Reinforcement Learning for Market Making: Leveraging Behavioral Cloning and Dynamic Weights for Optimal Trading Strategies

Candidate Number: 26678

Department of Statistics 2023-24

Submitted for the Master of Science, London School of Economics, University of London

Abstract

Market making plays a crucial role in financial markets by providing liquidity and facilitating smoother trading activities. Traditional market-making strategies often rely on static rules and heuristics, which can struggle to adapt to dynamic market conditions. This research explores the application of Reinforcement Learning (RL) to develop adaptive market-making strategies that balance profitability with risk management.

We present a study of various RL models, including Deep Q-Networks (DQN) and Advantage Actor-Critic (A2C), enhanced with Behavioral Cloning (BC) and Dynamic Weights (DW). BC improves the stability of trading strategies by reducing inventory fluctuations and mitigating the impact of volatile market conditions. DW further refines these strategies by dynamically balancing learning from policy optimization and expert behavior, leading to more consistent profitability.

Our experiments, conducted in both historical real market environments and simulated settings, demonstrate that RL agents significantly outperform traditional benchmark models in terms of profitability, stability, and inventory management. Notably, the DQN_BC_DW and A2C_BC_DW models exhibit superior performance, achieving high Sharpe Ratios and maintaining balanced inventories. These findings validate the effectiveness of integrating BC and DW into RL frameworks for market making.

This study underscores the potential of advanced RL techniques in developing robust and adaptive market-making strategies. Future work will explore incorporating market impact, multi-agent systems, and real-time implementation to enhance the applicability of these models in real-world trading scenarios.

Contents

1 Introduction **1**

2 Related Work **3**
- 2.1 Traditional Market Making Approaches . . . 3
- 2.2 Market Making via Reinforcement Learning . . . 4
- 2.3 Behavioral Cloning in Reinforcement Learning . . . 5

3 Preliminaries **7**
- 3.1 Limit order book . . . 7
- 3.2 State Space . . . 8
- 3.3 Actions . . . 9
- 3.4 Reward Function . . . 9

4 Methodology **12**
- 4.1 Deep Q-Network (DQN) . . . 12
- 4.2 Advantage Actor-Critic (A2C) . . . 13
- 4.3 Imitative Learning by Behavior Cloning . . . 14

5 Experiment **18**
- 5.1 Market Assumptions . . . 18
- 5.2 Data and Environment Setup . . . 19
- 5.3 Model Settings . . . 22
- 5.4 Experiment Settings . . . 23
- 5.5 Evaluation Criteria . . . 25

6 Results Interpretation **27**
- 6.1 Comparison of DQN, DQN_BC, and DQN_BC_DW Training Processes . . . 27
- 6.2 Comparison of A2C, A2C_BC, and A2C_BC_DW Training Processes . . . 31
- 6.3 Performance Comparison . . . 37

7 Conclusion **39**

1. Introduction

The financial markets are inherently dynamic and complex, characterized by continuous interactions between buyers and sellers. In this environment, market makers play a crucial role by providing liquidity, which is essential for ensuring smoother and more efficient trading. Market making involves continuously quoting bid and ask prices at which a market maker is willing to buy and sell assets, respectively. This activity allows other market participants to execute trades without causing significant price fluctuations.

However, the role of the market maker is not without challenges. One of the primary objectives of market making is to achieve maximum profit while managing the risk associated with holding inventory. Inventory risk arises from the potential depreciation in the value of assets held by the market maker due to adverse market movements. As highlighted by Avellaneda & Stoikov (2008), the balance between providing liquidity and managing inventory risk is delicate, requiring the market maker to strategically adjust their bid and ask prices in response to changing market conditions.

Traditional market making strategies often rely on predefined rules and heuristics, which may not adapt well to these rapidly changing market environments. This limitation underscores the need for more adaptive approaches that can dynamically respond to market conditions.

Reinforcement Learning (RL) has emerged as a powerful tool for developing adaptive trading strategies, including market making. RL algorithms enable agents to learn optimal policies through interactions with the environment, balancing exploration and exploitation to maximize long-term rewards. Recent advancements in deep learning have further enhanced the capabilities of RL, leading to the development of Deep Reinforcement Learning (DRL) techniques that can handle high-dimensional state and action spaces.

This study explores the application of RL to the market making problem, leveraging both historical and simulated market data to train and evaluate our models. We aim to develop trading agents that can dynamically adapt to real-time market conditions, making informed decisions that optimize profitability while managing inventory risk.

Additionally, we incorporate behavioral cloning to improve training efficiency and performance. Behavioral cloning involves training an agent to imitate the actions of a proficient expert, providing a strong initial policy that can be fine-tuned using RL. Within this framework, we introduce a novel method called dynamic weights, which dynamically adjusts the importance of expert demonstrations versus the reinforcement learning process during training. This method enhances the flexibility and efficiency of the training process, leading to superior performance.

The key contributions of this study include:

- Development of a market making framework using Reinforcement Learning (RL) techniques, with specific enhancements to the reward function. Two novel methods were proposed to improve the alignment of the reward function with the market making objective: a decay factor for unrealized PnL and an inventory penalty term. These enhancements make the reward function more robust and reflective of the true risks and rewards in market making.

- Development of a simulated exchange environment grounded in the Glosten-Milgrom model (Glosten & Milgrom, 1985), which effectively captures the dynamics of order flow and market microstructure.Additionally, historical data was employed by reconstructing the exchange using historical Limit Order Book (LOB) data. This dual approach ensures that the models are both theoretically sound and empirically validated, enabling robust performance in real-world market conditions.

- Introduction of a novel method within behavioral cloning called dynamic weights. This method dynamically adjusts the importance of expert demonstrations versus the reinforcement learning process during training. By adapting the weights based on the learning progress, the agent benefits from a more flexible and efficient training process, ultimately leading to better performance.

Through this research, we aim to demonstrate the potential of RL in developing adaptive and robust market-making strategies that can effectively navigate the complexities of financial markets. Additionally, by integrating novel methods such as Behavioral Cloning (BC) and Dynamic Weights (DW), we highlight how these enhancements significantly improve the stability and performance of RL agents, offering superior adaptability and profitability in varying market conditions.

2. Related Work

2.1 Traditional Market Making Approaches

Traditional market making has often been explored through analytical models, with one of the most influential being the Glosten-Milgrom (GM) model. This model analyzes how market makers set prices in the presence of informed and uninformed traders. The GM model posits that market makers adjust their bid and ask prices based on the information asymmetry between themselves and traders, which directly influences the spread. The model is grounded in the idea of adverse selection, where the risk of trading with an informed trader justifies the spread between the bid and ask prices.

In their foundational work, Glosten & Milgrom (1985) introduced the GM model, where the market maker does not have information about the true value of the asset but sets prices based on the order flow, reflecting the aggregate information of the market participants. This approach allows market makers to manage the risk of trading with better-informed agents by widening the spread when the risk is high and narrowing it when the risk is low. The GM model thus provides a framework for understanding the formation of spreads and the dynamics of inventory management by market makers.

Expanding on the GM model, Das (2005) developed a model of a learning market-maker. This extended model allows the market-maker to track the changing true value of a stock in settings with informed traders (with noisy signals) and liquidity traders, setting bid and ask prices based on its estimate of the true value. Through empirical evaluation in simulated markets, the study demonstrated the effectiveness of this approach and explored the impact of different market parameters on price processes, such as the relationship between volatility and the average spread.

Further refining the GM model, Kühn & Riedel (2012) analyzed the price-setting problem of market makers under risk neutrality and perfect competition, emphasizing the endogenous nature of the information filtration process. In their model, the true value of the asset is represented as a Markov process, observed by customers with noise at Poisson times. This rigorous mathematical approach provided deeper insights into how market makers set prices when the filtration process is endogenous, driven by the bid and ask prices they quote.

These foundational models laid the groundwork for understanding the dynamics of market making in the presence of information asymmetry, forming a critical basis for subsequent research, including the application of reinforcement learning techniques to market making.

However, while these model-based methods provide valuable theoretical insights, they often struggle to perfectly fit the complexities and nuances of real market conditions, where assumptions such as perfect rationality or fixed market dynamics may not hold. This limitation underscores the need for more adaptive and data-driven approaches, like reinforcement learning, which can better capture and respond to the ever-changing nature of financial markets.

2.2 Market Making via Reinforcement Learning

Previous research on the application of reinforcement learning to market making can be broadly categorized based on the experimental market settings and the type of algorithms employed, whether they are deep learning-based or rely on traditional reinforcement learning techniques.

For market settings, there are two primary approaches: simulation-based market environments and historical Limit Order Book(LOB) based environments.

In simulation-based market environments, an information-based approach is typically used to simulate market conditions. The primary advantage of this approach is the controlled environment it provides, which allows for clear observation of algorithm behavior and performance without the interference of external market noise. However, the downside is that these models may not fully capture the complexities and unpredictability of real-world markets.

Chan & Shelton (2001) introduced one of the first model-free RL approaches to optimal market making. In their framework, market makers adjust bid and ask prices based on their inventory levels and prevailing market conditions, such as order imbalances and overall market quality. This strategy is rooted in the Glosten–Milgrom information-based model, previously discussed, which includes three types of market participants: a monopolistic market maker, informed traders, and uninformed traders. This approach enables the market maker to navigate the stochastic processes representing the true values of underlying assets, which are unknown to them.

Building on this foundational work, Kim et al. (2002) advanced the field by integrating order flow dynamics, modeled through an input-output hidden Markov model, with the dynamics of traditional order books to create a more complex model. They implemented an RL algorithm that utilizes likelihood ratios to operate effectively in a partially observable market environment. These efforts mark significant early steps towards developing sophisticated automated trading systems capable of performing in real-world market conditions.

For historical Limit Order Book market environment, by reconstructing market scenarios from historical limit order book data, these models provide a more realistic environment that includes actual market reactions and participant behaviors. A significant limitation is the neglect of the order impact. They cannot account for the market impact of the agent's actions.

In their landmark work, Lim & Gorse (2018), which they claim to be "the first practical application of RL to optimal market making," Lim and Gorse utilized a tabular Q-learning-based approach with the constant absolute risk aversion (CARA) utility. This study demonstrated the efficacy of the RL approach in surpassing traditional analytical adaptive strategies (AS) and the zero tick offset benchmark in terms of cumulative profit and inventory metrics.

Expanding upon these foundational insights, Spooner et al. (2018) further explored the capabilities of RL in market making through the development of an agent using temporal-difference RL. They experimented with various state representations and reward function formulations, integrating the most effective components into a single agent that exhibited superior risk-adjusted performance. Notably, their best-performing solution incorporated

a linear combination of tile codings as a value function approximator and an asymmetrically dampened profit and loss (PnL) function as the reward mechanism.

Zhong et al. (2020),who utilized Q-learning coupled with state aggregation to derive an MM strategy. This approach was implemented via a simple lookup table, showcasing a straightforward yet effective method for MM in financial markets.

There are several works using deep Rl. Gavsperov & Kostanjvcar (2021a) explored deep reinforcement learning (DRL) for market making under a Hawkes process-based limit order book model. Their approach highlighted the advantages of using Monte Carlo backtesting to enhance the performance of DRL controllers. This study is notable for its comparison of DRL-based market making strategies against multiple benchmarks, demonstrating superior performance in various risk-reward metrics despite significant transaction costs.

Sadighian (2020) provided an end-to-end MM framework based on a state space comprising raw LOB data and additional trade and order flow imbalance features. Multiple goal-, risk-, and PnL-based reward functions were considered, and two state-of-the-art policy gradient-based algorithms, namely proximal policy optimization (PPO) and advantage actor–critic (A2C), were used to train the MM agent.

Gavsperov & Kostanjvcar (2021b) introduced an innovative deep reinforcement learning (DRL) framework specifically designed for market making, incorporating signals derived from adversarial reinforcement learning and neuroevolution techniques. This approach effectively addresses the limitations of traditional market-making models by utilizing a novel state space and action space formulation. Their experiments demonstrate that the resulting DRL agent significantly outperforms established benchmark strategies, achieving higher terminal wealth with reduced inventory risk. Additionally, the framework's design enhances the interpretability of the learned market-making policies, offering valuable insights into the decision-making process of the DRL agent.

2.3 Behavioral Cloning in Reinforcement Learning

Behavioral cloning, one of the earliest approaches to imitation learning, transforms the problem of learning to imitate a teacher into a supervised learning problem. In this approach, the agent learns to predict the most likely action given a state, $\arg\max P(a|s_t)$, based on a teacher's dataset. Specifically, the agent uses state-action pairs of demonstrations (s_t, a_t) at time t from a proficient source to learn to act as the source based on previously seen data. Behavioral cloning aims to approximate the agent's trajectory to that of its teacher. A trajectory is a coherent sequence of demonstrations or experiences from one cycle of the teacher or agent's interaction with the environment.

