Final Report

COMP 4801

Final Year Project

Market Making with Deep Reinforcement Learning

Student: Aidan Wong Weng Seng
UID: 3035868918

Supervisor:

Prof. Huang Zhiyi

Abstract

Market making is recognised as a critical function in financial markets, providing liquidity and facilitating smoother trading through the continuous buying and selling of securities. Traditionally, market making has relied on rule-based models grounded in strong market assumptions, such as the Glosten and Milgrom model [1], and more recently, the widely respected Avellaneda and Stoikov framework [2]. With the rapid advancement of artificial intelligence and machine learning, particularly deep learning, new opportunities have emerged for transforming financial decision-making. Techniques such as Long Short-Term Memory (LSTM) networks [3] and Soft Actor-Critic (SAC) reinforcement learning, when combined with order book dynamics, offer powerful tools for modelling complex market behaviour in continuous action spaces.

In this project, the stochastic control problem of market making is addressed through the development of a deep reinforcement learning agent capable of capturing intricate patterns in the limit order book. This objective is achieved through two main components: the environment in which the agent is trained and the agent itself. The limit order book environment is modelled using a Non-linear Multivariate Hawkes process, grounded in the weakly consistent LOB model of Law and Viens [19], and constructed using the OpenAI Gymnasium library. Three agent configurations are developed and evaluated: a standalone SAC agent, an SAC agent augmented with an LSTM network, and an SAC agent further enhanced with an Attention Mechanism.

The SAC agent demonstrates clear improvement over a fixed baseline quoting strategy, validating the viability of continuous-action deep reinforcement learning in a realistic LOB environment. However, the LSTM and attention-augmented models do not yield consistent improvement over the standalone SAC agent, attributed to the limited dimensionality of the observation space. An inverse relationship is observed between reward performance and inventory risk management across the three models, representing an important empirical finding. These results highlight the potential of deep reinforcement learning for intelligent market making, while identifying enriched state representations and multi-agent settings as key directions for future research.

Acknowledgement

Gratitude is sincerely expressed to my brother and sister, Bryan and Chloe, for their tremendous assistance and support throughout this project. Their unwavering encouragement has been vital to the successful completion of this work.

Additionally, appreciation is extended to my supervisor, Dr. Huang, for his efforts and guidance during this project. His comments and insights have proven invaluable in the development and refinement of the research.

Table of Contents

List of Figures

List of Tables

List of Equations

Abbreviations

MM	Market Making
LOB	Limit Order Book
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
MDP	Markov Decision Processes
DNN	Deep Neural Networks
LLM	Large Language Models
DDPG	Deep Deterministic Policy Gradient
LSTM	Long Short-Term Memory
DQN	Deep Q-Network
RNN	recurrent neural networks
PnL	Profit and Loss
ML	Machine Learning
SAC	Soft Actor Critic

1 Introduction

1.1 Project Background

Market making is a foundational function in financial markets, essential for maintaining liquidity and ensuring the continuous flow of trades. Market makers actively quote both buy and sell prices for financial instruments, thereby reducing transaction costs and enabling smoother execution for other market participants. By standing ready to transact at publicly posted prices, they help narrow bid-ask spreads, absorb temporary imbalances in supply and demand, and stabilize price movements.

Market makers typically generate profits through two primary mechanisms: capturing the bid-ask spread and engaging in high-frequency trading strategies. The spread represents the difference between the prices at which they buy and sell securities, and its efficient management is central to their profitability. In modern electronic markets, many market makers operate algorithmically, using sophisticated models to adjust quotes dynamically in response to real-time market conditions, order flow, and inventory levels.

Their role becomes especially critical during periods of market stress or volatility, where their ability to provide liquidity can prevent sharp price dislocation and support overall market resilience. A key element that Market makers utilize to gain understanding of financial market condition is the Limit Order Book (LOB).

