SIT 746 Complete Literature Review 2.1D

Nathan Peter Notaras

Student ID: s224824085

Deakin University

August 11, 2025

Abstract

This task involves drafting a complete literature review based on the research in sit 723 and sit 746.

1 Introduction

In recent years, deep reinforcement learning (DRL) has garnered significant attention since the introduction of the Deep Q-Network (DQN) by Mnih et al. (2013), which marked a turning point in the integration of deep learning with reinforcement learning. Although early applications of DRL were predominantly demonstrated in gaming environments, a growing body of literature has extended its use to complex real-world domains, notably in financial stock trading. This literature review focuses specifically on DRL applications in stock trading.

2 Overview of RL and DQNs

Before discussing the application of Deep Q-Networks (DQNs) in stock trading, it is essential to understand the fundamentals of reinforcement learning (RL) and the operation of DQNs. In RL, an agent interacts with an environment—a comprehensive space that includes all factors the agent can observe and influence. The environment is characterized by its state space, which in stock trading typically comprises observable data such as stock prices and possibly additional features (e.g., technical indicators derived from the price data). Based on its current state, the agent selects an action from a predefined action space. For stock trading applications, this action space is commonly discrete, with allowable actions being "buy," "hold," or "sell."

After performing an action, the agent receives feedback in the form of a reward, which is defined by a reward function; in trading, this is often the profit return on a portfolio given by $\frac{P_t - P_{t-1}}{P_{t-1}}$ where P_t is the portfolio value at time t. The primary objective of the RL agent is to learn a policy—an optimal mapping from states to actions—that maximizes the cumulative reward over time.

The vanilla DQN (Mnih et al., 2013) implement a value-based approach to policy learning. Instead of directly modeling the policy, DQNs estimate the action-value function (Q-function), which represents the expected cumulative reward for taking a particular action in a given state. At each time step, the DQN evaluates the Q-values of all possible actions and selects the action with the highest estimated value.

Several key components and hyperparameters are critical to the operation of DQNs:

- **Replay Buffer:** This is a memory module that stores past experiences, each typically represented as a tuple of (state, action, reward, next state). By sampling mini-batches of experiences from the replay buffer during training, the DQN mitigates correlations between consecutive samples and improves the stability and efficiency of the learning process.

- **Epsilon-Greedy Policy:** To balance exploration and exploitation, DQNs commonly employ an epsilon-greedy strategy. With probability ϵ, the agent selects a random action (exploration), and with probability $1 - \epsilon$, it selects the action that maximizes the current Q-value estimate (exploitation). Epsilon is typically decayed over time to reduce exploration as the agent's knowledge of the environment improves.
- **Key Hyperparameters:**
  - **Learning Rate:** Determines the step size at which the network's weights are updated during training.
  - **Discount Factor (γ):** Reflects the importance of future rewards relative to immediate rewards; a higher γ places more emphasis on long-term returns. For a stock trading agent using daily prices, higher gamma values are preferred.
  - **Batch Size:** Specifies the number of experience samples drawn from the replay buffer for each training update.
  - **Target Network Update Frequency:** In many DQN implementations, a separate target network is maintained to provide stable Q-value estimates; the target network's weights are periodically updated with the values from the main network.

By integrating these components, DQNs iteratively improve their Q-value estimates, thereby refining the underlying policy. This value-based approach enables the agent to approximate the optimal action-selection strategy through trial and error, ultimately learning to navigate the environment effectively, whether in gaming scenarios or in the more complex domain of stock trading.

3 Improvements to DQNs

Given the widespread popularity of DQNs in stock trading (Aken et al., 2023), it is imperative to understand the principal research contributions and various extensions to the DQN framework. Hessel et al. (2017) introduce the Rainbow DQN, which synthesizes multiple enhancements into a unified algorithm. This integration encompasses double DQN (DDQN) (van Hasselt et al., 2016), dueling network architectures (Wang et al., 2016), C51 distributional reinforcement learning (Bellemare et al., 2017), prioritized experience replay (PER) (Schaul et al., 2016), multi-step learning (Sutton & Barto, 1998), and noisy networks (Fortunato et al., 2018).

