Reinforcement Learning in Quantitative Trading: A Survey

Reinforcement Learning in Quantitative Trading: A Survey

Ali Alameer, *Member, IEEE,*, Haitham Saleh, *Member, IEEE,* Khaled Alshehri, *Member, IEEE*

***Abstract*—Quantitative trading through automated systems has been vastly growing in recent years. The advancement in machine learning algorithms has pushed that growth even further, where their capability in extracting high-level patterns within financial markets data is evident. Nonetheless, trading with supervised machine learning can be challenging since the system learns to predict the price to minimize the error rather than optimize a financial performance measure. Reinforcement Learning (RL), a machine learning paradigm that intersects with optimal control theory, could bridge that divide since it is a goal-oriented learning system that could perform the two main trading steps, market analysis and making decisions to optimize a financial measure, without explicitly predicting the future price movement. This survey reviews quantitative trading under the different main RL methods. We first begin by describing the trading process and how it suits the RL framework, and we briefly discuss the historical aspect of RL inception. We then abundantly discuss RL preliminaries, including the Markov Decision Process elements and the main approaches of extracting optimal policies under the RL framework. After that, we review the literature of QT under both tabular and function approximation RL. Finally, we propose directions for future research predominantly driven by the still open challenges in implementing RL on QT applications.**

***Index Terms*—reinforcement learning, quantitative trading, machine learning, control theory**

I. Introduction

Quantitative finance is a vast field concerned with applying mathematical and engineering tools to approach financial problems. The complex dynamics and severe stochasticity within financial markets motivate researchers and market participants to frame the financial problems under mathematical and engineering contexts. In the 1950s, Markowitz introduced a central concept in quantitative investing where he formulated the problem of portfolio management as a mathematical program under a single-stage optimization framework [1]. Quantitative finance also extended to the problems of market-making and cost-minimum execution of trades. Such problems where used to be approached through optimal control theory [2]–[5]. The emergence of the Reinforcement Learning (RL) [6], a machine learning paradigm that intersects with optimal control theory, and its successful application in real-life motivated researchers to explore its potency in the field of quantitative finance. Specifically, advanced recent RL algorithms have shown success in defeating world champions in the game of Go and Chess with RL algorithms such as "AlphaGo" and "AlphaZero" [7], [8], while the "MuZero" agent showed superior performance in a wide range of games including shogi and arcade [9]. That success of RL has led researchers to investigate its potential in solving the market-making, optimal trade execution, and investment problems [10]–[35].

Investment in financial markets, generally speaking, can be categorized into three main paradigms: trading a single asset, which can be referred to as Quantitative Trading (QT), assets allocation, and portfolio management. Transacting a single instrument is about buying and selling the asset repetitively in a short period, usually with an intraday interval. When applying auto-trading systems, one can also use High-Frequency Trading (HFT) strategies [36] where multi-trades can be performed within seconds or even microseconds. From an engineering perspective, the problem of assets allocation, where assets with various risk levels are traded at different investment capital weights, extends the problem of transacting a single asset. With that approach, one can consider re-balancing the investment capital at each time interval among risky (e.g., stocks) or risk-less (e.g., Treasury bonds) assets. The reflected interval in asset allocation problems is usually significantly larger than that of single-asset trading, where holding assets may range from a week to several months. For portfolio management problems, investors form a portfolio consisting of an arbitrary number of assets, where re-balancing the assets' weights can also occur at large time intervals.

Under all paradigms, the investor's ultimate objective is to optimize a financial performance measure that usually accounts for the potential profits against any associated risks with that investment. Under the Modern Portfolio Theory (MPT) proposed by Markowitz [37], the main objective is to maximize the expectation of investment returns while minimizing its variance, which he relates the latter to an investment potential risks [1]. Since then, the concept of associating return variance with investment risks has become common among researchers and practitioners. For example, Sharpe introduced the idea of Sharpe ratio as a performance measure, which is defined as the ratio of a risky asset's excess returns to its standard deviation [38]–[40]. That measure, however, is symmetric since it counts for both the upside and downside returns variance as an investment risk. An investor is usually concerned about the variance of the investment downside returns rather than the overall. Hence, the concept of downside risk as a performance measure was introduced by Sortino in [41]. After that, Rockafellar and Uryasev [42] introduced the notion of the Conditional Value-at-Risk (CVaR) that is only concerned about the tail of the downside return distribution instead of the overall as proposed by Sortino.

To meet the investment objective, all paradigms involve an iterative process of analyzing the market and making decisions. We note that the type of analysis differs depending on the followed investment approach. For trading a single asset, the preferred type of analysis among practitioners is the so-called technical analysis [43], [44], meanwhile fundamental analysis [45], [46] and macroeconomic variables [47], [48] may also be used for markets assessment in the case of assets allocation, and portfolio management. In its simplest definition, technical analysis is about generating temporal signals that are a function of the asset opening, low, high, and closing prices at a predefined number of past time intervals. Those prices, within a single time interval, form the well-known price candlestick [49]. We note here that the use of technical analysis has proven, theoretically and practically, its

A. Alameer and K. Alshehri are with the Control and Instrumentation Engineering Department, King Fahd University at Petroleum and Minerals (KFUPM). H. Saleh is with the Industrial and Systems Engineering Department, KFUPM. K. Alshehri and H. Saleh are also with the Interdisciplinary Research Center for Smart Mobility and Logistics, KFUPM. Email: {g201904230,haithamhs,kalshehri}@kfupm.edu.sa.

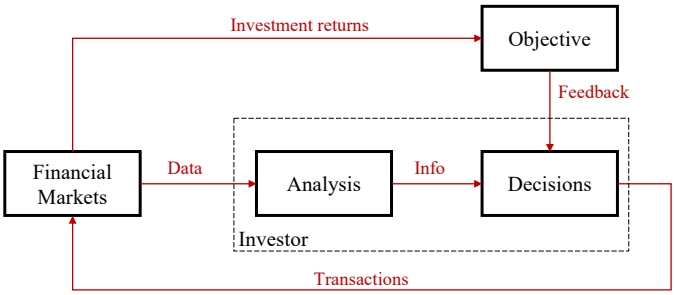


Fig. 1: A general block diagram of the trading process in financial markets

usefulness in extracting nonlinear regularities in noisy price signals [50], [51]. After generating those technical indicators, technicians use these temporal signals to form a prediction about the potential future movement of the asset's price and decide to either buy or sell the asset accordingly. Among the most common technical indicators are the Simple Moving Average (SMA), representing the closing price's mean over predefined past time steps, and the Relative Strength Index (RSI) where it indicates that the traded asset might be oversold (respectively overbought) and suggests a potential price increase (respectively decrease) over the near future. For other common technical indicators and technical analysis approaches, we refer the reader to these references [52]–[54].

After analyzing the market and forming a prediction about the asset's price movement, the investor has to make trading decisions based on that prediction. Under any investment approach, initiating a trade includes two possible decisions: to open long or short positions. A long position is about buying the asset with predicting the price to move higher over time. On the other hand, short selling is related to borrowing the asset and selling it to a third party when predicting a decline in its price. Later on, the asset is repurchased and returned to the lender in exchange for cash. In that sense, a short position is profitable in case of an asset's price drop. We note here that even closing the initiated position requires thorough analysis to ensure timing the closing optimally. Intuitively, the trader sells the asset to close a long position, whereas the trader "covers" its selling position. Note that each decision in the process involves transaction frictions. The clearest one is the commission a broker charges the trader of transacting assets, usually a percentage of the traded amount [3]. The other charge is the "spread" cost that results from the difference between the *bid* and *ask* prices of an asset. One can refer to Fig.1 to visualize the whole process that a trader may go through while trading within financial markets.

Reinforcement Learning is an experience-based and goal-oriented learning system that aims to optimize an agent's behavior within an environment. Therefore, it can be categorized as a machine learning paradigm other than supervised and unsupervised learning approaches. Trading through supervised machine learning can be challenging since one trains it to predict the price to minimize the prediction error rather than maximize a financial performance function, we refer the reader to these works that look at the problem with supervised learning [55]–[59]. To this end, one of the vital privileges that RL could introduce into approaching Quantitative Trading (QT) is its potential in performing the two main investment steps, market analysis and making decisions, without explicitly predicting the future price movement. More importantly, all of that may be formulated under an online multi-stage optimization framework where the agent interacts within the financial markets and learns an optimal policy (trading strategy) to meet the long-term investment objectives.

The root of the RL concept is backed to 1948 when the so-called "pleasure-pain system" was introduced by Turing in his report *Intelligent Machinery* [60]. In that report, Turing discussed the potential of computer-based artificial intelligence inspired by the biological theory of "reinforcing" animal behavior through a trial-and-error learning approach. On the other side, the research in the field of optimal control theory was progressing extensively at that time. In the 1950s, Richard Bellman proposed a pioneering notion in optimal control when he proposed the utilization of system states dynamics and control feedback signals to extract a control strategy such that a predefined function is optimized [61]. Bellman also proposed solving this function, which is known as the Bellman optimality equation, by Dynamic Programming (DP). At that time, computational intractability was imposing limitations in the use of DP especially when the number of state variables of a stochastic dynamical system is large. Another shortcoming in using DP for finding optimal behaviors is the requirement of complete knowledge about the system dynamics. These challenges perhaps motivated melting the concept of learning with DP. This was first devised by Werbos in 1977 in his efforts to develop advanced forecasting methods for coping with the global crisis [62] and he, therefore, introduced the concept of heuristic DP that works as an approximate approach for solving the problem. About ten years later, a seminal work in RL was introduced by Watkins whose completely leveraged DP and optimal control with learning and proposed the concept of model-free RL in his Ph.D. thesis [63]. In that context, one does not need to know the dynamical behavior of a system to extract an optimal control signal (or policy in RL framework), it is rather learned through estimating the Bellman equation by experience. Since then, the research in RL has been extensive and this led to diverse novel algorithms that in turn grab the attention of other engineering field researchers, this includes the field of quantitative finance. To our knowledge, the first two works that considered formulating the investment problem using RL were the ones by Neuneier [11], and Moody and Saffel [64]. Neuneier implemented the critic approach to solve the problem of assets allocation by extracting a controller's policy based on estimating value functions with Q-learning. Instead, Moody and Saffel explored the actor approach, direct policy search, which they called Direct Reinforcement Learning (DRL), and tested it on the Forex market. The reader can refer to Fig.2 to see the progress and the noticeable contributions of the QT research with RL since its inception in 1997.

A. Scope and Review Perspectives

The scope of this manuscript is surveying the application of RL on QT, a branch of quantitative finance. Under that scope, the objective is to provide the interested readership with an exhaustive review of how far the ongoing research of QT with RL reached the frontier of RL algorithms development. In light of that, we disseminate suggestions about future research directions endeavoring to prosper the field of QT with RL. We emphasize that we survey the literature considering the perspectives discussed below

- *Markov Decision Process design*: this is concerned about the overall setup of the Markov Decision Process (MDP), including states, actions, rewards. The MDP forms the fundamental structure of any problem approached by RL.
- *Handling markets' uncertainty and non-stationarity*: uncertainties in the financial assets price always exist. That mainly

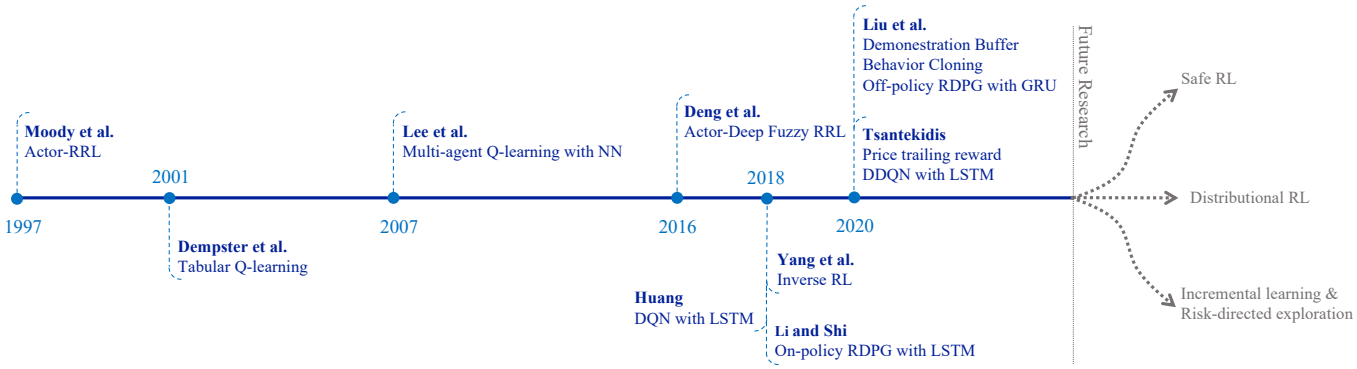


Fig. 2: A timeline that summarizes noticeable progress of QT with RL since the research inception in 1997. The gray dashed line represents the directions for future work we propose in this survey.

imposes serious challenges encountered when developing practical auto-trading systems. This survey evaluates the different existing approaches for handling such challenges. For example, we assess the designed methods of tackling RL's exploration versus exploitation dilemma.

- *RL's agent risk-sensitivity*: traders tend to follow risk-sensitive strategies. In that sense, they look for strategies that would result in excess returns while properly controlling risk measures. So, we assess the reviewed trading systems in terms of their agent's risk sensitivity.
- *Modelling QT frictions*: transaction, slippage, and spread are undesirable costs traders face. In addition, a trading strategy under frictions is dissimilar to a policy without frictions, i.e., considering frictions leads to path-dependent trading returns. In the RL context, the optimal policy under both cases, with or without frictions, is different since the mathematical structure of the performance measure would also vary.
- *Benchmark with baseline strategies*: Deploying RL-trading systems adds complexity compared to baseline strategies. That being said, we see it crucial for any proposed RL-based QT to be compared with a more straightforward approaches, for example, Buy&Hold, Dual Moving Average Corssover [65], Momentum Trading [66] (see [53], [65], [67]–[69] for other baseline strategies).
- *Testing evaluation metrics*: due to the importance of risk management while trading, risk measures along with total returns associated with back-testing a trading strategy are better to be reported. That includes, for example, Sharpe ratio, Sortino ratio, and maximum drawdown.

Noteworthy, there exist several surveys in the literature that discuss various quantitative investment strategies with the use of RL [70]–[72]. We need to emphasize that our work has main differences that we describe as follows:

- We amply discuss the historical aspects of RL, including its intersection with control theory, and how the research in implementing these two concepts on quantitative trading evolved with time along with the emergence of RL. To this end, we devote one section to discuss the literature of quantitative trading with the tools of control theory since it closely relates to RL.
- We endeavor to bind the investment problem with RL through a conceptual discussion about the problem and the fundamental mathematical formulation of RL.
- We abundantly review the literature to propose novel model formulations inspired by recently introduced RL algorithms that we believe would forwardly drive the research related to the problem.