The Limit Order Book is a fundamental component of financial markets. It records all outstanding buy (bid) and sell (ask) orders that have not yet been executed, offering a real-time snapshot of market activity across various price levels and depths. Researchers study the Limit Order Book extensively to understand price action, liquidity, and market microstructure. By analyzing order flow, queue dynamics, and trade execution patterns, they uncover how prices evolve in response to supply and demand imbalances.

For example, Hasbrouck [10] provides a comprehensive framework for interpreting Limit Order Book data and its role in price formation. Cont., Stoikov, and Talreja [9] developed a stochastic model that captures empirical properties of high-frequency trading behavior. Xie et al. [11] applied Markov queue theory to model the Limit Order Book and optimize market maker strategies in China's A-share market. These studies demonstrate how the Limit Order Book serves as a rich source of information for modelling short-term price movements, estimating volatility, and designing trading algorithms.

Among the many emerging technologies in today’s age. One can leverage the capability of Deep Reinforcement Learning (DRL) to help better understand the LOB dynamics. DRL is the merger of two powerful techniques. The first is Reinforcement Learning (RL), a trial-and-error-based approach for modelling and making sequential decisions under uncertainty. It has been

widely used to solve stochastic optimal control problems framed as a Markov Decision Processes (MDPs).

The second is Deep Neural Networks (DNNs), which play a crucial role in today's world — especially with the rise of Large Language Models (LLM) that have significantly influenced modern society. These networks, composed of multiple layers, can learn complex patterns and representations from data, allowing them to approximate functions that map inputs to outputs with high accuracy.

With DRL, many achievements have been seen, such as mastering the ancient board game of Go through AlphaGo's neural network and tree search architecture [4], optimizing multi-agent traffic navigation via crowdsourced hyperparameter tuning in the Deep Traffic project [5], and enabling continuous control in robotic systems using the Deep Deterministic Policy Gradient (DDPG) algorithm [6].

Acknowledging these advances in DRL, this research develops a deep reinforcement learning-based market-making model that leverages Long Short-Term Memory (LSTM) networks with an Attention Mechanism within a Soft Actor-Critic (SAC) framework. The model is designed to observe and learn temporal patterns in limit order book dynamics, enabling the agent to capture complex market behaviours and execute optimal quoting strategies in a continuous action space — a capability that discrete action frameworks such as DQN cannot natively support.

Current academic progress of machine learning in the financial industry has seen significant growth across various domains. Oyewola, Akinwunmi, and Omotehinwa employed Deep LSTM Q-learning to analyze price movements in the oil and gas sector [12]. Wang explored the integration of deep learning techniques to estimate fill probabilities within a Limit Order Book (LOB) [13]. In the area of optimal market making, several researchers have contributed notable advancements. Lim and Gorse claim to be pioneers in this field, having constructed a market-making agent trained using the LOB framework proposed by Cont., Stoikov, and Talreja [9], as detailed in their reinforcement learning study [14]. Gasperov and Kostanjčar extended this work by incorporating a more realistic LOB model based on Hawkes processes [15]. Kumar proposed a novel deviation by constructing a real-time exchange simulation of the LOB, moving beyond traditional statistical modelling [16].

Despite these contributions, no existing research has yet combined a Hawkes-based LOB model with a deep reinforcement learning (DRL) framework that utilizes LSTM architecture. This research paper aims to explore such integration and evaluate its performance against traditional market-making models, such as the one proposed by Avellaneda and Stoikov [2].

1.2 Project Motivation

With the rapid advancement of artificial intelligence and machine learning, deep learning has emerged as a transformative tool in the financial sector. This research aims to leverage these

technological advancements to address the shortcomings of conventional market-making strategies, which rely on rigid rule-based systems and fixed market conditions—often criticized for their simplistic and overly optimistic assumptions. The objective is to enhance trading strategies, improve quoting accuracy, and optimize inventory risk management.

A key breakthrough in this area was demonstrated by Lim and Gorse, who showed that reinforcement learning using a simple discrete Q-learning algorithm outperformed the widely respected Avellaneda and Stoikov model, which has long served as a benchmark in the field [14]. Building on this, Oyewola, Akinwunmi, and Omotehinwa introduced Long Short-Term Memory (LSTM) networks into a DRL framework, yielding promising results in capturing temporal dependencies in financial data. These developments have paved the way for modelling the Limit Order Book (LOB) using Hawkes processes to train DRL agents that leverage LSTM architectures. This approach aims to capture time-dependent patterns in LOB dynamics and address the challenge of optimal market making more effectively.