This approach, however, becomes costly for more complex scenarios, requiring a large number of samples and detailed information about the action's effects on the environment. For example, the number of samples needed to solve tasks involving a higher number of possible actions (e.g., continuous actions) or intricate dynamics is significantly higher compared to classic control tasks.

Bain & Sammut (1999) laid the groundwork for behavioral cloning by establishing a framework for this technique, demonstrating its application in various intelligent agent scenarios.

Newer approaches usually use behavioral cloning as a bootstrapping mechanism. Bootstrapping a policy means using a machine learning approach, such as supervised learning, to acquire knowledge from the environment or desired behavior before applying another learning technique to fine-tune the agent's performance. An example is the work from Daftry et al. Daftry et al. (2017), in which the researchers recorded themselves flying a drone, applied behavioral cloning to learn their flying behavior, and then used reinforcement learning so the agent could adapt to different seasons when the images looked different.

These approaches display some of the benefits of using behavioral cloning. The agent learns more efficiently offline (without requiring direct environment access) by applying this technique and, afterwards, using the acquired knowledge to reduce the number of steps required for less efficient learning approaches, such as reinforcement learning. Additionally, behavioral cloning coupled with bootstrapping can condition the policy to a more human-like behavior.

In recent years, the integration of behavioral cloning with deep reinforcement learning has shown promise in enhancing the efficiency and performance of RL agents in complex environments. By leveraging expert demonstrations, RL agents can achieve faster convergence and improved performance, particularly in tasks with high dimensionality and complex dynamics.

Market making is a prime example of such a task, characterized by a high-dimensional environment and a complex action space, where agents must continuously adjust bid and ask prices in response to rapidly changing market conditions. The complexity of this task makes it particularly well-suited for behavioral cloning, as the use of expert demonstrations can significantly reduce the exploration space and guide the agent toward more effective strategies.

Although there are several advantages to using behavioral cloning, there has been limited application of this technique in financial markets. One notable work is by Liu et al. (2020), who applied imitative learning to solve quantitative trading tasks. Considering the noisy financial data, they formulated the quantitative trading process as a Partially Observable Markov Decision Process (POMDP) and introduced imitation learning to leverage classical trading strategies, which are useful for balancing exploration and exploitation.

In our work, we apply imitation learning to the market-making task, recognizing its suitability for environments with high dimensionality and complex action spaces. By doing so, we aim to improve the efficiency and performance of market-making strategies through the integration of expert demonstrations and reinforcement learning.

3. Preliminaries

3.1 Limit order book

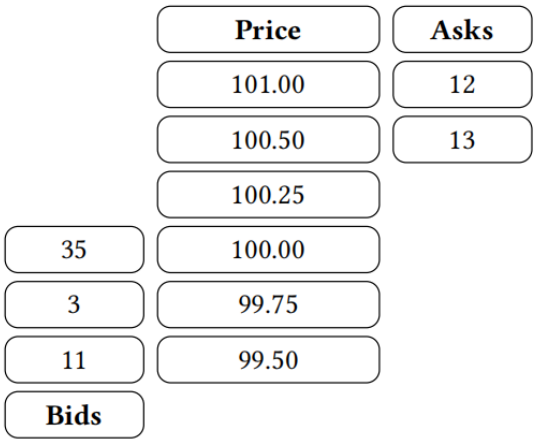


Figure 1: Snapshot of a limit order book with multiple price levels occupied by bid or ask orders.

A limit order is an instruction from a trader to buy or sell a certain amount of an asset at a specified price or better. The "or better" means that a buy order can be executed at the specified limit price or lower, and a sell order can be executed at the limit price or higher. These orders are not guaranteed to execute immediately, depending on market conditions and the order's position within the book.

As shown in Figure 1, an order book lists all the active buy and sell limit orders for an asset at various price levels. Orders are grouped together by price level, and within each level, they are typically arranged by the time they were placed, adhering to a first-come, first-served basis.

Key Components of a Limit Order Book

- **Bid Orders (Buy Orders)**: These orders are listed on the left side of the book. They are sorted by price in descending order, with the highest price (best bid) at the top. Each bid has a corresponding volume indicating the amount of the asset that buyers are willing to purchase at that price.

- **Ask Orders (Sell Orders)**: These orders are listed on the right side of the book. They are sorted by price in ascending order, with the lowest price (best ask) at the top. Each ask has a corresponding volume indicating the amount of the asset that sellers are willing to sell at that price.

- **Mid-Price**: This is calculated as the average of the best bid and best ask prices. It represents a hypothetical price that is exactly halfway between the most willing buyer's bid and the lowest seller's ask.

- **Market Orders**: While not displayed in the limit order book, market orders are instructions to buy or sell immediately at the current market price. A buy market order will execute at the lowest available ask price, and a sell market order will execute at the highest available bid price.

3.2 State Space

The state space in our trading environment is carefully designed to capture the key factors that influence market making decisions. It includes the following components:

1. **Inventory:** The current inventory level of the agent, representing the number of shares held. This is crucial for managing inventory risk, as maintaining a balanced inventory helps the agent avoid large, unhedged positions that could lead to significant losses if the market moves unfavorably. By tracking inventory, the agent can adjust its strategy to either reduce or increase holdings as needed.

2. **Order Imbalance:** The difference between the total bid and ask volumes in the order book. Order imbalance is a key indicator of market sentiment and potential short-term price movements. For instance, a high bid volume relative to ask volume may indicate buying pressure and a potential price increase, while the opposite suggests selling pressure. Including order imbalance in the state space allows the agent to anticipate and react to these market conditions.

3. **Spread:** The difference between the best bid and best ask prices. The spread is a fundamental aspect of market making, directly influencing the profitability of the agent's trading strategy. A narrow spread can lead to higher trade execution probabilities but lower profit per trade, while a wider spread increases profit margins but may reduce trade frequency. By monitoring the spread, the agent can make informed decisions about where to place its bid and ask orders.

4. **Mid-Price Move (Δm):** The change in the mid-price over time. The mid-price, being the average of the best bid and ask prices, serves as a reference point for price trends and volatility. Tracking its movement allows the agent to gauge market direction and adjust its trading strategy accordingly. For example, in a rising market, the agent might prioritize placing ask orders, while in a falling market, bid orders might be more advantageous.

5. **Current Best Bid Price:** The highest price at which buyers are willing to purchase shares. This component is critical for setting the agent's own bid price, ensuring it remains competitive while managing inventory and risk. It also helps the agent understand the current demand level in the market.

6. **Current Best Ask Price:** The lowest price at which sellers are willing to sell shares. Knowing the best ask price allows the agent to set its ask price effectively, balancing

the goal of profitable sales with the need to maintain market competitiveness. This component also provides insight into the supply side of the market.

By incorporating these components into the state space, the agent is equipped with a comprehensive view of the market dynamics that are essential for making informed and profitable trading decisions. This state space is designed to capture both the immediate market conditions and the underlying trends, enabling the agent to navigate the complexities of market making with greater effectiveness.

3.3 Actions

The action space in our market-making framework is designed to allow the agent to effectively manage its quoting strategy in response to real-time market conditions. At each timestep, the agent decides how far away from the mid-price to place its bid and ask limit orders, defined as a tuple $a = (d_b, d_a)$, where:

- $d_b \in \mathbb{Z}$ represents the distance, in multiples of 0.1 times the spread, below the mid-price for placing a bid limit order.
- $d_a \in \mathbb{Z}$ represents the distance, in multiples of 0.1 times the spread, above the mid-price for placing an ask limit order.

After determining these distances, the agent cancels any unexecuted orders from the previous timestep and submits new limit orders according to the action a, chosen based on the optimal policy derived from the current state. The order quantity, denoted as q, is a fixed parameter that can be set depending on the trading strategy or market conditions.

The bid and ask prices at time t_i are calculated as:

$$p_b(t_i) = \text{mid\_price}(t_i) - d_b \times \text{spread}(t_i) \times 0.1 \tag{1}$$

$$p_a(t_i) = \text{mid\_price}(t_i) + d_a \times \text{spread}(t_i) \times 0.1 \tag{2}$$

This approach allows the agent to dynamically adapt its quoting strategy based on the current market spread, ensuring that its actions are responsive to fluctuations in market conditions. By making d_b and d_a functions of the spread, the agent can maintain competitive pricing when spreads are narrow and manage risk effectively when spreads widen. This adaptability is crucial for optimizing the agent's performance across different market environments, balancing the goals of profitability and inventory management.

3.4 Reward Function

3.4.1 Basic Reward Function

The fundamental goal of a trading agent is to maximize its profit and loss (PnL) through interactions with the market. The natural reward function for this purpose is PnL, which quantifies the financial gain or loss resulting from trades. This concept draws inspiration from the reward function design discussed in Spooner et al. (2018), where the focus is on encouraging the agent to execute trades and maintain an inventory that will appreciate over time.

Formally, for a given time t_i, let $\text{Matched}_a(t_i)$ and $\text{Matched}_b(t_i)$ represent the volume executed against the agent's orders since the previous time step t_{i-1} in the ask and bid books, respectively, and let $m(t_i)$ denote the mid-price at time t_i. We define:

$$\psi_a(t_i) \equiv \text{Matched}_a(t_i) \cdot [p_a(t_i) - m(t_i)], \tag{3}$$

$$\psi_b(t_i) \equiv \text{Matched}_b(t_i) \cdot [m(t_i) - p_b(t_i)], \tag{4}$$

where $p_{a,b}(t_i)$ are determined by Eq. 2. These functions calculate the profit or loss from the execution of the agent's orders relative to the mid-price.

The total PnL is then computed as follows:

$$\text{realized\_pnl}(t_i) = \sum_{k=0}^{i} (\psi_a(t_k) + \psi_b(t_k)) \tag{5}$$

$$\text{unrealized\_pnl}(t_i) = (m(t_i) - m(t_{i-1})) \cdot \text{inventory}(t_i) \tag{6}$$

$$\text{pnl}(t_i) = \text{realized\_pnl}(t_i) + \text{unrealized\_pnl}(t_i) \tag{7}$$

3.4.2 Challenges with the Basic Reward Function

While the basic reward function is straightforward and aligns with the financial objectives of trading, it has some limitations that could lead to suboptimal agent behavior:

- **Sensitivity to Short-Term Price Fluctuations:** The unrealized PnL component, calculated as $\text{unrealized\_pnl}(t_i) = (m(t_i) - m(t_{i-1})) \cdot \text{inventory}(t_i)$, is highly sensitive to short-term price movements. This can result in excessive volatility in the rewards, especially when the agent holds a large inventory.
- **Absence of Inventory Risk Penalty:** The basic reward function does not penalize the agent for holding a large inventory, which may encourage the accumulation of significant positions without considering the associated risks. This could lead to unstable or risky trading strategies.

3.4.3 Enhancing the Reward Function

To address these challenges, we propose two key enhancements: applying a decay factor to the unrealized PnL and introducing an inventory penalty term.

1. Decay Factor for Unrealized PnL To mitigate the impact of short-term price volatility, we introduce a decay factor β to the unrealized PnL component. The modified unrealized PnL is expressed as:

$$\text{unrealized\_pnl}(t_i) = \beta \cdot (m(t_i) - m(t_{i-1})) \cdot \text{inventory}(t_i), \tag{8}$$

where $0 < \beta \leq 1$ represents the decay factor. By adjusting β, we can reduce the sensitivity of the reward function to temporary price fluctuations, resulting in a more stable and reliable reward signal.

2. Inventory Penalty Term To address the risks associated with holding large inventories, we incorporate an inventory penalty into the reward function. Two approaches are considered:

Linear Penalty:

The linear penalty approach imposes a direct penalty proportional to the size of the inventory:

$$\text{reward}(t_i) = \text{pnl}(t_i) - \alpha \cdot |\text{inventory}(t_i)|\,, \tag{9}$$

where α is a constant that determines the strength of the penalty.

Non-Linear Penalty:

The non-linear penalty approach applies a more aggressive penalty for larger inventories:

$$\text{reward}(t_i) = \text{pnl}(t_i) - \alpha \cdot |\text{inventory}(t_i)|^p\,, \tag{10}$$

where $p > 1$ controls the severity of the penalty, with larger p values leading to steeper penalties as the inventory size increases.