The paper is organized as follows: In Section II we discuss

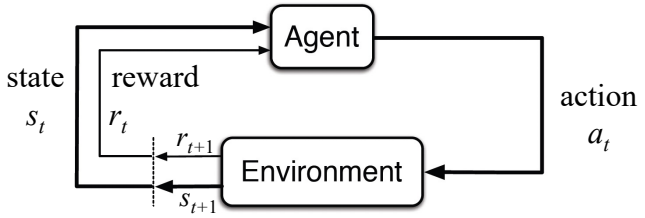


Fig. 3: The agent-environment interaction in reinforcement learning [6].

RL preliminaries, which mainly includes the MDP and its main elements, i.e., the state, actions, and rewards. We also discuss the general formulation of optimal value functions and policies that are attained by the agent's learning process. Following that, the different RL approaches are amply deliberated, including critic-, actor-, and actor-critic-based methods. Before proceeding in reviewing the existing literature of QT with RL, we shed light on the papers that use control theory to develop trading systems in Section III, and that is mainly driven by the fact that control is arguably foundational to the inception of RL notion. We expose our review of RL-based QT in Section IV where we focus on the trading systems developed under tabular RL, mainly with Q-learning algorithms. We then extend our review under function approximation RL in Section V, and we discuss the literature of QT with all RL paradigms. In light of that exhaustive review, we propose directions for future research that we see worth exploring in Section VI whereas the survey is concluded in Section VII.

II. Preliminaries: Reinforcement Learning

Here, we deliberate the fundamental elements of the RL framework, i.e., the environment and the agent and their interaction under the Markov Decision Process (MDP). We then move to discuss and provide general formulations of the different central RL approaches encompassing, critic-, actor- and actor-critic-based methods.

A. The Markov Decision Process

The RL notion is fundamentally based on the concept of MDPs [61], [73]. An MDP composes an environment and an agent where the interaction produces a tuple of temporal trajectories. When these interactions are endless in time, we classify the MDP as an infinite process. However, a finite MDP has a terminal state and is called episodic MDP. While

states describe the environment, the agent is represented by its policy that maps those states into actions. In that sense, the agent selects an action a, from a set of actions $\mathcal{A}$, based on the observed state $s \in \mathcal{S}$ following a stochastic policy denoted by $\pi \sim p(a|s)$ or a deterministic policy $\pi(s)$. The agent then receives a reward r that represents a "reinforcing signal" showing the action's effectiveness at s, and then the agent observes a new state s' as a subsequent of its action a at s. An observed trajectory can represent that whole sequence, denoted by $\tau = \{s, a, r, s'\}$. Those interactions between the agent and the environment are illustrated in Fig. 3.

The transition between states in the environment is statistically modeled with a probability distribution $s' \sim p(s'|s, a)$. Like s', the observed reward can also be modeled as $r \sim p(r|s, a)$. Combining the probability density of both s' and r results in a joint distribution that represents the environment's one-step-ahead dynamics

$$\psi_{S',R} := p(s', r|s, a) \tag{1}$$

We note here that the RL algorithm is model-based when the dynamics of the environment, described in (1) are known. With that, one can find an optimal policy by solving the well-known *Bellman equation* using DP techniques. Yet, we leave the discussion of obtaining optimal policies with DP under model-based RL since it is out of this work's scope; we instead refer the interested readers to related discussions that can be found in [6], [61]. If the MDP dynamics are unknown, like financial markets, one can solve the problem through different model-free RL approaches. Before discussing those approaches, let us see the fundamental formulation for finding optimal policies within a model-free MDP.

B. Optimal Value Functions and Policies

The main objective of the learning process, i.e., trial-and-error under model-free MDP, is to find a policy π that maps states into actions while simultaneously optimizes an action-value function[1] that counts for the policy's performance measure, and we denote it by $Q^{\pi}(s, a)$. Let us consider a finite MDP with a length of T and time steps indicated by t, then $Q^{\pi}(s, a)$ can be described by

$$Q^{\pi}(s, a) = \mathbb{E}[R_t|s = s_t, a = a_t] \tag{2}$$

With the cumulative reward R_t being defined as

$$R_t = \sum_{k=0}^{T-1} \gamma^k r_{t+k+1} \tag{3}$$

Where $\gamma \in [0, 1]$ is a discount factor. We now need to emphasize several notes. First, the action-value function governed by (2) and (3) indicate that the state-action value is the expected discounted cumulative reward when taking action a_t at s_t following the policy π from that time step and onward. To determine how farsighted the agent is, one can use the discount factor γ that tunes the observed reward's weights that contributes to the cumulative reward R_t.

Finally, a policy is called optimal, denote it by π^*, if it is associated with an optimal action-value function, that is

$$Q^{\pi^*} = \max_{\pi} Q^{\pi}(s, a); \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{4}$$

Where the controller can reach π^* through iterative experiential learning that can be performed with different model-free RL approaches, which we discuss in the next subsection.

[1]The value can also be a function of states only. That is precisely the case with model-based RL where the optimal policy is derived using DP. However, with model-free RL, it is better to represent the values with the state-action pairs, refer to [6].

C. Reinforcement Learning Approaches

1) Critic-Based Methods: Following this approach, the action-value function under policy π can be estimated by either tabular or function approximation methods [6]. With tabular methods, each state-action pair forms an entry to a table and is assigned a value. Clearly, with this method, the MDP can only be represented by discrete state and action spaces. On the other hand, function approximation methods allow for continuous state spaces where a vector of parameters approximates the values. The optimal value function, with tabular methods, can be obtained by SARSA (State-Action-Reward-State-Action) or Q-learning algorithms. Both of those approximate the solution of the *Bellman equation* by trial-and-error experience, which their proof of convergence to an optimal policy is deliberated in [74] and [75], respectively. Both algorithms are fundamentally based on the Temporal Difference (TD) notion [76]. However, they differ in the learning approach where SARSA learns with the so-called on-policy learning while Q-learning is off-policy [63]. To this end, with the SARSA algorithm, learning the value function depends on the pair value of the currently taken action and the next observed state. That is, the agent learns an optimal policy π^* with its associated optimal value function $Q^{\pi^*}(x, a)$ through an iterative one-step update rule given by

$$\begin{aligned} Q^{\pi}(s_t, a_t) \leftarrow & Q^{\pi}(s_t, a_t) + \\ & \alpha\left[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_t) - Q^{\pi}(s_t, a_t)\right] \end{aligned} \tag{5}$$

Where $\alpha \in (0, 1)$ is the learning rate. On the other hand, with Q-learning that follows the off-policy approach, the action-value update is carried out considering the action that would reveal maximum value, i.e., greedy, at the next observed state regardless of the currently taken action; the update rule is therefore given by:

$$\begin{aligned} Q^{\pi}(s_t, a_t) \leftarrow & Q^{\pi}(s_t, a_t) + \\ & \alpha\left[r_{t+1} + \gamma \max_a Q^{\pi}(s_{t+1}, a) - Q^{\pi}(s_t, a_t)\right] \end{aligned} \tag{6}$$

Under both algorithms, we note that all state-action pairs shall be visited frequently to ensure continuous updates for proper convergence [6]. Clearly, with that, the data and computational power required to find the optimal policy and having a satisfactory generalization performance grows significantly with the increase of state-action pairs, which leads to the well-known chronic dilemma of *curse of dimensionality*. To this end, function approximation has been introduced to improve the generalization performance in large state spaces since the agent can approximate the value of non-visited state-action pairs to nearby ones that were visited before. With a parameter vector $\Theta \in \mathbb{R}^{\kappa}$, one can assume

$$Q^{\pi}(s, a; \Theta) \approx Q^{\pi^*}(s, a) \tag{7}$$

and reach an approximation using linear functions or non-linear ones, similar to the pioneering work of Mnih et al. [77] where they introduced the concept of Q-network by utilizing a deep NN as the function approximation method, which they called DQN. Along with these approximations, one can iteratively learn the parameter vector Θ by minimizing the difference between an online and target networks in an off-policy learning approach, i.e.,

$$L_i(\Theta_i) = \mathbb{E}\left[\left(y_i^{DQN} - Q(s, a; \Theta_i)\right)^2\right] \tag{8}$$

Where

$$y_i^{DQN} = r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \Theta^-)$$

L_i and Θ_i count for the loss and the parameter vector, respectively, at the i^{th} iteration, while Θ^- is the target network parameter vector. Then, as proposed in [77], one can update the parameters in the direction that minimizes the loss in (8) using Gradient Descent (GD) methods [78]

$$\Delta\Theta = -\alpha \nabla_{\Theta_i} L_i(\Theta_i) \tag{9}$$

To enhance the data efficiency for the agent's learning process, Mnih implemented the concept of *memory replay* [79] where the agent not only learns from recent observations, but also learns from old experiences where they are stored in a memory buffer and used as mini-batches to update the online network. Moreover, Schaul et al. proposed an algorithm to further enhance the use of the memory replay, and that is by using a prioritized experiences along the learning process (see [80]).

Hasslet observed that an overoptimistic value estimation is usually attained with DQN [81]. He found out that is attributed to the fact that the same maximum operator is used twice during action evaluation, as shown in (8), and executing actions by the agent. Hasslet, as a consequence, proposed the concept of Double DQN (DDQN) in [82]. The DDQN algorithm is quite similar to DQN except for y_i^{DQN} where it becomes as follows in DDQN architecture

$$y_i^{DDQN} = r_{t+1} + \gamma Q(s_{t+1}, \max_a Q(s_{t+1}, a; \Theta_i); \theta^-)$$

Noteworthy, under either tabular or function approximation methods, ε-greedy policies are preferred when the agent's environment is non-stationary [6], that is precisely the financial market dynamics behavior. With that exploratory attitude, the agent takes greedy actions, i.e., $a_t = \max_a Q^\pi(x_t, a)$, with probability $1 - \varepsilon$, meanwhile it explores the financial market by taking random decisions with probability ε. Thus, selecting ε is crucial in having a fair balance between exploitative and exploratory decisions. In fact, exploration versus exploitation is one of the main challenges underlying the practical implementation of RL [6]. Nonetheless, we do not see ε-greedy policy is the best approach for exploration under QT, and we will discuss that further in the review sections IV and V.

2) Actor-Based Methods: These methods aim to directly learn a policy that optimizes an objective function independently from action-value estimation. The learning process involves a parametric policy whose parameters are updated to optimize the objective function. We refer the reader to the pivotal work of Williams that originally introduced these methods, see [83], [84]. Notwithstanding, here we discuss the formulation of actor-based approach following Moody and Saffell [64], and we call it DRL to be consistent with the literature. Under that approach, let

$$\pi(s; \Theta) \tag{10}$$

denote a deterministic policy function that has Θ as its decision vector. One can represent the function by a linear or non-linear universal approximators, for example, feed-forward NN, Long Short-Term Memory NN [85], [86], or even Convolutional NN (CNN) [87]. Note that the policy in (10) is a function of the environment's states. Obviously, with that formulation, the policy's actions are based on the sensed state of the system.

When DRL is applied to trading problems, one can evaluate the action efficacy, at each time step, by observing the achieved investment returns due to that action. Hence, we can define the reward with neglecting the discount factor, i.e., $\gamma = 0$ and consequently $R_t = r_{t+1}$. Based on that, let us now define a general performance measure that is a function of the received rewards within T, indicated by $\mathcal{U}_T(R)$; then one can set up the problem as

$$\max_{\Theta} \quad \mathcal{U}_T(R) \tag{11}$$

Where $R \in \mathbb{R}^T$ is the reward vector, and the objective is to find the decision vector Θ that maximizes[2] our performance measure $\mathcal{U}_T(R)$. To improve the policy with respect to the performance criterion in (11), the decision vector Θ can be updated in an offline manner. That is, after a full sweep on all trajectories in an episode, one can apply gradient ascent as follows

$$\Delta\Theta = \alpha \nabla_\Theta \mathcal{U}_T(R) \tag{12}$$

However, one may be interested in having an online algorithm where the parameters are updated at each time step t. That can be attained by redesigning the performance criterion to be a function of the received rewards up to and including time t, i.e.,

$$\mathcal{U}_t(R_1, R_2, ..., R_t) \tag{13}$$

In that sense, the policy is now a function of Θ_t, denoted by $\pi_t(s_t; \Theta_t)$, and the parameter vector can be updated through Stochastic Gradient (SG) methods [88]

$$\Theta_{t+1} = \Theta_t + \alpha \nabla_{\Theta_t} \mathcal{U}_t(R_1, R_2, ..., R_t) \tag{14}$$

We need to point out that optimization with SG has an attractive privilege. Since SG provides "noisy" gradient information, it enhances the exploratory behavior of the agent. That behavior can be altered by tuning the learning rate.

3) Actor-Critic Methods: The actor-critic method is an approach that is between critic- and actor-based methods. Under that paradigm, the agent learns an estimate for the value function while simultaneously follows a policy represented by a function. In that sense, the estimation error, i.e., the TD error, of the value function represents the critic part, whereas the actor part is used to update the agent's policy based on that error. The reader can refer to Fig.4 that illustrates the general learning process under this RL method. Following the fundamental formulation of this approach in [6], consider the action-value in (6) under off-policy learning, then one can define the value estimation error as

$$\delta_t = r_{t+1} + \gamma \max_a Q^\pi(s_{t+1}, a) - Q^\pi(s_t, a_t) \tag{15}$$

Also, considering a policy that, for example, is given by a *softmax* function

$$\pi_t(s, a) = \frac{e^{p(s,a)}}{\sum_b e^{p(s,b)}} \tag{16}$$

Then one can update the value and policy functions independently

$$\begin{aligned} Q^\pi(s_t, a_t) &\leftarrow Q^\pi(s_t, a_t) + \alpha \delta_t \\ p(s, a) &\leftarrow p(s, a) + \alpha_a \delta_t \end{aligned} \tag{17}$$

Where $\alpha_a \in (0, 1)$ is the learning rate for the actor part. With that, the tendency of taking a at s_t increases or decreases depending on the value of the error δ_t. In critic-based methods, besides their limited applications to discrete action-space problems, a small perturbation in the value estimation results in a significant change in the followed policy, which imposes serious convergent intricacies in many practical applications. The actor-critic algorithm can improve that unstable behavior by approximating the policy with a function whose parameters are updated with SG methods to improve the performance

[2]It also can be a minimization problem. That all depends on the defined performance measure.

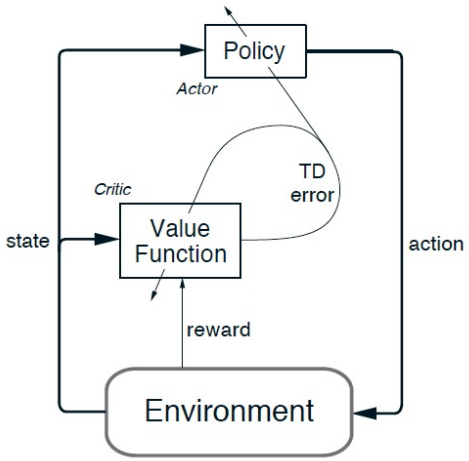


Fig. 4: The RL actor-critic architecture [6].

