Article

A Self-Rewarding Mechanism in Deep Reinforcement Learning for Trading Strategy Optimization

Yuling Huang [1], Chujin Zhou [2], Lin Zhang [3], Xiaoping Lu [2,*]

1 School of Computer Science and Software, Zhaoqing University, Zhaoqing 526060, China; huangyuling@zqu.edu.cn
2 School of Computer Science and Engineering, Macau University of Science and Technology, Taipa 999078, Macao, China; 3220002751@student.must.edu.mo
3 School of Accounting and Finance, Beijing Institute of Technology, Beijing 100811, China; 08110@bitzh.edu.cn
* Correspondence: xplu@must.edu.mo

Abstract: Reinforcement Learning (RL) is increasingly being applied to complex decision-making tasks such as financial trading. However, designing effective reward functions remains a significant challenge. Traditional static reward functions often fail to adapt to dynamic environments, leading to inefficiencies in learning. This paper presents a novel approach, called Self-Rewarding Deep Reinforcement Learning (SRDRL), which integrates a self-rewarding network within the RL framework. The SRDRL mechanism operates in two primary phases: First, supervised learning techniques are used to learn from expert knowledge by employing advanced time-series feature extraction models, including TimesNet and WFTNet. This step refines the self-rewarding network parameters by comparing predicted rewards with expert-labeled rewards, which are based on metrics such as Min-Max, Sharpe Ratio, and Return. In the second phase, the model selects the higher value between the expert-labeled and predicted rewards as the RL reward, storing it in the replay buffer. This combination of expert knowledge and predicted rewards enhances the performance of trading strategies. The proposed implementation, called Self-Rewarding Double DQN (SRDDQN), demonstrates that the self-rewarding mechanism improves learning and optimizes trading decisions. Experiments conducted on datasets including DJI, IXIC, and SP500 show that SRDDQN achieves a cumulative return of 1124.23% on the IXIC dataset, significantly outperforming the next best method, Fire (DQN-HER), which achieved 51.87%. SRDDQN also enhances the stability and efficiency of trading strategies, providing notable improvements over traditional RL methods. The integration of a self-rewarding mechanism within RL addresses a critical limitation in reward function design and offers a scalable, adaptable solution for complex, dynamic trading environments.

Keywords: deep reinforcement learning; self-rewarding mechanism; human alignment; trading strategy

MSC: 68T07

Citation: Huang, Y.; Zhou, C.; Zhang, L.; Lu, X. A Self-Rewarding Mechanism in Deep Reinforcement Learning for Trading Strategy Optimization. *Mathematics* **2024**, *12*, 4020. https://doi.org/10.3390/math12244020

Academic Editor: Ioannis Tsoulos

Received: 11 November 2024
Revised: 16 December 2024
Accepted: 19 December 2024
Published: 22 December 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Reinforcement Learning (RL), a key technology in the field of Artificial Intelligence, has shown great potential in solving complex decision-making tasks in recent years, especially in the fields of gaming, fine-tuning large language models, and algorithmic trading. However, a central challenge in reinforcement learning is designing reward functions that are both dynamic and effective. Reward functions, which are a core component of reinforcement learning, have a decisive impact on the learning process and the development of the final strategy. Traditional reinforcement learning methods typically rely on predefined static reward functions [1–10]. Although these approaches can achieve the desired goal to some extent, their limitations become apparent when faced with unstable and complex environments such as financial markets. Predefined reward functions often lack sufficient flexibility, making it difficult to adapt to dynamic changes in the environment, thus limiting the performance and adaptability of the algorithm.

To overcome this challenge, researchers have begun to explore more adaptive approaches, such as utilizing neural networks to learn reward functions. This approach allows the reward function to be adjusted in real-time according to the dynamics of the environment by allowing the neural network to automatically learn the reward function from the data [11–14]. This dynamic reward function not only improves the adaptability of the algorithm, but also enables it to better handle complex relationships and achieve better generalization performance across various datasets. In addition, recent research has focused on combining domain knowledge with reinforcement learning. Techniques such as reward shaping enable domain knowledge to be effectively incorporated into the reinforcement learning process, improving the efficiency and performance of algorithms. Meanwhile, the proposal of the self-supervised reinforcement learning (SSRL) framework provides new avenues for reinforcement learning. This framework helps reinforcement learning by extracting useful information from data using unsupervised learning methods, further enhancing the adaptability and generalization capabilities of the algorithm [15].

With the continuous development of large language models, various fine-tuning methods have emerged. For example, human feedback-based fine-tuning methods optimize model behavior by incorporating the knowledge of human experts, enabling the model to better match human expectations and needs [16,17]. In addition, methods such as self-rewarding language models [18] and Direct Large Model Alignment (DLMA) [19] provide new ideas for reinforcement learning. These methods enhance the quality and efficiency of training results by enabling models to iteratively refine their reward systems.

Inspired by these advances, this paper introduces an innovative Self-Rewarding Deep Reinforcement Learning (SRDRL) mechanism that incorporates a self-rewarding system into various RL algorithms, including value-based, policy-based, and actor-critic approaches. The proposed mechanism allows the model to update its RL strategy by selecting the higher reward from either its own predicted rewards or expert-labeled rewards (expert knowledge) during the training process. This ensures that the agents effectively combine expert knowledge with learned predictions, leading to informed and profitable trading decisions while optimizing overall strategy performance. Based on experimental comparisons, we conclude that the value-based approach is more applicable to the financial domain. Consequently, this paper introduces a self-rewarding mechanism with Double Deep Q-Network (DDQN), named Self-Rewarding Double Deep Q-Network (SRDDQN), specifically for financial trading strategies to optimize reward allocation based on different expert labels (e.g., metrics such as Min-Max, Sharpe Ratio, and Return). Additionally, advanced time-series feature extraction networks (e.g., TimesNet, WFTNet, NLinear) are utilized to approximate the self-rewarding network. This method improves performance and scalability by comparing predicted rewards with expert-defined reward labels, leveraging an advanced time-series feature extraction network to enable the model to self-adjust its reward strategy, thus proposing a trading strategy that can adapt under complex market conditions.

The major contributions of this study are outlined as follows:

- The introduction of a novel self-rewarding mechanism for Deep Reinforcement Learning (DRL), named the SRDRL mechanism, which dynamically adjusts reward structures in real-time based on direct market feedback. This mechanism is applicable across a variety of RL algorithms, including value-based, policy-based, and actor-critic methods. SRDDQN, as the implementation of SRDRL's value-based RL approach, demonstrates that the self-rewarding training mechanism refines the learning process and drives the agent to make informed and profitable trading decisions while optimizing the overall strategy performance.
- A novel RL reward mechanism is designed, which selects the higher of the expert-defined reward and the self-predicted reward generated by the self-rewarding network. This hybrid reward approach ensures that agents effectively combine expert

domain knowledge with their own learned predictions, leading to more informed and profitable trading decisions while improving the overall strategy performance.

- The development of SRDDQN, an implementation of the self-rewarding mechanism within the Double Deep Q-Network (DQN) framework, addresses the overestimation bias common in standard DQN models. By integrating a self-rewarding network into the double Q-learning process, the model enhances stability and improves the agent's ability to make decisions in dynamic, uncertain financial markets.
- The integration of advanced time-series feature extraction techniques, including networks such as TimesNet, WFTNet, and NLinear, within the self-rewarding framework significantly enhances the accuracy of reward predictions. This fusion of advanced feature extraction with the self-rewarding architecture strengthens the system's ability to adapt to volatile market conditions and deliver more precise trading strategies.

This paper is organized as follows: In Section 2, previous studies on the topic are reviewed. Section 3 details the proposed method. Experimental results are described and analyzed in Section 4. Finally, Section 5 summarizes the findings and discusses future work.

2. Related Work

To deepen the understanding of the connections and distinctions in the literature related to the proposed methodology, this section closely examines two key research areas: reward functions in RL and deep reinforcement learning in algorithmic trading. Discussing and reviewing these related studies aims to provide a focused and organized review of current research trends.

2.1. Reward Functions in Reinforcement Learning

Traditional reward functions in reinforcement learning have predominantly been designed based on human experience and fixed formulas, focusing on calculating the reward value for each step in single-asset trading strategies. The design of reward functions is a critical aspect of RL as it directly influences the agent's behavior and the overall performance of the learning algorithm. The design of reward functions can vary significantly depending on the specific objectives and preferences of the investors involved in trading strategies [1–5,7,9,10,20–26].

For investors who are risk-averse, incorporating risk measures into the reward function is essential. A common approach is to use the Sharpe Ratio, which balances returns against the volatility of those returns, thereby providing a measure of risk-adjusted performance. This allows the RL agent to optimize not just for profitability but also for stability in returns, aligning with the preferences of risk-averse investors [2,3,10,21,24]. For example, Rodinos et al. proposed a reward-shaping method based on the Sharpe ratio, optimizing DRL agents by considering both Profit & Loss (PnL) and the Sharpe ratio. This method aims to improve overall portfolio performance by reducing the risk stemming from the agent's decisions [27]. However, some investors prioritize short-term gains and may prefer reward functions that emphasize immediate returns. By focusing on immediate returns, the RL agent can be guided to make decisions that maximize short-term profitability, which is particularly appealing in fast-paced trading environments where quick gains are valued [1,7,9,26,28]. Furthermore, there are investors who adopt a multi-objective perspective, seeking to balance both short-term returns and risk measures such as the Sharpe Ratio. In such cases, the reward function is designed to incorporate multiple objectives, allowing the RL agent to optimize for a combination of immediate profitability and risk-adjusted-performance. This multi-objective approach provides a more comprehensive framework for strategy optimization, catering to investors with diverse preferences. For example, Cornalba et al. investigated the effectiveness of multi-objective deep reinforcement learning in stock and cryptocurrency trading, dynamically incorporating reward functions and discount factors into the learning process [29].

However, reliance on human feedback and fixed formulas often limits the scalability and adaptability of models, especially in algorithmic trading, where market conditions are dynamic. Recent advancements include learning reward functions through neural networks, which offer adaptability to new data, the capability to model complex relationships and improved generalization. Grzes and Kudenko explored the enhancement of RL techniques through the incorporation of domain knowledge, addressing the challenges faced by RL in complex domains due to scalability issues [11]. Their study introduced the application of high-level STRIPS operator knowledge in reward shaping to refine the search towards the optimal policy. Proper and Tumer introduced a method for modeling global rewards through function approximation, enabling the fast computation of shaped difference rewards [12]. Li et al. developed enhanced versions of the DQN and Asynchronous Advantage Actor-critic (A3C) algorithms, applying them to the trading market [13]. Their reward function was designed around the Sharpe ratio to optimize risk-adjusted returns from trading behavior. Huang and Jin discussed the application and impact of reward shaping in self-organizing systems, especially in the context of multi-agent reinforcement learning [14]. Nair et al. introduced an innovative approach to representation learning, enhancing the ability of policies and reward functions to generalize across new domains [30]. This advancement means that in instances where a directly learned policy may not immediately apply to an unfamiliar situation (zero-shot generalization failure), a generalized reward function can still be employed to refine the robot's policy. This ensures adaptability and continued performance improvement when facing previously unseen challenges. Moreover, the inclusion of market sensitivity and sentiment analysis in reward functions has been noted, with models being developed that respond dynamically to market sentiments and fluctuations [31,32]. This is particularly beneficial in trading, where real-time feedback is crucial.

Emerging trends show a move toward self-supervised frameworks. Zhang et al. proposed a Self-Supervised Reinforcement Learning (SSRL) framework with a dual-reward mechanism to enhance recommendation accuracy and provide explanations by navigating knowledge graphs [15]. Traditional RL methods in this domain often rely on a singular, short-term reward, which may trap the learning process in local optima and overlook valuable paths. Yuan et al. proposed Self-Rewarding Language Models, in which the language model provides its own rewards during training through "LLM-as-a-Judge" prompts [18]. This method not only improved instruction-following ability during iterative Direct Preference Optimization (DPO) training but also increased the quality of the rewards provided by the model itself. Liu et al. introduced a novel method for automatically aligning large language models, DLMA [19]. The process involves using contrastive prompt pairs to autonomously generate preference data, which are further assessed to derive a self-rewarding score. The final step employs the DPO algorithm, which incorporates this self-rewarding score to efficiently align large language models.

