremental Forecasting Fusion Deep Reinforcement Learning (IFF-DRL), which significantly improves the trading performance of DRL agent in financial environment. In the proposed method, self-supervised learning helps model exploit abundant unlabeled data to uncover complex structures and temporal dependencies and incremental learning enables models to continuously adjust with new data to improve prediction accuracy. By incorporating a time-series prediction network AutoConNet to predict future price and generate daily & weekly data based on observed and predicted price data, the agent can not only see the potential price change pattern in the future, but also observe the market in a more macro level. In this case, deep reinforcement learning model can extract better latent expression from the data. Moreover, an incremental learning algorithm, Elastic Weight Consolidation (EWC) is introduced during testing phase, which enable the time-series network to have the ability to continue learning and also improve its prediction accuracy. This method's efficacy is validated through extensive testing across various datasets, including DJI, FCHI, HSI, SP500, N225, and KS11.

Key contributions of this paper include:

1. Introducing a novel framework that allow deep reinforcement learning agent can not only see the future price change but also observe price data in a macro level. As a result, the agent within the proposed framework can better extract useful latent expression and generate more optimal trading strategies.
2. The introduction of incremental learning algorithm improves the overall performance of IFF-DRL significantly. Due to the fact that financial market is ever changing, the price evolving pattern that time-series prediction network learned from training set may not be sufficient when it comes to testing set. In this case, incremental learning algorithm like EWC that being used in this paper enable the neural network keep learning during testing phase. As result, the overall prediction MSE of AutoConNet achieved an average reduction of 5.93 % across six datasets, which leads to the improvement of IFF-DRL's trading performance.
3. Incorporating self-supervised time-series prediction network AutoConNet to predict future financial price data. Self-supervised learning enable model to explore uncover complex structures and temporal dependencies, improving model generalization and pattern detection. As a result, AutoConNet provides precise prediction of future data that help improve trading performance.
4. Developing a new types of data, namely daily & weekly data, that provide a more macro-level perspectives to reinforcement learning agent, which significantly improves the trading efficiency.
5. The effectiveness of the proposed framework has been well demonstrated by combining with different deep reinforcement learning algorithm. Although Algorithm 1 used DDQN to demonstrate the basic idea of the proposed method, it does not mean that IFF-DRL can only combined with DDQN. The following experiment result in Section 4 showed that the proposed method can be combined with policy-based method like PPO or Actor-Critic based method like A2C.

Despite its strengths, the proposed framework still encounter some limitations that need to be addressed in the future research. Future work will focus on expanding the scope of experiments, incorporating additional data sources, and exploring the integration of different types of incremental learning algorithms to further improve the performance of the proposed methods. Furthermore, the incorporation of more financial indicators will be essential for further advancing the capabilities of financial reinforcement learning models.

Data availability

The data that has been used is confidential.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Chujin Zhou: Conceptualization, Methodology, Software, Validation, Writing - original draft; **Yuling Huang:** Writing - original draft, Conceptualization; **Yuting Kong:** Writing - original draft; **Xiaoping Lu:** Writing - review & editing, Supervision.

Acknowledgements

We would like to express our sincere gratitude to the anonymous reviewers for their insightful comments and valuable suggestions, which have significantly improved the quality and presentation of this work. This work are supported by the Science and Technology Development Fund, Macau SAR (File no. 0096/2022/A), and the Zhaoqing University Natural Science Fund (File no. 2024BSZ018 and qn202529).

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S. et al. (2023). Gpt-4 technical report. arXiv:2303.08774

de Azevedo Takara, L., Santos, A. A. P., Mariani, V. C., & dos Santos Coelho, L. (2024). Deep reinforcement learning applied to a sparse-reward trading environment with intraday data. *Expert Systems with Applications*, *238*, 121897.

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271

Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (vol. 4). Springer.

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, *7*(3), 1247–1250.