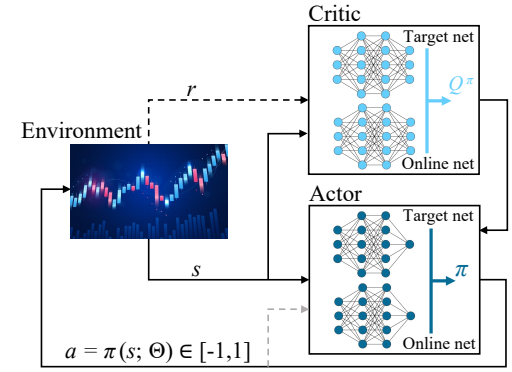
measure, i.e., the estimation of the value function. These methods have been extended to function approximation where separate parameter vectors approximate the critic and actor parts independently; for example, see the work of Konda and Tsitsiklis [89]. Further, Sutton et al. [90] proposed an actor-critic approach under function approximation that is based on stochastic policy gradient. Recently, Silver et al. [91] proposed the deterministic policy gradient (DPG) algorithm that is significantly more sample-efficient than its stochastic counter algorithm, especially in problems with large continuous action spaces. Gradient computation in DPG only requires integration over the state space, whereas the stochastic algorithm necessitates integration over both state and action space. Note here that even policy gradient algorithms do have on-policy and off-policy learning, and the details of those can be found here [91]. Let us consider, under on-policy learning, a performance measure associated with a parametric policy $\pi(s;\Theta)$, denoted by $J\big(\pi(s;\Theta)\big)$, that counts for the reward expectation, as the right hand side of (2), then the policy gradient with respect to theta considering a stochastic policy can be given by [91]

$$\nabla_\Theta J\big(\pi(s;\Theta)\big) = \mathbb{E}_{s'\sim p(s'|s,a),a\sim\pi}\big[\nabla_\Theta \log \pi(a|s;\Theta) Q^\pi(s,a)\big] \tag{18}$$

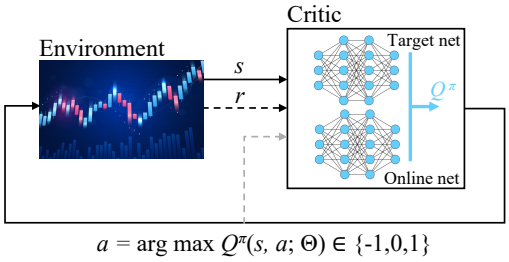
While if one considers a deterministic policy, then the gradient would be

$$\nabla_\Theta J\big(\pi(s;\Theta)\big) = \mathbb{E}_{s'\sim p(s'|s,a)}\big[\nabla_\Theta \pi(s;\Theta) \nabla_a Q^\pi(s,a)|_{a=\pi}\big] \tag{19}$$

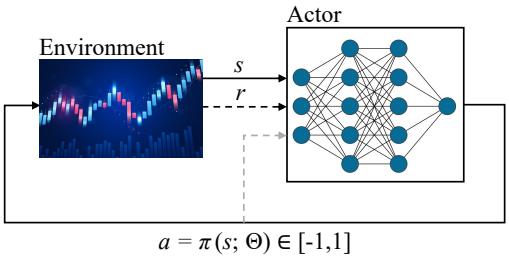
While the critic can be updated through (9), one can then update the policy's parameter Θ in the ascent direction of the performance measure using (18) and (19) for stochastic and deterministic policies, respectively. Note that for computing the gradients in (18) and (19) one would need to estimate the value functions, similar to what we described in (7). Noteworthy, various advancements related to actor-critic were exposed in the literature since the work of Silver, mainly inspired by the successful enhancements implemented on DQN. For instance, Lillicrap et al. [92] extended the concepts of memory replay and target networks to the DPG framework, which he called Deep DPG (DDPG). Moreover, Mnih et al. [93] proposed different asynchronous deep RL algorithms, and the most remarkable performance was found in the asynchronous actor-critic, which they called A3C. The A3C is an on-policy learning method that to some extents



(a) Actor-Critic



(b) Critic



(c) Actor

Fig. 5: Quantitative trading systems based on the main, and common, function approximation RL approaches; (a) Illustrates the configuration of the system under actor-critic with DPG; (b) Shows the configuration of a critic with DQN architecture; (c) Demonstrates the actor part under feedforward NN as the function approximation for the agent's policy. The dashed grey signals under all figures represent the possibility of the recurrent structure where the agent remembers its last action (or holding), that is to learn a path-dependent policy and hence reduce transaction costs.

has similar impact to memory replay on learning from data. During the learning process in A3C, instead of memory replay, A3C asynchronously executes multiple agents in parallel on various states of each. In that sense, the de-correlation of samples is evident, similar to the impact of the memory replay concept while enhancing the use of data. Further, DDPG is similar to DQN in the overestimation of the value function since the evaluation and execution of actions are based on the greedy policy. Therefore, Fujimoto et al. [94] overcome this challenge by implementing double DDPG, which they called twin delayed deep deterministic RL (TD3). The most current state-of-art actor-critic method under function approximation is the Proximal Policy Optimization (PPO). Instead of learning from one sample at each gradient step in DPG, PPO offers a

novel performance function that allows multi-epoch of mini-batch updates; it thus has better empirical sample complexity [95]. Moreover, Haarnoja et al. [96] proposed a Soft Actor-Critic (SAC) algorithm that deploys the concept of maximum entropy RL [97] under the actor-critic framework. That was shown to boost data efficiency during the learning process due to enhancing the exploratory behavior of the agent. At the same time, it also uses target networks and memory replay over the learning process.

We need to underscore that auto QT systems can be developed using the different RL approaches we discussed earlier. For visualizing those systems, the reader can refer to Fig.5 where it shows the interaction of the agent and financial markets during the trading process under the common RL approaches. Before discussing the literature related to QT with RL, let us first shed light on the QT research using control theory in the next section.

III. Control-Theoretic Approach

Essentially, one can view financial markets as dynamical systems in which temporal aspects govern their behavior. Control theory is a very attractive tool for decision making in dynamical systems, however, it often requires good knowledge of the dynamics governing how the states evolve with time. In control, the dynamics are assumed to take a certain form and the reward needs to be well-defined, and then, the controller (or agent, in the context of this paper) is designed. Given the severe stochasticity of financial markets, it often becomes challenging to have a clear description of the dynamics. Nevertheless, there are techniques in control theory that proved useful for trading under uncertainty. From a control perspective, computing the policy in reinforcement learning can be viewed as an approximation to the optimal control policy, which maximizes the reward over the entire horizon subject to system dynamics and constraints.

In [98], a risk-free stochastic control approach was considered for optimal pairs trading, and the optimal control policy was derived via the Hamilton-Jacobi-Bellman equation. Uncertain parameters were computed via maximum-likelihood estimation. For mean-reverting assets, conditions for the optimal control policy to buy low and sell high were derived in [99]. A different paradigm in [100] was considered, where the control design is model-free, that is, there is no parametrization nor estimation of the stock price dynamics. The work provides robust performance guarantees, but for a frictionless market. For Simultaneous Long-Short trading, the performance limits of feedback control were discussed in [101] and the work in [102] demonstrated that linear feedback control, which works well for time-invariant price dynamics (not realistic in financial markets), and has good potential for time-varying dynamics.

To deal with uncertainty, one can also use what is called robust control [103], a control paradigm which optimizes under the worst-case scenarios of uncertain parameters. For risk-averse stock trading, the work in [104] looked into mean-variance considerations, and concluded that mean-variance measures might be inappropriate when feedback control is used, which motivates the use of other risk measures.

On another note, control methods can be useful in portfolio optimization. For example, in [105], the multi-period portfolio optimization problem was addressed by using linear control policies, which is suboptimal but can be computed by a convex quadratic program, making its computation attractive. Furthermore, in [106], stochastic interest rates were considered, and explicit solutions to control strategies were provided.

For real implementation of control methods, reinforcement learning provides an attractive approximation to optimal control policies, and in this section, we have provided examples from the control literature to shed light into some interesting efforts, which remain mostly idealized.

IV. Quantitative Trading via Tabular Reinforcement Learning

In the context of QT, the market's environment is complex and requires large and continuous state space for appropriate representation. As we discussed earlier, QT practitioners have a common approach of using temporal data to generate trading signals; this includes technical indicators, past price returns, or any financial metrics related to fundamental analysis. Although we believe that tabular methods are with limited practical use for QT. However, we still offer a detailed discussion about noticeable works, while providing a general overview of many other relevant papers.

A summary of all reviewed papers that use RL with tabular methods is presented in Table I. The work of Dempster et al. in [107] was instrumental in integrating QT with RL through implementing the notion of Q-learning [113]. In their work, the environment is represented by technical indicators that form a string with sixteen bits while the actions follow a discrete-deterministic policy $\pi(s) \in \{0, 1\}$ where 0 indicates selling signal whereas 1 represents indicates buying. Moreover, their action-value function is modeled as the expectation of the multiplicative returns, rather than the additive described in (3),

$$Q(s_t, a_t) = \mathbb{E}\left[\prod_{t=1}^{T-1}\left(\frac{p_{t+1}}{p_t}\right)^{\pi(s_t)}(1-c)^{|\pi(s_t)-\pi(s_{t-1})|}\right] \quad (20)$$

Where p_t represents the asset's price at time t, and c is a fixed percentage representing the transaction cost. That expectation in (20) is approximated using Q-learning where the update rule takes the back-recursion form described in (6).

They compared the performance of their trading system with strategies extracted from Genetic Algorithm, Markov-Chain Linear Programming approximation, and simple heuristic. Interestingly, in their back-testing of the trading strategy, they observed significant adverse change in the result when transaction cost is imposed.

We see that Yang et al. in [110] had a novel contribution in the field of QT using RL where they implemented the concept of Gaussian Process Inverse Reinforcement Learning (GPIRL) to generate trading signals. The Inverse Reinforcement Learning problem was originally formulated by Russell in [114], and Ng et al. in [115]. In the IRL context, rather than finding an optimal policy from a sequence of observations, we aim to extract a reward function by observing a given policy in an MDP. The concept was enhanced further by Ramachandran et al. in [116] when he formulated the idea of IRL in the Bayesian framework. Qiao et al. in [117] then extended this work by discussing GPIRL to handle the problem of partially observed optimal behaviour (policy) in IRL setup (For further reading related to IRL and recent advancements, we refer the reader to [97], [118], [119], [120], [121] and [122]). Back to Yang's work in [110], the authors used the extracted reward function by GPIRL as features space to train a supervised-learning-based trading system. Noteworthy, IRL could help in overcoming the challenge of a handcrafted reward function, and that may be combined with RL to train an agent based on the learned reward by the IRL algorithm [123].

It is observed that most of the works are using model-free Q-learning algorithm to estimate the action-state value function, except for Chen et al. in [112] where the authors used SARSA and Yang et al. in [110] where a model-based algorithm is used based on IRL. We have observed the limited

TABLE I: Summary of Quantitative Trading with Reinforcement Learning - Tabular Methods

Reference	RL Algorithm	Policy		Value Function		Assets	QT Frictions			Evaluation Metrics			
		Deterministic	Stochastic	Risk-Neutral	Risk-Averse		Trans.	Spread	Slippage	Total Returns	Sharpe Ratio	Sortino Ratio	Max. Drawdown
Dempster et al. [107]	Model-free, Q-learning	*		*		Forex [15m]	*		*	*	*	*	*
Zhu et al. [108]	Model-free, Q-learning		*	*		USETF [days]				*			
Jangmin et al. [109]	Model-free, Q-learning		*	*		KSM [1d]	*	*	*	*			
Yang et al. [110]	Model-based	*			*	USETF [15m]	*			*	*	*	*
Elder et al. [111]	Model-free, Q-learning	*		*	*	USI [15m]				*			*
Chen et al. [112]	Model-free, SARSA		*	*		JSM [1d]				*			

Notes:
1) US Exchange Traded Fund (USETF); Korea Stock Market (KSM); US Indexes (USI); Japan Stock Market (JSM)
2) m = minute; d = day

use of the eligibility traces concept in estimating the Q-function [124], [125]. The eligibility traces concept is useful when the table entries of state-action values are large. These traces track recently visited entries and give more weights to them when they are updated. The use of eligibility traces weights while learning enhances the computation efficiency; it also provides better sample complexity. Generally speaking, the states implemented to represent the environment satisfy the Markovian property to some extent, where we found that most of the papers use technical indicators or price signals. All these states are temporal data that summarizes the price trends over predefined past time steps, and they also can be related to the agent's external environment. With formulating QT following RL framework in case of retail investors, we recommend to include agent's internal states, for example, the agent holding positions and its performance while holding as it is seen in the work of Elder in [111]. When we use internal states, we make the agent more self-aware by having its action at s_t cause a partial influence on the next observed state s_{t+1}. Moreover, we have noticed that most presented algorithms implement stochastic policies to produce trading signals. Specifically, the most common policy used in these works follows the concept of ε-greedy policy in which the exploratory actions are taken with pure randomness; this approach is also called the undirected exploration method as stated by Thrun in [126]. With that policy design, we can say the dilemma of exploitation versus exploration is moderately addressed. However, as stated earlier in this survey, price signals of financial assets usually experience uncertainty that can be characterized as a non-stationary stochastic process. To this end, we believe that environments with financial assets require more robust policy designs in which exploration actions are also taken strategically, see for example the work of Agostini et al. where they advocated the use of directed exploration strategy using the variance of action-state value estimates in non-stationary environments [127]. The value function design was found mostly as risk-neutral, that is the expected return with a different selection of the discount rate constant γ. Elder in [111], however, used a risk-averse design that accounts for the agent's reward variance. From the trading aspect, when looking at modeling QT frictions during policy learning, it can be observed that it is well-addressed in the work of Jangmin et al. in [109] where they modeled all possible frictions, including transaction, slippage, and spread costs. Elder in [111], on the other hand, included QT frictions while back-testing instead of during learning. As we emphasized earlier, modeling frictions while learning the policy is preferred since it helps the agent to learn path-dependent trading policy. Finally, most of the results are reported using the annualized return as a performance metric. We, however, observed that Yang focused more on reporting risk measures such as Sharpe Ratio, Sortino Ratio, and maximum drawdown [110].

V. Quantitative Trading via Reinforcement Learning with Function Approximations

QT with function approximation solutions is observed to be dominant over tabular approaches in the literature. That perhaps is attributed to their potential use with deep learning [128]. In a sense, integrating the RL notion with deep learning reveals novel insights into QT applications. One aspect is the capability of function approximation to better represents the complexity of financial assets' environments. The ability to develop a more feasible representation for financial instruments' price signals using function approximation and deep learning lay in their potency of capturing complex patterns in large and continuous state spaces. In principle, an agent designed to trade financial assets should have a flexible strategy that can work with never-encountered situations, and to our belief, this is the significance of function approximation with deep learning in the field of QT. To this end, we confer a detailed review for function approximation methods with all RL paradigms, and we discuss them in separate subsections. Similar to our discussion in the tabular methods section, here we focus our review on selected works that we see have novel strides in the field of QT following the RL function approximation paradigm under each approach, and we give an overall discussion about the remaining related literature.