This evolving landscape highlights a shift from traditional, static reward functions to dynamic, self-rewarding models that promise more responsive and profitable outcomes in algorithmic trading by enabling algorithms to autonomously refine their strategies based on continuous learning and real-time market feedback. This shift not only overcomes the limitations imposed by human-based reward systems but also opens new possibilities for autonomous learning and self-improvement in various applications.

2.2. Deep Reinforcement Learning in Algorithmic Trading

Algorithmic trading employs computer programs to execute high-speed, high-volume trades based on predefined rules and parameters, quickly reacting to market changes and capitalizing on short-term price fluctuations [33]. This approach has grown with advancements in technology, data availability, and the rise of high-frequency trading, thereby enhancing market liquidity and reshaping trading platforms [34–36].

The integration of machine learning algorithms into algorithmic trading offers a more adaptable, data-driven, and unbiased method, optimizing returns and managing risks [22]. These algorithms process vast datasets, eliminating emotional bias and increasing trading speed, which are crucial in markets such as equities, futures, and foreign exchange [37,38].

In the application of DRL to algorithmic trading, the process typically involves defining a trading environment where the agent interacts with market data to make buy, sell, or hold decisions. The DRL agent receives state information from the market, such as price movements, volume, and other relevant indicators, and learns to maximize cumulative rewards, which are often tied to profitability or risk-adjusted returns. This learning process involves exploring various trading strategies and exploiting the most effective ones based on feedback from the environment. Key challenges in applying DRL to trading include the non-stationary nature of financial markets, where the underlying data distribution can change over time, making it difficult for the agent to generalize from past experiences. Additionally, the high dimensionality and noise present in financial data can hinder the learning process, requiring sophisticated techniques for feature extraction and noise reduction. Another significant challenge is ensuring the stability and convergence of the DRL algorithms, as financial markets are highly volatile and can lead to unstable training dynamics. Furthermore, the risk of overfitting historical data poses a threat to the robustness of the developed trading strategies, necessitating robust validation methods to ensure performance in live trading scenarios.

Recent research has explored various aspects of algorithmic trading, including risk management, post-trade analysis, and trading in different assets like cryptocurrencies and single stocks [39–42]. Innovations in DRL have led to the development of models that significantly enhance the efficiency and accuracy of trading strategies [43,44].

Significant advancements have been made in the application of DRL to stock trading. For instance, Kong and So introduced the Remake Ensemble, a combination of multiple DRL methods, thereby enhancing automated stock trading performance [45]. Kochliaridis et al. combined DRL with rule-based systems and advanced neural architectures like transformers and U-Nets, improving adaptability to market conditions [32,46]. Zou et al. proposed a model using cascaded Long-Short Term Memory (LSTM) networks to better handle the noisy financial data typical of stock markets [47].

Further, Avramelou et al. introduced models that integrate diverse data types, such as market sentiment and price information, with DRL to create adaptive trading strategies without the need for retraining [48,49]. Park et al. demonstrated the efficacy of DRL using a robot trained with varied datasets to perform comparative stock investment analyses [50]. Lastly, Takara et al. developed the Extended Trading Deep Q-Network (ETDQN) model, which adjusts its learning process to trade efficiently under varying market conditions and employs distributed learning to enhance its DRL capabilities [51].

In the realm of algorithmic trading, advancements in DRL are significantly enhancing trading strategies through the integration of machine learning algorithms. These innovations are optimizing returns and managing risks by processing vast, complex datasets to make unbiased, high-speed trading decisions.

2.3. Motivation

Despite the advancements in DRL for algorithmic trading, several challenges persist that hinder the optimization of trading strategies. Traditional DRL approaches often struggle with the high dimensionality and non-stationarity of financial data, leading to difficulties in capturing complex market dynamics. Additionally, these methods may suffer from overfitting, where models perform well on historical data but fail to generalize to unseen market conditions. Furthermore, the balance between exploration and exploitation in trading strategies remains a critical issue, as excessive exploration can lead to significant financial losses, while insufficient exploration may result in suboptimal strategy performance.

Motivated by these challenges, our research introduces a self-rewarding mechanism within the DRL framework to enhance trading strategy optimization. The self-rewarding mechanism is designed to dynamically adjust reward signals based on market conditions and trading performance, thereby improving the agent's ability to adapt to changing environments and mitigate the risk of overfitting. By incorporating this mechanism, we aim to develop more robust and efficient trading strategies that can better navigate the complexities of financial markets. This motivation underscores the necessity for innovative approaches in DRL to achieve superior performance and reliability in algorithmic trading.

3. Method

This paper formalizes single-asset trading as a Markov decision process within an RL framework. The agent generates optimal trading strategies and executions by exploring a series of actions in a trading environment. The state consists of the opening, high, low, closing prices, and trading volume from the past 20 days and 20 weeks at time t. Actions are defined discretely as buy, sell, and hold. More details are provided in [52]. Self-rewarding function labels use Min-Max [52], Sharpe Ratio [44], and Return [53] indicators.

However, in exploring the intricacies of stock trading decisions, the need to establish the necessary assumptions to confirm the reliability of the conclusions is emphasized. The model assumes market efficiency, where prices reflect all available information, without being affected by external market influences. Building upon the Efficient Markets Hypothesis (EMH), which posits that financial markets are highly efficient in reflecting information, we incorporate the Adaptive Markets Hypothesis (AMH) to account for the dynamic and evolving nature of market participants and conditions. AMH suggests that market efficiency is not static but adapts based on environmental changes and the learning behaviors of investors, providing a more flexible and realistic framework, especially in less liquid markets or during periods of high volatility. It also assumes that individual trading orders by small investors do not significantly impact market prices due to the market's high liquidity and that there is no slippage in order execution. These assumptions are necessary for the simulation environment but may not hold under all market conditions, which should be considered when applying this model to real-world scenarios.

3.1. Overview of the Proposed Method

Inspired by self-rewarding language models, this paper proposes an innovative self-rewarding deep reinforcement learning (SRDRL) approach that integrates self-rewarding mechanisms into an RL framework, aiming to enhance the framework's capabilities in complex decision-making scenarios. As shown in Figure 1, the process is divided into two key phases aimed at refining the learning process and improving the efficiency of the trading strategies formulated by the RL method.

Figure 1 provides a comprehensive schematic of the SRDRL framework, illustrating the flow of data and interaction between the main components. The SRDRL framework comprises several main components: raw data processing, environment, replay memory, and the trading agent, which includes the Q Network, Target Network, and Reward Network. The raw data module processes input data such as daily and weekly OHLC (Open, High, Low, Close) prices and volume data to construct state features. These features are then fed into the environment, which dynamically tracks parameters like prices, trading volume, capital, position, and trading costs. The environment generates state and reward information that drives the reinforcement learning process. The trading agent consists of three interconnected networks: the Q Network and Target Network employ reinforcement learning algorithms to refine policy optimization through iterative updates of network parameters. Leveraging supervised learning techniques, the Reward Network predicts rewards based on advanced time-series feature extraction models such as TimesNet, WFT-Net, and NLinear. It approximates expert-labeled reward functions, enabling the system to evaluate the quality of actions taken. The computed actions and associated rewards are stored in the Replay Memory, which is instrumental in experience replay and batch sampling during training.

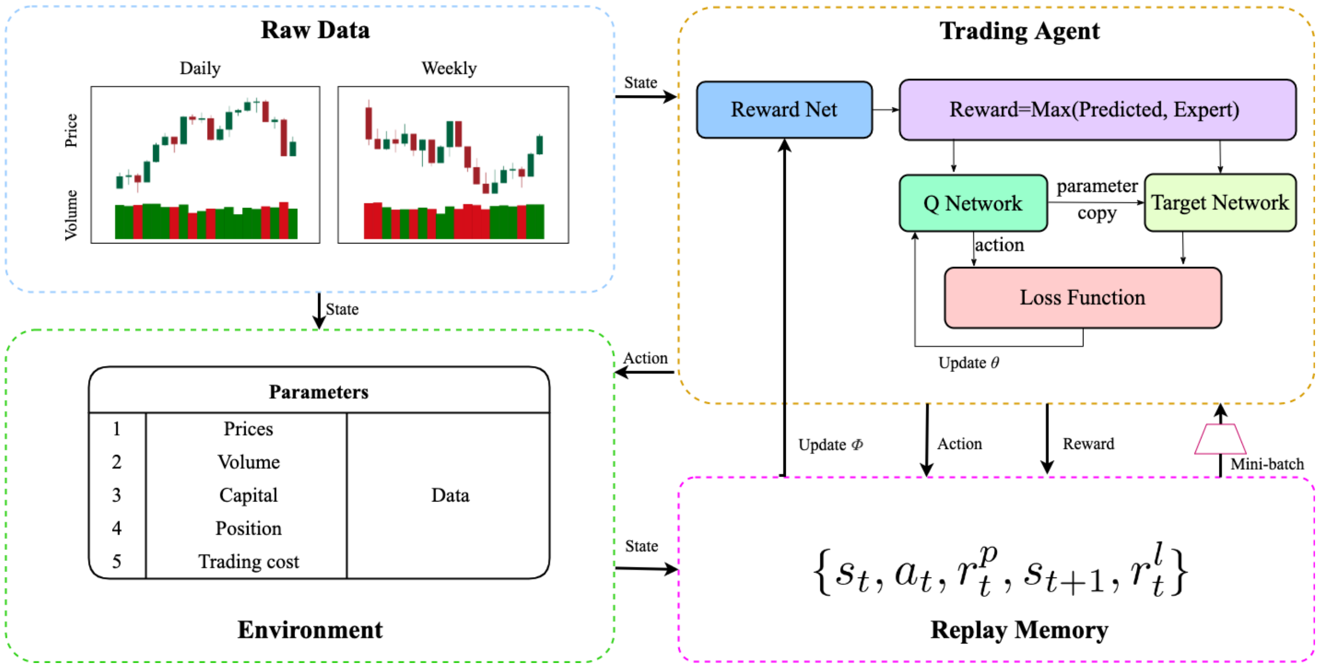


Figure 1. The structure of the proposed SRDDQN. The SRDRL framework comprises four components. The raw data module processes input data such as daily and weekly OHLC (Open, High, Low, Close) prices and volume data to construct state features. These features are then fed into the environment, which dynamically tracks parameters like prices, trading volume, capital, position, and trading costs. The environment generates state and reward information that drives the reinforcement learning process. The trading agent consists of three interconnected networks: the Q Network and Target Network employ reinforcement learning algorithms to refine policy optimization through iterative updates of network parameters. Leveraging supervised learning techniques, the Reward Network predicts rewards based on advanced time-series feature extraction models such as TimesNet, WFTNet, and NLinear. It approximates expert-labeled reward functions, enabling the system to evaluate the quality of actions taken. The computed actions and associated rewards are stored in the Replay Memory, which is instrumental in experience replay and batch sampling during training.

The SRDRL framework operates in two phases. In the first phase, the model employs supervised learning techniques to learn expert experience by approximating the self-rewarding function using advanced time-series feature extraction networks (e.g., TimesNet, WFTNet, and NLinear). This approach allows the model to update the self-rewarding function parameters by comparing its own predicted rewards with the reward labels set by experts based on various metrics such as Min-Max, Sharpe Ratio, and Return during the training process, selecting the better rewards as labels for supervised learning. This method allows the model to self-tune its reward strategy, which improves performance and scalability. The goal is to build a more generalized reward network that is adept at navigating the multifaceted nature of financial markets and capturing the wide range of market dynamics and trader intuition contained in expert reward functions.

In the second stage, the higher between the expert reward label and the self-rewarding network predicted reward is selected, which is then used as the reward for the RL framework and stored in the replay buffer. This method dynamically adjusts the learning trajectory based on a comparative analysis of the rewards generated by the expert reward and self-rewarding networks. By prioritizing higher rewards, the system inherently tends to make more profitable and strategically sound trading decisions, thus optimizing the overall performance of the trading strategy. The focus of this paper is to develop a self-rewarding Double DQN designed specifically for financial trading strategies to optimize reward allocation strategies based on different expert labels.