These enhancements can be used individually or in combination, depending on the specific requirements of the trading strategy. By integrating these modifications, we create a more balanced and risk-aware reward function that better aligns with the goals of stable and profitable trading.

4. Methodology

In this section, the methodologies employed to develop and evaluate market-making strategies are outlined. The approach integrates reinforcement learning techniques with deep learning models to address the complexities of high-frequency trading environments. Two primary reinforcement learning algorithms are utilized: Deep Q-Network (DQN) and Advantage Actor-Critic (A2C), both of which have been adapted to better accommodate the dynamic nature of financial markets. Additionally, these models are enhanced with imitative learning techniques through behavior cloning to improve training efficiency and stability. This combination of reinforcement learning and behavior cloning enables the agent to learn not only from its interactions with the environment but also from expert strategies, leading to more robust and adaptive market-making policies.

A key innovation in this methodology is the introduction of **dynamic weights** in both the A2C and DQN models. Unlike traditional approaches that employ fixed weightings between the policy learning and behavior cloning objectives, this method dynamically adjusts these weights during training. This adjustment allows the model to balance exploration of the environment with exploitation of expert knowledge more effectively. As the agent's performance improves, the dynamic weights shift to place greater emphasis on behavior cloning, thereby fine-tuning the policy and mitigating overfitting. This innovation ensures a more adaptive learning process, resulting in more stable and higher-performing trading strategies.

4.1 Deep Q-Network (DQN)

Deep Q-Network (DQN) is a value-based, model-free reinforcement learning algorithm that combines Q-Learning with deep neural networks. In DQN, the action-value function, or Q-value, $Q(s, a; \theta)$, is approximated by a neural network with parameters θ. This neural network takes the current state s as input and outputs the Q-values for all possible actions a.

The key innovation in DQN is the use of experience replay and target networks to stabilize training. The neural network's output, $Q(s_t, a_t; \theta)$, is trained to minimize the difference between the predicted Q-value and the target Q-value, which is calculated using the Bellman equation:

$$Q(s_t, a_t; \theta) = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-), \tag{11}$$

where θ^- are the parameters of the target network, which are periodically updated with the parameters θ of the main network.

The neural network is updated by applying gradient descent to minimize the loss function $L(\theta)$. This involves computing the gradient of the loss function with respect to the network parameters θ and adjusting θ in the direction that reduces the loss:

$$\nabla_\theta L(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}} \left[\left(y_t - Q(s_t, a_t; \theta) \right)^2 \right], \tag{12}$$

where y_t is the target value given by:

$$y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-). \tag{13}$$

The parameters θ are then updated using gradient descent:

$$\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta), \tag{14}$$

where α is the learning rate.

Experience replay involves storing transitions (s_t, a_t, r_t, s_{t+1}) in a replay buffer $\mathcal{D}$ and sampling mini-batches of transitions to update the network. This approach breaks the temporal correlations between consecutive transitions and leads to more stable training.

Target networks are used to provide a stable target for the Q-value updates. The parameters of the target network θ^- are copied from the main network θ at regular intervals, thereby reducing the risk of divergence during training.

4.2 Advantage Actor-Critic (A2C)

The Actor-Critic algorithm, as described by Konda & Tsitsiklis (2000), is a hybrid reinforcement learning approach that combines the strengths of both policy-based (actor) and value-based (critic) methods. The actor is responsible for selecting actions based on a parameterized policy $\pi(a_t|s_t;\theta)$, while the critic evaluates the actions taken by estimating the value function $V(s_t;\theta_v)$. This dual structure allows the algorithm to balance the exploration-exploitation trade-off more effectively.

In the Actor-Critic framework, the critic uses Temporal Difference (TD) learning to approximate the value function. This estimate is then used to inform the actor on how to adjust the policy parameters in a way that improves future rewards. Specifically, the advantage function $A(s_t, a_t; \theta, \theta_v)$ is used, which is the difference between the action-value $Q(s_t|a_t)$ and the state value $V(s_t)$. The advantage function is calculated as:

$$A(s_t, a_t; \theta, \theta_v) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}, \theta_v) - V(s_t, \theta_v), \tag{15}$$

where k is the discount factor, and γ is the discount rate.

The actor updates its policy parameters θ in the direction of the gradient of the expected reward, calculated as:

$$\nabla_\theta J(\theta) = \nabla_\theta \log \pi(a_t|s_t;\theta') A(s_t, a_t; \theta, \theta_v), \tag{16}$$

where $\nabla_\theta \log \pi(a_t|s_t;\theta')$ is the gradient of the log-policy.

This method is particularly useful in environments with large or continuous state and action spaces, where pure policy-based or value-based methods may struggle. The critic provides a more stable learning signal to the actor, which helps reduce the variance in policy updates and can lead to faster convergence.

4.3 Imitative Learning by Behavior Cloning

Behavior cloning is a technique in reinforcement learning where an agent learns to replicate the actions of an expert by mimicking their demonstrated behavior. In this study, the expert strategy, referred to as the Prophet Expert Strategy, is employed to assess the agent's ability to integrate learning from both expert demonstrations and its interactions with the environment.

4.3.1 Prophet Expert Strategy

The Prophet Expert Strategy assumes knowledge of all future true prices, allowing the optimal market-making strategy to involve placing a bid order at a price slightly less than half the spread below the next true price and an ask order slightly less than half the spread above it. This approach maximizes the likelihood of executing profitable trades while maintaining a balanced inventory.

In Figure 2, the expert strategy is illustrated. The top panel shows how the expert bid and ask prices are strategically positioned around the true price, which is estimated by the mid-price. The dashed lines represent the expert bid and ask actions, set at slightly less than half the current spread distance from the next mid-price. This placement ensures that as the market fluctuates, the expert's orders are optimally positioned to capture profitable trades.

The inventory over time plot demonstrates how the expert strategy dynamically adjusts its position, increasing or decreasing inventory as trades are executed. This adjustment is crucial for maintaining balance and avoiding excessive exposure to market risk.

The rewards and PnL over time plots provide insight into the profitability of the strategy. Positive rewards and an upward-trending PnL indicate successful execution of trades aligned with the market's true price movement, showcasing the effectiveness of the expert actions.

While this strategy demonstrates the benefits of precise order placement based on true price knowledge, it does not account for the prediction of future market trends using current information—a critical capability that the agent must develop through interaction with the environment. Thus, the expert strategy is an effective tool for assessing the agent's ability to learn effectively from both expert guidance and its environment.

4.3.2 Behavior Cloning in A2C and DQN

To guide each trading action, expert actions are introduced, denoted as $\bar{a}$. During each training step, the behavior cloning technique Ross & Bagnell (2010) is employed to measure the discrepancy between the agent's actions and those of the expert:

$$L' = -\mathbb{E}\left[\left\|\mu^{\theta}(h_t^i) - \bar{a}_t^i\right\|^2\right]. \tag{17}$$

The behavior cloning loss L' serves as an auxiliary loss to refine the agent's updates. In A2C, the modified policy gradient, $\nabla_\theta \tilde{J}$, is computed as:

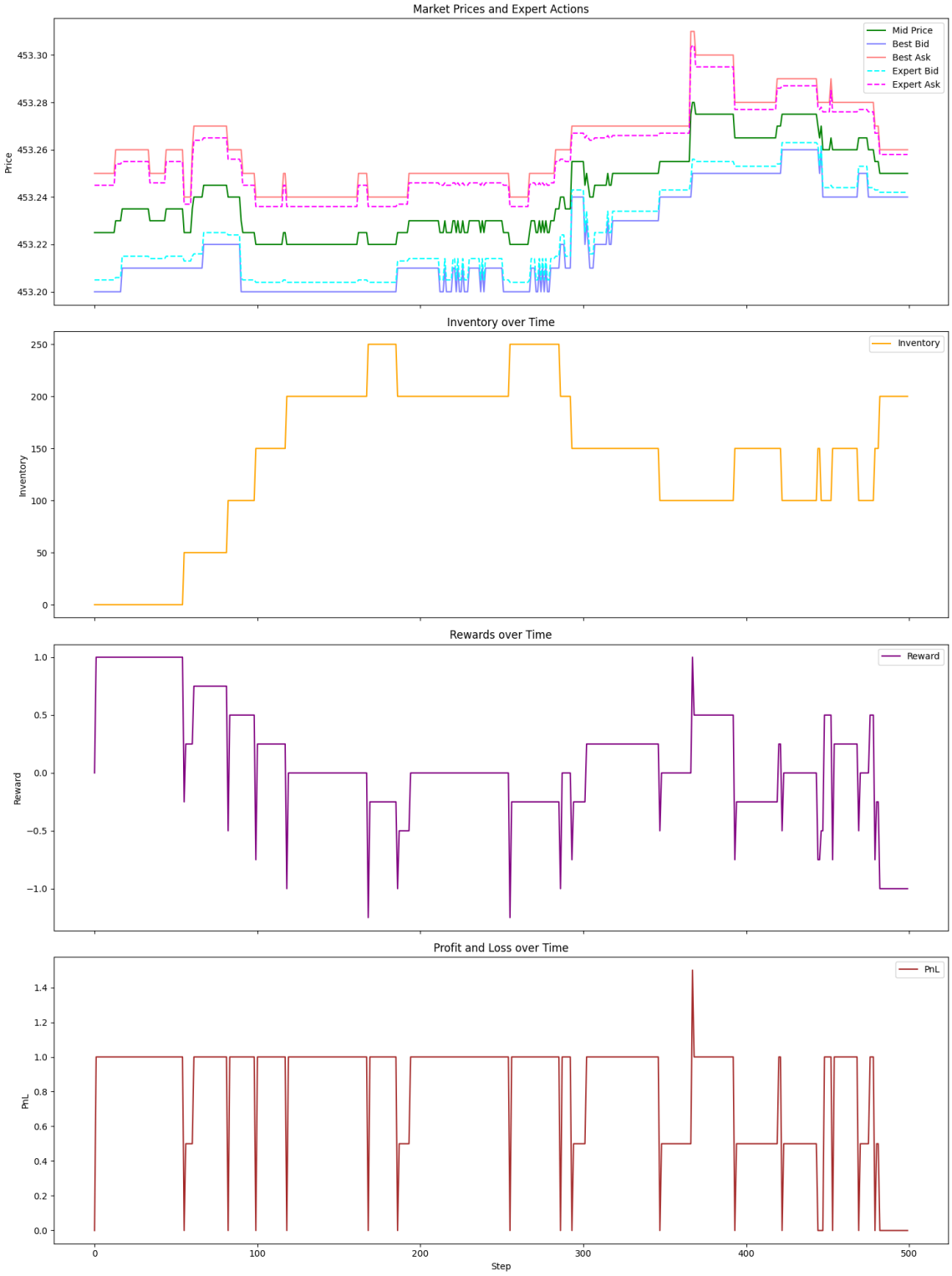


Figure 2: Demonstration of the Expert Strategy with Market Prices and Expert Actions.

$$\nabla_\theta \tilde{J}_{\text{A2C}} = \lambda_1 \nabla_\theta J_{\text{A2C}} + \lambda_2 \nabla_\theta L', \tag{18}$$

where $\nabla_\theta J_{\text{A2C}}$ is the policy gradient from the earlier equation 16. The parameters λ_1 and λ_2 control the influence of the policy gradient and the behavior cloning loss, respectively.

In the case of DQN, the behavior cloning loss L' similarly contributes to the gradient used to update the neural network parameters. The gradient of the combined loss function for DQN is then given by:

$$\nabla_\theta \tilde{L}_{\text{DQN}}(\theta) = \lambda_1 \nabla_\theta L(\theta) + \lambda_2 \nabla_\theta L', \tag{19}$$

where $\nabla_\theta L(\theta)$ is the gradient of the original loss function. The parameter λ_2 controls the influence of the behavior cloning loss in the gradient update.

The neural network parameters θ are updated using this gradient to minimize the overall loss, balancing between the original Q-learning objective and the behavior cloning objective.

4.3.3 Dynamic Weights in A2C and DQN:

In practice, it is crucial to consider scenarios where the loss functions for the agent's policy and behavior cloning move in different directions. This divergence can arise due to several factors:

- **Conflicting Objectives:** The agent's primary objective—maximizing rewards—may at times conflict with the expert's demonstrated behavior. This conflict is particularly evident if the expert's strategy is suboptimal or if there have been significant changes in the environment since the expert data was collected.

- **Exploration vs. Exploitation:** The agent may discover more efficient strategies that diverge from the expert's behavior, creating tension between the need to exploit known strategies (behavior cloning) and the need to explore potentially superior alternatives.