Double DQN addresses the overestimation bias by decoupling the processes of action selection and evaluation, dueling network architectures improve learning efficiency by separately estimating the state value and the advantage of each action, and distributional reinforcement learning captures the full probability distribution of returns, providing richer feedback than a simple expectation.

Schaul et al. (2016) introduced PER to refine the selection process of replay experiences. In the vanilla DQN, transitions are selected randomly, whereas PER prioritizes transitions with the highest temporal difference (TD) error, enabling the agent to focus on the most informative experiences. Similarly, multi-step learning aggregates rewards over multiple time steps, beyond the single step used in the vanilla DQN, which facilitates faster and more stable learning by providing a more comprehensive feedback signal on sequences of actions. Distributional reinforcement learning aims to approximate the distribution of returns instead of the expected return as in traditional DQNs.

Hessel et al. (2017) suggests that integrating these DQN extensions is crucial for achieving improved performance in reinforcement learning applications. Through rigorous ablation studies, they demonstrate that the combination of all extensions, except for dueling networks or Double DQN, yields superior performance relative to any individual extension. Their findings reveal that the removal of multi-step learning, PER, or C51 distributional results in a significant decline in performance, thereby indicating the essential nature of these three components.

4 State Space Baseline

A common feature added to the state space in reinforcement learning for stock trading are technical indicators. Technical indicators are derived from various combinations of stock price data, such as closing, opening, high, and low prices, and provide unique insights into the dynamics of the price time series, including momentum and trend strength. Many studies generate multiple technical indicator time series and integrate them into the state space to enrich the information available for analysis and forecasting.

Li et al. (2022) conducted ablation experiments on Google stock by evaluating three distinct data configurations: (i) stock data augmented with technical indicators, (ii) stock data represented through candlestick chart images, and (iii) the combined use of both data sources. Their findings indicate that the fusion of technical indicator data with candlestick chart information outperformed the other configurations, demonstrating that a hybrid representation enhances the predictive capabilities of reinforcement learning models.

Pourahmadi et al. (2024) compared two configurations of input data for their RL trading model: one using only raw price data (open, close, and volume) and another augmented with three widely used technical analysis indicators, RSI, ROC, and OBV. Evaluated over 34 different periods using cross-validation, the model incorporating technical indicators consistently achieved higher average returns than the model based solely on price data.

Yasin & Gill (2024) extended this line of inquiry by augmenting the input state of their DQN model with a comprehensive set of 20 technical indicators, including SMA, OBV, RSI, among others, to capture a richer and more nuanced view of market dynamics. Their study compared this enhanced configuration against a baseline that relied solely on raw price data (open, close, high, low, and volume).

Shi et al. (2021) address the potential issue of multicollinearity when incorporating an excessive number of technical indicators. To mitigate this, they opted to use candlestick chart information, specifically the open, close, high, low, and volume data, transformed into log returns as inputs. Their results, using a DDQN architecture with convolutional neural network (CNN) layers, showed significant performance improvements over baseline models, including support vector machines (SVM), long short-term memory (LSTM) networks, and conventional buy-and-hold strategies.

Although these studies demonstrate that adding technical indicators can improve trading performance, few offer a principled rationale for selecting one set of indicators over another and the sheer variety of choices makes it impossible to identify an optimal combination. Moreover, introducing too many often highly correlated indicators risks multicollinearity and overfitting. This underscores the need for a clear standard baseline representation in deep RL stock trading.

Empirically, these studies have shown that stock data (open, high, low, close and volume) consistently capture core market dynamics, making them an ideal foundation for the state space. Building on this candlestick based baseline, we now turn to feature engineering strategies and architectural considerations that can further enhance agent performance.

5 Feature engineering

Feature engineering refers to the extraction and transformation of raw data into informative inputs for a learning algorithm. In deep reinforcement learning this means processing the state space data beyond simple normalization before it enters the network. The aforementioned studies apply only min max scaling or standardization to raw stock time series and then feed it directly into the model. Since financial time series are notoriously noisy, there is a clear need for techniques that can remove spurious fluctuations and reveal the underlying signal.