A. Critic-Based Methods

As we stated earlier, function approximation under critic RL aims to approximate the action-value function to handle large and continuous state space, which is usually the case in financial markets. The value function plays the role of the performance index that the agent shall learn to optimize with experiential trial and error. In most cases, the index is the expectation of the returns obtained over the learning process; however, this leads to a risk-neutral agent that lacks dealing with risks associated with QT. The agent is usually greedy in taking the action that would reveal the highest action-state pair value. At the same time, it can explore the environment with random activities from time to time, i.e., the well-known ε-greedy policy we discussed earlier. Exploring with random actions may lead to unnecessary losses. During our review, we will see how some works creatively strategize the exploratory behavior of the agent rather than the pure randomness associated with the ε-greedy policy. Next, we review some foundational works based on the perspectives we discussed earlier, then state our general notes and remarks.

The summary of QT under critic-methods with function approximations is demonstrated in Table II. Lee et al. in [129]

TABLE II: Summary of Quantitative Trading with Reinforcement Learning - Critic-Based Methods

Reference	RL Algorithm	Policy		Value Function		Assets	QT Frictions			Evaluation Metrics			
		Deterministic	Stochastic	Risk-Neutral	Risk-Averse		Trans.	Spread	Slippage	Total Returns	Sharpe Ratio	Sortino Ratio	Max. Drawdown
Lee at el. [129]	Q-learning with NN		ε-greedy	*		KSM [1d]	*		*	*			
Bertoluzzo and Corazza [130]	Q-learning with linear function		ε-greedy		*	ISM [1d]				*	*		
Eilers et al. [131]	Q-learning with NN		ε-greedy	*		USI, GI [1mo]				*			*
Cumming [132]	Least-Square Temporal Difference [133]		ε-greedy	*		Forex [1m]				*			
Corazza and Sangalli [134]	Q-learning and SARSA with linear function		ε-greedy		*	ISM [1d]	*			*			
Carapucco [135]	Q-learning with NN		ε-greedy	*		Forex [2h]		*		*			*
Huang [136]	DQN with LSTM	*		*		Forex [15m]		*		*	*	*	*
Gao [137]	DQN with LSTM, GRN, CNN		ε-greedy	*		APS				*			
Chen and Gao [138]	DQN, DQN with LSTM		ε-greedy	*		USETF [1d]				*			
Lucarelli and Borrotti [139]	DDQN and Duelling DDQN with CNN		ε-greedy		*	Crypto [1m]	*			*	*		
Sornmayura [140]	DQN		ε-greedy	*		Forex [1d]				*	*		
Tsantekidis et al. [141]	DDQN with LSTM		ε-greedy	*		Forex [60m]	*			*	*		*
Carta et al. [142]	Duelling DDQN		ε-greedy	*		USSM [60m]	*			*			*
Theate and Ernst [143]	DDQN		ε-greedy	*		VSM [1d]	*			*	*	*	*

Notes:
1) US Stock Market (USSM); Various Stock Markets (VSM) ;Korea Stock Market (KSM); Italy Stock Market (ISM); German Indexes (GI); US Indexes (USI); US Exchange Traded Fund (USETF); Artificial Price Signal (APS)
2) m = minute; h = hour; d = day

presented a unique methodology of trading with a buy-only strategy with critic-based methods where they applied it on the Korean stock market. The primary approach is to train four agents that collaboratively learn the following sequential tasks: agent to generate a buy signal, agent to advise the best buy price, agent to generate a sell signal after the purchase is done, and finally, an agent to recommend the best-selling price. All agents are developed with the Q-learning algorithm that estimates the state-action values using a feedforwad neural network. Interestingly, all states and actions are specified using an encoding scheme, and each agent has its own state-action space. The states of signal generation agents are matrices with a binary representation for their elements. Fibonacci numbers [144] are used to represent predefined and discrete past periods within those matrices columns. In contrast, the rows represent the change in the price within Fibonacci intervals, forming what they call the Turning Point matrices. Note that the state of the selling signal agent is the unrealized profit resulting from holding a specific asset. The action of the buy signal agent is whether to buy or not, whereas the sell signal is to sell or hold. For order agents, the states are Japanese Candlesticks' bodies, upper and lower shadows, and technical indicators. The actions of order agents are to determine the best price for execution with discrete fractions of the simple moving average for the closing prices, and they are also encoded through binary representation. The main reward for the buying agent is simply the resultant profit of the whole buy/sell process of all agents after discounting the transaction costs, including slippage. Interestingly, the slippage cost is represented by random perturbation of the prices. They also performed a trading cost analysis to evaluate the performance at different transactions and slippage costs.

We observed that most of the recent works consider deep Q-learning at different architectures tested on various financial markets. One remarkable work was carried out by Tsantekidis et al. [141] where they implemented DDQN with a recurrent structure through the use of an LSTM network [145]. With their temporal signals of technical indicators as the states, the implementation of DDQN with an LSTM enhances the environment's representation by capturing time-dependency within the data and, more importantly, avoiding overfitting due to the high noise embedded within the financial markets. Moreover, Huang in [136] also implemented a Recurrent DQN (RDQN) with LSTM as the function approximator and applied it on the forex market. One exciting finding by Huang is that a substantial small replay memory size, around a few hundred experiences, would perform better than a large one. That is perhaps attributed to the non-stationary behavior of the market, where training samples from older experiences may not be representative of the current market regime. Carta et al. in [142] further investigated the performance of DDQN where they implemented the notion of duality proposed by Wang et al. [146] to build an agent that trades within the US stock market. The central concept behind duel DDQN is estimating the action values based on separate estimates of the state value function and the action advantage function. We note here that the key advantage of the dueling architecture lies in avoiding the unnecessary evaluation of some actions at particular states. In trading within financial markets, it is always recommended to be aware of the consequences of all actions at all states, and that is for better risk management. Some of the works compared the performance of different DQN algorithms. Gao [137] made a comprehensive comparison related to the performance of DQN with various function approximators, i.e., CNN, GRU, LSTM at multiple architectures. The comparison, however, is applied to artificial price signals. On the other hand, Chen and Gao in [138] assessed the performance of DQN against RDQN with LSTM in the US stock market. They found that RDQN outperformed DQN in terms of total achieved profits; however, it was under neglected trading commissions. Lucarelli and

Borrotti in [139] compared DDQN with dueling DDQN on bitcoin data. As expected, the experiment revealed that the performance of DDQN outweighs the dueling algorithm in terms of the total generated profits.

As for the exploration versus exploitation dilemma, Cumming [132] in his work proposed an exponentially decaying ε with time

$$\varepsilon_t = e^{-t/\tau}(\varepsilon_{max} - \varepsilon_{min}) + \varepsilon_{min} \tag{21}$$

Where τ governs the decaying rate, and $\varepsilon_{max}, \varepsilon_{min}$ determine the exploration range one would use in training the agent. Following that approach, one may ensure sufficient exploration initially with proper annealing, i.e., less exploratory agent, as time progresses. As that may help improve the training process, we see that it could eventually lead to degraded trading performance as time evolves due to the non-stationary nature of financial markets. Wang et al. in [147], Huang in [136], and Theate and Ernst in [143] also proposed an exploration strategy that prevents taking random actions by the agent when exploring the market. Their idea is to update the action-value function for taken and non-taken actions. That can be achieved in trading problems since knowing the asset price change allows one to observe the reward for all possible actions, then update all action-state pair values accordingly. Due to the inherent nature of Q-learning, most of the works use discrete action space that may eventually lead to risky trading behavior. We, however, observed that Jeong and Kim in [148] used a softmax function as the output of the DQN to determine how many shares to trade. With that approach, to some extent, one may regulate the size of the trades that would ultimately lead to better risk management.

For the reward and action-value designs, Corazza and Sangalli in [134], and Bertoluzzo and Corazza [130] implemented the Sharpe Ratio as the primary agent objective to maximize, which in turn leads to a risk-sensitive agent. We need to emphasize the novel reward design done by Tsantekidis et al. [141]. They shaped the function with what they called "Price Trailing" reward. That price trailing reward can be described as

$$r_t^{\text{Trail}} = 1 - \frac{|p_{t_a} - p_t|}{m p_t} \tag{22}$$

Where p_{t_a} is a controllable price that is assigned by the agent's actions, and m is a margin parameter that determines the distance from the closing price p_t. The price-trail reward is positive when the agent's price is within the margin, while negative when the agent's price is outside the range. Note that Tsantekidis's proposed the final reward as a combination between the profit/loss, price-trailing, and incurred transaction costs. The overall reward function, generally after normalizing each component for consistency, therefore, becomes

$$r_t = \pi_{t-1}\Delta p + r_t^{\text{Trail}} - c|\pi_t - \pi_{t-1}| \tag{23}$$

Where $\Delta p = p_t - p_{t-1}$ is the asset's return at time t. Albeit the authors implemented that reward design on the forex market and showed that it could lead to better performance over convectional profit/loss reward, that novel design can be generalized and applied with different RL paradigms. We see that exploring it under different RL algorithms and financial markets is worth further investigation.

We observed a few works that compare their RL trading system to those simple strategies. Carta et al. [142], Huang [136], and Chen [112], for example, compared their method with Buy&Hold. The more comprehensive comparison we found in the work of Theate and Ernst in [143] where they compared their method with benchmark strategies, for example, Buy&Hold and trend-following moving averages. Eilers et al. [131] compared their NN-based RL with Buy&Hold and what they called "Static" strategy where the trader buys one day before a key event that may impact the asset's prices and sells two days later at the closing price. We also noticed some RL systems being compared with other machine learning-based systems. For example, Carta et al. compared against supervised systems proposed by Sezer [149] and Calvi [150]. Generally speaking, most of the recent advancement in RL algorithms has been investigated under different markets. Nonetheless, the focus is on testing those algorithms on forex and stock financial markets with no empirical literature on trading commodities. Most of the proposed trading systems use the conventional ε-greedy strategy where the agent is only greedy with a probability $1-\varepsilon$; otherwise, it takes completely random actions. To overcome the issue of random exploration, we suggest to follow the method of Huang [136] we discussed earlier. Further, the reward design and the value function, in general, are found to be risk-neutral, where it is represented by the expectation of the achieved trading returns. We note that most of the trading systems performed subtly amid fair trading frictions, which is a positive indication of the usefulness of RL trading systems under realistic market conditions. That is obviously reflected in the evaluation metrics used to assess the system's ability to generate profits.

B. Actor-Based Methods

A summary of all reviewed papers that follow the actor-based RL paradigm is tabulated in Table III. Moody et al. were the first who introduced the notion of RL with actor-based paradigm into QT in their seminal works presented in [64], [167], [168]. Moody in his work represented the environment's states with the price signal past returns, which is a standard financial metric in quantitative finance and its statistical behaviour has been widely investigated in the finance literature [169]–[171]. In addition, Moody developed a Recurrent RL (RRL) structure for the policy where in that case the agent's trading signal at time t is influenced by the one generated at $t-1$. In a sense, when the agent "remembers" its last taken action, then its next decision would influence the trades turnover rate and consequently minimizing transaction costs. The function that is used to derive a policy is a Neural Network (NN) with a single hidden neuron that has a hyperbolic tangent as its activation function. The general policy defined in (10), from Moody's work perspective, thus becomes

$$\pi_t(s_t, \pi_{t-1}; \Theta) = \tanh(w s_t + u\pi_{t-1} + b) \tag{24}$$

Where $w, u, b \in \Theta$. We can see that the expression in (24) implies that agent's action at t is bounded since $\pi_t \in [-1, 1]$. Noteworthy, Moody implements the policy in (24) while the agent is learning only to ensure smooth differentiable policy. During generating trading signals the agent however can only take discrete actions, i.e. $\pi_t \in \{-1, 1\}$. With this, the agent's trading signal would be to long the asset when $\pi_t > 0$, and to short it if $\pi_t < 0$. Moody also endeavored to design subtle utility functions that enhance the agent's risk awareness when trading. To demonstrate this, let us first discuss his general reward structure, and then proceed to confer two of his utility designs that account for financial risk. Moody's reward signal design at time t, considering an immediate reward with $\gamma = 0$, is described by

$$R_t = \pi_{t-1}\Delta p - c|\pi_t - \pi_{t-1}| \tag{25}$$

TABLE III: Summary of Quantitative Trading with Reinforcement Learning - Actor-Based Methods

Reference	RL Algorithm	Policy		Performance Function		Assets	QT Frictions			Evaluation Metrics			
		Deterministic	Stochastic	Risk-Neutral	Risk-Averse		Trans.	Spread	Slippage	Total Returns	Sharpe Ratio	Sortino Ratio	Max. Drawdown
Moody et al. [64]	RRL	*			*	Forex [30m]	*	*		*	*		
Gold [151]	RRL	*			*	Forex [30m]		*		*	*		
Dempster et al. [152]	RRL	*			*	Forex [1m]		*		*			
Bertoluzzo et al. [153]	RRL	*			*	VSM [1d]	*			*			*
Maringer et al. [154]	RRL	*			*	USSM [1d]	*			*	*		
Gorse [155]	RRL		*	*		USI [1d, 1w, 1mo]	*			*			
Zhang et al. [156], [157]	RRL	*			*	USSM [1d]	*			*	*		
Gabrielsson et al. [158]	RRL	*			*	USIF [1m]		*		*	*		
Deng et al. [159]	RRL	*		*		CIF [1/2s]	*	*	*	*	*		
Deng et al. [160]	Fuzzy deep RRL	*		*	*	CIF, COMF [1m]	*	*	*	*	*		
Lu [161]	Deep LSTM	*			*	Forex [30m]				*			
Lei et al. [162]	Deep GRU		*	*		USSM [1d]	*			*	*		
Weng et al. [163]	Deep CNN	*		*		Crypto [30m]				*	*		*
Sattarov et al. [164]	Deep NN		*	*		Crypto [60m]	*			*			
Fengqian et al. [165]	Deep NN		*	*		COMF, CFI [1m]	*		*	*			
Alameer and Alshehri [166]	NN	*			*	USETF [15m]	*			*			*

Notes:
1) Various Stock Markets (VSM); US Stock Market (USSM); US Indexes (USI); US Indexes Futures (USIF); China Indexes Futures (CFI), Commodities Futures (COMF); US Exchage Traded Fund (USETF)
2) s = second; m = minute; d = day; w = week; mo = month

Moody's first utility design is the Sharpe ratio. When applying the definition of Sharpe ratio on the received rewards by the agent, the utility in (11) can be defined as

$$\mathcal{U}_T = \frac{\mu_R}{\sigma_R} \tag{26}$$

Where μ_R, σ_R are the mean and standard deviation of the reward, respectively. In that sense, Sharpe ratio is a symmetric risk measure since it counts for both upside and downside risks. Moody, therefore, strove to develop an asymmetric risk utility that only captures the downside risk, which ultimately penalizes only negative returns. In his work, he used the well-known downside risk measure called *downside deviation* (DD), which was primarily introduced by Sortino et al. in [41] for portfolio management applications. The downside deviation is mathematically represented by taking the root mean square of all negative returns. In the context of QT with RL, it can be calculated as

$$DD_T = \sqrt{\frac{1}{T}\sum_{t=1}^{T} \min\{R_t, 0\}^2} \tag{27}$$

Moody used (27) with the mean of rewards μ_R to have a utility that eventually discourages the agent from executing trades result in high variation in the downside returns. Hence, he defined the following utility

$$\mathcal{U}_T = \frac{\mu_R}{DD_T} \tag{28}$$

When we demand the agent to learn a policy that maximizes the utility in (26) or (28) in an online manner, then we need to know the impact of the most recent observed reward R_t on the marginal utility $\mathcal{U}_t$ defined in (13). This can be attained by computing the gradient $d\mathcal{U}_t/dR_t$. Yet, the computation of this gradient is intractable and this consequently motivated Moody to introduce the notion of *differential Sharpe ratio* and *differential downside deviation*. With that concept, he proposed to compute the term $d\mathcal{U}_t/dR_t$ recursively by approximating it through the use of exponential moving averages for the statistical quantities in (26) or (28), and then expanding them to a first order adaptation rate of η; where η controls the influence of R_t on the marginal utility $\mathcal{U}_t$. We observed that there has been significant advancement in the research of QT following the notion of Moody from different verticals.