- **Non-stationary Environments:** In dynamic or non-stationary environments, optimal strategies may shift over time, causing the agent's learned policy to diverge from static expert demonstrations.

To address the challenges of balancing agent learning with expert guidance, this research proposes a dynamic weight adjustment strategy for the parameters λ_1 and λ_2. Rather than maintaining fixed values, this approach adapts over the course of training, facilitating a more effective learning process.

In the initial phase of learning, the strategy sets a lower λ_1 and a higher λ_2, reflecting the agent's limited knowledge at this stage. By prioritizing the expert demonstrations, the agent is guided away from excessive and potentially unproductive exploration, allowing it to establish a solid foundation of competent behavior.

As training progresses, a shift occurs: λ_1 is gradually increased, while λ_2 is decreased. This transition is triggered by the diminishing gradients of the behavior cloning loss,

$\nabla_\theta L'(\theta)$, which indicate that the agent's actions are increasingly aligned with the expert's behavior. As the need for direct expert guidance wanes, the agent is given more autonomy to learn from its interactions with the environment, allowing it to explore and refine strategies that may even surpass the expert's initial performance.

This dynamic adjustment process ensures a balanced and adaptive learning trajectory. In the early stages, the agent effectively leverages the expertise encoded in the demonstrations, while in later stages, it capitalizes on the experience gained from its environment. This approach facilitates convergence toward an optimal strategy that integrates the strengths of both expert-guided actions and environment-driven learning.

5. Experiment

5.1 Market Assumptions

The following market assumptions are fundamental to both the historical data reconstruction and the simulated market environments used in our experiments. These assumptions guide how our market-making agent interacts with the market and are critical to the design and evaluation of our strategies.

5.1.1 General Market Assumptions

- **No Transaction Costs:** There are no transaction costs, allowing for a clearer evaluation of the strategy's performance.
- **No Information Leakage:** There is no information leakage from the agent's actions, meaning other market participants do not adapt their strategies based on the agent's trades.
- **Efficient Market Hypothesis:** The market is assumed to be efficient, ensuring that bid prices are always lower than offer prices, thereby preventing any instances where orders cross. This reflects a no-arbitrage condition in the market, where bid-offer spreads are always positive. When reconstructing from historical data, this condition is consistently met.
- **Exchange-Traded Instruments:** The agent operates in an environment where all trading occurs with exchange-styled instruments, rather than over-the-counter (OTC) instruments. Thus, all quotes are public and available to the entire market, reflecting the transparency of a centralized exchange.
- **Liquid Market with Negligible Market Impact:** We assume the market is highly liquid, with the agent's small-sized orders having negligible impact on market prices. This simplifies the modeling by ignoring the potential price movements caused by the agent's trades. However, in more advanced setups, this assumption could be relaxed to study the relationship between traded volume, order side, price movement, spread, and time of day. A simple regression analysis could be used to quantify these relationships if required.

5.1.2 Agent Order Assumptions

- **Two-Way Price Quotation:** The market-making agent always quotes two-way prices, providing both a bid and an offer. This means the agent simultaneously quotes a price at which it is willing to buy (bid) and a price at which it is willing to sell (offer), thereby providing liquidity to both sides of the market.
- **Negligible Latency and Firm Quotes:** It is assumed that there is negligible latency in the market, meaning that the quotes provided by the agent are firm and executable upon arrival in the market. This simplifies the model by eliminating the complexities associated with latency and quote adjustments during transmission.

- **Execution Conditions for Quoted Prices:** The execution of the agent's quotes is contingent upon their position within the Limit Order Book (LOB):
  - **Best Bid/Offer Level:** If the agent's quoted price becomes the best bid or offer and maintains this position for a certain number of market updates (denoted as "primary updates"), the quote will be executed. These updates represent changes in the bid/ask prices or order sizes. To introduce variability, the number of primary updates can be modeled using a Poisson distribution, reflecting the inherent randomness of real-world trading.
  - **Better Than Best Bid/Offer:** If the agent's quoted price is better than the current best bid (i.e., a higher bid) or the best offer (i.e., a lower ask), the quote will be fully executed immediately. This reflects the market's tendency to accept orders that offer more favorable prices, leading to immediate execution.
- **Order Management:** The agent is designed to maintain a limited number of active orders in the market, referred to as "active quotes." If the agent reaches the maximum allowed number of active quotes, it must cancel an existing order before placing a new one. In practical scenarios, strategies involving cancelation and replacement are common; however, for simplicity, the agent in this model places only one bid or offer at a time. Adjusting the bid/ask price is therefore equivalent to canceling the previous order and placing a new one.

5.2 Data and Environment Setup

In our experiments, we employ two distinct approaches to setting up the market environment: historical data reconstruction and simulated data creation. This dual approach ensures the robustness and comprehensiveness of our results, allowing us to validate the effectiveness of our models under both realistic and controlled conditions.

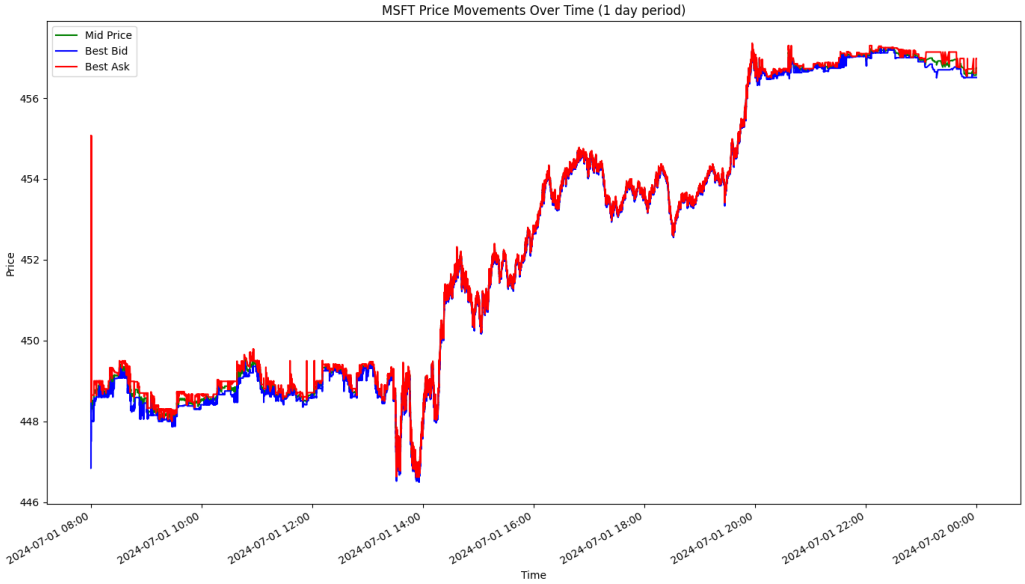
5.2.1 Historical Data Reconstruction

Data Source: The study utilizes high-frequency Limit Order Book (LOB) data from major financial exchanges, spanning several months. This dataset provides detailed information on 10 levels of bid and ask prices, order volumes, and timestamps. Specifically, the data is drawn from two different liquidity stocks:

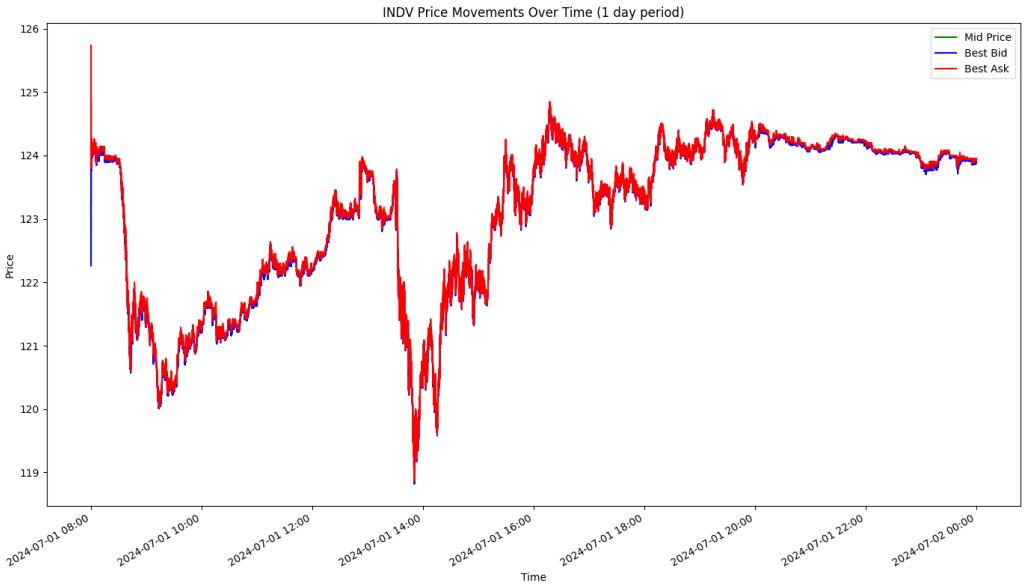
- MSFT and NVDA from Nasdaq.

The data is sourced from an open-source dataset available at Databento. As illustrated in Figure 3, these two assets exhibit distinct levels of volatility and spread.

Environment Setup: The environment is set up by replaying the historical Limit Order Book (LOB) data to simulate a live trading environment. In doing so, the reconstruction adheres to the market assumptions outlined in the previous subsection. The agent interacts with this environment by placing bid and ask orders based on



((a)) MSFT Price Data



((b)) INDV Price Data

Figure 3: Price Data for MSFT and INDV Over 1 Day

the current state of the LOB. Feedback is provided through filled orders, updated LOB states, and reward signals based on profit and loss (PnL).

The market reconstruction process assumes that the agent's trading volume is negligible, meaning that its actions do not impact market prices. This ensures that the historical integrity of the market data is maintained, allowing the agent to develop strategies that are robust and applicable to real-world trading conditions. The setup, therefore, provides a realistic yet controlled environment for evaluating the performance of the market-making agent.

5.2.2 Simulated Data Implementation

The simulated market environment is meticulously designed to emulate the behavior of different market participants, including informed traders, uninformed traders, and a market maker. The implementation of these participants is central to creating a realistic and challenging environment in which the market-making agent operates.

Informed and Uninformed Traders: In the simulation, traders are categorized into informed and uninformed groups. Informed traders are equipped with predictive capabilities that allow them to anticipate future price movements. These traders use advanced models to make decisions based on the predicted direction of the market, reflecting their access to superior information. In contrast, uninformed traders, often referred to as noise traders, make decisions randomly without any specific knowledge of future prices. Their random actions add liquidity and unpredictability to the market, making it more challenging for the market maker to differentiate between informed and uninformed trading activity.

Market Maker: The market maker in the simulation sets bid and ask prices based on real-time market data, including the observed order flow and its own inventory levels. The market maker's primary objective is to maintain a balanced inventory while profiting from the spread between bid and ask prices. To achieve this, the market maker dynamically adjusts prices in response to changing market conditions. For instance, if the market maker accumulates too much inventory, it may lower ask prices to encourage sales, or raise bid prices to attract more purchases, depending on the direction needed to balance the inventory.

Contribution to Simulated Data Approach: The interplay between informed traders, uninformed traders, and the market maker creates a dynamic and responsive market environment. This environment serves as a testing ground for the market-making agent, challenging it to develop strategies that are effective under various market conditions, such as high volatility or periods of low liquidity.

The simulation allows for precise control over market parameters, enabling the exploration of different scenarios. For example, the proportion of informed to uninformed traders can be adjusted to simulate markets with varying levels of information asymmetry. Similarly, market volatility and liquidity can be modified to assess how the agent performs under different stress conditions.

By carefully designing the behaviors of the traders and the market maker, the simulation provides a robust platform for testing and refining market-making strategies. This approach ensures that the agent is not only trained in a controlled environment but also exposed to a wide range of market conditions that it may encounter in real-world trading. The flexibility of this simulated environment makes it an invaluable tool for validating the effectiveness and adaptability of the market-making strategies developed in this research.

The full implementation details, including the code for these market participants, are available in the appendix or upon request, providing transparency and allowing for reproducibility of the simulation and its results.