Dastgerdi and Mercorelli (2022) compare two denoising approaches, Kalman filtering and wavelet transforms, against an unfiltered baseline. They apply each method to five major stock indices and

then train an LSTM for price prediction. Using four evaluation metrics (MAPE, RMSE, R squared, and Theil's U), they show that both denoising techniques consistently improve forecast accuracy, with Kalman filtering outperforming wavelet transforms in 60 percent of cases.

Yashaswi (2021) builds on this idea by combining Kalman filtering with latent feature extraction. After denoising open, high, low, and close data using a Kalman filter, they pass the smoothed series through an autoencoder, a zoomSVD module, and a Boltzmann machine. By avoiding a high dimensional space of technical indicators, they mitigate the curse of dimensionality. Their experimental results for deep reinforcement learning based portfolio optimization demonstrate that this two step filtering and feature learning pipeline outperforms all benchmark methods.

Together, these studies underscore the value of denoising in financial time series, and in particular the effectiveness of the Kalman filter. Yet, in the realm of deep Q networks for stock trading, the impact of integrating a Kalman filter into the state-preprocessing stage has not been explored.

6 Architecture Design

When utilising deep neural networks with time series data in DRL these two specialised architectures appear often: long short term memory (LSTM) networks and convolutional neural networks (CNN). LSTM networks excel at capturing temporal dependencies, whereas CNN can identify multi-dimensional patterns and exploit any spatial structure embedded in the input.

Cui et al. (2023) note that, although LSTM is well suited to sequential data, stock market series also exhibit spatial relationships among features. To address this, they propose a multi scale CNN feature extractor coupled with a double deep Q network (DDQN) agent. Their MS CNN+DDQN architecture stacks multiple convolutional layers at varying scales and outperforms all benchmarks on Apple, General Electric and Dow Jones Industrial Average data.

Yashaswi (2021) corroborates the superior efficacy of CNN architectures in financial deep reinforcement learning (DRL). In a portfolio optimization task, they compare a CNN model against both an LSTM network and a standard deep neural network (DNN). The CNN consistently delivers higher returns and greater stability than either alternative.

Lin et al. (2025) advance this paradigm by embedding a dual layer CNN encoder within a deep deterministic policy gradient (DDPG) framework enhanced by quantum price level indicators. Their system achieves substantially higher returns and Sharpe ratios than all reference models and converges more rapidly during training, demonstrating both superior performance and improved sample efficiency.

Together, these studies indicate that CNN architectures offer the optimal balance of representational power and training capacity for financial DRL tasks. As Cui et al. observe, "In DRL, when the feature extraction network is complex, it often makes the entire model difficult to train. However, deeper neural network models can better extract high level data feature information." This insight motivates the design of CNN architectures that are sufficiently deep to capture complex state spaces yet tractable enough for stable learning.

7 Conclusion

The literature on deep Q-networks in stock trading makes clear that modern DQN variants offer significant gains over the vanilla algorithm. Rainbow DQN's integration of double DQN, dueling networks, C51 distributional learning, prioritized experience replay, multi-step targets, and noisy layers delivers markedly more stable, data-efficient learning. Ablation studies confirm that multi-step learning, PER and C51 distributional learning (MPC) are indispensable for strong performance.

Despite the prevalence of technical indicators in state representations, the field still lacks a

consensus on which features to include, and excessive, correlated inputs invite multicollinearity and overfitting. Empirical results show that the raw candlestick quintet, open, high, low, close, and volume (OHLCV), captures the essential dynamics of equity time series (Shi et al., 2021). This candlestick-chart baseline provides a clear, robust foundation for any deep-RL trading agent.

On the feature-engineering front, denoising remains underutilized: Kalman filtering substantially improves forecast accuracy by suppressing noise, yet no study has woven Kalman-filtered OHLCV data into a DQN or DDQN trading framework. Incorporating such a filter before deep Q-learning could reveal richer, more stable state signals and enhance policy learning.

When it comes to network architecture, convolutional encoders consistently outshine LSTM or vanilla feed-forward models in financial DRL. The previous studies convey how CNNs deliver higher returns, tighter risk control, and faster convergence by capturing both temporal fluctuations and spatial correlations in market data.