We earlier discussed the intricacies associated with the dynamics of financial assets' environments. That as a result motivated researchers to explore various approaches to enhance the sensory part of the trading system, i.e., scouting better environment's representation. Deng et al. [159] introduced the notion of Sparse-Coding [172], [173] into the DRL framework leading to their novel algorithm sparse coding-inspired optimal trading (SCOT). In that sense, the infusion of sparse coding leads to an enhanced representation for the market's environment since it shows robustness in dealing with noisy signals [174], [175]. Moreover, Deng et al. were the first who introduced the notion of deep learning [176] and fuzzy learning [177], [178] in their seminal work [160] to enhance the sensory part of their RL-based trading system. The deep learning part plays the role of detecting patterns in the complex assets' price signals, while the fuzzy part is implemented to mitigate the inherent uncertainties underling the data. In their context, the first layer of the deep network is the fuzzy layer followed by the deep structure. Let us denote the set of parameters related to the deep learning part with θ, and we define the parameters of the fuzzy membership functions as ϑ. Denoting the input of the network as f_t, then the final states used to represent the environment at time t can be given by:

$$s_t = g[v(\mathrm{f}_t; \vartheta); \theta] \tag{29}$$

Where $g(\cdot)$ and $v(\cdot)$ are the deep structure activation and fuzzy membership functions, respectively. With representing the market as in (29), then the used policy for generating trading signals is similar to that of Moody's in (24). The learning process is, however, different under the framework of Deng where now an optimal representation for the market as well as lucrative trading decisions should be carried out. Therefore, during the learning process, all parameters Θ, θ and ϑ should be learned simultaneously. The optimization problem

defined in (11) hence becomes

$$\max_{(\Theta,\theta,\vartheta)} \quad \mathcal{U}_T(R) \tag{30}$$

Since the work of Deng, the infusion of deep learning with RL in the field of QT has noticeably progressed. Lu [161] for example implemented LSTM network architecture to detect short- and long-temporal dependencies within the price signals. Weng et al. [163] used deep CNN to enhance the sensory part of the trading system with the use of XGBoost [179] to optimize the selection process of the network's input. Lei et al. [162] also enhanced the environment representation by extracting features from the price signals using deep structure of Gated Recurrent Unit (GRU) [180]. They used technical indicators and cointegration of price signals [181] to reduce the environment noise. Apart from deep learning, we observed several works that consider simpler, but effective, approaches for the environment's representation. Gold [151] for example extended the policy of Moody to have two hidden units NN and applied it on the Forex market. Furthermore, Dempster et al. [152] used technical indicators in additions to price returns as the environment's states. Zhang et al. [156], [157] further enhanced the process by using genetic algorithm [182] to optimally represent the environment's states by selecting the most representative technical, fundamental [183] and volatility [184] indicators. To extract high level information from the price signals, Gabrielsson et al. [158] utilized the Japanese candlesticks information for the sensory part of the environment in their trading system. Fengqian et al. [165] also used candlesticks to reduce data noise and then implemented clustering techniques, such as K-mean [185], fuzzy c-means [186], and data density online clustering [187] with deep neural network structure to represent the environment.

The other vertical that has been explored by researchers is the policy's design. An interesting design of the policy is observed in the work of Maringer et al. [154], [188] where they introduced the concept of regime-switching DRL. In their proposed design, two trading strategies are learned by two different policies. In a sense, Maringer proposed to follow different trading strategies based on the current market regime. The regime switching is detected by an external-temporal signal q_t that is continuously compared with a threshold b. In the mathematical context, Moody's deterministic policy in (24) is modified as:

$$\pi_t = \pi_t^1 I_{\{q_t > b\}} + \pi_t^2 (1 - I_{\{q_t > b\}}) \tag{31}$$

Where I is the indicator function. They also proposed a smooth policy transition, based on the regime-switching signal, by replacing the the indicator function in (31) with a bounded function $g \in [0, 1]$. The regime-switching signal used is the volatility since a surge in it could indicate a negative autocorrelation in the price returns, see [189], [190]. We see that Maringer's methods indeed adds an additional adaptive dimension to the problem. To enhance the exploratory behaviour of the agent, Gorse [155] extended the work of Moody by considering a stochastic policy rather than the deterministic approach.

The last main dimension that grabs the attention of researchers is the design of the financial utility. For instance, Bertoluzzo et al. [153] investigated a utility that maximizes the upside returns while minimizing the downside risk:

$$\mathcal{U}_T = \frac{\sum_{t=1}^{T} \max\{R_t, 0\}}{\sum_{t=1}^{T} |\min\{R_t, 0\}|} \tag{32}$$

In that sense, the learned policy will be eager to take actions that maximize the positive returns, rather than the average returns as in (26) and (28), while simultaneously avoid decisions that may result in losses. As we discussed earlier, one can develop a further risk-averse agent by using CVaR, see the work of Alameer and Alshehri in [166] where they modeled for the function $\mathcal{U}_T$ as

$$\mathcal{U}_T = -\mathsf{CVaR}_{\zeta}[-R] \tag{33}$$

Where ζ is the risk-aversion parameter, with the special case of $\mathsf{CVaR}_{\zeta}[-R] = \mathbb{E}[-R]$ when $\zeta = 0$. In that sense, the agent becomes aware of trading decisions at particular states where those actions may lead to catastrophic losses. An additional advantage for the use of CVaR is that one can conveniently control the risk-aversion level of the agent by tuning ζ.

We generally observe that the literature related to the design of the trading system's sensory part is quite ample. That includes examining different well-recognized financial data as states while also feeding them to learning systems that can detect complex patterns within these data, such as deep RNN, LSTM, CNN, and GA. Although we see proposals for enhancing the policy's design, we here underscore the importance of further investigating the implementation of strategic stochastic policies to improve the exploratory behavior of the trading agent. The advantage of the actor-based approach in extending a risk-neutral performance function to a risk-averse one through the financial utility function is well-exploited. Despite that the DRL approach has been reconnoitered with several financial instruments, we noticed a limited number of trading systems tested under different financial markets. As for modeling trading frictions, we see that they are adequately addressed, proving the practicality of the proposed models to an acceptable extent. Nevertheless, we encourage to extend the research related to frictional trading to specifically investigate the impacts these frictions could have on the optimization problem. Trading frictions affect the mathematical characteristics of the financial utility and this consequently may alter the optimal solution. Noteworthy, the impact of trading frictions on the optimal portfolio problem is well-investigated in the finance literature see, for example, [3], [191]–[193]. For the performance metrics, we see that most of the proposed systems are evaluated by at least one risk measure. Notably, we see that the most comprehensive comparison is found in the work of Deng et al. [160] where they compare their system's performance with baseline strategies, Moody's original work, and supervised-learning based trading systems, which their trading system outperformed all of those.

C. Actor-Critic Based Methods

We underscore that the QT systems developed using the actor-critic method are noticeably limited. We may attribute that to their inherent complexity where function approximators represent both the actor and critic parts. Again, we need to stress out that most of the algorithms of actor-critic are established under considering the expectation of the returns as the performance index, as we discussed in Section II. In that sense, most of the works we found are developing trading agents with a risk-neutral attitude. We next review the works that considered QT with actor-critic, then articulate our general observations. Table IV summarizes the our survey related to QT under actor-critic approach. We note that various deep actor-critic architectures were assessed in the literature. Li and Shi [196] used on-policy Recurrent DPG (RDPG) [200] with LSTM on the Chinese stock market. Although financial markets are considered as a noisy environment, hence on-policy learning may trigger exploration; we see that noise impacts the agent's performance if not dealt with properly since it

TABLE IV: Summary of Quantitative Trading with Reinforcement Learning - Actor-Critic-Based Methods

Reference	RL Algorithm	Policy		Value Function		Assets	QT Frictions			Evaluation Metrics			
		Deterministic	Stochastic	Risk-Neutral	Risk-Averse		Trans.	Spread	Slippage	Total Returns	Sharpe Ratio	Sortino Ratio	Max. Drawdown
Mabu et al. [194]	GNP		*	*		JSM [5d]				*			
Bekiros [195]	Adaptive Fuzzy	*		*		VSM [1d]	*			*	*		
Li and Shi [196]	On-policy RDPG with LSTM	*		*		CSM	*			*	*		
Ponomarev et al. [197]	A3C with LSTM		*	*		RSM [1m]	*			*	*		
Liu et al. [198]	Off-policy RDPG with GRU	*			*	CSM [1m]	*		*	*	*		*
Briola et al. [199]	PPO with NN		*	*		USSM [tick]		*		*			

Notes:
1) Japan Stock Market (JSM); Various Stock Markets (VSM); China Stock Market (CSM); Russia Stock Market (RSM); US Stock Market (USSM)
2) m = minute; d = day

would ultimately lead to random exploration. Ponomarev [197] assessed the performance of A3C architecture with LSTM on the Russia stock market. Although the reported performance was acceptable, a comparison with other actor-critic methods was not evident, for example, with DDPG and memory replay. Most remarkably, Liu et al. [198] implemented an RDPG with GRU while carrying trades under off-policy learning. Their reported performance outweighs the DDPG with a feedforward neural network in terms of both total returns and Sharpe ratio.

As for the environment's states, Mabu et al. [194] implemented GNP to optimize the environment representation through the evolution of selecting a subset of technical indicators. Bekiros [195] implemented a fuzzy learning system where the inference rules represent the environment based on the expected return and conditional volatility of the financial assets price signal. In [198], the environment is represented by two parts: one to represent the market through technical indicators, and the other to represent the agent's internal state, that is, for example, the accumulative account profit.

Most of the works, if not all, implement a discrete action space, particularly buying or selling the assets. Li and Shi in [196] used a bagging trading execution method where both the actor and critic collaboratively contribute to the final trading decision. If we denote the actor's decision at time t by a_t^a and the critic's by a_t^c, then the final decision is executed as

$$a_t = \begin{cases} -1 & a_t^a + a_t^c < 0 \\ 0 & a_t^a + a_t^c = 0 \\ 1 & a_t^a + a_t^c \geq 0 \end{cases} \tag{34}$$

It is noteworthy that Li followed a similar exploration technique as Wang et al. [146] where they can explore the value of all possible actions other than the actual executed with storing them in the memory buffer to be used while training. Liu et al. in [198] exposed a creative exploration strategy that prevents the agent from random actions when exploring the environment, that is, by applying *imitation learning*. Their concept is developed based on Demonstration Buffer [201] and Behavior Cloning [202]. With Demonstration Buffer, the agent is pre-trained to learn a baseline trading strategy that is called Dual Thrust [203] through an initial replay buffer that stores experiences based on that baseline technique. In that sense, the agent first learns a fundamental trading strategy and enhances it based on new observations through actual interaction. With Behavior Cloning, the performance of the agent is compared with a prophetic expert trader that always takes a long position at low prices and short at high prices. In that sense, an auxiliary loss is introduced based on *Q-Filter* concept [204]

$$J' = -\mathbb{E}\Big[||a_t - \bar{a}_t||^2 \mathbf{1}_{Q(s_t,\bar{a}_t) > Q(s_t,a_t)}\Big] \tag{35}$$

Where $\bar{a}_t$ denotes the expert action and $Q(s_t, \bar{a}_t)$ represents its value. With that, the gradient of the overall function that is used to update the policy is given by

$$\nabla_\Theta \bar{J} = \lambda_1 \nabla_\Theta J + \lambda_2 \nabla_\Theta J' \tag{36}$$

Where $\nabla_\Theta J$ is the policy gradient for the central objective, for example the one in (19), and λ_1, λ_2 are the weights to tune the importance of each loss.

For the reward design, Mabu et al. [194] and Bekiros [195] used the reward as the prediction accuracy of the price action, whether it will move up or down. Yet, Liu et al. [198] implemented the concept of differential Sharpe ratio introduced by Moody [64] in which the agent can learn a risk-sensitive trading strategy.

Liu et al. [198] compared their method with Buy&Hold and DDPG, where they showed that they have a better trading system than those two. Bekiros [195], on the other hand, performed a more comprehensive comparison with other methods, e.g., Buy&Hold, Markov-Switching model [205], and supervised Recurrent NN, which shows superior performance over other models in terms of profit generation. Since most of the works apply their trading systems on stock markets, it would be interesting to investigate the performance of actor-critic based methods on other financial markets, for example, commodities or forex trading.

We believe that the actor-critic approach is the most method yet to be sufficiently assessed for QT applications. Specifically, we see the algorithms TD3 that overcomes optimistic estimation of value functions and SAC that uses maximum entropy in the reward function are worth research under the scope or RL-based QT. These recent advancements in RL may be thoroughly investigated under different financial markets, including stock markets, commodities, and forex. Moreover, most trading agents are trained under a risk-neutral performance index, similar to critic-based methods. Although that may be justifiable since actor-critic methods are fundamentally based on maximizing expectation, that would lead to incompetent agents in terms of risk management during trading. We see that the key advantage of actor-critic methods in dealing with large action space may be considered and assessed, which indeed aid in training agents to size their trading positions properly. We emphasize on the off-policy learning implemented by Liu [198] while executing strategic exploration with *imitation learning*. That exploration vs. exploitation strategy is worth testing on QT applications under the various actor-critic methods. As the literature in critic-methods, modeling market friction was adequately considered when testing the proposed trading systems.