5.3 Model Settings

In our experiments, we evaluate seven different models to assess the efficacy of various enhancements such as Behavior Cloning (BC) and Dynamic Weights (DW). Each model is described below:

1. **Benchmark Model:** The benchmark model is a static, rule-based model rather than a reinforcement learning agent. It operates by placing bid and ask orders at a fixed ratio(0.5) of the spread away from the current mid-price. This approach does not adapt to market conditions but provides a stable reference point against which the performance of more dynamic models can be compared.
2. **DQN:** Deep Q-Network, a standard reinforcement learning model using deep neural networks to approximate the Q-value function.
3. **DQN_BC:** DQN enhanced with Behavior Cloning (BC). This model leverages expert trajectories to guide the learning process, potentially speeding up the convergence and improving the policy robustness.
4. **DQN_BC_DW:** DQN with Behavior Cloning and Dynamic Weights. This model not only incorporates expert behavior but also adapts the importance of the cloning mechanism dynamically throughout training, optimizing the balance between learning from the expert and exploring new strategies.
5. **A2C:** Advantage Actor-Critic, an on-policy model that uses separate networks for the policy and value function, enabling more stable updates.
6. **A2C_BC:** A2C enhanced with Behavior Cloning. By mimicking expert decisions, this model aims to avoid poor local minima and accelerate learning.
7. **A2C_BC_DW:** A2C with Behavior Cloning and Dynamic Weights. It utilizes dynamic adjustments in the weight given to the behavior cloning loss to fine-tune the learning process based on the model's performance and convergence rate.

5.4 Experiment Settings

This section outlines the detailed settings and hyperparameters used in our experiments for both the Deep Q-Network (DQN) and Advantage Actor-Critic (A2C) models. It also covers the configurations specific to Behavior Cloning (BC) and Dynamic Weights (DW) adjustments, as well as the environment and other experimental settings.

5.4.1 Hyperparameters for DQN and A2C

The following tables 1 and 2 summarize the hyperparameters used for both DQN and A2C, respectively. These hyperparameters were determined by a grid search. To focus on comparing the impact of different algorithms on training efficiency and performance, the neural network structure was kept the same across both algorithms.

Table 1: Key Hyperparameters for Deep Q-Networks (DQN)

Hyperparameter	**Description**
Learning Rate (α)	0.001
Discount Factor (γ)	0.99
Epsilon (ε)	For ε-greedy exploration (initial: 1.0, final: 0.01)
Replay Buffer Size	10,000
Hidden Layer Size	64 neurons per layer
Number of Hidden Layers	3
Batch Size	64
Target Network Update Frequency	Every 10,000 steps
Activation Functions	ReLU for hidden, Linear for output
Optimizer	Adam

Table 2: Key Hyperparameters for Advantage Actor-Critic (A2C)

Hyperparameter	**Description**
Learning Rate (α)	0.001
Discount Factor (γ)	0.99
Entropy Coefficient (β)	0.01
Number of Steps (n)	5
Hidden Layer Size	64 neurons per layer
Number of Hidden Layers	3
Batch Size	64
Activation Functions	ReLU for hidden, Linear for critic, Softmax for actor
Optimizer	Adam

5.4.2 Behavior Cloning Parameters

For models enhanced with Behavior Cloning, the parameters λ_1 and λ_2 are used to control the influence of the policy gradient and behavior cloning loss. These parameters are set differently for static and dynamic weight adjustments, as shown below:

Setting	λ_1	λ_2
Static Weights	0.5	0.5
Dynamic Weights (Initial Values)	0.2	0.8

Table 3: Behavior Cloning Parameters (λ_1 and λ_2)

Dynamic Weights Adjustment: During training, the dynamic weights λ_1 and λ_2 are adjusted according to the behavior cloning loss as follows. The threshold (Threshold) is typically a very small positive value, used to determine when significant changes in the behavior cloning loss occur. In practice, the loss rarely falls below this threshold, so the adjustment process generally involves gradually increasing λ_1 and decreasing λ_2:

Condition	**Adjustment**
$\Delta L' < \text{Threshold}$	$\lambda_2 \leftarrow \min(0.9, \lambda_2 \times 1.05)$ $\lambda_1 \leftarrow \max(0.1, \lambda_1 \times 0.95)$
$\Delta L' \geq \text{Threshold}$	$\lambda_1 \leftarrow \min(0.9, \lambda_1 \times 1.05)$ $\lambda_2 \leftarrow \max(0.1, \lambda_2 \times 0.95)$

Table 4: Dynamic Weights Adjustment Logic

5.4.3 Environment Settings

As mentioned in Section 3.4, we have two types of reward functions: linear inventory penalty and non-linear penalty. After testing both, it was found that using the linear penalty provided a better balance between PnL and inventory management.

Regarding the action space, the trading quantity was fixed at a volume of 50 units per trade for both the simulation and historical environments. This fixed volume was chosen to ensure that the agent's trades would not significantly impact the market, thereby making our assumptions more practical and aligned with real-world trading conditions.

Parameter	**Value**
Reward Function Type	Linear Inventory Penalty
Trading Quantity per Action	50 units

Table 5: Environment Setting

5.4.4 Other Experimental Settings

Setting	Value
Time steps within an iteration	500
Training Duration	300 iterations
Evaluation Frequency	Every 10 iterations

Table 6: Other Experimental Settings

Thus, there will be a total of 500 * 300 time steps during the entire training process. In each time step, the agent will receive a reward and compute the realized PnL (profit and loss) for that specific time step.

5.5 Evaluation Criteria

The evaluation of the models is conducted using three key metrics: training efficiency, profit and loss (PnL), and inventory management.

- **Training Efficiency:** Training efficiency is typically measured by the number of iterations required for the loss function to converge, where convergence is defined as the point where the loss stabilizes with minimal fluctuations over successive iterations. However, due to the differing loss functions in our models, we fixed the number of iterations and instead focused on the evolution of PnL and rewards throughout the training process.
- **Profit and Loss (PnL):** PnL is a key indicator of each model's performance, reflecting its ability to make profitable decisions in the simulated market environment. Specifically, we focus on the PnL per round, calculated as:

  $$\text{PnL per round} = \frac{\text{Total Realized PnL}}{\text{Number of Time Steps}}$$

  Higher PnL values suggest more effective trading strategies. Additionally, we employ the Sharpe ratio, calculated as:

  $$\text{Sharpe Ratio} = \frac{\text{Realized PnL} + \text{Unrealized PnL}}{\text{Variance of PnL}}$$

  to assess how inventory impacts unrealized PnL, providing a more comprehensive evaluation of the model's risk-adjusted returns.
- **Inventory Management:** Effective inventory management is crucial in market making, as it mitigates the risks associated with large positions. We monitor inventory levels to evaluate each model's ability to balance risk while optimizing trading opportunities.

Both PnL and inventory management performance will be evaluated based on the best model identified during training. This evaluation will be conducted across 100 new episodes, with the mean and standard deviation recorded.

These criteria provide a comprehensive framework for assessing the impact of enhancements like Behavior Cloning and Dynamic Weights on both the learning process and the models' practical performance in trading scenarios.

6. Results Interpretation

6.1 Comparison of DQN, DQN_BC, and DQN_BC_DW Training Processes

This section provides a comparative analysis of the training processes for the three models: Deep Q-Network (DQN), DQN with Behavior Cloning (DQN_BC), and DQN with Behavior Cloning and Dynamic Weights (DQN_BC_DW). The focus is on the convergence of rewards, Profit and Loss (PnL), and loss functions over time, as illustrated in the respective figures. In the figures, the x-axis represents either the episode number or the cumulative updated time, calculated as the product of episodes and the number of time steps within each episode. This dual representation allows for a detailed understanding of how these models perform over both discrete training episodes and continuous time-based updates.

DQN Training Process Figure 4 presents the training dynamics of the DQN model. The episode rewards demonstrate significant volatility, with occasional sharp drops and spikes, indicating that the agent's decision-making process lacks stability. Although there is a general downward trend in the overall training loss, it remains relatively high throughout the training process, signaling that the agent struggles to consistently optimize its strategy. The PnL also fluctuates considerably, further underscoring the instability in the agent's trading decisions. The Epsilon value decreases over time as expected, which allows the agent to transition from exploration to exploitation; however, the instability in rewards and PnL suggests that the model may not be effectively capitalizing on the learned policy.

DQN with Behavior Cloning (DQN_BC) In Figure 5, the incorporation of Behavior Cloning (BC) into the DQN model shows a noticeable improvement in the training process. The episode rewards and PnLs still exhibit fluctuations, but they show a more pronounced trend towards stabilization, particularly in the latter stages of training. This suggests that Behavior Cloning aids the agent in learning more stable and effective strategies by imitating the expert's actions. The Q-Learning loss decreases more consistently compared to the standard DQN, and the BC loss converges rapidly, indicating that the agent successfully integrates the expert's behavior into its policy.

DQN with Behavior Cloning and Dynamic Weights (DQN_BC_DW) Figure 6 and 7 depict the training results for the DQN_BC_DW model. This model exhibits the most stable training process among the three, with both the episode rewards and PnLs showing smoother trends towards convergence. The dynamic adjustment of the lambda weights (λ_1 and λ_2) for Q-learning and Behavior Cloning plays a crucial role in stabilizing the training process. As training progresses, the dynamic weights shift the model's focus from imitating the expert's behavior to refining its own Q-learning strategy. This results in a more balanced and robust learning process, with a consistent decrease in both Q-Learning and Behavior Cloning losses, leading to improved performance in terms of stable rewards and PnLs.

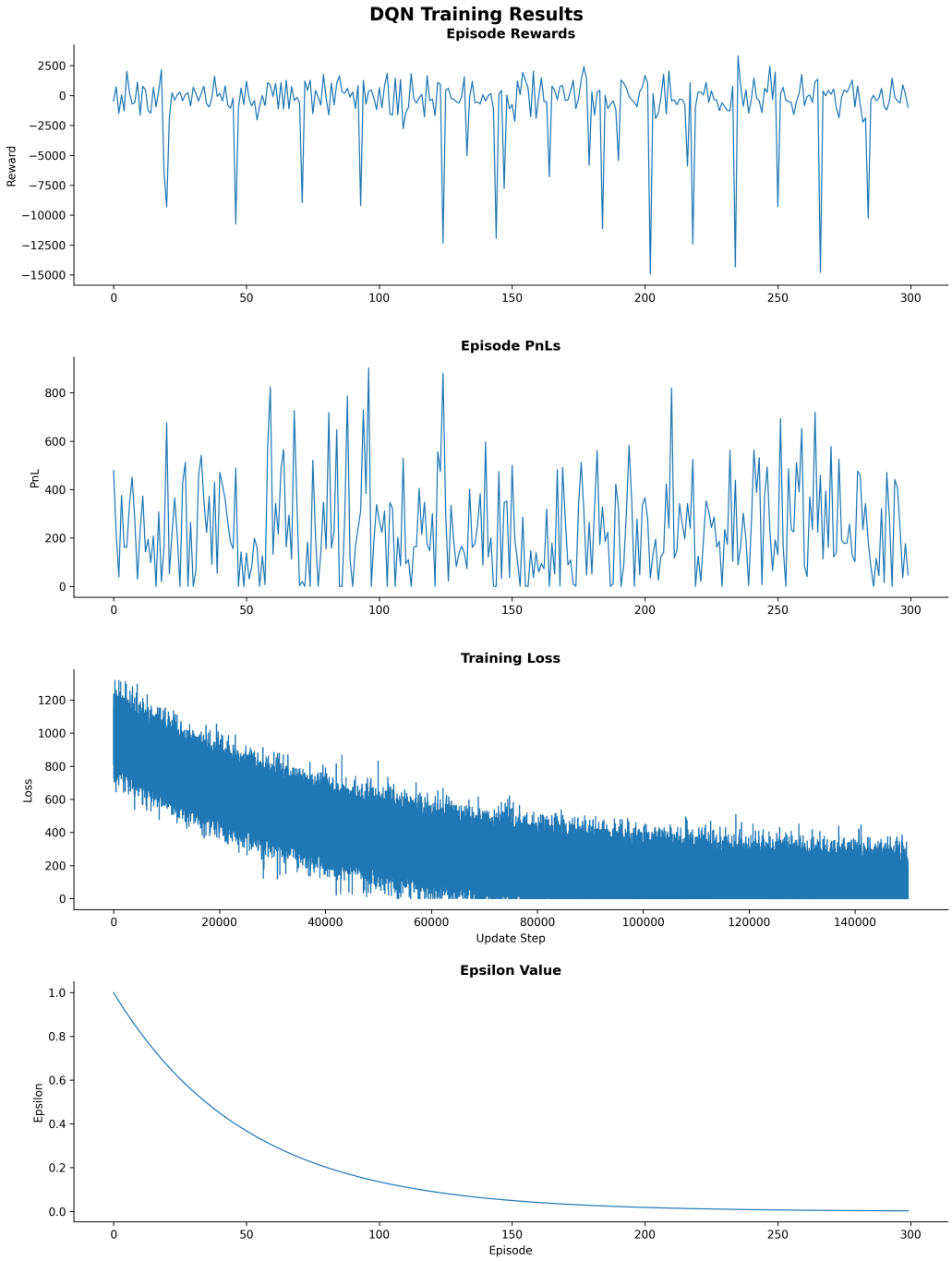


Figure 4: DQN Training Results: Rewards, PnLs, Loss, and Epsilon over Time.