Chen, L., & Gao, Q. (2019). Application of deep reinforcement learning on automated stock trading. In *2019 IEEE 10th International conference on software engineering and service science (ICSESS)* (pp. 29–33). IEEE.

Cheng, L.-C., & Sun, J.-S. (2024). Multiagent-based deep reinforcement learning framework for multi-asset adaptive trading and portfolio management. *Neurocomputing*, *594*, 127800.

Cornalba, F., Disselkamp, C., Scassola, D., & Helf, C. (2024). Multi-objective reward generalization: Improving performance of deep reinforcement learning for applications in single-asset trading. *Neural Computing and Applications*, *36*(2), 619–637.

Dong, J., Wu, H., Zhang, H., Zhang, L., Wang, J., & Long, M. (2024). SimMTM: A simple pre-training framework for masked time-series modeling. *Advances in Neural Information Processing Systems*, *36*, 29996–30025.

Ehret, B., Henning, C., Cervera, M. R., Meulemans, A., Von Oswald, J., & Grewe, B. F. (2020). Continual learning in recurrent neural networks. arXiv:2006.12109

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, *3*(4), 128–135.

Friedman, E., & Fontaine, F. (2018). Generalizing across multi-objective reward functions in deep reinforcement learning. arXiv:1809.06364

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., & Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events, . *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 29996–30025. IEEE.

Ghania, M. U., Awaisa, M., & Muzammula, M. (2019). Stock market prediction using machine learning (ML) algorithms. *ADCAIJ: Adv Distrib Comput Artif Intell*, *8*(4), 97–116.

Grossberg, S. T. (2012). Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control, . *70*.

Gulcehre, C., Chandar, S., & Bengio, Y. (2017). Memory augmented neural networks with wormhole connections. arXiv:1701.08718

Gupta, S. C., Singh, S., Kumar, D. et al. (2022). Stock prediction using machine learning algorithms. *Grenze International Journal of Engineering & Technology (GIJET)*, *8*(1), 405–414.

Hodson, T. O. (2022). Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not. *Geoscientific Model Development Discussions*, *2022*, 1–10.

Huang, Y., Wan, X., Zhang, L., & Lu, X. (2024a). A novel deep reinforcement learning framework with biLSTM-attention networks for algorithmic trading. *Expert Systems with Applications*, *240*, 122581.

Huang, Y., Zhou, C., Cui, K., & Lu, X. (2024b). Improving algorithmic trading consistency via human alignment and imitation learning. *Expert Systems with Applications*, *253*, p. 124350.

Huang, Y., Zhou, C., Cui, K., & Lu, X. (2024c). A multi-agent reinforcement learning framework for optimizing financial trading strategies based on timesnet. *Expert Systems with Applications*, *237*, 121502.

Jing, B., Wang, Y., Sui, G., Hong, J., He, J., Yang, Y., Li, D., & Ren, K. (2024). Automated contrastive learning strategy search for time series. arXiv:2403.12641

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S. et al. (2019). Model-based reinforcement learning for atari. arXiv:1903.00374

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., & Choo, J. (2021). Reversible instance normalization for accurate time-series forecasting against distribution shift, . *International conference on learning representations*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A. et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526.

Lample, G., & Chaplot, D. S. (2017). Playing FPS games with deep reinforcement learning, . *Proceedings of the AAAI conference on artificial intelligence*, *31*(1).

Li, J., & Dai, Q. (2019). A new dual weights optimization incremental learning algorithm for time series forecasting. *Applied Intelligence*, *49*, 3668–3693.

Li, J., Dai, Q., & Ye, R. (2019). A novel double incremental learning algorithm for time series prediction. *Neural Computing and Applications*, *31*, 6055–6077.

Li, Y., Ni, P., & Chang, V. (2020). Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing*, *102*(6), 1305–1322.

Liu, Q., Zhuang, C., Gao, P., & Qin, J. (2024). CDFormer: When degradation prediction embraces diffusion model for blind image super-resolution, . *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, *195*, (pp. 7455–7464).