Through this research, it is hoped that the adoption of more advanced machine learning techniques will be inspired by demonstrating their potential applications within the financial industry.

1.3 Project Objective

The primary objective of this research is to design and implement an intelligent agent-based market maker using deep reinforcement learning techniques, capable of interpreting and reacting to the complex dynamics of a financial limit order book. The agent is trained to analyse key market microstructure features including order flow dynamics, bid-ask spread, and inventory position. The overarching goal is to build a robust market-making model that leverages Long Short-Term Memory (LSTM) networks with an Attention Mechanism to capture temporal dependencies in order book movements, while employing a Soft Actor-Critic (SAC) framework to learn optimal quoting strategies in a continuous action space. The agent aims to maximise profitability, maintain quoting precision, manage inventory efficiently, and mitigate exposure to market risk.

1.4 Project Scope and Deliverables

The primary deliverable of this study is a comprehensive research report that compares a modernised market-making approach, powered by deep reinforcement learning (DRL), with a fixed baseline quoting strategy. To achieve this, the project is organised around six key milestones: first, a literature review on stochastic models for the limit order book (LOB) is conducted; second, parameter estimation for the LOB Hawkes model takes place. Next, a simulated training environment is developed using the OpenAI Gymnasium library, followed by the development and evaluation of three agent configurations: a standalone SAC agent, an SAC agent augmented with an LSTM network, and an SAC agent further enhanced with an Attention Mechanism. The project then includes benchmarking of all three agents against the fixed baseline quoting strategy, culminating in a comprehensive analysis of reward

performance, inventory risk management, and the trade-offs between model complexity and profitability.

1.5 Report Outline

Section 2 outlines the methodology, detailing the models, tools, and rationale behind the experimental setup. Section 3 presents the implementation of both the environment and the three agent configurations developed in this research. Section 4 presents experimental results, including a hyperparameter study, training convergence analysis, and a comprehensive evaluation of agent performance. Section 5 concludes the study with a summary of key findings and directions for future research.

2 Project Methodology

In the realm of reinforcement learning, two fundamental components are essential: the environment and the agent. This section will elucidate the specific characteristics and rationale underlying the selection of the chosen environment.

2.1 Environment

The construction of the agent simulation environment for repetitive training is modelled based on the Hawkes Process theory proposed by Lu and Abergel [18]. This model was selected because it effectively replicates real-world continuous order book dynamics by capturing short-term dependencies and changes in the limit order book, unlike the model by Cont., Stoikov, and Talreja [9], which does not account for time-dependent properties of the financial market. Consequently, in comparison to traditional stochastic limit order book models, the Hawkes Process model provides a more accurate representation of the real-time limit order book by capturing both excitation and inhibition effects among different types of orders, thus offering a more realistic depiction of market behaviour.

As shown in Figure 1, reproduced from [18], this diagram highlights the significance of employing the Hawkes process in modelling the order book. The Hawkes process matches closely with the theoretical order book reaction, whereas the independent Poisson process deviates more strongly from the theoretical model. This result serves as a strong justification for modelling the order book using the theory proposed by Lu and Abergel [18].