Together, these gaps suggest a clear path forward: we select Cui et al.'s candle-based MS-CNN + DDQN framework as our baseline because it not only uses the standardized OHLCV state space and a CNN encoder tuned for temporal and spatial patterns, but also tackles stock trading directly with DQN variants, matching our research focus. We simplify their overcomplex multi-scale extractor and augment it with MPC and a pre-processing Kalman denoiser. By implementing only the three most impactful Rainbow components (multi-step targets, prioritized replay, and C51 distributional learning), we strike a balance between algorithmic simplicity and peak performance. This combined approach leverages the most powerful DQN extensions, establishes a standardized state space, and applies advanced feature engineering to create a more robust, efficient trading agent.

References

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning [arXiv preprint arXiv:1312.5602]. https://doi.org/10.48550/arXiv.1312.5602

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay [arXiv preprint arXiv:1511.05952]. https://doi.org/10.48550/arXiv.1511.05952

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2017). Rainbow: Combining improvements in deep reinforcement learning [arXiv preprint arXiv:1710.02298]. https://doi.org/10.48550/arXiv.1710.02298

van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning [arXiv preprint arXiv:1509.06461]. https://doi.org/10.48550/arXiv.1509.06461

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2016). Dueling network architectures for deep reinforcement learning [arXiv preprint arXiv:1511.06581]. https://doi.org/10.48550/arXiv.1511.06581

Fortunato, M., Gheshlaghi Azar, M., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., & Legg, S. (2018). Noisy networks for exploration [arXiv preprint arXiv:1706.10295]. https://doi.org/10.48550/arXiv.1706.10295

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A distributional perspective on reinforcement learning [arXiv preprint arXiv:1707.06887]. https://doi.org/10.48550/arXiv.1707.06887

R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," in IEEE Transactions on Neural Networks, vol. 9, no. 5, pp. 1054-1054, Sept. 1998, doi: 10.1109/TNN.1998.712192.

Li, Y., Liu, P., & Wang, Z. (2022). Stock trading strategies based on deep reinforcement learning. Scientific Programming, 2022, 1–15. https://doi.org/10.1155/2022/4698656

Aken, J., Liang, D., Lin, Z., & Wang, C. (2023). The application of deep reinforcement learning in stock trading models. In Proceedings of the 7th International Conference on Economic Management and Green Development. https://doi.org/10.54254/2754-1169/39/20231973

Shi, Y., Li, W., Zhu, L., Guo, K., & Cambria, E. (2021). Stock trading rule discovery with double deep Q-network. Applied Soft Computing, 107, 107320. https://doi.org/10.1016/j.asoc.2021.107320

Yasin, A. S., & Gill, P. S. (2024). Reinforcement learning framework for quantitative trading. arXiv preprint arXiv:2411.07585. https://doi.org/10.48550/arXiv.2411.07585

Pourahmadi, Z., Fareed, D., & Mirzaei, H. R. (2024). A novel stock trading model based on reinforcement learning and technical analysis. Annals of Data Science, 11, 1653–1674. https://doi.org/10.1007/s40745-023-00469-1

Karimi Dastgerdi, A., & Mercorelli, P. (2022). Investigating the effect of noise elimination on LSTM models for financial markets prediction using Kalman filter and wavelet transform. WSEAS Transactions on Business and Economics, 19, Article 39. https://doi.org/10.37394/23207.2022.19.39

Yashaswi, K. (2021). Deep reinforcement learning for portfolio optimization using latent feature state space (LFSS) module [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2102.06233

Cui, K., Hao, R., Huang, Y., Li, J., & Song, Y. (2023). A novel convolutional neural networks for stock trading based on DDQN algorithm. IEEE Access, 11, 1–1. https://doi.org/10.1109/ACCESS.2023.3259424

Lin, R., Xing, Z., Ma, M., & Lee, R. S. T. (2023). Dynamic portfolio optimization via augmented DDPG with quantum price levels-based trading strategy. In Proceedings of the 2023 International Joint Conference on Neural Networks (IJCNN) (pp. 1–8). IEEE. https://doi.org/10.1109/IJCNN54540.2023.10191785