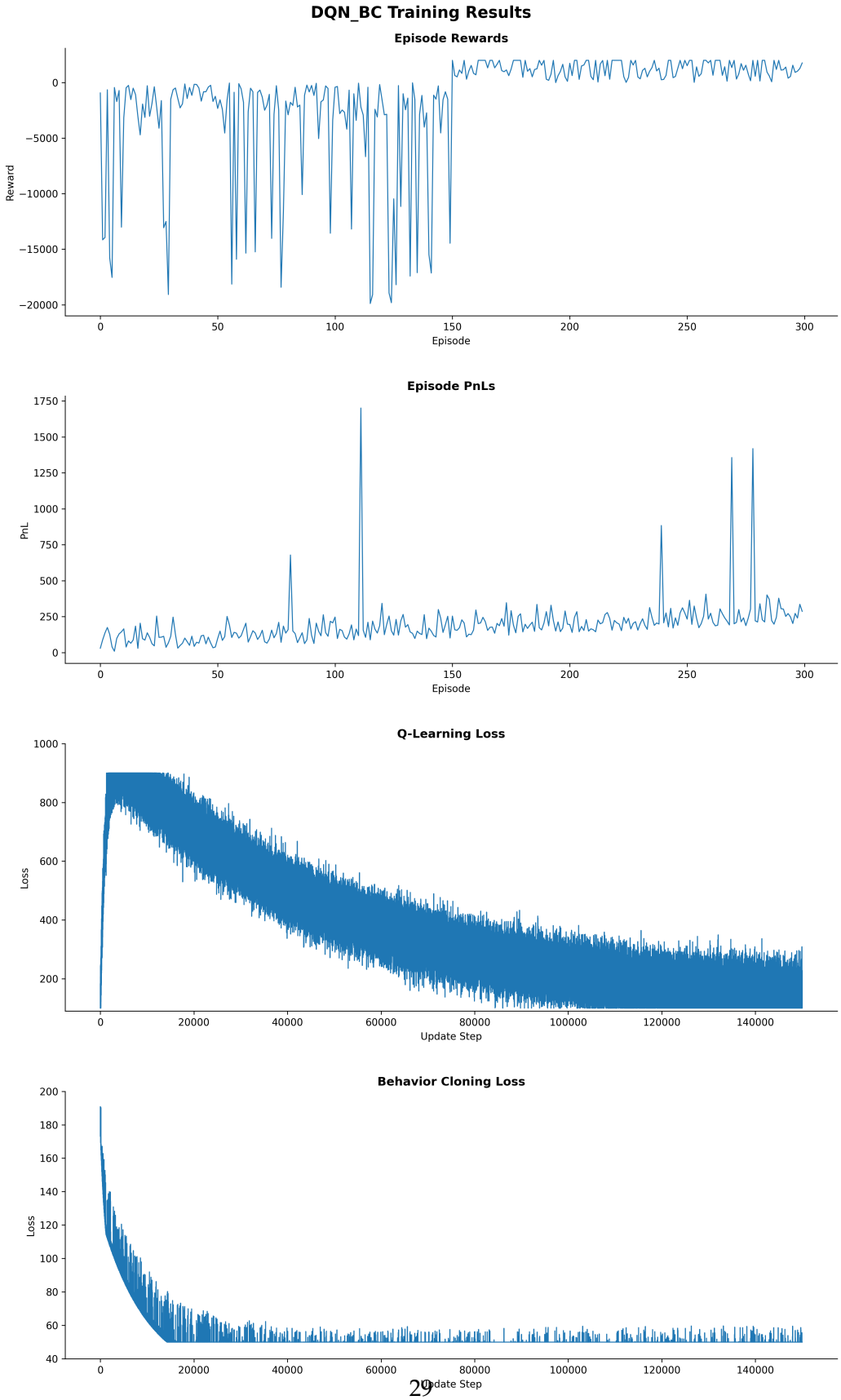


Figure 5: DQN_BC Training Results: Rewards, PnLs, Q-Learning Loss, and Behavior Cloning Loss.

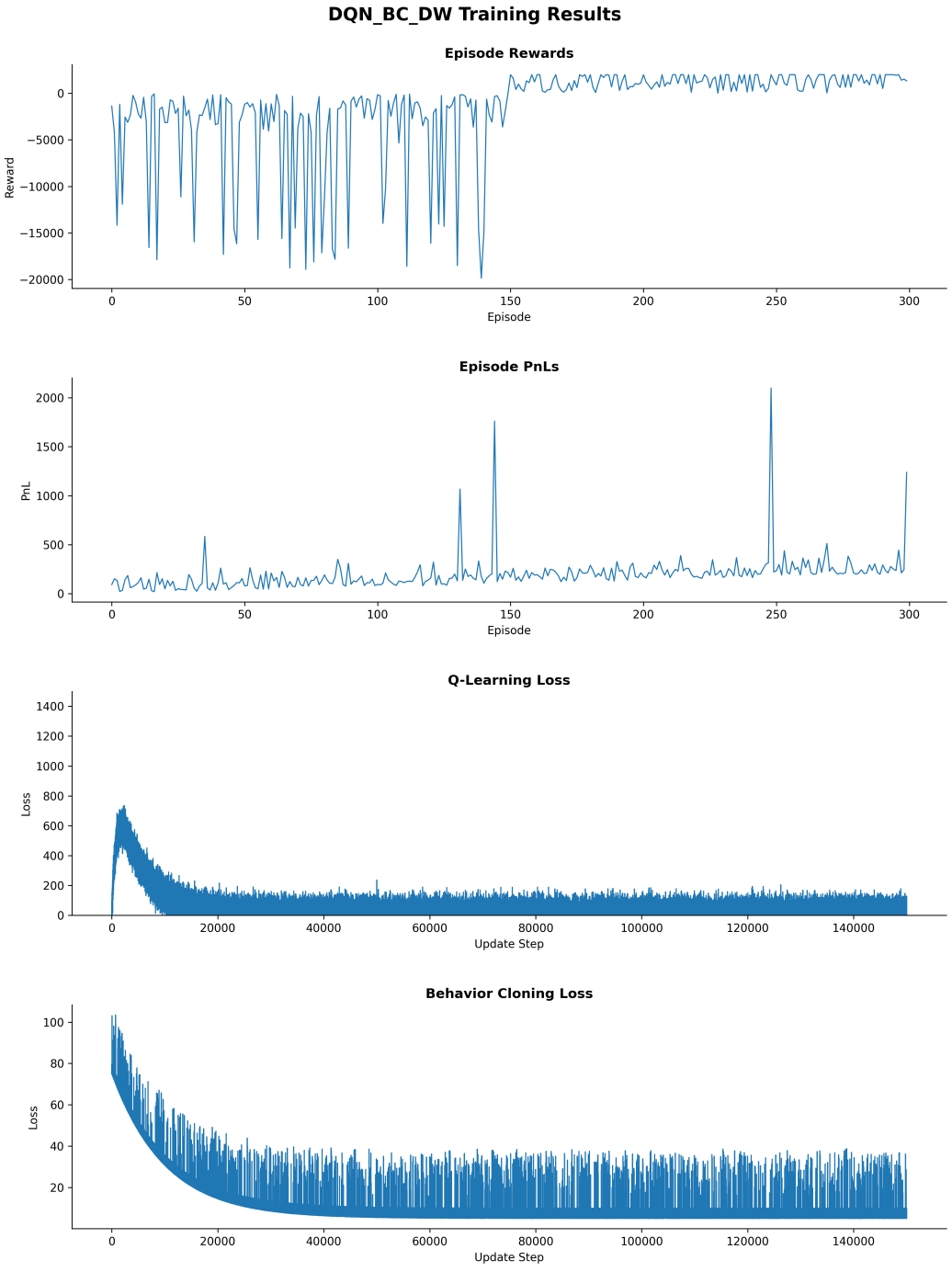


Figure 6: DQN_BC_DW Training Results: Rewards, PnLs, Q-Learning Loss, and Behavior Cloning Loss.

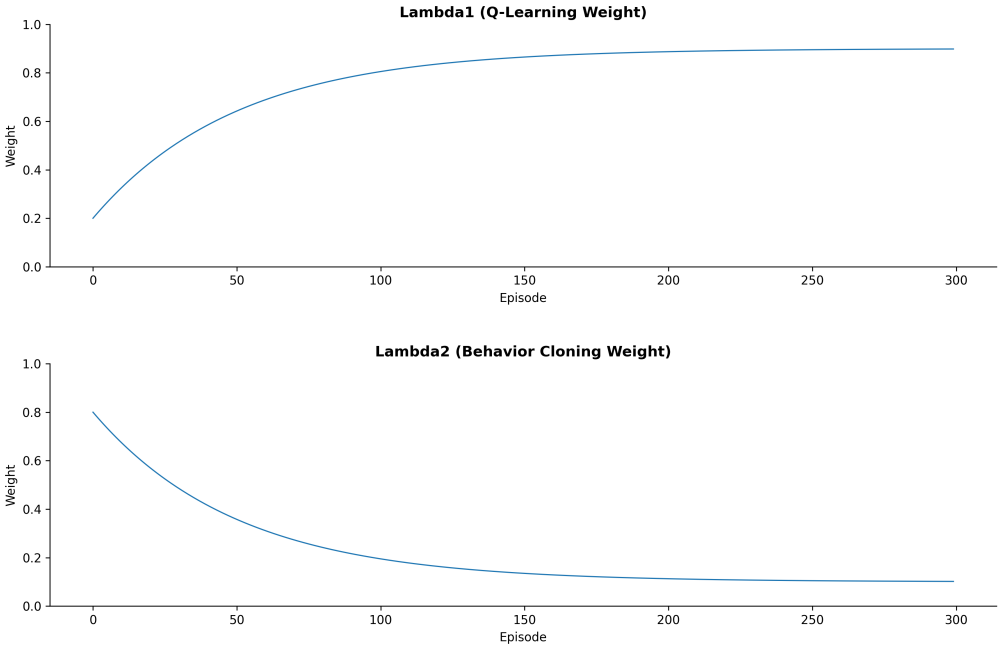


Figure 7: DQN_BC_DW Training Results: Dynamic Weights over Time.

Summary The DQN_BC_DW model demonstrates the best stability and performance, with smoother learning curves and fewer fluctuations in rewards and PnL compared to the other models. The dynamic weighting mechanism effectively balances the influence of Behavior Cloning and Q-learning, leading to a more robust and consistent training process. In contrast, the DQN model shows the most instability, with high volatility in rewards and PnL, while the DQN_BC model offers some improvement, though not to the extent achieved by the DQN_BC_DW model. These results underscore the advantages of incorporating both Behavior Cloning and dynamic weighting in enhancing the stability and efficiency of the learning process.

6.2 Comparison of A2C, A2C_BC, and A2C_BC_DW Training Processes

To compare the training processes for the A2C, A2C_BC, and A2C_BC_DW models, key aspects such as reward consistency, PnL (Profit and Loss) stability, and the behavior of losses (Actor Loss, Critic Loss, and BC Loss) are considered. For BC Loss, the x-axis represents each update time, providing a detailed view of how the behavior cloning loss evolves over time. For Actor and Critic Losses, the x-axis is shown by episode, highlighting the changes in these losses as training progresses through different episodes.

A2C Training Process The rewards and PnLs show significant volatility throughout the training process, as illustrated in Figure 8. This suggests that the A2C model struggles to consistently optimize its strategy. Large spikes and drops in the graph indicate that the model might be exploring various strategies without consistently

finding a profitable approach. The Actor and Critic losses decrease gradually, indicating some learning, but the fluctuations suggest instability in the learning process.