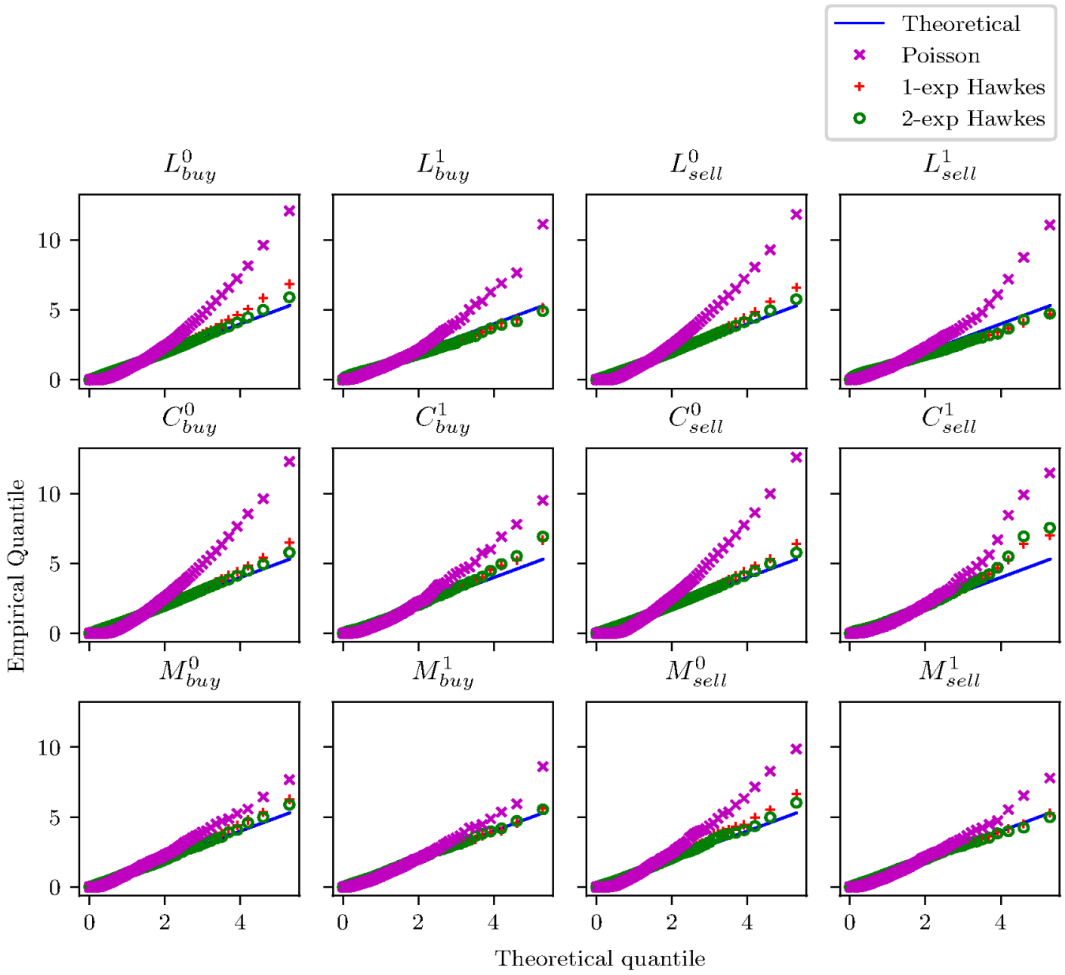

Figure 1: Q-Q plot goodness of fit tests for order book. [18]

To implement the environment, the OpenAI Gymnasium library is utilised. This open-source Python module enables researchers and data scientists to build reinforcement learning environments with ease, providing easily integrated custom environments that allow benchmarking across a wide range of tasks and comparing performance using consistent metrics. This enables accurate and consistent comparisons of different agent models and market-making strategies, which is essential for validating new approaches.

2.2 Agent

To construct a reliable and effective model, deep learning networks are incorporated into the framework. These additional layers possess the capability to learn intricate patterns from order book dynamics, thereby enabling the agent to identify hidden temporal dependencies between each action.

The agent is built using LSTM layers within a Soft Actor-Critic (SAC) framework to capture long and short-term dependencies present in the training data. LSTM layers offer the agent the capability of maintaining a memory of past observations, providing more context through each

decision-making process. This further enhances the agent's capability of identifying hidden temporal relationships between different past and present states.