The two-phase integration of the self-rewarding mechanism ultimately creates a self-rewarding deep reinforcement learning environment. This environment not only allows for continuous improvement through the RL process but also facilitates the development of more complex and powerful reward networks. The fusion of advanced feature extraction, expert knowledge introduction, and dynamic reward selection results in a learning mechanism that is both adaptable and robust.

3.2. Self-Rewarding Mechanism

In traditional RL, the reward function is often manually designed and fixed, which can be limiting in dynamic environments such as financial markets. This subsection introduces a novel self-rewarding mechanism tailored for deep reinforcement learning, aimed to enhance the performance of optimized single-asset trading strategies. A self-rewarding mechanism allows the agent to adapt its reward function based on the environment and learning process, making it more flexible and responsive to changes. The RL agent can dynamically generate its own reward signals, instead of relying solely on predefined, static reward functions provided by human experts. As shown in Figure 2, the mechanism employs a training method similar to regression analysis. It assigns reward values to various states based on potential actions, identifies the highest rewards from these actions, and updates the learning based on the mean-square error with respect to the expert-defined rewards. The proposed method uses supervised learning to combine the insights of experienced market practitioners with advanced time series feature extraction networks, such as TimesNet [54], NLinear [55], and WFTNet [56]. These networks approximate a self-rewarding function, predict reward values, and then use expert-defined reward values as labels. Training through a backpropagation process guided by mean square error computation helps improve the accuracy of reward prediction.

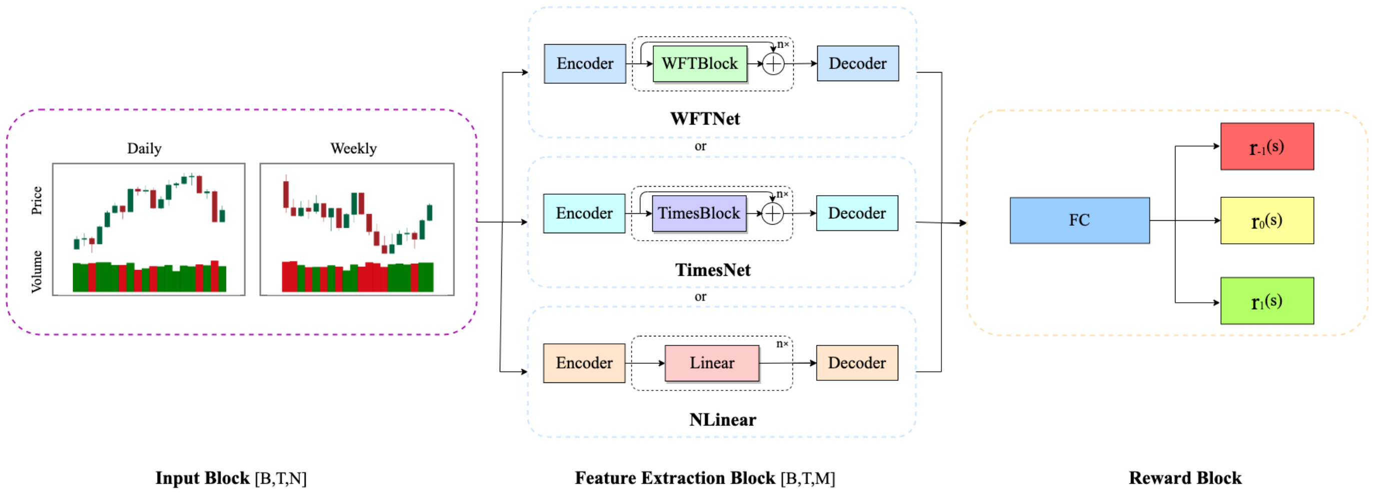


Figure 2. The schematic representation of the Reward Net. The Reward Net comprises three components: Input Block, Feature Extraction Block, and the Reward Block. This network uses supervised learning to acquire knowledge from experts by mimicking the self-rewarding function. It achieves this through the use of sophisticated time-series feature extraction models such as TimesNet, WFTNet, and NLinear.

Moreover, the mechanism can select different reward functions designated by experts—such as Min-Max, Sharpe Ratio, and Return—as labels to guide the training of the reward network via supervised learning. By comparing the predicted rewards with the expert-defined labels, the model selects the most favorable rewards. This process allows the self-rewarding network to adapt effectively to diverse market conditions.

Short-term Sharpe ratio reward: The Sharpe ratio, proposed by Sharpe et al. [57], evaluates strategy performance by assessing both return and risk. This paper employs a short-term Sharpe ratio as a reward function to measure the immediate influence of actions over a future time span denoted by k days, as defined in Equations (1)–(3):

$$r_t^{\mathrm{SSR}} = \mathrm{POS}_t \times R_t^*, \tag{1}$$

$$R_t^* = \frac{\mathrm{mean}(R_t^k)}{\mathrm{std}(R_t^k)}, \tag{2}$$

$$R_t^k = \left[\frac{p_{t+1} - p_t}{p_t}, \frac{p_{t+2} - p_t}{p_t}, \dots, \frac{p_{t+k} - p_t}{p_t}\right], \tag{3}$$

where POS_t denotes the position held at time t, R_t^k denotes the return for k days in the future, computed as the percentage difference in price (p_t) from time t to $t + k$.

Return reward: The return-based reward function quantifies the financial gain from each action [53]. It is formulated to capture the direct financial impact of a decision, providing immediate feedback on the financial outcomes of the agent's trading decisions. It is mathematically expressed in Equation (4):

$$r_t^{\mathrm{Return}} = \mathrm{POS}_t \times \frac{P_t - P_{t-1}}{P_{t-1}} \times 100, \tag{4}$$

where POS_t denotes the position held at time t, and P_t represents the asset's closing price at time t.

Min-Max reward: The Min-Max reward function, tailored to reflect market dynamics over various durations, maximizes profit by considering the greatest benefits from different stock positions [52]. This reward function's temporal adaptability allows it to capture market dynamics across different durations, maximizing profit. The function's customization potential enables its fine-tuning to align with specific market tendencies. It is formally defined in Equations (5)–(7):

$$r_t^{\text{Min-Max}} = \begin{cases} \mathrm{POS}_t * \mathrm{maxR}, & \text{if maxR} > 0 \text{ or} \\ \mathrm{maxR} + \mathrm{minR} > 0, & \\ \mathrm{POS}_t * \mathrm{minR}, & \text{if minR} < 0 \text{ or} \\ \mathrm{maxR} + \mathrm{minR} < 0, & \\ \text{const-(maxR-minR)}, & \text{otherwise.} \end{cases} \tag{5}$$

where POS_t denotes the position held at time t, and

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

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

with $r_{t+i} = \frac{p_{t+i} - p_t}{p_t} \times 100$.

Accordingly, the immediate reward $r_t \in \mathbb{R}^{1\times1}$ is computed as follows:

$$r_t = \begin{cases} r_t^s[a_t], & \text{if } r_t^s[a_t] \geq r_t^e[a_t]; \\ r_t^e[a_t], & \text{otherwise,} \end{cases} \tag{8}$$

where $r_t^s \in \mathbb{R}^{3\times1}$ is the output from the Reward Net, and $r_t^e \in \mathbb{R}^{3\times1}$ is the expert-defined reward.

3.3. Self-Rewarding Double Deep Q-Network

The self-rewarding mechanism extends across various RL algorithms, including value-based methods like DQN, policy-based methods like Proximal Policy Optimization (PPO), and actor-critic methods like Advantage Actor-Critic (A2C). This paper focuses on the development of a Self-Rewarding Double DQN (SRDDQN) specifically designed for financial trading strategies, aiming to optimize reward allocation using expert labels and learned predictions.

The Double DQN framework addresses the overestimation bias inherent in DQN models, where Q-values may be inaccurately inflated, leading to suboptimal decisions. Double DQN separates the action selection process from value estimation by maintaining two distinct neural networks: the main network (θ) and the target network (θ^-). The main network predicts Q-values for selecting actions, while the target network is used for value updates, reducing bias and stabilizing learning.

For each action, a transition tuple (s_t, a_t, r_t, s_{t+1}) is created, where s_t is the current state, a_t is the action taken, r_t is the reward obtained, and s_{t+1} is the next state. The reward r_t is determined by comparing the predicted reward from the self-rewarding network (r_t^s) with the expert-defined reward (r_t^e), selecting the higher of the two:

$$r_t = \max(r_t^s[a_t], r_t^e[a_t]). \tag{9}$$

The self-rewarding network (more details can be seen in Figure 2) estimates reward values for each action. These predicted rewards are compared against expert-labeled rewards (expert knowledge), which are based on financial metrics like Min-Max, Sharpe Ratio, and Return.

The Q-value update is computed using the following equation:

$$Q(s_t, a_t; \theta) \leftarrow Q(s_t, a_t; \theta) + \alpha\left[r_t + \gamma Q(s_{t+1}, a_{\max}(s_{t+1}; \theta^-)) - Q(s_t, a_t; \theta)\right], \tag{10}$$

where α is the learning rate, and γ is the discount factor. The term $a_{\max}(s_{t+1}; \theta) = \arg\max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta)$ represents the action that maximizes the Q-value in the next state s_{t+1}.

The two networks (main and target) are updated asynchronously to ensure stability, where the target network (θ^-) is updated to match the main network (θ) every N^- steps.

3.4. SRDDQN Training

The training process of SRDDQN is systematically outlined in Algorithm 1. The algorithm is based on Double DQN to train an optimal trading strategy. At each time step, the agent selects an action a_t based on the current state s_t using an ϵ-greedy policy. After executing the action, the agent observes the next state s_{t+1} and calculates the corresponding reward. The reward network and the expert reward generation function calculate reward values for all possible actions. The two sets of rewards, r_t^s (predicted by the self-rewarding network) and r_t^e (expert-labeled rewards), are compared. The higher reward is selected as the final reward r_t for training. This reward is used both to update the agent's Q-values and to train the self-rewarding network. When the number of transitions stored in the replay buffer exceeds the batch size, a batch of transitions of size N_b is randomly sampled from the buffer for training both the agent and the reward network. The agent's Q-network is updated using the TD target, while the reward network is updated using the selected reward r_t^l. Notably, the agent's parameters (θ) and the reward network's parameters (ϕ) are updated synchronously during training.