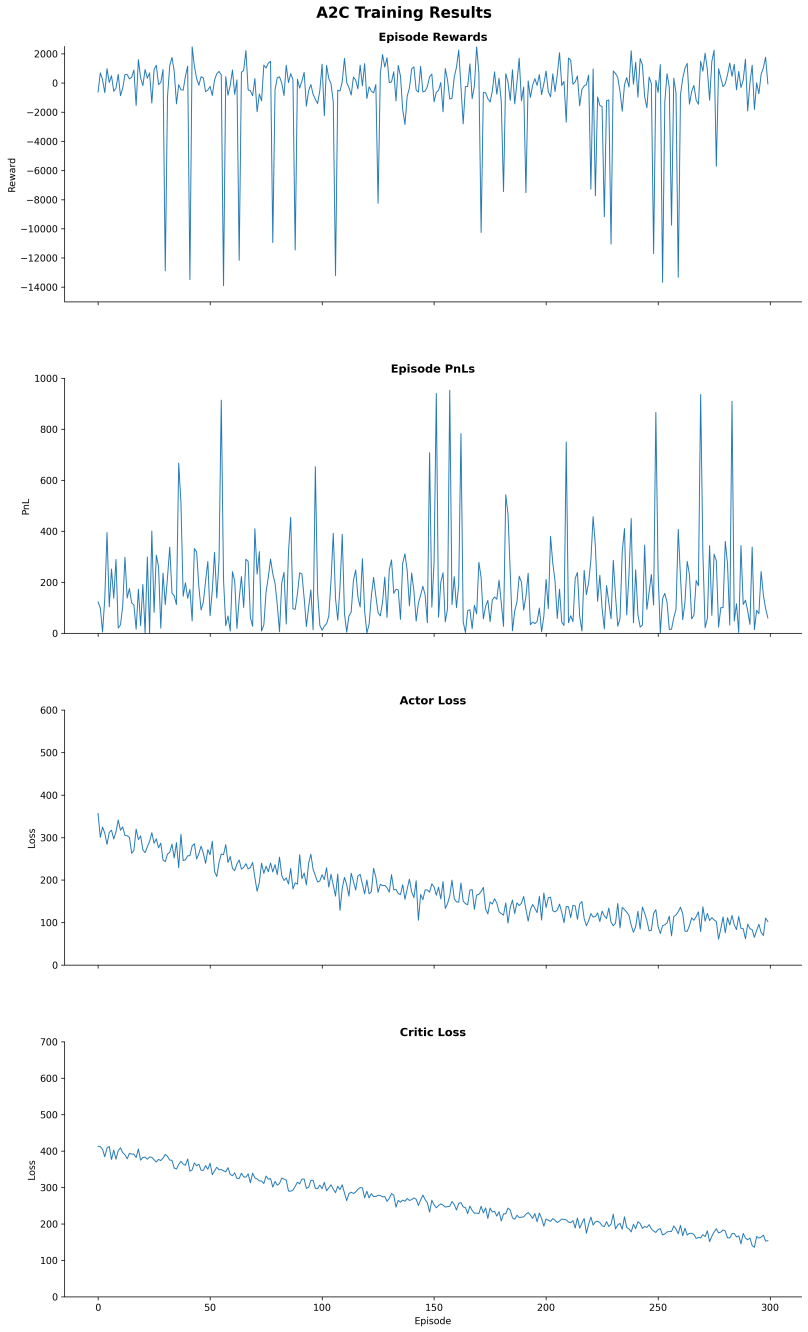


Figure 8: Training results for A2C model: Episode Rewards, PnL, Actor Loss, and Critic Loss.

A2C with Behavior Cloning (A2C_BC) Figure 9 shows that the introduction of Behavior Cloning into the A2C model significantly reduces the initial instability observed in rewards and PnLs. While fluctuations are still present, there is a clear trend towards greater stabilization as training progresses. The consistent decrease in BC Loss reflects the model's growing proficiency in replicating expert actions, contributing to more reliable performance over time.

A2C with Behavior Cloning and Dynamic Weights (A2C_BC_DW) The A2C_BC_DW model, as depicted in Figures 10 and 11, shows the most stable rewards and PnLs, with fewer dramatic spikes and a smoother learning curve. This indicates that the dynamic weighting of behavior cloning and policy learning is effective in stabilizing the learning process. The Actor and Critic losses reduce steadily, similar to the A2C_BC model, but with even less fluctuation, particularly in the BC Loss. The dynamic weights (Lambda1 for A2C and Lambda2 for BC) adjust over time, allowing the model to focus more on either A2C or BC as needed throughout training.

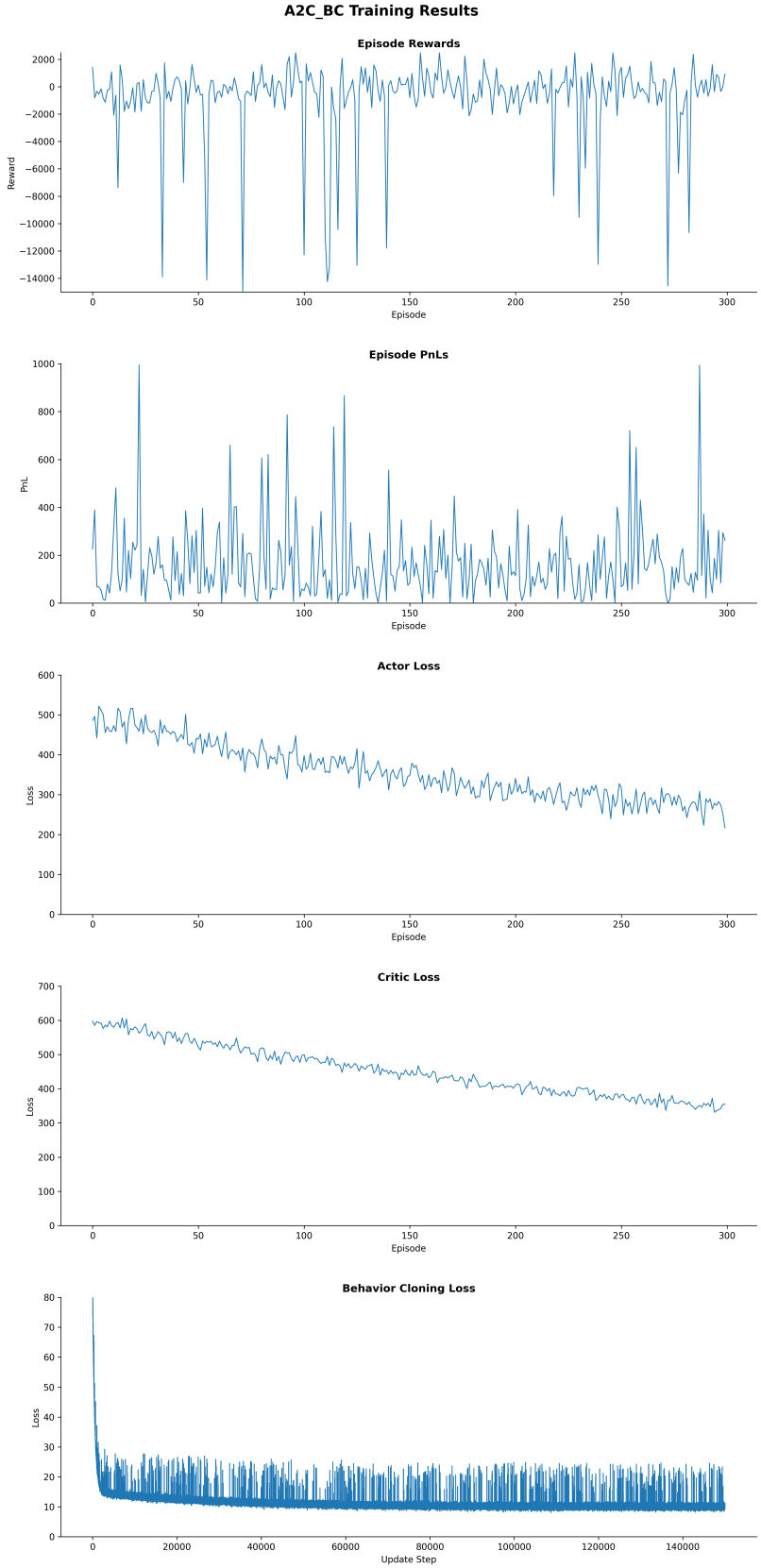


Figure 9: Training results for A2C_BC model: Episode Rewards, PnL, Actor Loss, Critic Loss, and BC Loss.

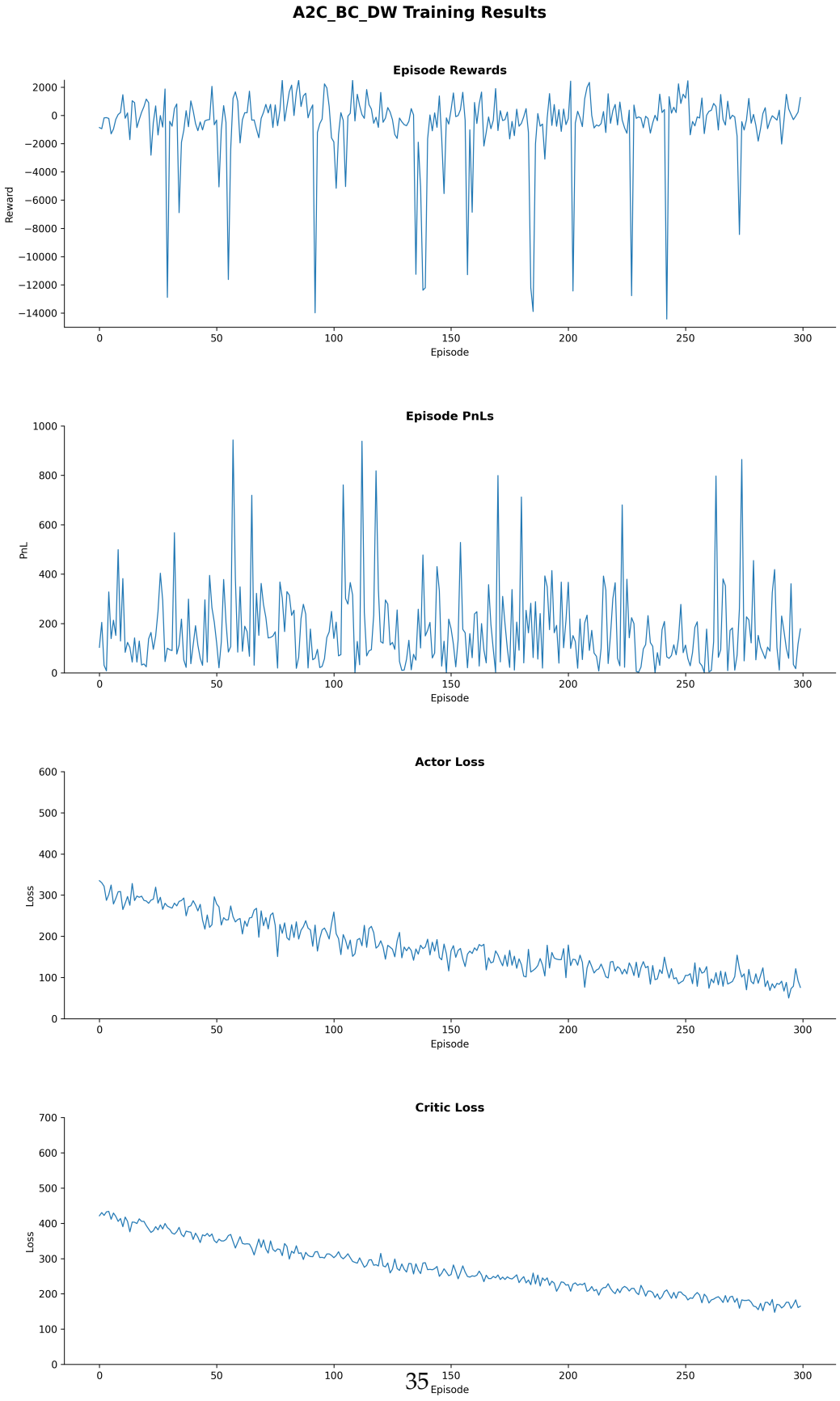


Figure 10: Episode Rewards, Episode PnLs, Actor Loss, and Critic Loss for A2C_BC_DW