Financial markets are widely argued to contain high levels of noisy data, often influencing model decision-making. Introducing LSTM with an Attention Mechanism improves performance by allowing the network to focus on the most relevant parts of the input sequence. Proposed by Sang and Li [8], this model has yielded positive results in modelling temporal dependencies in financial markets. As shown in Figures 2a and 2b, Sang and Li [8] demonstrate that an LSTM with an attention mechanism outperforms the normal LSTM in capturing the predicted closing price.

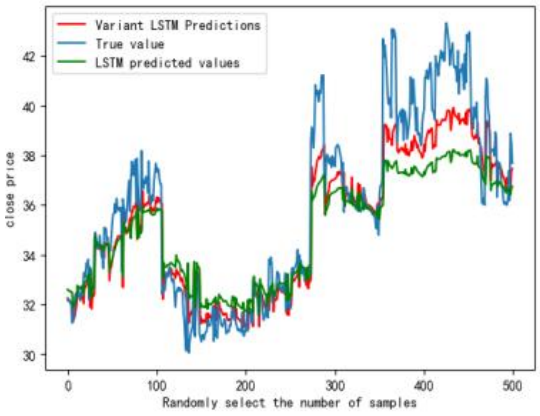

Figure 2a: LSTM [8]

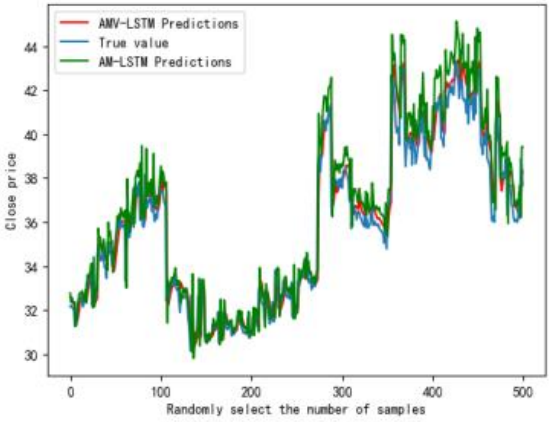

Figure 2b: LSTM with Attention [8]

When modelling the reward function, the agent considers factors such as profit and loss (PnL) and inventory risk to guide optimal quoting decisions. The Constant Absolute Risk Aversion (CARA) utility function, implemented by [14], is used to shape the reward structure. This approach introduces risk sensitivity into the agent's decision-making process and serves as the foundation for constructing the policy that drives optimal market-making behaviour.

3 Implementation

As outlined in the methodology section, there are two primary components in reinforcement learning research that are essential: the environment in which the agent trains and the agent itself. In the following section, the development of the agent will be elaborated upon as part of the current progress.

3.1 Environment Development

Following the completion of the initial literature review, attention is directed toward the development of the environment. Given the financial structure of a limit order book, accurate modeling of this system is essential for replicating the real-world financial dynamics present in the equities market.

To facilitate this, the Hawkes process has been employed to mathematically model the arrival of orders in a mutually dependent manner. Subsequently, the environment was coded, as illustrated in Figure 3 below.

environment.py > Environment > event_occurred

```
# importing required packages
import numpy as np
import random
import matplotlib.pyplot as plt
import gymnasium as gym
import stable_baselines3 as sb3
from stable_baselines3 import SAC, TD3
import scipy.stats as ss
from gymnasium import spaces, logger
import tensorflow as tf
import yaml

class Environment(gym.Env):

    def __init__(self, config): ···

    def initiate_config(self): ···

    def initiate_parameters(self): ···

    def order_validity_check(self, propsed_bid, propsed_ask): ···

    def update_state(self, order_type): ···

    def event_occurred(self, event_type):

        # 0 : Market Buy Aggressive
        # 1 : Market Sell Aggressive
        # 2 : Limit Buy
        # 3 : Limit Sell
        # 4 : Cancel Buy
        # 5 : Cancel Sell
        # 6 : Market Buy Non-Aggressive
        # 7 : Market Sell Non-Aggressive

        if event_type == 0:
            jumpsize = np.random.exponential(self.exp_dispar) + 0.01
            self.ask_price += jumpsize
            self.mid_price += (self.ask_price + self.bid_price) / 2
            self.spread = np.round(self.ask_price - self.bid_price, 2)
            self.intensities += np.array(self.excitation_matrix[0])

            if self.outstanding_ask_order < self.ask_price:
                prob_execution = 1 # as we do not have order book depth, assume full execution
```

Figure 3: Code of Environment

As demonstrated above, the current progress in mathematically modeling the order arrivals within the limit order book has been achieved. Furthermore, the code for the environment has been completed using the OpenAI Gym library.