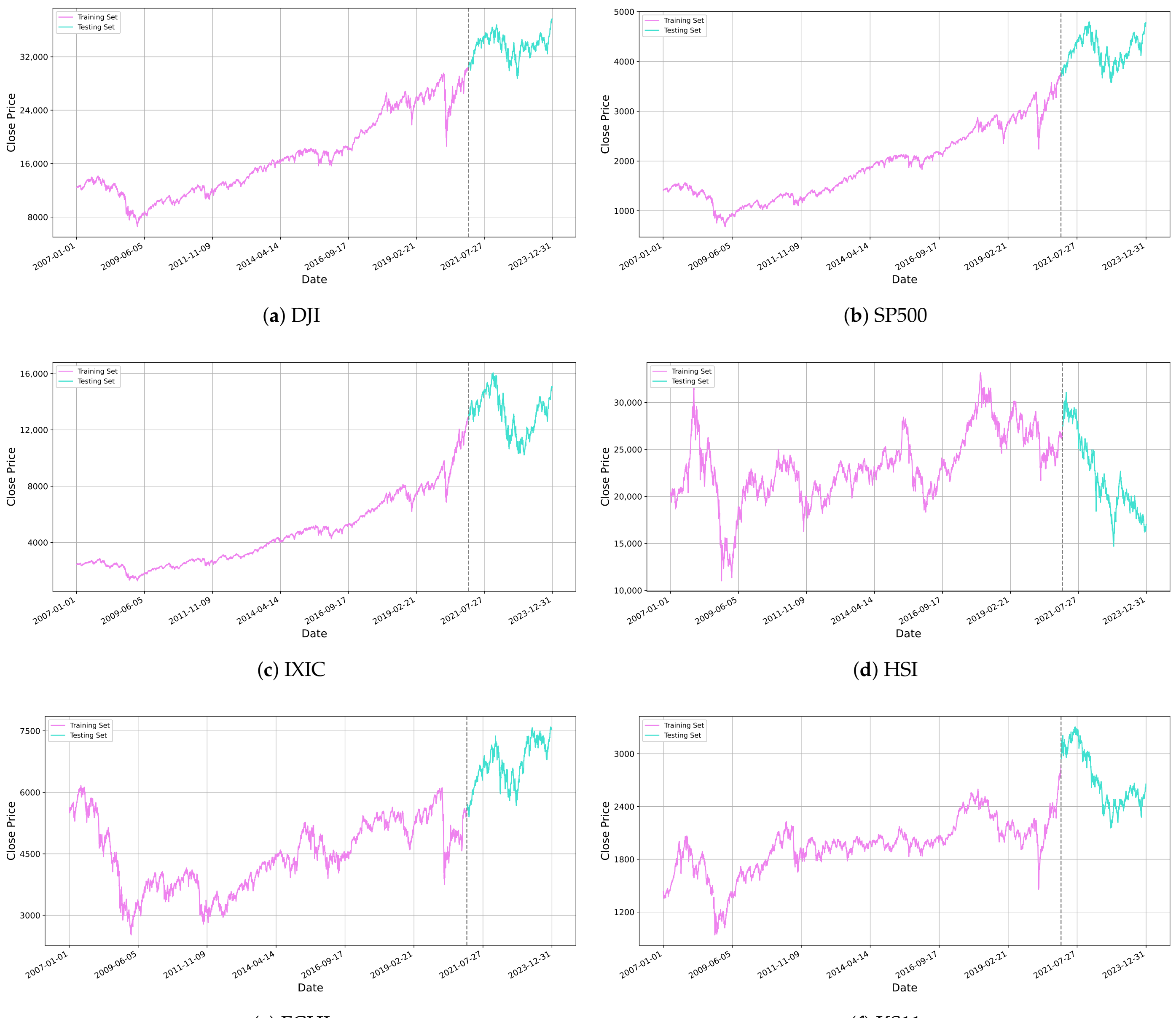
Algorithm 1 SRDDQN algorithm.

```
1: Input: $\mathcal{B}$—empty replay buffer; $\theta$—initial policy network parameters, $\theta^-$—target network parameters from copying $\theta$; $N_r$—replay buffer maximum size; $N_b$—training batch size; $N^-$—target network replacement frequency, $\phi$—self-rewarding function network $R_p$ parameter, $\alpha$—learning rate for policy network, $\beta$—learning rate for self-rewarding network $R_p$, $R_e$—Expert reward generation function;
2: Initialize the policy network and target network with random initial weights $\theta$ and $\theta^-$.
3: Initialize self-rewarding function network parameter with random initial weights $\phi$.
4: for episode $e = \{1, 2, \cdots, M\}$ do
5:     for $t$ = 1, T do
6:         Initialize sequence $s_1 = \{x_1\}$ and preprocessed state $\omega_1 = \omega(s_1)$.
7:         Select $a_t$ with $\epsilon$-greedy method; otherwise, select $a_t = \text{argmax}_a Q(\omega(s_t), a_t; \theta)$.
8:         Execute action $a_t$ in the environment and observe next state $s_{t+1}, \omega_{t+1} = \omega(s_{t+1})$.
9:         Compute predicted reward for all actions $r^s_t = R_p(s_t; \phi)$ by self-rewarding network.
10:        Compute expert reward for all actions $r^e_t = R_e(s_t)$ by expert reward generation function.
11:        Set:
               $$r^l_t = \begin{cases} r^s_t, & \text{if } r^s_t[a_t] \geq r^e_t[a_t]; \\ r^e_t, & \text{otherwise.} \end{cases}$$
12:        Reward received by agent at time $t$: $r_t = r^l_t[a_t]$.
13:        Store the transition $(s_t, a_t, r_t, s_{t+1}, r^l_t)$ in $\mathcal{B}$.
14:        If there is sufficient data in $\mathcal{B}$, sample a minibatch of $N_b$ tuples $(s_i, a_i, r_i, s_{i+1}, r^l_i) \sim \text{Unif}(\mathcal{B})$.
15:        Set:
               $$y_i = \begin{cases} r_i, & \text{if } s_{i+1} \text{ is terminal;} \\ r_i + \gamma Q(s_{i+1}, \text{argmax}_a Q(s_{i+1}, a; \theta); \theta^-), & \text{otherwise.} \end{cases}$$
16:        Perform gradient descent on the loss function of the policy network: $L_\theta = \mathbb{E}[|y_i - Q(s_i, a_i; \theta)|^2]$.
17:        Update the parameters of the policy network $\theta$ with $\theta \leftarrow \theta - \alpha * \frac{\partial L_\theta}{\partial \theta}$.
18:        Replace target network parameters $\theta^- \leftarrow \theta$ every $N^-$ steps.
19:        Perform gradient descent on the loss function of the self-rewarding network: $L_\phi = \mathbb{E}[|r^l_i - R_p(s_i; \phi)|^2]$.
20:        Update the parameters of the self-rewarding network $\phi$ with $\phi \leftarrow \phi - \beta * \frac{\partial L_\phi}{\partial \phi}$.
21:        Set $s_t \leftarrow s_{t+1}$.
22:    end for
23: end for
```

4. Experiments

4.1. Dataset

To demonstrate the robustness of the proposed model, six major stock indices from different regions are used for evaluation purposes, including the Hang Seng (HSI) in Hong Kong, CAC40 (FCHI) in France, KOSPI (KS11) in Korea, as well as the NASDAQ Composite (IXIC), S&P 500 (SP500), and Dow Jones Industrial Average (DJI) in the United States. By incorporating multiple indices from diverse geographical regions, we ensure that the model is exposed to a wide range of market conditions, thereby enhancing its ability to generalize and thereby reducing the risk of overfitting to any single market environment. Additionally, the data range covers an extended period, from 1 January 2007 to 31 December 2023, allowing for comprehensive training and testing of the proposed model. The dataset prior to 31 December 2020 is the training set, and the data from 1 January 2021 to 31 December 2023 is the test set, as shown in Figure 3. This temporal split ensures that the model is evaluated on out-of-sample data, providing a realistic assessment of its performance in unseen market conditions. Each index in the dataset has five key features: opening price, lowest price, highest price, closing price, and trading volume.



(**a**) DJI (**b**) SP500 (**c**) IXIC (**d**) HSI (**e**) FCHI (**f**) KS11

Figure 3. Close price curve of six stock indices.

4.2. Evaluation Metrics

To objectively and comprehensively assess the proposed method, this paper utilizes four key investment performance metrics. The first and foremost metric is the Cumulative Return (CR), which reflects the overall return on investments over a given time horizon. Secondly, the Annualized Return (AR) is employed to show the average level of return on investments over a given time horizon, presented as an annual percentage. Furthermore, the Sharpe Ratio (SR) is utilized to assess the risk-adjusted return of the investment, which measures the excess return per unit of volatility or total risk relative to the risk-free rate. Finally, Maximum Drawdown (MDD) is also considered, a metric that reveals the potential downside risk to which an investment or portfolio may be exposed, as reflected by a measure of the maximum decline in invested capital from a high to a low in a given period of time. These metrics combine multiple dimensions of return, risk, and volatility, and together provide a comprehensive assessment of investment performance.

4.3. Baseline Methods

This section provides a comparative analysis of the SRDDQN algorithm against established Deep Reinforcement Learning (DRL) models and traditional trading strategies. The DRL models considered include TDQN, DQN-Vanilla, and Fire (DQN-HER), while the traditional trading strategies are Buy and Hold (B&H), Sell and Hold (S&H), Mean Reversion with Moving Averages (MR), and Trend Following with Moving Averages (TF).

Specifically, the **B&H** strategy involves an investor purchasing an asset and holding it throughout the investment period, disregarding price fluctuations. Conversely, the **S&H** strategy entails an investor short-selling an asset at the beginning of the period and maintaining this position until the end. **MR** strategies operate on the assumption that prices will return to their historical averages; this study employs a 10-day Simple Moving Average (SMA) to detect such reversion opportunities. **TF** strategies, often used in algorithmic trading, utilize moving averages to identify trading signals. For this analysis, 10-day and 20-day SMAs are implemented to discern potential trading activities. The **TDQN**, proposed by Théate and Ernst [58], focuses on optimizing trading positions through a five-layer fully connected Deep Q-Network. **DQN-Vanilla**, developed by Taghian et al. [59], uses a simpler two-layer fully connected network to formulate trading decisions based on stock OHLC data. Additionally, **Fire (DQN-HER)**, presented by Cornalba et al. [29], explores multi-objective deep reinforcement learning for effective trading in both stock and cryptocurrency markets, incorporating dynamic reward functions and discount factors to enhance learning outcomes.

4.4. Experimental Setup

The proposed method is trained individually on six different stock indices datasets, and after the training phase, the performance of the model is evaluated using a test set associated with each dataset. This evaluation method measures the accuracy of the model in predicting various actions of different stock markets by training on multiple datasets. Table 1 lists the hyperparameters involved in the three time series feature extraction networks in this experiment. Table 2 details the hyperparameters for the three reinforcement learning methods. The initial capital is \$500,000, and the transaction cost is 0.3% of the total amount of each transaction. The hardware and software used are as follows: the processor is 13th Gen Intel(R) Core(TM) i7-13700KF, GPU is NVIDIA RTX 4090 with 24 GB memory. The software versions used are as follows: Python v3.10, Pytorch v1.12, and CUDA v12.1. The system is running on Windows 10.

Moreover, Figures 4 and 5 show the training results of the self-reward network and SRDDQN agent on the DJI and HSI datasets. Figure 4 shows the loss curves of the self-rewarding network trained with TimesNet and Min-Max reward labels. The loss steadily decreases as training progresses, indicating that the network is learning effectively. By the end of 100 episodes, the loss values on both DJI and HSI datasets converge to near-zero values, suggesting that the self-rewarding network has successfully minimized prediction errors and learned the reward patterns from the market data. Figure 5 illustrates the accumulated reward of the SRDDQN agents during the 100 training episodes. The accumulated reward represents the total reward obtained by the agent over a complete trajectory. As shown in the figure, the reward increases rapidly in the early episodes, demonstrating the agent's ability to improve its performance as training progresses. After approximately 20 episodes for both datasets, the accumulated reward reaches a stable plateau, with no significant drops or sudden spikes in the subsequent episodes. This stability suggests that the SRDDQN agent has successfully converged to an optimal strategy.

Table 1. The hyperparameters of various reward function networks.

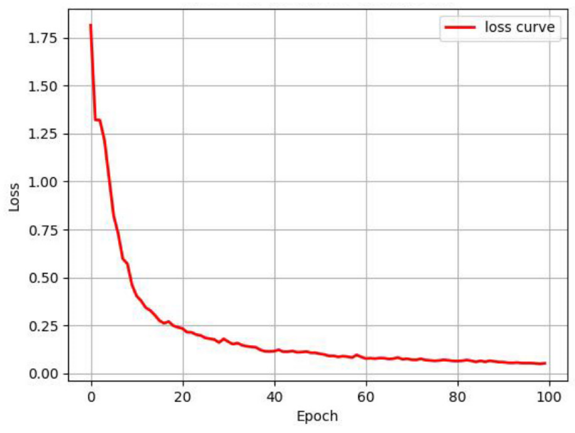
Hyperparameter	NLinear	WFTNet	TimesNet
Network layer	2MLP	13CNN+2MLP	13CNN+2MLP
Activation function	-	GELU	GELU
Batch size	32	32	32
Learning rate (α)	0.0001	0.0001	0.0001
Optimizer	Adam	Adam	Adam
Window size	20	20	20

The - of this two tables means the algorithm or the network does not have this parameter.