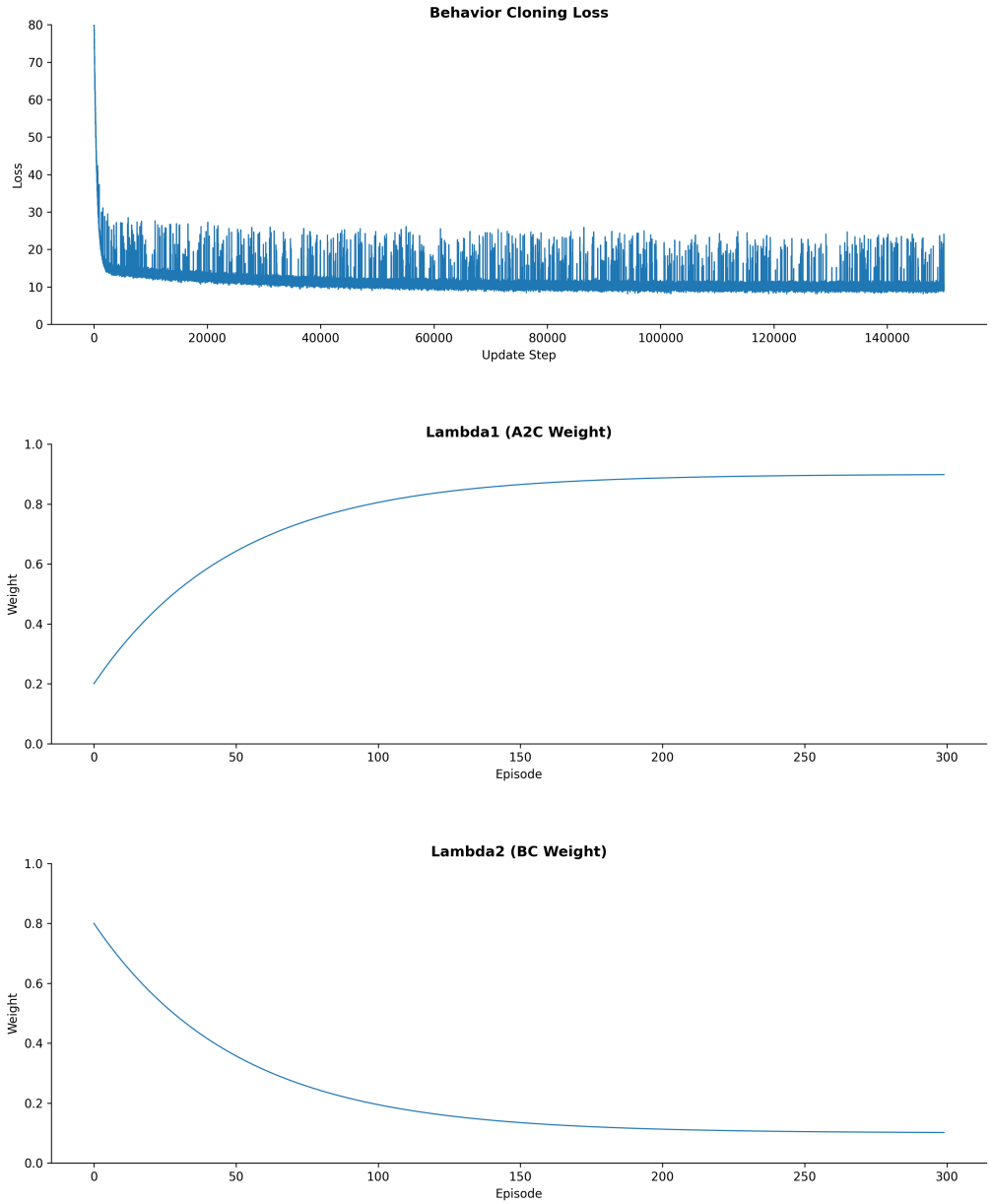


Figure 11: Behavior Cloning Loss, and Lambda Weights for A2C_BC_DW

Summary The A2C_BC_DW model demonstrates superior stability and learning performance compared to the other models, primarily due to its dynamic weighting mechanism, which effectively balances the influence of behavior cloning and policy learning. This results in smoother and more consistent improvements throughout training. While the introduction of behavior cloning in the A2C_BC model already enhances learning consistency, the dynamic weights in the A2C_BC_DW model further amplify these benefits, leading to a more robust and efficient training process.

Overall, the A2C_BC_DW model excels in developing a market-making strategy with reduced volatility and more consistent profits.

6.3 Performance Comparison

Models	PnL Mean	PnL Std	Inventory Mean	Inventory Std	Sharpe Ratio
Benchmark	1.45	0.55	1.93	1.57	2.81
DQN	1.96	0.95	0.18	0.08	4.42
DQN_BC	1.95	0.88	0.18	0.06	4.43
DQN_BC_DW	1.97	0.89	**0.17**	0.07	**4.63**
A2C	**2.00**	0.99	0.20	0.09	3.93
A2C_BC	1.96	0.87	0.21	0.65	4.20
A2C_BC_DW	1.98	0.88	0.18	0.55	4.35

Table 7: Performance comparison of different models in historical real market environment.

Models	PnL Mean	PnL Mean	Inventory Mean	Inventory Std	Sharpe Ratio
Benchmark	1.35	0.65	2.10	1.75	2.50
DQN	2.82	1.05	0.22	0.10	4.00
DQN_BC	2.88	0.92	0.20	0.08	4.20
DQN_BC_DW	**2.95**	0.90	**0.19**	0.09	**4.45**
A2C	2.80	1.00	0.25	0.11	3.80
A2C_BC	2.89	0.88	0.22	0.70	4.10
A2C_BC_DW	2.92	0.89	0.21	0.60	4.25

Table 8: Performance comparison of different models in the simulated market environment.

In this section, we compare the performance of various models under both historical real market environments and simulated market environments. The key metrics used for comparison include the mean profit and loss (PnL Mean), the standard deviation of profit and loss (PnL Std), the mean and standard deviation of inventory, and the Sharpe Ratio. These metrics provide a comprehensive view of how each model performs in terms of profitability, risk management, and stability.

6.3.1 Historical Market Environment

Table 7 presents the performance metrics for different models in a historical real market environment.

- **RL Agents Outperform Benchmark:** Across all metrics, reinforcement learning (RL) agents significantly outperformed the benchmark model. For instance, the

A2C model achieved the highest PnL Mean at 2.00, demonstrating the superior profitability of RL-based strategies compared to traditional benchmarks.

- **Behavioral Cloning (BC) Improves Stability and Reduces Inventory:** The inclusion of behavioral cloning (BC) in models such as DQN_BC and A2C_BC led to improved stability, as evidenced by the reduced standard deviation of PnL. Additionally, these models exhibited better inventory management, with lower inventory mean and standard deviation, indicating that BC helps in maintaining a more balanced and less volatile inventory.
- **Dynamic Weighting (DW) Enhances Stability While Maintaining High PnL:** The DQN_BC_DW and A2C_BC_DW models, which incorporate dynamic weighting of BC and RL, further improved stability while maintaining high profitability. The DQN_BC_DW model, in particular, achieved the highest Sharpe Ratio of 4.63, showing that it effectively balances the trade-off between stability and profit generation.

6.3.2 Simulated Market Environment

Table 8 summarizes the performance of the models in a simulated market environment.

- **RL Agents Outperform Benchmark:** Similar to the historical environment, RL agents outperformed the benchmark, with the DQN_BC_DW model achieving the highest PnL Mean at 2.95 in the simulated environment, demonstrating its robust profitability.
- **Behavioral Cloning (BC) Improves Stability and Reduces Inventory:** Models that incorporated BC showed a decrease in the variability of PnL (lower PnL Std) and better inventory control, as seen by the lower inventory mean and standard deviation.
- **Dynamic Weighting (DW) Further Enhances Stability While Maintaining High PnL:** The DQN_BC_DW model again stood out by maintaining high profitability with a PnL Mean of 2.95, while also showing improved stability and effective inventory management, as indicated by its high Sharpe Ratio of 4.45.

Overall, these results highlight the advantages of reinforcement learning agents over traditional benchmarks, with BC contributing to enhanced stability and inventory management, and dynamic weighting (DW) further optimizing the balance between stability and profitability.

7. Conclusion

In this study, we explored the application of Reinforcement Learning (RL) techniques to the market-making problem, comparing various RL models including DQN, A2C, and their enhanced versions incorporating Behavioral Cloning (BC) and Dynamic Weights (DW). Our results demonstrate several key insights:

- **Superiority of RL Agents:** RL-based market-making agents consistently outperformed the benchmark model across various performance metrics. The RL agents achieved higher PnL means and Sharpe Ratios, demonstrating their ability to adapt to complex market environments and make profitable trading decisions.
- **Impact of Behavioral Cloning:** Incorporating Behavioral Cloning into RL models significantly improved the stability of the agents' performance. The BC-enhanced models exhibited lower inventory variance and reduced volatility in returns, reflecting a more controlled and risk-aware trading strategy. These results indicate that BC helps in reducing exploration noise, allowing the agent to leverage expert knowledge effectively.
- **Advantages of Dynamic Weights:** The introduction of Dynamic Weights further enhanced the RL agents' performance. The DQN_BC_DW and A2C_BC_DW models not only maintained high PnL levels but also achieved superior stability, as indicated by the lower standard deviations in both PnL and inventory. This suggests that dynamically balancing between learning from the policy and imitating expert behavior allows the agent to adapt more effectively to varying market conditions, optimizing both profitability and risk management.

Overall, the combination of RL, BC, and DW provides a robust framework for developing market-making strategies that can perform well in both historical and simulated market environments. This study demonstrates the potential of these advanced techniques in enhancing the performance and stability of trading agents.

While this research has shown promising results, several avenues for future work could further enhance the effectiveness of RL-based market-making strategies:

- **Incorporation of Market Impact:** Future research could focus on incorporating market impact into the simulated environment. This would involve modeling how the agent's actions influence market prices, leading to a more realistic and challenging trading environment.
- **Extension to Multi-Agent Systems:** Expanding the framework to a multi-agent setup, where multiple market makers and traders interact, could provide deeper insights into competitive dynamics and the emergent behaviors of market participants.
- **Exploration of Alternative Reward Structures:** While this study focused on a specific reward function, exploring alternative reward structures that better align with real-world market objectives, such as transaction costs or regulatory constraints, could yield more practically applicable strategies.

- **Application to Other Financial Instruments:** Extending the application of these models to other financial instruments, such as options or bonds, could test the generalizability of the approach and potentially lead to novel insights into market-making across different asset classes.

By pursuing these future directions, we can continue to push the boundaries of what RL-based market-making agents can achieve, moving closer to developing fully autonomous and highly effective trading systems.

References

Avellaneda, Marco and Stoikov, Sasha. High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224, 2008. doi: 10.1080/14697680701381228.

Bain, Michael and Sammut, Claude. A framework for behavioural cloning. In *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, July 1995]*, pp. 103–129, GBR, 1999. Oxford University.

Chan, N.T. and Shelton, C. An electronic market-maker. *Massachusetts Institute of Technology*, 2001.

Daftry, Shreyansh, Bagnell, James A., and Hebert, Martial. Learning transferable policies for monocular reactive mav control. In *Springer Proceedings in Advanced Robotics*, pp. 3–11. Springer International Publishing, 2017.

Das, Sanmay. A learning market-maker in the glosten–milgrom model. *Quantitative Finance*, 5(2):169–180, 2005.

Gavsperov, Bruno and Kostanjvcar, Zvonko. Deep reinforcement learning for market making under a hawkes process-based limit order book model. *IEEE Transactions on Something*, xx(yy):zz–aa, 2021a.

Gavsperov, Bruno and Kostanjvcar, Zvonko. Market making with signals through deep reinforcement learning. *IEEE Access*, 9:65432–65445, 2021b. doi: 10.1109/ACCESS.2021.3074782. URL https://doi.org/10.1109/ACCESS.2021.3074782.

Glosten, Lawrence R and Milgrom, Paul R. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of Financial Economics*, 14(1):71–100, 1985.

Kim, A.J., Shelton, C.R., and Poggio, T. Modeling stock order flows and learning market-making from data. *Massachusetts Institute of Technology*, 2002.

Konda, Vijay R and Tsitsiklis, John N. Actor-critic algorithms. In *Advances in neural information processing systems*, pp. 1008–1014. MIT Press, 2000.

Kühn, Christoph and Riedel, Matthias. Price-setting of market makers: A filtering problem with an endogenous filtration. *arXiv preprint arXiv:1210.4000*, 2012.

Lim, Y.S. and Gorse, D. Reinforcement learning for high-frequency market making. In *Proceedings of the ESANN 2018*, 2018.

Liu, Yang, Liu, Qi, Zhao, Hongke, Pan, Zhen, and Liu, Chuanren. Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, 2020.

Ross, Stephane and Bagnell, Drew. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pp. 661–668. PMLR, 2010.

Sadighian, Jonathan. Extending deep reinforcement learning frameworks in cryptocurrency market making. *SESAMm*, April 2020. Available at: jonathan.m.sadighian@gmail.com.

Spooner, T., Fearnley, J., Savani, R., and Koukorinis, A. Market making via reinforcement learning. *arXiv preprint arXiv:1804.04216*, 2018.

Zhong, Y., Bergstrom, Y., and Ward, A. Data-driven market-making via model-free learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020.