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., & Dustdar, S. (2021). Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting.

Luenberger, D. (2009). Investment science: International edition. OUP Catalogue.

Magdon-Ismail, M., & Atiya, A. F. (2004). Maximum drawdown. *Risk Magazine*, *17*(10), 99–102.

Melgar-García, L., Gutiérrez-Avilés, D., Rubio-Escudero, C., & Troncoso, A. (2023). A novel distributed forecasting method based on information fusion and incremental learning for streaming time series. *Information Fusion*, *95*, 163–173.

Moustakidis, V., Passalis, N., & Tefas, A. (2024). Online probabilistic knowledge distillation on cryptocurrency trading using deep reinforcement learning. *Pattern Recognition Letters*, *186*, 243–249.

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2022). A time series is worth 64 words: Long-term forecasting with transformers. arXiv:2211.14730

Park, J., Gwak, D., Choo, J., & Choi, E. (2024). Self-supervised contrastive forecasting. arXiv:2402.02023

Sharpe, W. F. (1994). The sharpe ratio. *Journal of portfolio management*, *21*(1), 49–58.

Sun, Q., Wei, X., & Yang, X. (2024). GraphSAGE with deep reinforcement learning for financial portfolio optimization. *Expert Systems with Applications*, *238*, 122027.

Taghian, M., Asadi, A., & Safabakhsh, R. (2022). Learning financial asset-specific trading rules via deep reinforcement learning. *Expert Systems with Applications*, (p. 116523).

Théate, T., & Ernst, D. (2021). An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, *173*, 114632.

Vijayarani, S., Suganya, E., & Jeevitha, T. (2020). Predicting stock market using machine learning algorithms. *IRJMETS, December-2020*.

Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., & Xiao, Y. (2023). MICN: Multi-scale local and global context modeling for long-term series forecasting. *The eleventh international conference on learning representation*

Wang, S., Wu, H., Shi, X., Hu, T., Luo, H., Ma, L., Zhang, J. Y., & Zhou, J. (2024). TimeMixer: Decomposable multiscale mixing for time series forecasting. arXiv:2405.14616

Wang, Y., & Yan, G. (2021). Survey on the application of deep learning in algorithmic trading. *Data Science in Finance and Economics*, *1*(4), 345–361.

Weng, B., Ahmed, M. A., & Megahed, F. M. (2017). Stock market one-day ahead movement prediction using disparate data sources. *Expert Systems with Applications*, *79*, 153–163.

Weng, B., Lu, L., Wang, X., Megahed, F. M., & Martinez, W. (2018). Predicting short-term stock prices using ensemble methods and online data sources. *Expert Systems with Applications*, *112*, 258–273.

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research*, *30*(1), 79–82.

Woo, G., Liu, C., Sahoo, D., Kumar, A., & Hoi, S. (2022). Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. arXiv:2202.01575

Wu, X., Chen, H., Wang, J., Troiano, L., Loia, V., & Fujita, H. (2020). Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences*, *538*, 142–158.

Yang, H., Liu, X.-Y., Zhong, S., & Walid, A. (2020). Deep reinforcement learning for automated stock trading: An ensemble strategy, . (pp. 1–8).

Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2023). Are transformers effective for time series forecasting? *Proceedings of the AAAI conference on artificial intelligence*, *37*(9), 11121–11128.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z. et al. (2023). A survey of large language models. arXiv:2303.18223

Zhou, C., Huang, Y., Cui, K., & Lu, X. (2024). R-DDQN: Optimizing algorithmic trading strategies using a reward network in a double DQN. *Mathematics*, *12*(11), 1621.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021a). Informer: Beyond efficient transformer for long sequence time-series forecasting. *35*(12), 11106–11115.

Zhou, W. J., Subagdja, B., Tan, A.-H., & Ong, D. W.-S. (2021b). Hierarchical control of multi-agent reinforcement learning team in real-time strategy (RTS) games. *Expert Systems with Applications*, *186*, 115707.