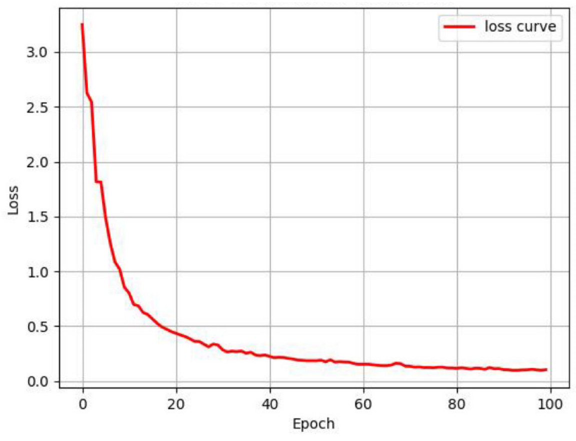
Table 2. The hyperparameters of various deep reinforcement learning methods.

Hyperparameter	A2C	PPO	SRDDQN
Agent network	TimesNet	TimesNet	TimesNet
Discount factor γ	0.9	0.9	0.9
Learning rate (α)	0.0001	0.0001	0.0001
Episode	100	100	100
Replay memory	1000	1000	1000
Batch size	32	32	32
Greedy	-	-	0.9
Clip	-	0.1	-

The - of this two tables means the algorithm or the network does not have this parameter.

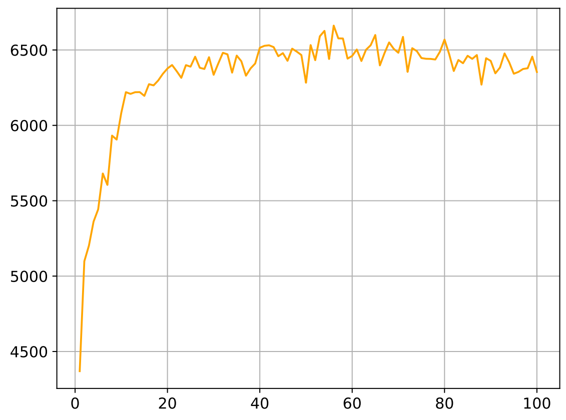


(**a**) DJI

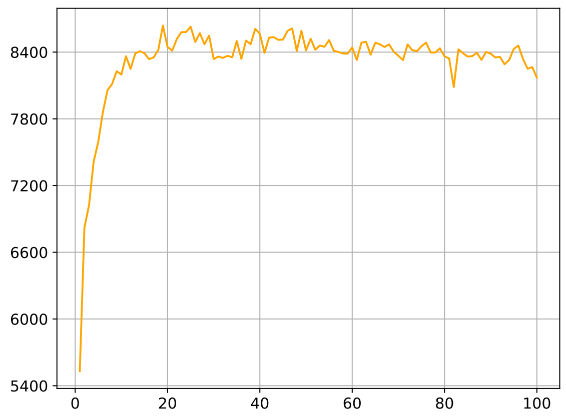


(**b**) HSI

Figure 4. The loss of self-rewarding networks based on TimesNet and Min-Max reward labels.



(**a**) DJI



(**b**) HSI

Figure 5. Accumulated reward of the SRDDQN agents on DJI and HSI.

4.5. Experimental Results

4.5.1. Comparison with Baselines

To assess the effectiveness of the proposed method, we conducted a comparative evaluation of the SRDDQN model against various baseline models across six different datasets. The experimental results demonstrate that the SRDDQN model significantly outperforms the other models on four major financial metrics: CR, AR, SR, and MDD, as shown in Table 3.

Table 3. Performance comparison of various methods on six datasets.

Datasets	Metrics	B&H	S&H	MR	TF	TDQN	DQN-Vanilla	Fire (DQN-HER)	SRDDQN
DJI	CR↑	4.12%	−4.12%	1.53%	−3.08%	−43.04%	58.42%	76.79%	**305.43%**
	AR↑	10.55%	−10.38%	4.84%	−7.92%	−22.51%	21.06%	19.11%	**57.83%**
	SR↑	0.54	−0.40	0.27	−0.44	−1.26	1.28	1.51	**3.94**
	MDD↓	20.23%	25.37%	17.95%	23.78%	45.57%	10.96%	6.92%	**5.03%**
IXIC	CR↑	1.93%	−1.93%	−5.15%	−4.18%	−17.35%	14.25%	51.87%	**1124.23%**
	AR↑	8.04%	−2.11%	−12.58%	−9.55%	−4.91%	9.85%	14.50%	**92.62%**
	SR↑	0.25	−0.07	−0.37	−0.31	−0.25	0.31	0.89	**4.43**
	MDD↓	34.50%	27.41%	35.85%	30.28%	28.38%	38.34%	10.74%	**5.55%**
SP500	CR↑	0.54%	−0.54%	1.46%	−1.57%	−37.85%	26.22%	90.21%	**455.74%**
	AR↑	4.89%	1.90%	7.91%	−4.06%	−25.03%	18.37%	21.24%	**68.40%**
	SR↑	0.17	0.05	0.30	−0.15	−1.09	0.76	1.37	**4.11**
	MDD↓	25.05%	26.70%	16.06%	24.28%	47.15%	20.80%	11.69%	**3.89%**
HSI	CR↑	−7.07%	7.07%	−3.47%	−2.01%	37.11%	86.61%	46.92%	**1302.76%**
	AR↑	−22.11%	16.28%	−6.32%	−2.51%	11.05%	29.25%	13.46%	**100.05%**
	SR↑	−0.65	0.84	−0.18	−0.08	0.67	1.19	0.70	**4.51**
	MDD↓	45.38%	17.05%	43.65%	32.69%	33.49%	17.62%	18.72%	**5.41%**
FCHI	CR↑	7.83%	−7.83%	0.34%	3.91%	40.86%	127.84%	34.03%	**371.92%**
	AR↑	18.28%	−23.17%	2.94%	10.89%	11.57%	32.00%	10.48%	**65.37%**
	SR↑	0.77	−0.51	0.12	0.48	0.75	1.95	0.69	**3.99**
	MDD↓	21.95%	39.57%	23.41%	16.08%	21.11%	7.01%	12.50%	**1.98%**
KS11	CR↑	−2.69%	2.69%	-0.36%	4.70%	−5.00%	−9.04%	18.71%	**444.20%**
	AR↑	−5.86%	7.70%	0.86%	12.20%	−0.46%	−2.69%	6.33%	**69.60%**
	SR↑	−0.24	0.40	0.04	0.57	−0.03	−0.11	0.35	**4.46**
	MDD↓↓	34.23%	13.29%	29.59%	15.47%	24.98%	33.65%	21.60%	**4.68%**

↑: A higher value of this metric is preferred. ↓: A lower value of this metric is preferred. Bold: The best experimental result.

SRDDQN not only surpassed traditional models like B&H and S&H but also outperformed more sophisticated models such as MR, TF, and other deep reinforcement learning approaches like TDQN, DQN-Vanilla, and Fire (DQN-HER). Remarkably, SRDDQN achieved a CR of 305.43% on DJI, vastly surpassing Fire's 76.79%, the next best model. This trend of dominance is consistent across all datasets, with IXIC, SP500, and HSI showing exponential improvements in both CR and AR. SRDDQN displayed extraordinary resilience and profitability, with its lowest AR being 57.83% on DJI and its highest reaching an astounding 100.05% on HSI. This is in sharp contrast to the baseline models, where none exceeded a 40% AR. Such a wide margin underscores SRDDQN's robustness in diverse market conditions.

Additionally, SRDDQN's Sharpe Ratio (SR) is significantly higher across all datasets, indicating a higher return per unit of risk. On the DJI dataset, the model's SR peaks at 3.94, which is notably higher than the highest baseline Fire's 1.51. In terms of risk management, as measured by Maximum Drawdown (MDD), SRDDQN exhibits an exceptional ability to minimize losses, with MDD on the DJI dataset at only 5.03%, while TDQN's MDD is as high as 45.57%. The lower MDD suggests that SRDDQN not only enhances returns but also reduces potential losses, emphasizing its efficacy in mitigating risk during market declines.

Figure 6 depicts the cumulative return curves of SRDDQN across six stock indices. The vertical axis represents cumulative returns, while the horizontal axis represents the days over which the strategy was run. Among the indices, HSI shows the highest cumulative return, reaching above 1302.76% by the end of the period. IXIC also exhibits strong performance, though not as high as HSI. These findings further validate SRDDQN's superiority in managing the complexities of stock market dynamics and offer valuable insights for developing trading strategies.

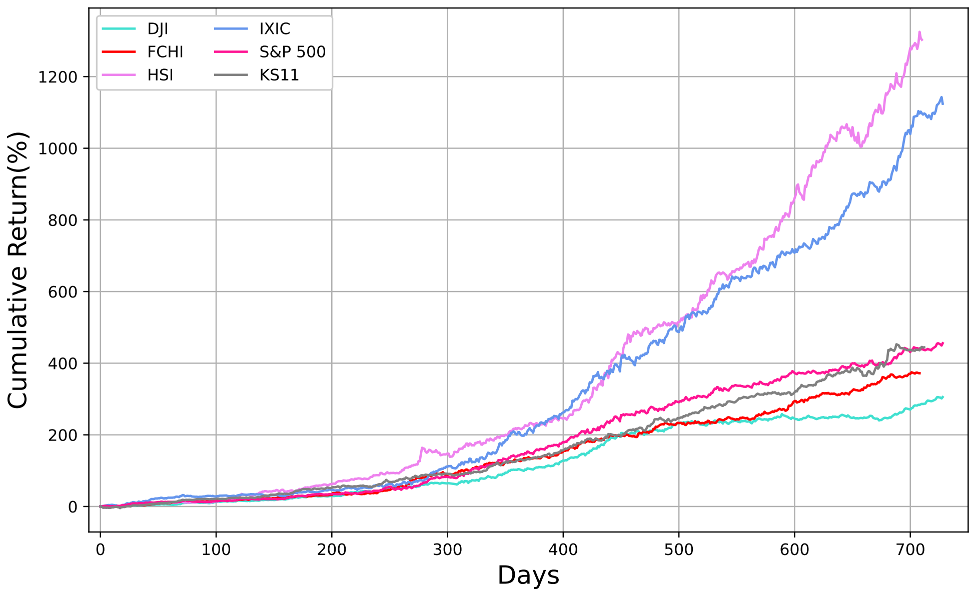


Figure 6. Cumulative return curves of SRDDQN across six stock indices.

4.5.2. Analysis of the Self-Rewarding Learning Paradigm

This section analyzes the impact of employing DDQN with shared buffers and synchronous training strategies on model performance using the DJI dataset. Various configurations of buffer sharing and synchronized training steps were tested, and are summarized in Table 4. The results indicate a clear advantage when using a shared buffer with synchronized training steps. Specifically, the configuration where both the buffer is shared and the training steps are synchronized (N = 1, M = 1) outperformed all other configurations, achieving a CR of 305.43%, AR of 57.83%, and SR of 3.94. This shows that synchronous learning with a shared experience buffer significantly enhances trading performance.

Table 4. Performance comparison of various training methods with DDQN on the DJI dataset.

Buffer	Training	CR↑	AR↑	SR↑	MDD↓
Independent	synchronized	297.62%	57.19%	3.85	6.92%
Shared	N = 1, M = 1	**305.43%**	**57.83%**	**3.94**	5.03%
	N = 100, M = 5	284.1%	56.01%	3.77	3.97%
	N = 100, M = 10	267.11%	54.42%	3.76	4.92%
	N = 100, M = 15	237.55%	51.57%	3.40	6.62%
	N = 100, M = 20	262.78%	54.09%	3.56	4.92%
	N = 50, M = 15	278.55%	55.52%	3.72	5.43%
	N = 150, M = 15	265.98%	54.34%	3.68	**3.86%**
	N = 200, M = 15	258.45%	53.63%	3.61	4.21%

↑: A higher value of this metric is preferred. ↓: A lower value of this metric is preferred. Bold: The best experimental result.

In contrast, increasing the intervals between learning reward functions (e.g., N = 100, M = 15 or N = 200, M = 15) generally led to reduced performance metrics across all indicators.

For example, increasing M values in configurations with N = 100 resulted in performance degradation, showcasing diminishing returns with less frequent reward updates.