TABLE V: Comparison of the different RL approaches in QT

Reference	RL Algorithm	Financial Market	Evaluation Metrics	Remarks
Moody and Saffell [64]	Actor with RRL vs. Critic Q-learning with NN	US Futures/US Treasury-bill	Total return, Sharpe ratio	Actor approach outperformed the critic method in terms of total profit and Sharpe ratio over a testing period of 25 years while considering market frictions
Tsantekidis et al. [141]	Critic DDQN with LSTM vs. Actor-critic PPO with LSTM	Forex	Total return Sharpe ratio Max. drawdown	The actor-critic PPO significantly outperformed its critic DDQN counterpart over one-year testing period, that is in terms of total profit and Sharpe ratio. At the same they both almost have the same max. drawdown over the same period.
Li et al. [206]	Critic DQN with LSTM vs. Actor-critic A3C with LSTM	US Stocks and Futures China Futures	Annual return Sharpe ratio	Over about one-year testin period, the actor-critic defeated the critic in terms of annual returns and Sharpe ratio, that is under all tested classes of financial assets
Zhang et al. [207]	Critic DQN vs. On-policy actor-critic stochastic policy gradient vs. advantage actor-critic (All with LSTM)	Futures of commodity, equity index, fixed income, and forex	Average return, Sharpe ratio, Sortino ratio, max. drawdown	DQN was found to outperform other methods over a testing period of nine years under transaction costs. It outperformed over all tested financial assets
Wu et al. [208]	Actor deep RNN vs. Critic DQN with GRU vs. Actor-critic DDPG with GRU	US, UK, and Chines Stocks	Total return Sharpe ratio	Over a testing period of 3 years on the fifteen stocks, the actor-critic performed best in nine stocks, while the critic outperformed the others in 6 stocks. That is all under neglecting market frictions.

D. Comparison among Different RL Approaches

We dedicate one subsection to gain insights on how the different RL methods perform under QT applications. The main reason for that is the difficulty in obtaining fair insights about the suitable approach while usually each proposed trading system is compared with counterpart methods using a different dataset of the original work. With that, one cannot emanate a clear conclusion since different data could lead to dissimilar results. That is especially the case under QT since the testing data could have various resolutions, markets, volatility, and price trend depending on the market regime. Nevertheless, it is imperative to gain insights related to the most suitable RL method for QT applications. To this end, in this subsection, we review papers that dedicate part of their work to compare the performance of trading systems with different RL methods over the same dataset. We refer the reader to Table V for a summary of that comparison.

Moody and Safell [64] found that the learning process of their recurrent actor method is more stable than that of the Q-learning, which they found susceptible to the financial market noise. That led to higher turnover rate of the Q-learning algorithm and hence higher drained profits due to market frictions. They also implemented sensitivity analysis to quantify the importance of each input to the actions. Unlike the actor method, they found it difficult to establish a sensitivity framework for Q-learning. Also, the authors argued that the actor method is more computationally efficient since the critic method requires training two networks, the online and the target. Yet, we argue that their experiment was conducted long before the notions of experience replay and double DQN, improving the learning process of Q-learning agents.

Most comprehensive comparison was made by Zhang et al. [207] where they compared all RL methods with several baseline strategies, including returns momentum [66] and MACD signal [209], under various financial assets. Interestingly, they found out that DQN has a low turnover rate which caused it to outperform in most of the assets. Their use of experience replay and LSTM architecture could probably improve the performance observed by Moody and Saffell when their Q-learning trader experienced a high turnover rate. Yet, needless to say, we emphasize that Moody's dataset is different from Zhang's work. Moreover, Zhang et al. noted that the second-best performance, overall, was observed in the trading strategy of the A2C algorithm (synchronous version of the A3C).

The comparison of Tsantekidis et al. [141] is more focused on agent's performance under their proposed trailing reward in (23) or without, rather than comparing the two RL methods, PPO and DDQN. Nevertheless, from their results, one could observe that the learning process of the PPO is remarkably more stable, and that is reflected in the higher Sharpe ratio that indicates lower returns variance during the trading horizon.

In general, we can observe that whenever an actor-critic trading agent is compared with other methods, it either outperforms or performs very well and is close to the best strategy. That is perhaps because these methods have a more stable policy over the learning process than both actor, which usually experiences high variance in their gradient, and the critic that has a policy that is highly sensitive to the value function updates [90]. In addition, they also can perform continuous actions in which the agent can adequately adjust its trading volume. However, that superior performance of actor-critic methods comes at the expense of higher computation costs, especially with deep learning for the actor and critic parts, where four networks are involved in the learning process.

VI. Directions for Future Research

This section proposes directions for future research predominantly driven by the still open challenges in implementing RL on QT applications. As we saw in our discussion of the current literature in sections IV and V, other than the actor approach, most of the critic and actor-critic trading systems rely on maximizing the expectation of the agent's cumulative reward; hence they are risk-insensitive. We see that justifiable since those RL algorithms are originally developed to be insensitive to risk. Nonetheless, most of the time, market practitioners prefer trading strategies that are risk-aware and robust against market uncertainties. Moreover, the non-stationarity associated with the financial market's data may necessitate unique learning paradigms to extend the trading algorithms' practicality when implemented in real-time trading. To this end, we believe that safe RL, distributional RL, adaptive learning, risk-directed exploration, which we will discuss next, are active research areas in the machine learning literature worth probing by researchers developing RL-based trading systems.

A. Safe Reinforcement Learning

Safe RL is an active research area concerned with bringing risk dimensions into the learning process of RL-based agents. The central motive of this research is the nature of many real-life problems that necessitate proper risk management while making decisions, and we see QT as one of those applications. Following the notion of Safe RL discussed amply by Garcia and Fernandez in [210], a risk-aware agent can be developed with several approaches, including risk-sensitive and risk-constrained learning, among others. With the former, one can incorporate risk in the value function, while the latter can introduce risk awareness by adding a constraint to the objective to be optimized. In the following subsections, we discuss several models that consider both approaches with the

different RL paradigms, which we see worth researching in the future when it comes to the development of RL-based trading systems.

1) Risk-Sensitive Agent: The most common risk-sensitive model, generally speaking, is the mean-variance, i.e., the Markowitz approach. In that sense, one endeavor to maximize the difference between the mean and variance of the trading returns. One could solve the following optimization problem when approaching that using actor RL

$$\max_{\Theta} \quad \mathbb{E}[R] - \kappa \mathbb{V}[R] \tag{37}$$

Where $\mathbb{E}[\cdot]$ and $\mathbb{V}[\cdot]$ represent the expectation and variance operators, respectively, and $\kappa < 0$ counts for the risk tolerance. For efficient computation and online learning, one can consider Moody and Saffell [64] technique in finding the marginal change in (37) through recursive computation of the first and second moments using Taylor expansion. Finally, one can update the policy in the ascent direction of the objective gradient using SG methods.

The mean-variance problem can also be approached through critic methods. Consider the following objective

$$\max_{\pi} \quad \mathbb{E}[R] - \kappa \mathbb{V}[R] \tag{38}$$

As proposed by Ritter [211], if we consider the reward r_t as the single time step agent's return from trading a financial asset, then we can reformulate the reward function as

$$r_t^{mv} = r_t - \kappa(r_t - \hat{\mu})^2 \tag{39}$$

Where $\hat{\mu}$ is an estimate for the sample mean. With that, maximizing the expectation of the cumulative reward defined in (3), which a standard critic RL does, is equivalent to maximizing (38) when considering the single-step reward in (39) and a unity discount factor γ. One then can follow tabular or any function approximation algorithms we discussed in Section II to derive the optimal risk-sensitive policy under critic methods.

2) Risk-Constrained Agent: A risk-constrained agent aims to maximize the expectation of the returns while meeting a constraint imposed on a risk measure. We saw that a risk-sensitive agent would be aware of the symmetrical risk, whereas one can develop an agent that cares about asymmetrical risk during the learning process. Under that approach, one can consider the following general problem

$$\begin{aligned} \max \quad & \mathbb{E}[R] \\ \text{s.t.} \quad & \beta(R) \leq \Gamma \end{aligned} \tag{40}$$

Where $\beta(\cdot)$ is a risk measure, and Γ is a strictly positive constant. If one is only concerned about rare events that may incur significant losses, then β can be represented by the Conditional Value-at-Risk [42], [212]; otherwise, one can use the variance of the negative returns as β.

Problem (40) can be solved through actor RL with maximizing the objective with respect to the policy's parameter Θ. With that approach, one can compute the optimal policy through unfolding the constrained objective to an unconstrained version through either Lagrange multipliers [213], [214] or the log-barrier method [215]. Finally, we can recursively estimate the policy parameters with gradient-based learning.

For solving (40) through Q-learning, Borkar [216] introduced an online algorithm along with its convergence proof to an optimal risk-constrained policy under the tabular approach. Furthermore, the risk-constrained problem considering CVaR is also approachable using an actor-critic algorithm. With that, we refer to the seminal work of Chow et al. [217] who proposed an algorithm based on the Lagrange multiplier and proved its convergence to a locally optimal policy.

B. Distributional RL

Distributional RL is about learning the distribution of the value function rather than the single-point estimate of its expectation [218]. Following Bellemare et al. [219], let Z^π denote a random variable that maps state-action pairs over the distribution of the returns when following policy π. If one considers a discrete distribution with N atoms, then we have

$$Z^\pi(s,a) = z_i \quad \text{w.p.} \quad p_i(s,a) \quad \forall i \in [0, N) \tag{41}$$

Bellemare proposed estimating the distribution using a parameter vector $\Theta \in \mathbb{R}^N$ in a DQN architecture, he called it *categorical DQN*, where the output of the network is *softmax* function that represents the probability of each atom, i.e., $p_i(s,a)$. Those probabilities are then updated using Bellman projection with gradient descent while considering the cross-entropy of the Kullback–Leibler divergence as the loss. For risk-insensitive learning, one then can take greedy actions with respect to the empirical expectation of Z^π

$$a = \arg\max \left(\sum_{i=0}^{N-1} z_i p_i(s_{t+1}, a) \right) \tag{42}$$

Even though the learning process does not involve a risk dimension, Bellemare found that distributional learning is more stable than the standard approach. Further, the categorial algorithm remarkably outperformed the standard DQN when applied to games from the Arcade Learning Environment [220]. Nevertheless, we note here that with knowing the probability distribution of Z^π, one can easily extend to distributional risk-sensitive learning where the actions, for example, are taken based on minimizing CVaR, as proposed by Morimura et al. [221] or maximizing the mean-variance.

C. Learning Methods

1) Adaptive Incremental Learning: RL is an online learning system by its nature. Yet, in QT with RL, if one gives up updating the model after training, validation, and testing, the real-time trading system's performance would probably deteriorate over time due to the non-stationary behavior of the markets. Another terminology for non-stationarity in the machine learning literature is the *concept drift* where it refers to the evolvement of the data underlying distribution over time [222]–[224]. There are established learning algorithms in the literature to handle the concept drift within the data. One of those is adaptive incremental learning, where the model can be updated in real-time based on the most recent observation or mini-batch past instances, and that all depend on the computational resources and the nature of the application [225]. We note here that Moody and Saffell [64] and Deng et al. [160] implemented rolling training/validation window to handle the concept drift within financial markets, while Al-Ameer and Alshehri [166] and Huang [136] implemented the incremental learning approach for a continuous model update. We nonetheless encourage researchers to further explore those learning approaches in QT under different RL algorithms since we insight their potency in extending the practicality of trading systems performance in real-time.

2) Risk-Directed Exploration: We discussed earlier that random exploration in QT might cause undesirable losses, so strategizing the agent's exploration process would improve its performance. Gehring and Precup [226] proposed a risk-directed exploration where it depends on the expected absolute deviation of the temporal difference, and they define it as

the *controllability* of a state-action pair. With Q-learning, that controllability measure is define as

$$C^{\pi}(s,a) = -\mathbb{E}[|\delta_t||s_t = s, a_t = a] \tag{43}$$

In that sense, the controllability of the state-action pair is inversely related to the variability of the temporal difference. One then can update the controllability, under tabular method, every time a state-action pair is visited as follows

$$C(s_t,a_t) \leftarrow C(s_t,a_t) - \alpha'\big[|\delta_t| + C(s_t,a_t)\big] \tag{44}$$

Wehre $\alpha' \in (0,1)$ is the learning rate. For the updates following function approximations, one may refer to the original work of Gehring and Precup in [226]. Finally, the risk-oriented Q-learning algorithm can take actions that are greedy to

$$Q(s_t,a_t) + \omega C(s_t,a_t) \tag{45}$$

Where ω is a weighing parameter to emphasize on the variability of the temporal difference when taking actions.

VII. Conclusion

The learning process for an optimal agent's policy, or trading strategy under QT context, can undergo different approaches, i.e., critic-, actor-, and actor-critic-based methods. In this survey, we abundantly reviewed the literature of QT following all those methods, that is, under tabular and function approximation RL with different learning architectures. Under all, generally speaking, we saw that RL is an attractive tool for developing auto-trading systems since with it, one can have a trading system that learns online under its interaction within financial markets. The review outcome shows that actor-critic trading agents can outperform other methods because they have a more stable learning process than both actors and critics, although we see that the more advanced algorithm such as the TD3 and SAC are yet to be assessed under QT applications. However, the superior performance of actor-critic methods comes at higher computation costs, especially under the implementation of deep learning. From our literature review, we observe that most of the critic and actor-critic trading systems rely on the expectation of the agent's cumulative reward; thus, they are risk-neutral. Therefore, we propose to research the application of Safe and distributional RL on QT to introduce risk dimension into the learning process, besides investigating risk-directed exploration techniques such as measuring the controllability of state-action pairs. Finally, we encourage researchers to consider adaptive incremental learning due to its potency in enhancing real-time trading systems' performance.

References

[1] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952. [Online]. Available: http://www.jstor.org/stable/2975974

[2] M. J. Magill and G. M. Constantinides, "Portfolio selection with transactions costs," *Journal of economic theory*, vol. 13, no. 2, pp. 245–263, 1976.

[3] M. H. Davis and A. R. Norman, "Portfolio selection with transaction costs," *Mathematics of operations research*, vol. 15, no. 4, pp. 676–713, 1990.

[4] Y. Amihud and H. Mendelson, "Dealership market: Market-making with inventory," *Journal of financial economics*, vol. 8, no. 1, pp. 31–53, 1980.

[5] D. Bertsimas and A. W. Lo, "Optimal control of execution costs," *Journal of Financial Markets*, vol. 1, no. 1, pp. 1–50, 1998.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: MIT Press, 2015.

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[8] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[9] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[10] R. Neuneier, "Optimal asset allocation using adaptive dynamic programming," *Advances in Neural Information Processing Systems*, pp. 952–958, 1996.

[11] ——, "Enhancing q-learning for optimal asset allocation," in *Advances in neural information processing systems*, 1998, pp. 936–942.