The simulation environment is implemented as a custom OpenAI Gymnasium environment providing a realistic LOB framework for agent training. The environment models continuous-time market dynamics across an episode of length T, discretised into time steps of size dt. At

each step, the agent places limit orders, market events occur stochastically via the Hawkes process, and the environment transitions to the next state.

3.1.1 Order Book Dynamics – Nonlinear Multivariate Hawkes Process

Order arrivals are governed by a **Non-linear Multivariate Hawkes Process**. The arrival intensity of order type i at time t is defined as:

$$\lambda_i(t) = \phi_i\left(\mu_i + \sum_{j=1}^{d} \int_0^t h_{ij}\,(t-s)\,dN_j(s)\right)$$

Equation 1: Non-linear Multivariate Hawkes Process Intensity Function

where μ_i is the baseline arrival rate of order type i, φ_i is a non-linear link function, and $h_{ij}(t-s)$ is the kernel function capturing the influence of past events of type j on future arrivals of type i. The kernel takes an exponential form:

$$h_{ij}(t) = \alpha_{ij} e^{-\beta_{ij} t} \cdot \mathbf{1}_{\{t \geq 0\}}$$

Equation 2: Exponential Kernel Function

where α_{ij} is the excitation coefficient (the excitation matrix entry) and β_{ij} is the exponential decay rate. This formulation captures both **excitation** (an aggressive market buy triggering further buy activity) and **inhibition** effects across the 8 mutually dependent order types modelled:

Index	Order Type
0	Market Buy Aggressive
1	Market Sell Aggressive
2	Limit Buy
3	Limit Sell
4	Cancel Buy
5	Cancel Sell
6	Market Buy Non-Aggressive
7	Market Sell Non-Aggressive

Table 1: Order Event Types

Between agent steps, events are sampled using **Ogata's thinning algorithm**, which accepts or rejects candidate events based on the ratio of current to previous total intensity — ensuring statistical correctness of the simulated point process. The inter-event time is drawn from an exponential distribution with rate $\Sigma\lambda_i(t)$.

3.1.2 Price Dynamics – Jump Process

The mid-price evolves according to a **jump process** driven by aggressive market orders:

$$P_t = P_0 + \left(\sum_{e \in \mathbb{E}_{\text{inc}}} J_e - \sum_{e \in \mathbb{E}_{\text{dec}}} J_e\right) \cdot \frac{\delta}{2}$$

Equation 3: Jump process of price from aggressive orders

where $\mathbb{E}$_inc is the set of price-increasing events (aggressive market buys), $\mathbb{E}$_dec is the set of price-decreasing events (aggressive market sells), J_e is the jump size drawn from an exponential distribution with parameter exp_dispar, and $\delta/2$ scales the half-spread impact per jump. This ensures that price movements are directly coupled to aggressive order flow — correcting the independence assumption in the original Avellaneda-Stoikov model, as formalised by Law & Viens [19].

3.1.3 State and Action Space

State Space:
The agent observes a 3-dimensional continuous state vector, normalised to [−1, 1]:

#	Feature	Formula	Intuition
1	Inventory ratio	q / q_max	Normalised long/short position
2	Spread	s / 10	Current bid-ask spread
3	Order flow imbalance	$(\lambda_0 + \lambda_6 - \lambda_1 - \lambda_7) / 5$	Net buy vs sell pressure from Hawkes intensities

Table 2: State Space

The **order flow imbalance** feature is particularly important — derived directly from the Hawkes intensities, it encodes the current directional pressure of the market, giving the agent a forward-looking signal about likely price movements.