Moreover, the experimental data demonstrate that while increasing the number of initial synchronized steps (e.g., N = 100) without frequent reward updates (e.g., higher M values) leads to moderate performance decline, excessively high values (e.g., N = 200, M = 15) correlate with further reductions in CR and AR, alongside slight improvements in MDD. This pattern highlights the trade-off between learning efficiency and risk management in algorithmic trading environments.

4.5.3. Analysis of Self-Rewarding with Different Networks

The proposed method integrates the self-rewarding learning mechanism into the double DQN approach. This section evaluates the effectiveness of the self-rewarding mechanism in combination with three different networks—TimesNet, WFNet, and NLinear—known for their robust performance in financial time series forecasting. The results, detailed in Table 5, highlight the comparative performance of each network when employing the self-rewarding mechanism versus a pre-trained reward mechanism on the DJI dataset.

Table 5. Performance of self-rewarding networks and pre-trained reward networks with DDQN on the DJI dataset.

Network	Reward	CR↑	AR↑	SR↑	MDD↓
NLinear	self-rewarding	298.58%	57.26%	3.87	6.92%
	pre-trained	275.82%	55.27%	3.72	3.83%
WFTNet	self-rewarding	264.58%	54.21%	3.67	**3.65%**
	pre-trained	240.22%	51.84%	3.43	5.99%
TimesNet	self-rewarding	**305.43%**	**57.83%**	**3.94**	5.03%
	pre-trained	209.35%	48.45%	3.25	4.21%

↑: A higher value of this metric is preferred. ↓: A lower value of this metric is preferred. Bold: The best experimental result.

The results show that the self-rewarding mechanism consistently outperforms the pre-trained reward model across all metrics. Specifically, TimesNet achieved the highest Cumulative Return (CR) of 305.43% and the highest Annualized Return (AR) of 57.83% under the self-rewarding mechanism. This performance is notable compared to its pre-trained counterpart, which yielded a CR of 209.35% and an AR of 48.45%. TimesNet also exhibited a significant improvement in SR, reaching 3.94 under the self-rewarding paradigm, compared to 3.25 with the pre-trained reward. This indicates superior risk-adjusted returns.

WFNet also demonstrated substantial gains under the self-rewarding configuration, achieving an impressive CR of 264.58% and SR of 3.67, along with the lowest MDD of 3.65%, emphasizing its resilience during adverse market conditions.

4.5.4. Analysis of Self-Rewarding with Different Reward Labels

The effectiveness of the self-rewarding mechanism in the SRDDQN method is evaluated across three different reward labels: formula-based reward, and self-rewarding using the IXIC dataset. In Table 6, the results show that the self-rewarding mechanism significantly outperforms the formula-based approach. Specifically, under the self-rewarding mechanism, the CR increased from 816.32% to 1124.23% in the best case (Min-Max), reflecting a gain of approximately 300 percentage points. Similarly, AR exhibited marked improvements.

Furthermore, the SR, a measure of risk-adjusted return, further supports the superiority of the self-rewarding method. The self-rewarding mechanism achieved a Sharpe ratio of 4.43, compared to 3.93 under the formula-based method in the Min-Max label, indicating more efficient performance relative to the risk undertaken.

Table 6. Performance comparison of self-rewarding for different reward labels with DDQN on the IXIC dataset.

Reward Type	Metrics	Sharpe Ratio	Return	Min-Max
Formula	CR↑	921.68%	421.83%	816.32%
	AR↑	87.58%	67.23%	84.36%
	SR↑	4.03	2.96	3.93
	MDD↓	5.70%	14.02%	5.42%
Self-rewarding	CR↑	**1035.83%**	**646.25%**	**1124.23%**
	AR↑	**90.56%**	**78.34%**	**92.62%**
	SR↑	**4.25**	**3.53**	**4.43**
	MDD↓	5.70%	**6.60%**	**5.55%**

↑: A higher value of this metric is preferred. ↓: A lower value of this metric is preferred. Bold: The best experimental result.

4.5.5. Analysis of Self-Rewarding in Different RL Methods

This section provides insights into the effectiveness of the self-rewarding mechanism under various RL frameworks, particularly for the A2C, PPO, and DDQN algorithms. A comprehensive comparison of the performance of these methods under two different reward types—a formula-based reward mechanism and a self-rewarding mechanism—is presented in Table 7, with both being tested on the DJI dataset. The self-rewarding mechanism significantly improves the cumulative reward (CR) for all tested methods. Notably, DDQN's CR increased from 295.16% to 305.43%, A2C's CR improved from 177.23% to 242.90%, and PPO's CR rose from 251.13% to 256.47%.

Table 7. Comparison of self-rewarding for A2C, PPO, and DDQN on the DJI Dataset.

Reward Type	Metrics	A2C	PPO	DDQN
Formula-Based	CR↑	177.23%	251.13%	295.16%
	AR↑	44.55%	52.95%	57.00%
	SRR↑	2.84	3.49	3.80
	MDD↓	**4.89%**	**3.93%**	6.32%
Self-Rewarding	CR↑	**242.90%**	**256.47%**	**305.43%**
	AR↑	**52.19%**	**53.44%**	**57.83%**
	SR↑	**3.30**	**3.60**	**3.94**
	MDD↓	5.22%	4.19%	**5.03%**

↑: A higher value of this metric is preferred. ↓: A lower value of this metric is preferred. Bold: The best experimental result.

Similarly, the annual return (AR) is positively affected by the self-rewarding strategy. DDQN's AR increased from 57.00% to 57.83%. The increase in the SR (Sharpe Ratio) suggests a more favorable risk-adjusted return profile when employing the self-rewarding feature. DDQN's SR rose from 3.80 to 3.94, indicating the highest risk-adjusted performance among all the methods. While the maximum drawdown (MDD) for A2C and PPO increased slightly when using the self-rewarding mechanism, DDQN's MDD decreased, suggesting that the self-rewarding mechanism has the potential to reduce significant losses and improve the risk management of these methods.

4.5.6. Analysis of Trading Strategy

After analyzing the effectiveness of SRDDQN across different experiments, further observations were made on the agent's actions in all datasets. Figures 7–11 show the actions performed by the SRDDQN agent in six datasets. The red box highlights the zoomed-in section of the figure. These actions are labeled on the stock price trend charts to illustrate how the agent made trading decisions based on price movements.

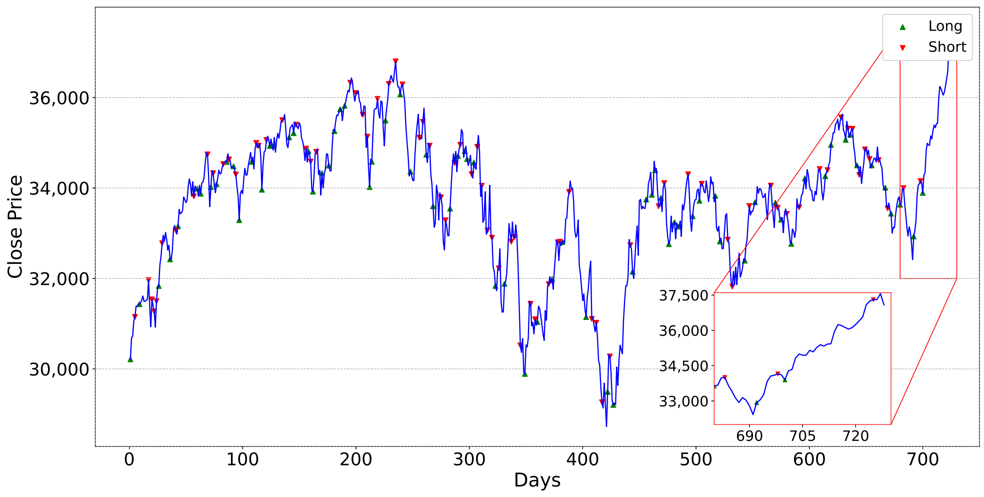


Figure 7. Actions of SRDDQN Agent in DJI.

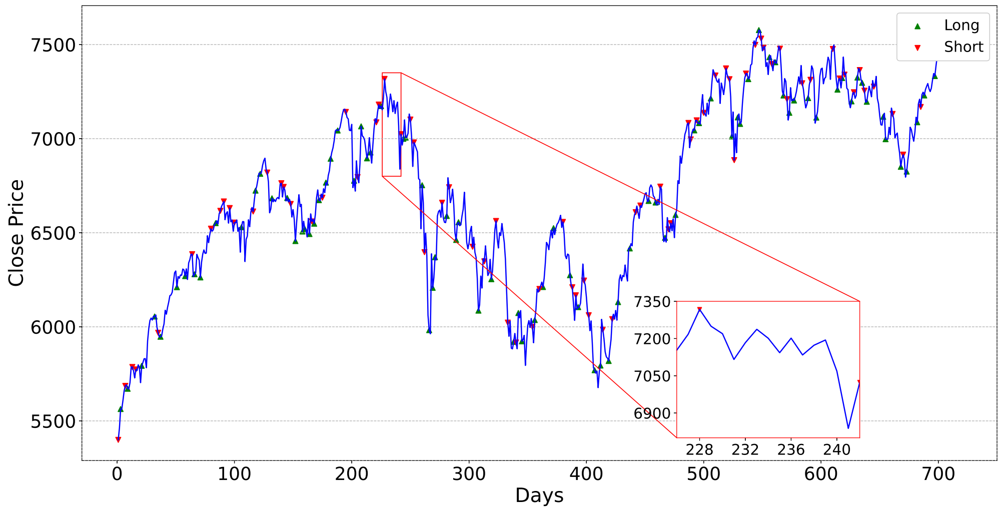


Figure 8. Actions of SRDDQN Agent in FCHI.

First, the trading actions undertaken by the SRDDQN agent within the DJI and FCHI datasets are illustrated in Figures 7 and 8. Upon closer examination, it becomes evident that the SRDDQN agent exhibits astute trading decisions. For instance, in Figure 7, around the 690th day, the agent identified an upward trend in stock price and executed a "long" trade. By the 700th day, as the stock price approached a peak, the agent swiftly switched to a "short" position. A similar pattern is observed in Figure 8, where the agent made a "short" decision at the peak around the 228th day, anticipating a subsequent downward trend.

This pattern of decisive actions is also evident in Figure 10, where the agent adeptly navigated significant market fluctuations, consistently making profitable trades.

Second, the trading behavior of the SRDDQN agent in the HSI and KS11 datasets is visualized in Figures 9 and 11. In Figure 9, around the 24th day, the agent strategically executed a "long" trade anticipating an upward movement. Shortly thereafter, it transitioned to a "short" position, followed by another "short" trade due to a continued downward trend. This demonstrates the agent's high-frequency trading approach during periods of frequent oscillations, aiming to optimize returns. A similar pattern is seen in Figure 11, where the agent demonstrated precise timing for each trade during high-frequency oscillations in stock prices.

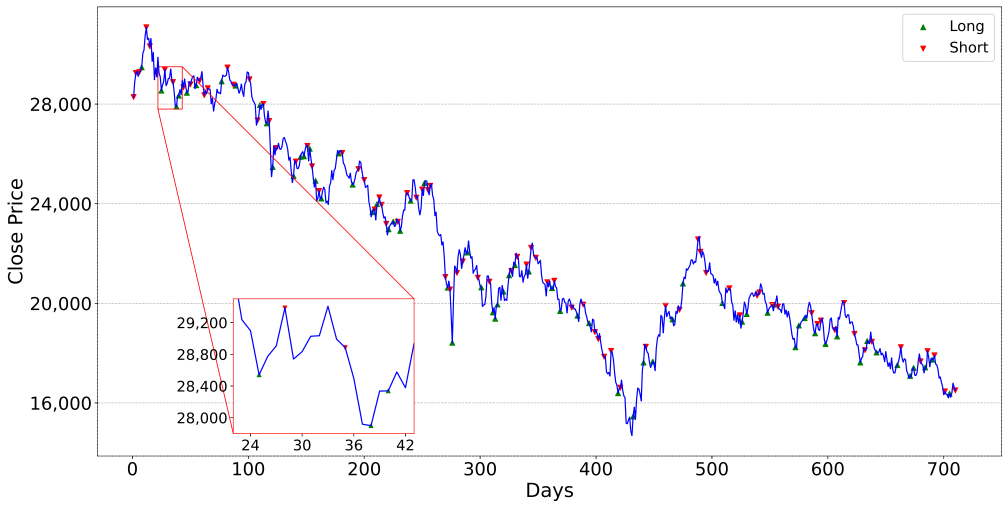


Figure 9. Actions of SRDDQN Agent in HSI.