[12] X. Gao and L. Chan, "An algorithm for trading and portfolio management using q-learning and sharpe ratio maximization," in *Proceedings of the international conference on neural information processing*. Citeseer, 2000, pp. 832–837.

[13] C. R. Shelton, "Importance sampling for reinforcement learning with multiple objectives," 2001.

[14] Y. Nevmyvaka, Y. Feng, and M. Kearns, "Reinforcement learning for optimized trade execution," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 673–680.

[15] T. Hens and P. Wöhrmann, "Strategic asset allocation and market timing: a reinforcement learning approach," *Computational Economics*, vol. 29, no. 3, pp. 369–381, 2007.

[16] O. Guéant and I. Manziuk, "Deep reinforcement learning for market making in corporate bonds: beating the curse of dimensionality," *Applied Mathematical Finance*, vol. 26, no. 5, pp. 387–452, 2019.

[17] B. Gašperov and Z. Kostanjčar, "Market making with signals through deep reinforcement learning," *IEEE Access*, vol. 9, pp. 61 611–61 622, 2021.

[18] O. Jin and H. El-Saawy, "Portfolio management using reinforcement learning," *Stanford University*, 2016.

[19] X. Du, J. Zhai, and K. Lv, "Algorithm trading using q-learning and recurrent reinforcement learning," *positions*, vol. 1, p. 1, 2016.

[20] Z. Jiang and J. Liang, "Cryptocurrency portfolio management with deep reinforcement learning," in *2017 Intelligent Systems Conference (IntelliSys)*. IEEE, 2017, pp. 905–913.

[21] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," *arXiv preprint arXiv:1706.10059*, 2017.

[22] P. C. Pendharkar and P. Cusatis, "Trading financial indices with reinforcement learning agents," *Expert Systems With Applications*, vol. 103, pp. 1–13, 2018.

[23] Z. Liang, H. Chen, J. Zhu, K. Jiang, and Y. Li, "Adversarial deep reinforcement learning in portfolio management," *arXiv preprint arXiv:1808.09940*, 2018.

[24] S. Almahdi and S. Y. Yang, "An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown," *Expert Systems with Applications*, vol. 87, pp. 267–279, 2017.

[25] ——, "A constrained portfolio trading system using particle swarm algorithm and recurrent reinforcement learning," *Expert Systems with Applications*, vol. 130, pp. 145–156, 2019.

[26] M. García-Galicia, A. A. Carsteanu, and J. B. Clempner, "Continuous-time reinforcement learning approach for portfolio management with time penalization," *Expert Systems with Applications*, vol. 129, pp. 27–36, 2019.

[27] N. Kanwar, "Deep reinforcement learning-based portfolio management," Ph.D. dissertation, The University of Texas at Arlington, 2019.

[28] H. Park, M. K. Sim, and D. G. Choi, "An intelligent financial portfolio trading strategy using deep q-learning," *Expert Systems with Applications*, vol. 158, p. 113573, 2020.

[29] Z. Xiong, X.-Y. Liu, S. Zhong, H. Yang, and A. Walid, "Practical deep reinforcement learning approach for stock trading," *arXiv preprint arXiv:1811.07522*, 2018.

[30] L. Conegundes and A. C. M. Pereira, "Beating the stock market with a deep reinforcement learning day trading system," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[31] F. Heryanto, B. Handari, and G. Hertono, "Trading financial assets with actor critic using kronecker-factored trust region (acktr)," in *AIP Conference Proceedings*, vol. 2242, no. 1. AIP Publishing LLC, 2020, p. 030001.

[32] H. Yang, X.-Y. Liu, S. Zhong, and A. Walid, "Deep reinforcement learning for automated stock trading: An ensemble strategy," *Available at SSRN*, 2020.

[33] Q. Kang, H. Zhou, and Y. Kang, "An asynchronous advantage actor-critic reinforcement learning method for stock selection and portfolio management," in *Proceedings of the 2nd International Conference on Big Data Research*, 2018, pp. 141–145.

[34] A. Filos, "Reinforcement learning for portfolio management," *arXiv preprint arXiv:1909.09571*, 2019.

[35] Y. Ye, H. Pei, B. Wang, P.-Y. Chen, Y. Zhu, J. Xiao, and B. Li, "Reinforcement-learning based portfolio management with augmented asset movement prediction states," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 1112–1119.

[36] A. Kirilenko, A. S. Kyle, M. Samadi, and T. Tuzun, "The flash crash: High-frequency trading in an electronic market," *The Journal of Finance*, vol. 72, no. 3, pp. 967–998, 2017.

[37] E. J. Elton and M. J. Gruber, "Modern portfolio theory, 1950 to date," *Journal of banking & finance*, vol. 21, no. 11-12, pp. 1743–1759, 1997.

[38] W. F. Sharpe, "Mutual fund performance," *The Journal of business*, vol. 39, no. 1, pp. 119–138, 1966.

[39] ——, "Asset allocation: Management style and performance measurement," *Journal of portfolio Management*, vol. 18, no. 2, pp. 7–19, 1992.

[40] ——, "The sharpe ratio," *Journal of portfolio management*, vol. 21, no. 1, pp. 49–58, 1994.

[41] F. A. Sortino and R. Van Der Meer, "Downside risk," *Journal of portfolio Management*, vol. 17, no. 4, p. 27, 1991.

[42] R. T. Rockafellar, S. Uryasev *et al.*, "Optimization of conditional value-at-risk," *Journal of risk*, vol. 2, pp. 21–42, 2000.

[43] J. D. Schwager, *The new market wizards: Conversations with America's top traders*. John Wiley & Sons, 2012, vol. 95.

[44] C. J. Neely, D. E. Rapach, J. Tu, and G. Zhou, "Forecasting the equity risk premium: the role of technical indicators," *Management science*, vol. 60, no. 7, pp. 1772–1791, 2014.

[45] J. S. Abarbanell and B. J. Bushee, "Abnormal returns to a fundamental analysis strategy," *Accounting Review*, pp. 19–45, 1998.

[46] P. M. Dechow, A. P. Hutton, L. Meulbroek, and R. G. Sloan, "Short-sellers, fundamental analysis, and stock returns," *Journal of financial Economics*, vol. 61, no. 1, pp. 77–106, 2001.

[47] C. Gan, M. Lee, H. H. A. Yong, and J. Zhang, "Macroeconomic variables and stock market interactions: New zealand evidence," *Investment management and financial innovations*, no. 3, Iss. 4, pp. 89–101, 2006.

[48] A. Humpe and P. Macmillan, "Can macroeconomic variables explain long-term stock market movements? a comparison of the us and japan," *Applied financial economics*, vol. 19, no. 2, pp. 111–119, 2009.

[49] S. Nison, *Beyond candlesticks: New Japanese charting techniques revealed*. John Wiley & Sons, 1994, vol. 56.

[50] A. W. Lo, H. Mamaysky, and J. Wang, "Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation," *The journal of finance*, vol. 55, no. 4, pp. 1705–1765, 2000.

[51] M. J. Pring, *Study guide for technical analysis explained*. McGraw-Hill New York, 2002, vol. 5.

[52] R. W. Colby, *The encyclopedia of technical market indicators*. McGraw-Hill, 2003.

[53] T. N. Bulkowski, *Trading classic chart patterns*. John Wiley & Sons, 2003, vol. 176.

[54] J. L. Person, *A complete guide to technical trading tactics: how to profit using pivot points, candlesticks & other indicators*. John Wiley & Sons, 2012, vol. 217.

[55] Y. Li, J. Wu, and H. Bu, "When quantitative trading meets machine learning: A pilot survey," in *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*. IEEE, 2016, pp. 1–6.

[56] O. Ican, T. B. Celik *et al.*, "Stock market prediction performance of neural networks: A literature review," *International Journal of Economics and Finance*, vol. 9, no. 11, pp. 100–108, 2017.

[57] M. Fabbri and G. Moro, "Dow jones trading with deep learning: The unreasonable effectiveness of recurrent neural networks." in *Data*, 2018, pp. 142–153.

[58] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," *IEEE Transactions on neural networks*, vol. 9, no. 6, pp. 1456–1470, 1998.

[59] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 5, pp. 1407–1418, 2018.

[60] A. M. Turing, "Intelligent machinery," 1948.

[61] R. Bellman, "The theory of dynamic programming," Rand corp santa monica ca, Tech. Rep., 1954.

[62] P. Werbos, "Advanced forecasting methods for global crisis warning and models of intelligence," *General System Yearbook*, pp. 25–38, 1977.

[63] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[64] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE transactions on neural Networks*, vol. 12, no. 4, pp. 875–889, 2001.

[65] A. C. Szakmary, Q. Shen, and S. C. Sharma, "Trend-following trading strategies in commodity futures: A re-examination," *Journal of Banking & Finance*, vol. 34, no. 2, pp. 409–426, 2010.

[66] T. J. Moskowitz, Y. H. Ooi, and L. H. Pedersen, "Time series momentum," *Journal of financial economics*, vol. 104, no. 2, pp. 228–250, 2012.

[67] M. H. Miller, J. Muthuswamy, and R. E. Whaley, "Mean reversion of standard & poor's 500 index basis changes: Arbitrage-induced or statistical illusion?" *The Journal of Finance*, vol. 49, no. 2, pp. 479–513, 1994.

[68] T. T.-L. Chong and W.-K. Ng, "Technical analysis and the london stock exchange: testing the macd and rsi rules using the ft30," *Applied Economics Letters*, vol. 15, no. 14, pp. 1111–1114, 2008.

[69] J. O. Katz and D. L. McCORMICK, *The encyclopedia of trading strategies*. McGraw-Hill New York, 2000.

[70] T. G. Fischer, "Reinforcement learning in financial markets-a survey," FAU Discussion Papers in Economics, Tech. Rep., 2018.

[71] Y. Sato, "Model-free reinforcement learning for financial portfolios: a brief survey," *arXiv preprint arXiv:1904.04973*, 2019.

[72] T. L. Meng and M. Khushi, "Reinforcement learning in financial markets," *Data*, vol. 4, no. 3, p. 110, 2019.

[73] L. S. Shapley, "Stochastic games," *Proceedings of the national academy of sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.

[74] F. S. Melo, "Convergence of Q-learning: A simple proof," *Institute Of Systems and Robotics, Tech. Rep*, pp. 1–4, 2001.

[75] P. Dayan and T. J. Sejnowski, "Td (λ) converges with probability 1," *Machine Learning*, vol. 14, no. 3, pp. 295–301, 1994.

[76] G. Tesauro *et al.*, "Temporal difference learning and td-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

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

[78] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[79] L.-J. Lin, *Reinforcement learning for robots using neural networks*. Carnegie Mellon University, 1992.

[80] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[81] H. Hasselt, "Double q-learning," *Advances in neural information processing systems*, vol. 23, pp. 2613–2621, 2010.

[82] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[83] R. J. Williams, "On the use of backpropagation in associative reinforcement learning." in *ICNN*, 1988, pp. 263–270.

[84] ——, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[85] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[86] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[87] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[88] L. Bottou, "Stochastic gradient learning in neural networks," *Proceedings of Neuro-Nımes*, vol. 91, no. 8, p. 12, 1991.

[89] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[90] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation." in *NIPs*, vol. 99. Citeseer, 1999, pp. 1057–1063.

[91] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. PMLR, 2014, pp. 387–395.

[92] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[93] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[94] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[95] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[96] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[97] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[98] S. Mudchanatongsuk, J. A. Primbs, and W. Wong, "Optimal pairs trading: A stochastic control approach," in *2008 American control conference*. IEEE, 2008, pp. 1035–1039.

[99] H. Zhang and Q. Zhang, "Trading a mean-reverting asset: Buy low and sell high," *Automatica*, vol. 44, no. 6, pp. 1511–1518, 2008.

[100] B. R. Barmish and J. A. Primbs, "On a new paradigm for stock trading via a model-free feedback controller," *IEEE Transactions on Automatic Control*, vol. 61, no. 3, pp. 662–676, 2015.

[101] B. R. Barmish, "On performance limits of feedback control-based stock trading strategies," in *Proceedings of the 2011 American Control Conference*. IEEE, 2011, pp. 3874–3879.

[102] J. A. Primbs and B. R. Barmish, "On robustness of simultaneous long-short stock trading control with time-varying price dynamics," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 267–12 272, 2017.

[103] B. R. Barmish, "On trading of equities: A robust control paradigm," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 1621–1626, 2008.

[104] S. Malekpour and B. R. Barmish, "How useful are mean-variance considerations in stock trading via feedback control?" in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 2110–2115.

[105] G. C. Calafiore, "Multi-period portfolio optimization with linear control policies," *Automatica*, vol. 44, no. 10, pp. 2463–2473, 2008.

[106] R. Korn and H. Kraft, "A stochastic control approach to portfolio problems with stochastic interest rates," *SIAM Journal on Control and Optimization*, vol. 40, no. 4, pp. 1250–1269, 2002.

[107] M. A. Dempster, T. W. Payne, Y. Romahi, and G. W. Thompson, "Computational learning techniques for intraday fx trading using popular technical indicators," *IEEE Transactions on neural networks*, vol. 12, no. 4, pp. 744–754, 2001.

[108] Y. Zhu, H. Yang, J. Jiang, and Q. Huang, "An adaptive box-normalization stock index trading strategy based on reinforcement learning," *International Conference on Neural Information Processing*, p. 335–346, 2018.

[109] J. O, J. Lee, J. W. Lee, and B.-T. Zhang, "Adaptive stock trading with dynamic asset allocation using reinforcement learning," *Information Sciences*, vol. 176, pp. 2121–2147, 2006.

[110] S. Y. Yang, Y. Yu, and S. Almahdi, "An investor sentiment reward-based trading system using gaussian inverse reinforcement learning algorithm," *Expert Systems with Applications*, vol. 114, pp. 388–401, 2018.

[111] T. Elder, "Creating algorithmic traders with hierarchical reinforcement learning," Edinburgh, UK, 2008.

[112] Y. Chen, S. Mabu, K. Hirasawa, and J. Hu, "Genetic network programming with sarsa learning and its application to creating stock trading rules," in *2007 IEEE Congress on Evolutionary Computation*. IEEE, 2007, pp. 220–227.

[113] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[114] S. Russell, "Learning agents for uncertain environments," *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 101–103, 1998.

[115] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," *Proceedings of the Seventeenth International Conference on Machine Learning*, vol. 1, pp. 663–670, 2000.

[116] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning." in *IJCAI*, vol. 7, 2007, pp. 2586–2591.

[117] Q. Qiao and P. A. Beling, "Inverse reinforcement learning with gaussian process," in *Proceedings of the 2011 American control conference*. IEEE, 2011, pp. 113–118.

[118] J. Choi and K.-E. Kim, "Inverse reinforcement learning in partially observable environments," *Journal of Machine Learning Research*, vol. 12, pp. 691–730, 2011.