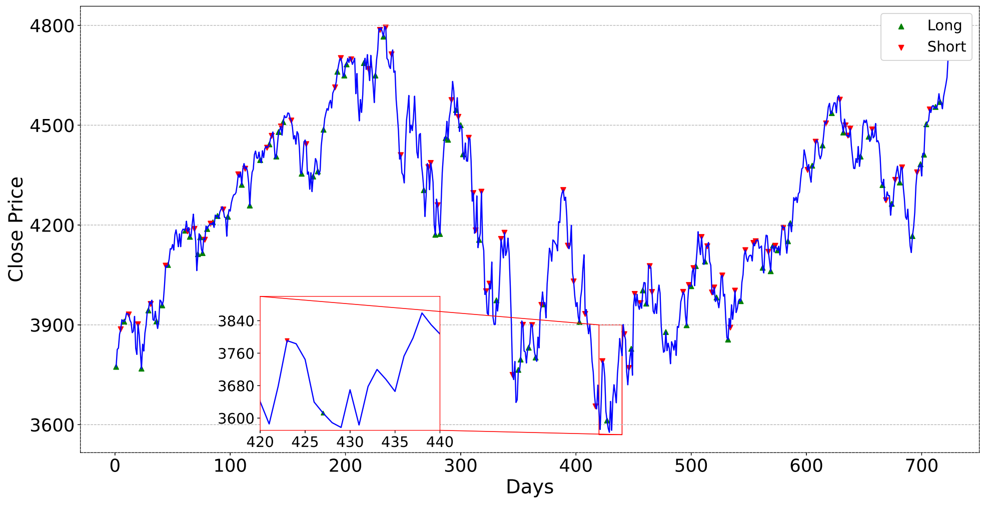


Figure 10. Actions of SRDDQN agent in SP500.

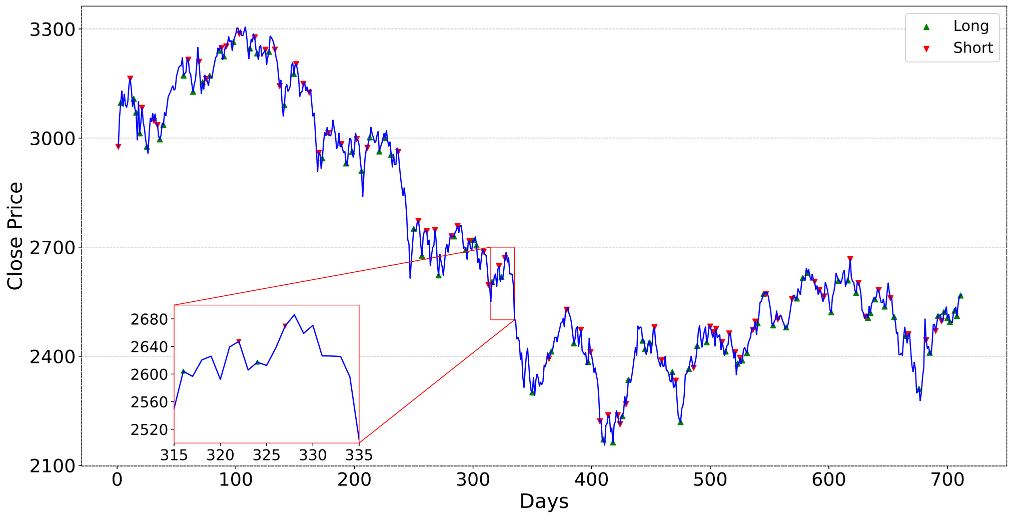


Figure 11. Actions of SRDDQN agent in KS11.

4.5.7. Discussion

This comprehensive analysis of SRDDQN demonstrates its consistent outperformance of baseline strategies across various datasets and scenarios, as measured by CR, AR, SR, and MDD. The superior performance of SRDDQN, as detailed in Table 3, can be attributed to its novel self-rewarding mechanism, which dynamically adjusts the reward structure based on real-time market feedback. This approach not only capitalizes on prevailing market trends but also effectively mitigates potential losses, as evidenced by significantly lower MDD values.

Section 4.5.2 provides an analysis of DDQN configurations using shared buffers and synchronous training strategies, highlighting the benefits of synchronized steps in training regimes. The configurations tested vary in the extent of buffer sharing and synchronization of training steps, demonstrating clear advantages of employing a shared buffer system in enhancing consistency and reliability in volatile financial markets.

Further analysis in Section 4.5.3 reveals that the integration of self-rewarding mechanisms not only boosts key financial metrics but also significantly reduces risk exposure. This comparative study vividly illustrates the advantages of employing self-rewarding systems in optimizing the training of neural networks for complex financial time series applications, suggesting that these techniques could be pivotal in advancing the field of algorithmic trading.

The experiments discussed in Section 4.5.4, using the IXIC dataset, consistently show improvements across all financial metrics, reinforcing the effectiveness of the self-rewarding mechanism. This dynamic adaptability to changing market conditions marks a significant improvement over traditional formula-based reward functions, underscoring the potential of self-rewarding RL in algorithmic trading.

While the results are promising, reliance on historical data and simulations presents limitations in predicting future market behavior. Future research should explore the application of SRDDQN in live trading environments to validate its effectiveness under real-world conditions. Additionally, the integration of macroeconomic indicators into the reward system could further enhance the model by accounting for broader economic impacts on market dynamics.

This work opens several avenues for future exploration, including the potential for real-time adaptation of the self-rewarding mechanism to sudden market shifts and the exploration of hybrid models that combine traditional financial analysis with RL techniques. The continued refinement and testing of these systems will be crucial in advancing the practical applications of machine learning in financial markets.

5. Conclusions

This paper introduces the innovative Self-Rewarding Deep Reinforcement Learning (SRDRL) mechanism, which significantly enhances reinforcement learning applications through the integration of a self-rewarding function. By addressing a key limitation in reward function design, the self-rewarding mechanism offers a scalable and adaptable framework, proving highly beneficial in complex and dynamic trading environments. Focused on the financial domain, this paper employs a novel Self-Rewarding Double DQN (SRDDQN) to optimize reward structures in algorithmic trading strategies based on expert-defined metrics such as Min-Max, Sharpe Ratio, and Returns. Advanced time series feature extraction networks, including TimesNet, WFTNet, and NLinear, are utilized to refine the self-rewarding function, enabling the dynamic adjustment of reward strategies in response to market conditions. The efficacy of this method is validated through extensive testing across various datasets, including DJI, IXIC, SP500, HSI, FCHI, and KS11.

The key contributions of this paper include the following:

1. Introducing a self-rewarding mechanism within a DDQN framework; the proposed SRDRL system synchronizes self-rewarding learning with the reinforcement learning process, allowing for real-time adjustments to the reward structure based on direct market feedback. This synchronized iterative learning architecture not only adapts

swiftly to market trends but also mitigates potential losses, significantly aligning learning outcomes with optimal trading strategies.

2. By employing expert labels such as Min-Max, Sharpe Ratio, and Returns to refine the self-rewarding functions, and leveraging advanced time series extraction networks, the proposed method exhibits superior scalability. This application of DDQN further optimizes the reward mechanism, enriching the model's capacity to react to financial market fluctuations.
3. Demonstrating the effectiveness of the self-rewarding function under various RL frameworks, particularly for two different reward types (formula-based reward mechanism and self-rewarding mechanism), tested with the A2C, PPO, and DDQN algorithms on the DJI dataset. The self-rewarding mechanism significantly improves all metrics for all tested methods.
4. The integration of DDQN with shared experience buffers and synchronous training protocols markedly improves trading performance. Our experiments demonstrate that this approach substantially boosts the algorithm's efficiency and effectiveness over traditional methods. Remarkably, SRDDQN achieved a cumulative return (CR) of 436.01% on the DJI dataset, vastly surpassing DQN-Vanilla's 58.42%, the next-best model. This trend of dominance is consistent across all datasets, with IXIC, SP500, and HSI showcasing exponential improvements in both CR and annualized return (AR).

Despite its strengths, the framework encounters certain limitations that need addressing. Future research will focus on refining the integration of self-rewarding mechanisms in more complex trading environments, enhancing the adaptability of the model under diverse market conditions, and further exploring the potential of multi-agent learning systems within the SRDRL framework. Additionally, large pre-trained language models (e.g., GPT, FinBERT, etc.) are capable of accurately interpreting market trends due to their strong capabilities in sentiment analysis, market prediction, and natural language processing [60–62]. Therefore, we will include LLM models in our future work to carry out the research of sentiment analysis in financial markets.

Author Contributions: Conceptualization, methodology, data curation, writing—original draft preparation. Y.H.; software, validation, visualization, investigation, C.Z. and L.Z.; supervision, writing—review & editing, project administration, X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Faculty Research Grant, Macau University of Science and Technology (Project No. FRG-22-001-INT), Experimental Teaching Demonstration Center for Intelligent Financial Talents, Beijing Institute of Technology, Zhuhai (2023006ZLGC), and the Research Base for Intelligent Financial Governance and National Economic Security, Beijing Institute of Technology, Zhuhai (2023WZJD009).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Our stock data is available for download at http://finance.yahoo.com (accessed on 10 November 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
DQN	Deep Q-Network
DDQN	Double Deep Q-Network
DLMA	Direct Large Model Alignment
SSRL	Self-Supervised Reinforcement Learning
SRDRL	Self-Rewarding Deep Reinforcement Learning

SRDDQN	Self-Rewarding Double Deep Q-Network
PPO	Proximal Policy Optimization
A2C	Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-critic
CR	Cumulative Return
AR	Annualized Return
SR	Sharpe Ratio
MDD	Maximum Drawdown
PnL	Profit & Loss
DPO	Direct Preference Optimization
LSTM	Long-Short Term Memory
ETDQN	Extended Trading Deep Q-Network
B&H	Buy and Hold
S&H	Sell and Hold
MR	Mean Reversion with Moving Averages
TF	Trend Following with Moving Averages