[119] D. S. Brown, Y. Cui, and S. Niekum, "Risk-aware active inverse reinforcement learning," in *Conference on Robot Learning*. PMLR, 2018, pp. 362–372.

[120] G. Dexter, K. Bello, and J. Honorio, "Inverse reinforcement learning in the continuous setting with formal guarantees," *arXiv preprint arXiv:2102.07937*, 2021.

[121] Q. Qiao and X. Lin, "Gaussian processes non-linear inverse reinforcement learning," *IET Cyber-Systems and Robotics*, 2021.

[122] X. Chen, L. Yao, X. Wang, A. Sun, W. Zhang, and Q. Z. Sheng, "Generative adversarial reward learning for generalized behavior tendency inference," *arXiv preprint arXiv:2105.00822*, 2021.

[123] M. Dixon and I. Halperin, "G-learner and girl: Goal based wealth management with reinforcement learning," *arXiv preprint arXiv:2002.10990*, 2020.

[124] S. P. Singh and R. S. Sutton, "Reinforcement learning with replacing eligibility traces," *Machine learning*, vol. 22, no. 1, pp. 123–158, 1996.

[125] A. H. Klopf, *Brain function and adaptive systems: a heterostatic theory*. Air Force Cambridge Research Laboratories, Air Force Systems Command, United …, 1972, no. 133.

[126] S. B. Thrun, "The role of exploration in learning control," 1992.

[127] A. Agostini and E. Celaya, "Reinforcement learning with a gaussian mixture model," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2010, pp. 1–8.

[128] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[129] J. W. Lee, J. Park, O. Jangmin, J. Lee, and E. Hong, "A multiagent approach to Q-learning for daily stock trading," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 37, no. 6, pp. 864–877, 2007.

[130] F. Bertoluzzo and M. Corazza, "Testing different reinforcement learning configurations for financial trading: Introduction and applications," *Procedia Economics and Finance*, vol. 3, pp. 68–77, 2012.

[131] D. Eilers, C. L. Dunis, H.-J. von Mettenheim, and M. H. Breitner, "Intelligent trading of seasonal effects: A decision support algorithm based on reinforcement learning," *Decision support systems*, vol. 64, pp. 100–108, 2014.

[132] J. Cumming, D. D. Alrajeh, and L. Dickens, "An investigation into the use of reinforcement learning techniques within the algorithmic trading domain," *Imperial College London: London, UK*, 2015.

[133] S. J. Bradtke and A. G. Barto, "Linear least-squares algorithms for temporal difference learning," *Machine learning*, vol. 22, no. 1, pp. 33–57, 1996.

[134] M. Corazza and A. Sangalli, "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*, vol. 15, 2015.

[135] J. Carapuço, R. Neves, and N. Horta, "Reinforcement learning applied to forex trading," *Applied Soft Computing*, vol. 73, pp. 783–794, 2018.

[136] C. Y. Huang, "Financial trading as a game: A deep reinforcement learning approach," *arXiv preprint arXiv:1807.02787*, 2018.

[137] X. Gao, "Deep reinforcement learning for time series: playing idealized trading games," *arXiv preprint arXiv:1803.03916*, 2018.

[138] L. Chen and Q. Gao, "Application of deep reinforcement learning on automated stock trading," in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2019, pp. 29–33.

[139] G. Lucarelli and M. Borrotti, "A deep reinforcement learning approach for automated cryptocurrency trading," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2019, pp. 247–258.

[140] S. Sornmayura, "Robust forex trading with deep q network (dqn)," *ABAC Journal*, vol. 39, no. 1, 2019.

[141] A. Tsantekidis, N. Passalis, A.-S. Toufa, K. Saitas-Zarkias, S. Chairistanidis, and A. Tefas, "Price trailing for financial trading using deep reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[142] S. Carta, A. Corriga, A. Ferreira, A. S. Podda, and D. R. Recupero, "A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning," *Applied Intelligence*, vol. 51, no. 2, pp. 889–905, 2021.

[143] T. Théate and D. Ernst, "An application of deep reinforcement learning to algorithmic trading," *Expert Systems with Applications*, vol. 173, p. 114632, 2021.

[144] R. Fischer, *Fibonacci applications and strategies for traders*. John Wiley & Sons, 1993, vol. 4.

[145] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable mdps," in *2015 aaai fall symposium series*, 2015.

[146] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[147] Y. Wang, D. Wang, S. Zhang, Y. Feng, S. Li, and Q. Zhou, "Deep q-trading," *cslt. riit. tsinghua. edu. cn*, 2017.

[148] G. Jeong and H. Y. Kim, "Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning," *Expert Systems with Applications*, vol. 117, pp. 125–138, 2019.

[149] O. B. Sezer and A. M. Ozbayoglu, "Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach," *Applied Soft Computing*, vol. 70, pp. 525–538, 2018.

[150] G. G. Calvi, V. Lucic, and D. P. Mandic, "Support tensor machine for financial forecasting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 8152–8156.

[151] C. Gold, "Fx trading via recurrent reinforcement learning," in *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.* IEEE, 2003, pp. 363–370.

[152] M. A. Dempster and V. Leemans, "An automated fx trading system using adaptive reinforcement learning," *Expert Systems with Applications*, vol. 30, no. 3, pp. 543–552, 2006.

[153] F. Bertoluzzo and M. Corazza, "Making financial trading by recurrent reinforcement learning," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 2007, pp. 619–626.

[154] D. Maringer and T. Ramtohul, "Regime-switching recurrent reinforcement learning for investment decision making," *Computational Management Science*, vol. 9, no. 1, pp. 89–107, 2012.

[155] D. Gorse, "Application of stochastic recurrent reinforcement learning to index trading." ESANN, 2011.

[156] J. Zhang and D. Maringer, "Indicator selection for daily equity trading with recurrent reinforcement learning," in *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, 2013, pp. 1757–1758.

[157] ——, "Using a genetic algorithm to improve recurrent reinforcement learning for equity trading," *Computational Economics*, vol. 47, no. 4, pp. 551–567, 2016.

[158] P. Gabrielsson and U. Johansson, "High-frequency equity index futures trading using recurrent reinforcement learning with candlesticks," in *2015 IEEE Symposium Series on Computational Intelligence*. IEEE, 2015, pp. 734–741.

[159] Y. Deng, Y. Kong, F. Bao, and Q. Dai, "Sparse coding-inspired optimal trading system for hft industry," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 2, pp. 467–475, 2015.

[160] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 653–664, 2016.

[161] D. W. Lu, "Agent inspired trading using recurrent reinforcement learning and lstm neural networks," *arXiv preprint arXiv:1707.07338*, 2017.

[162] K. Lei, B. Zhang, Y. Li, M. Yang, and Y. Shen, "Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading," *Expert Systems with Applications*, vol. 140, p. 112872, 2020.

[163] L. Weng, X. Sun, M. Xia, J. Liu, and Y. Xu, "Portfolio trading system of digital currencies: A deep reinforcement learning with multidimensional attention gating mechanism," *Neurocomputing*, vol. 402, pp. 171–182, 2020.

[164] O. Sattarov, A. Muminov, C. W. Lee, H. K. Kang, R. Oh, J. Ahn, H. J. Oh, and H. S. Jeon, "Recommending cryptocurrency trading points with deep reinforcement learning approach," *Applied Sciences*, vol. 10, no. 4, p. 1506, 2020.

[165] D. Fengqian and L. Chao, "An adaptive financial trading system using deep reinforcement learning with candlestick decomposing features," *IEEE Access*, vol. 8, pp. 63 666–63 678, 2020.

[166] A. AlAmeer and K. Alshehri, "Conditional value-at-risk for quantitative trading: A direct reinforcement learning approach," *arXiv preprint arXiv:2109.14438*, 2021.

[167] J. Moody and L. Wu, "Optimization of trading systems and portfolios," in *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*. IEEE, 1997, pp. 300–307.

[168] J. Moody, L. Wu, Y. Liao, and M. Saffell, "Performance functions and reinforcement learning for trading systems and portfolios," *Journal of Forecasting*, vol. 17, no. 5-6, pp. 441–470, 1998.

[169] D. B. Nelson, "Conditional heteroskedasticity in asset returns: A new approach," *Econometrica: Journal of the Econometric Society*, pp. 347–370, 1991.

[170] R. Cont, "Empirical properties of asset returns: stylized facts and statistical issues," 2001.

[171] P. Carr, H. Geman, D. B. Madan, and M. Yor, "The fine structure of asset returns: An empirical investigation," *The Journal of Business*, vol. 75, no. 2, pp. 305–332, 2002.

[172] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[173] W. E. Vinje and J. L. Gallant, "Sparse coding and decorrelation in primary visual cortex during natural vision," *Science*, vol. 287, no. 5456, pp. 1273–1276, 2000.

[174] T. Bai and Y. Li, "Robust visual tracking using flexible structured sparse representation," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 538–547, 2014.

[175] M. Elad, M. A. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 972–982, 2010.

[176] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[177] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic*. Prentice hall New Jersey, 1995, vol. 4.

[178] N. R. Pal and J. C. Bezdek, "Measuring fuzzy uncertainty," *IEEE Transactions on Fuzzy Systems*, vol. 2, no. 2, pp. 107–118, 1994.

[179] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[180] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[181] J. J. Kremers, N. R. Ericsson, and J. J. Dolado, "The power of cointegration tests," *Oxford bulletin of economics and statistics*, vol. 54, no. 3, pp. 325–348, 1992.

[182] J. H. Holland *et al.*, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[183] B. Graham, *The intelligent investor*. Prabhat Prakashan, 1965.

[184] R. T. Baillie and R. P. DeGennaro, "Stock returns and volatility," *Journal of financial and Quantitative Analysis*, pp. 203–214, 1990.

[185] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[186] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[187] X. Gu, P. P. Angelov, A. M. Ali, W. A. Gruver, and G. Gaydadjiev, "Online evolving fuzzy rule-based prediction model for high frequency trading financial data stream," in *2016 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*. IEEE, 2016, pp. 169–175.

[188] D. Maringer and T. Ramtohul, "Threshold recurrent reinforcement learning model for automated trading," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2010, pp. 212–221.

[189] G. Koutmos, "Feedback trading and the autocorrelation pattern of stock returns: further empirical evidence," *Journal of international money and finance*, vol. 16, no. 4, pp. 625–636, 1997.

[190] E. Sentana and S. Wadhwani, "Feedback traders and stock return autocorrelations: evidence from a century of daily data," *The Economic Journal*, vol. 102, no. 411, pp. 415–425, 1992.

[191] A. J. Morton and S. R. Pliska, "Optimal portfolio management with fixed transaction costs," *Mathematical Finance*, vol. 5, no. 4, pp. 337–356, 1995.

[192] H. Liu and M. Loewenstein, "Optimal portfolio selection with transaction costs and finite horizons," *The Review of Financial Studies*, vol. 15, no. 3, pp. 805–835, 2002.

[193] M. S. Lobo, M. Fazel, and S. Boyd, "Portfolio optimization with linear and fixed transaction costs," *Annals of Operations Research*, vol. 152, no. 1, pp. 341–365, 2007.

[194] S. Mabu, Y. Chen, K. Hirasawa, and J. Hu, "Stock trading rules using genetic network programming with actor-critic," in *2007 IEEE Congress on Evolutionary Computation*. IEEE, 2007, pp. 508–515.

[195] S. D. Bekiros, "Heterogeneous trading strategies with adaptive fuzzy actor–critic reinforcement learning: A behavioral approach," *Journal of Economic Dynamics and Control*, vol. 34, no. 6, pp. 1153–1170, 2010.

[196] J. Li, R. Rao, and J. Shi, "Learning to trade with deep actor critic methods," in *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2. IEEE, 2018, pp. 66–71.

[197] E. Ponomarev, I. V. Oseledets, and A. Cichocki, "Using reinforcement learning in the algorithmic trading problem," *Journal of Communications Technology and Electronics*, vol. 64, no. 12, pp. 1450–1457, 2019.

[198] Y. Liu, Q. Liu, H. Zhao, Z. Pan, and C. Liu, "Adaptive quantitative trading: An imitative deep reinforcement learning approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, 2020, pp. 2128–2135.

[199] A. Briola, J. Turiel, R. Marcaccioli, and T. Aste, "Deep reinforcement learning for active high frequency trading," *arXiv preprint arXiv:2101.07107*, 2021.

[200] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," *arXiv preprint arXiv:1512.04455*, 2015.

[201] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[202] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[203] G. Pruitt and J. R. Hill, *Building Winning Trading Systems with Tradestation,+ Website*. John Wiley & Sons, 2012, vol. 542.

[204] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[205] J. D. Hamilton, "A new approach to the economic analysis of nonstationary time series and the business cycle," *Econometrica: Journal of the econometric society*, pp. 357–384, 1989.

[206] Y. Li, W. Zheng, and Z. Zheng, "Deep robust reinforcement learning for practical algorithmic trading," *IEEE Access*, vol. 7, pp. 108 014–108 022, 2019.

[207] Z. Zhang, S. Zohren, and S. Roberts, "Deep reinforcement learning for trading," *The Journal of Financial Data Science*, vol. 2, no. 2, pp. 25–40, 2020.

[208] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, and H. Fujita, "Adaptive stock trading strategies with deep reinforcement learning methods," *Information Sciences*, vol. 538, pp. 142–158, 2020.

[209] J. Baz, N. Granger, C. R. Harvey, N. Le Roux, and S. Rattray, "Dissecting investment strategies in the cross section and time series," *Available at SSRN 2695101*, 2015.

[210] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[211] G. Ritter, "Machine learning for trading," *Available at SSRN 3015609*, 2017.

[212] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath, "Coherent measures of risk," *Mathematical finance*, vol. 9, no. 3, pp. 203–228, 1999.

[213] R. Bellman, "Dynamic programming and lagrange multipliers," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 42, no. 10, p. 767, 1956.

[214] R. T. Rockafellar, "Lagrange multipliers and optimality," *SIAM review*, vol. 35, no. 2, pp. 183–238, 1993.

[215] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[216] V. Borkar and R. Jain, "Risk-constrained markov decision processes," *IEEE Transactions on Automatic Control*, vol. 59, no. 9, pp. 2574–2579, 2014.

[217] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6070–6120, 2017.

[218] R. Dearden, N. Friedman, and S. Russell, "Bayesian q-learning," in *Aaai/iaai*, 1998, pp. 761–768.

[219] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 449–458.
[220] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.
[221] T. Morimura, M. Sugiyama, H. Kashima, H. Hachiya, and T. Tanaka, "Nonparametric return distribution approximation for reinforcement learning," in *ICML*, 2010.
[222] J. C. Schlimmer and R. H. Granger, "Incremental learning from noisy data," *Machine learning*, vol. 1, no. 3, pp. 317–354, 1986.
[223] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.
[224] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.
[225] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.
[226] C. Gehring and D. Precup, "Smart exploration in reinforcement learning using absolute temporal difference errors," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 1037–1044.