References

1. Chakole, J.; Kurhekar, M. Trend following deep Q-Learning strategy for stock trading. *Expert Syst.* **2020**, *37*, e12514. [CrossRef]
2. Corazza, M.; Sangalli, A. Q-Learning and SARSA: A Comparison between Two Intelligent Stochastic Control Approaches for Financial Trading. University Ca' Foscari of Venice, Dept. of Economics Research Paper Series No. No. 15/WP/2015. Available online: https://ssrn.com/abstract=2617630 (accessed on 11 November 2024).
3. Cornalba, F.; Disselkamp, C.; Scassola, D.; Helf, C. Multi-Objective reward generalization: Improving performance of Deep Reinforcement Learning for selected applications in stock and cryptocurrency trading. *arXiv* **2022**, arXiv:2203.04579.
4. Huang, C.Y. Financial trading as a game: A deep reinforcement learning approach. *arXiv* **2018**, arXiv:1807.02787.
5. Jeong, G.H.; Kim, H.Y. Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Syst. Appl.* **2019**, *117*, 125–138. [CrossRef]
6. Jiang, W.; Liu, M.; Xu, M.; Chen, S.; Shi, K.; Liu, P.; Zhang, C.; Zhao, F. New reinforcement learning based on representation transfer for portfolio management. *Knowl.-Based Syst.* **2024**, *293*, 111697. [CrossRef]
7. Ma, C.; Zhang, J.; Liu, J.; Ji, L.; Gao, F. A Parallel Multi-module Deep Reinforcement Learning Algorithm for Stock Trading. *Neurocomputing* **2021**, 449, 290–302. [CrossRef]
8. Oyewola, D.O.; Akinwunmi, S.A.; Omotehinwa, T.O. Deep LSTM and LSTM-Attention Q-learning based reinforcement learning in oil and gas sector prediction. *Knowl.-Based Syst.* **2024**, *284*, 111290. [CrossRef]
9. Si, W.; Li, J.; Ding, P.; Rao, R. A multi-objective deep reinforcement learning approach for stock index future's intraday trading. In Proceedings of the 2017 10th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 9–10 December 2017; Volume 2, pp. 431–436.
10. Tran, M.; Pham-Hi, D.; Bui, M. Optimizing Automated Trading Systems with Deep Reinforcement Learning. *Algorithms* **2023**, *16*, 23. [CrossRef]
11. Grzes, M.; Kudenko, D. Plan-based reward shaping for reinforcement learning. In Proceedings of the 2008 4th International IEEE Conference Intelligent Systems, Varna, Bulgaria, 6–8 September 2008; IEEE: Piscataway, NJ, USA, 2008; Volume 2, pp. 10–22.
12. Proper, S.; Tumer, K. Modeling difference rewards for multiagent learning. In Proceedings of the AAMAS, Valencia, Spain, 4–8 June 2012; pp. 1397–1398.
13. Li, Y.; Zheng, W.; Zheng, Z. Deep robust reinforcement learning for practical algorithmic trading. *IEEE Access* **2019**, *7*, 108014–108022. [CrossRef]
14. Huang, B.; Jin, Y. Social learning in self-organizing systems for complex assembly tasks. *Adv. Eng. Inform.* **2023**, *57*, 102109. [CrossRef]
15. Zhang, W.; Lin, Y.; Liu, Y.; You, H.; Wu, P.; Lin, F.; Zhou, X. Self-Supervised Reinforcement Learning with dual-reward for knowledge-aware recommendation. *Appl. Soft Comput.* **2022**, *131*, 109745. [CrossRef]
16. Casper, S.; Davies, X.; Shi, C.; Gilbert, T.K.; Scheurer, J.; Rando, J.; Freedman, R.; Korbak, T.; Lindner, D.; Freire, P.; et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv* **2023**, arXiv:2307.15217.
17. Kaufmann, T.; Weng, P.; Bengs, V.; Hüllermeier, E. A survey of reinforcement learning from human feedback. *arXiv* **2023**, arXiv:2312.14925.
18. Yuan, W.; Pang, R.Y.; Cho, K.; Sukhbaatar, S.; Xu, J.; Weston, J. Self-rewarding language models. *arXiv* **2024**, arXiv:2401.10020.
19. Liu, A.; Bai, H.; Lu, Z.; Kong, X.; Wang, S.; Shan, J.; Cao, M.; Wen, L. Direct Large Language Model Alignment Through Self-Rewarding Contrastive Prompt Distillation. *arXiv* **2024**, arXiv:2402.11907.
20. Chen, L.; Gao, Q. Application of Deep Reinforcement Learning on Automated Stock Trading. In Proceedings of the 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 18–20 October 2019; pp. 29–33.
21. Corazza, M.; Fasano, G.; Gusso, R.; Pesenti, R. A comparison among Reinforcement Learning algorithms in financial trading systems. *Univ. Ca'Foscari Venice Dept. Econ. Res. Pap. Ser. No* **2019**, *33*. [CrossRef]

22. Dang, Q.V. Reinforcement learning in stock trading. In *Advanced Computational Methods for Knowledge Engineering: Proceedings of the 6th International Conference on Computer Science, Applied Mathematics and Applications, ICCSAMA 2019, Hanoi, Vietnam, 19–20 December 2019*; Springer International Publishing: Cham, Switzerland, 2020; pp. 311–322.
23. Gao, X. Deep reinforcement learning for time series: Playing idealized trading games. *arXiv* **2018**, arXiv:1803.03916.
24. Li, Y.; Liu, P.; Wang, Z. Stock Trading Strategies Based on Deep Reinforcement Learning. *Sci. Program.* **2022**, *2022*, 698656. [CrossRef]
25. Liu, P.; Zhang, Y.; Bao, F.; Yao, X.; Zhang, C. Multi-type data fusion framework based on deep reinforcement learning for algorithmic trading. *Appl. Intell.* **2023**, *53*, 1683–1706. [CrossRef]
26. Xiao, X. Quantitative Investment Decision Model Based on PPO Algorithm. *Highlights Sci. Eng. Technol.* **2023**, *34*, 16–24. [CrossRef]
27. Rodinos, G.; Nousi, P.; Passalis, N.; Tefas, A. A Sharpe Ratio based reward scheme in Deep Reinforcement Learning for financial trading. In *Artificial Intelligence Applications and Innovations. AIAI 2023. IFIP Advances in Information and Communication Technology*; Springer: Cham, Switzerland, 2023; pp. 15–23.
28. Chakraborty, S. Capturing Financial markets to apply Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1907.04373.
29. Cornalba, F.; Disselkamp, C.; Scassola, D.; Helf, C. Multi-objective reward generalization: Improving performance of Deep Reinforcement Learning for applications in single-asset trading. *Neural Comput. Appl.* **2024**, *36*, 619–637. [CrossRef]
30. Nair, A.; Zhu, B.; Narayanan, G.; Solowjow, E.; Levine, S. Learning on the job: Self-rewarding offline-to-online finetuning for industrial insertion of novel connectors from vision. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 7154–7161.
31. Koratamaddi, P.; Wadhwani, K.; Gupta, M.; Sanjeevi, S.G. Market sentiment-aware deep reinforcement learning approach for stock portfolio allocation. *Eng. Sci. Technol. Int. J.* **2021**, *24*, 848–859. [CrossRef]
32. Yang, B.; Liang, T.; Xiong, J.; Zhong, C. Deep reinforcement learning based on transformer and U-Net framework for stock trading. *Knowl.-Based Syst.* **2023**, *262*, 110211. [CrossRef]
33. Wang, Y.; Yan, G. Survey on the application of deep learning in algorithmic trading. *Data Sci. Financ. Econ.* **2021**, *1*, 345–361. [CrossRef]
34. Hendershott, T.; Jones, C.M.; Menkveld, A.J. Does algorithmic trading improve liquidity? *J. Financ.* **2011**, *66*, 1–33. [CrossRef]
35. Nuti, G.; Mirghaemi, M.; Treleaven, P.; Yingsaeree, C. Algorithmic trading. *Computer* **2011**, *44*, 61–69. [CrossRef]
36. Treleaven, P.; Galas, M.; Lalchand, V. Algorithmic trading review. *Commun. ACM* **2013**, *56*, 76–85. [CrossRef]
37. Amirzadeh, R.; Nazari, A.; Thiruvady, D. Applying artificial intelligence in cryptocurrency markets: A survey. *Algorithms* **2022**, *15*, 428. [CrossRef]
38. Nikolova, V.; Trinidad Segovia, J.E.; Fernández-Martínez, M.; Sánchez-Granero, M.A. A novel methodology to calculate the probability of volatility clusters in financial series: An application to cryptocurrency markets. *Mathematics* **2020**, *8*, 1216. [CrossRef]
39. Guzmán, A.; Pinto-Gutiérrez, C.; Trujillo, M.A. Trading cryptocurrencies as a pandemic pastime: COVID-19 lockdowns and bitcoin volume. *Mathematics* **2021**, *9*, 1771. [CrossRef]
40. Huang, Y.; Lu, X.; Zhou, C.; Song, Y. DADE-DQN: Dual Action and Dual Environment Deep Q-Network for Enhancing Stock Trading Strategy. *Mathematics* **2023**, *11*, 3626. [CrossRef]
41. Rosati, R.; Romeo, L.; Goday, C.A.; Menga, T.; Frontoni, E. Machine learning in capital markets: Decision support system for outcome analysis. *IEEE Access* **2020**, *8*, 109080–109091. [CrossRef]
42. Teng, T.; Ma, L. Deep learning-based risk management of financial market in smart grid. *Comput. Electr. Eng.* **2022**, *99*, 107844. [CrossRef]
43. Huang, Y.; Song, Y. A new hybrid method of recurrent reinforcement learning and BiLSTM for algorithmic trading. *J. Intell. Fuzzy Syst.* **2023**, *45*, 1939–1951. [CrossRef]
44. Huang, Y.; Wan, X.; Zhang, L.; Lu, X. A novel deep reinforcement learning framework with BiLSTM-Attention networks for algorithmic trading. *Expert Syst. Appl.* **2024**, *240*, 122581. [CrossRef]
45. Kong, M.; So, J. Empirical analysis of automated stock trading using deep reinforcement learning. *Appl. Sci.* **2023**, *13*, 633. [CrossRef]
46. Kochliaridis, V.; Kouloumpris, E.; Vlahavas, I. Combining deep reinforcement learning with technical analysis and trend monitoring on cryptocurrency markets. *Neural Comput. Appl.* **2023**, *35*, 21445–21462. [CrossRef]
47. Zou, J.; Lou, J.; Wang, B.; Liu, S. A novel deep reinforcement learning based automated stock trading system using cascaded lstm networks. *Expert Syst. Appl.* **2024**, *242*, 122801. [CrossRef]
48. Avramelou, L.; Nousi, P.; Passalis, N.; Tefas, A. Deep reinforcement learning for financial trading using multi-modal features. *Expert Syst. Appl.* **2024**, *238*, 121849. [CrossRef]
49. Kwon, Y.; Lee, Z. A hybrid decision support system for adaptive trading strategies: Combining a rule-based expert system with a deep reinforcement learning strategy. *Decis. Support Syst.* **2024**, *177*, 114100. [CrossRef]
50. Park, J.H.; Kim, J.H.; Huh, J.H. Deep Reinforcement Learning Robots for Algorithmic Trading: Considering Stock Market Conditions and US Interest Rates. *IEEE Access* **2024**, *12*, 20705–20725. [CrossRef]
51. de Azevedo Takara, L.; Santos, A.A.P.; Mariani, V.C.; dos Santos Coelho, L. Deep reinforcement learning applied to a sparse-reward trading environment with intraday data. *Expert Syst. Appl.* **2024**, *238*, 121897. [CrossRef]

52. Huang, Y.; Zhou, C.; Cui, K.; Lu, X. Improving Algorithmic Trading Consistency via Human Alignment and Imitation Learning. *Expert Syst. Appl.* **2024**, *253*, 124350. [CrossRef]
53. Carta, S. A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. *Appl. Intell. Int. J. Artif. Intell. Neural Netw. Complex Probl.-Solving Technol.* **2021**, *51*, 889–905. [CrossRef]
54. Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; Long, M. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. *arXiv* **2023**, arXiv:2210.02186.
55. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are transformers effective for time series forecasting? *Proc. AAAI Conf. Artif. Intell.* **2023**, *37*, 11121–11128. [CrossRef]
56. Liu, P.; Wu, B.; Li, N.; Dai, T.; Lei, F.; Bao, J.; Jiang, Y.; Xia, S. WFTNet: Exploiting Global and Local Periodicity in Long-term Time Series Forecasting. *arXiv* **2023**, arXiv:2309.11319.
57. Sharpe.; William, F. The Sharpe Ratio. *J. Portf. Manag.* **1994**, *21*, 49–58. [CrossRef]
58. Théate, T.; Ernst, D. An application of deep reinforcement learning to algorithmic trading. *Expert Syst. Appl.* **2021**, *173*, 114632. [CrossRef]
59. Taghian, M.; Asadi, A.; Safabakhsh, R. Learning financial asset-specific trading rules via deep reinforcement learning. *Expert Syst. Appl.* **2022**, *195*, 116523. [CrossRef]
60. Liu, Z.; Huang, D.; Huang, K.; Li, Z.; Zhao, J. Finbert: A pre-trained financial language representation model for financial text mining. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, Yokohama, Japan, 7–15 January 2021; pp. 4513–4519.
61. Li, Y.; Wang, S.; Ding, H.; Chen, H. Large language models in finance: A survey. In Proceedings of the Fourth ACM International Conference on AI in Finance, Brooklyn, NY, USA, 27–29 November 2023; pp. 374–382.
62. Dong, M.M.; Stratopoulos, T.C.; Wang, V.X. A scoping review of ChatGPT research in accounting and finance. *Int. J. Account. Inf. Syst.* **2024**, *55*, 100715. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.