Action Space:
The action space is **continuous** with two outputs in [−1, 1]:

#	Action	Interpretation
a_1	Ask offset	Posted ask = best ask + a_1
a_2	Bid offset	Posted bid = best bid − a_2

Table 3: Action Space

The agent quotes relative to the current best bid/ask rather than absolute prices, making the policy scale-invariant. This continuous action design is a primary motivation for choosing **SAC over DQN** — DQN cannot natively represent this quoting decision without discretisation that

loses precision. An **order validity check** prevents inventory breaches and self-crossing orders at every step.

Episode Structure:
Each episode runs from t = 0 to t = T. At each discrete step dt, the agent: (1) cancels outstanding orders, (2) places new bid/ask limit orders, (3) checks for immediate order crossing (which triggers a market order), and (4) processes all Hawkes-driven events occurring within the dt window via Ogata thinning. The episode terminates when t ≥ T.

Reward Function
The reward is the change in **mark-to-market wealth** minus a continuous inventory risk penalty:

$$r_t = (C_t + q_t \cdot S_t) - (C_{t-1} + q_{t-1} \cdot S_{t-1}) - \gamma \cdot \Delta t \cdot | q_t |$$

Equation 4: Reward function

where C_t is cash, q_t is inventory, S_t is the mid-price, γ is the risk aversion parameter, and Δt is elapsed time. The penalty term $\gamma \cdot \Delta t \cdot |q_t|$ penalises holding large inventory over time, incentivising balanced two-sided quoting. Executed maker orders earn a fee rebate (mm_fee) while taker executions incur a fee (mt_fee), consistent with real exchange fee structures.

3.2 Agent Model

The agent is responsible for learning an optimal market-making policy by interacting with the Hawkes process LOB environment described in Section 3.1. The agent development follows a progressive architecture: beginning with a standalone Soft Actor-Critic (SAC) agent, extended with a Long Short-Term Memory (LSTM) network, and finally augmented with an Attention Mechanism. Each addition is motivated by a specific limitation of the preceding model, forming a natural ablation study.

3.2.1 Why SAC over DQN

Prior work in reinforcement learning for market making, such as Lim and Gorse [14], employed Deep Q-Networks (DQN) with a discrete action space. However, the market-making problem in this research requires the agent to continuously quote bid and ask offsets — a fundamentally continuous action space that DQN cannot represent without lossy discretisation.

Soft Actor-Critic (SAC) [20] is an off-policy, maximum entropy deep reinforcement learning algorithm designed specifically for continuous action spaces. It optimises a modified objective that balances expected cumulative reward with policy entropy:

$$J(\pi) = \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|\, s_t))]$$

Equation 5: SAC objective function

where α is the temperature parameter controlling the trade-off between reward maximisation and entropy $H(\pi(\cdot|s_t))$. The entropy term encourages the agent to maintain a stochastic policy, which provides three key advantages in the LOB setting:

- **Continuous action space**: SAC outputs a Gaussian distribution over bid/ask offsets, enabling precise, fine-grained quoting decisions
- **Exploration under partial observability**: The LOB environment is partially observable — the agent cannot see the full order book depth. Entropy maximisation naturally encourages broader exploration under this uncertainty, preventing premature convergence to suboptimal quoting strategies
- **Sample efficiency**: SAC is off-policy and uses an experience replay buffer, making it significantly more sample efficient than on-policy alternatives such as PPO, which is important given the computational cost of Hawkes process simulation

The entropy temperature α is automatically tuned during training by minimising:

$$J(\alpha) = \mathbb{E}_{a_t\sim\pi}[-\alpha\log\,\pi(a_t \mid s_t) - \alpha\bar{\mathcal{H}}]$$

Equation 6: SAC Entropy temperature update

where $\bar{H}$ is the target entropy, typically set to $-\dim(A)$. As observed in the training results, α decays from 1.0 to near zero over 100,000 update steps, confirming that the agent transitions from broad exploration early in training to a more deterministic exploitation policy as it converges.

3.2.2 Baseline Agent — Fixed Quote Policy

Before training a learning-based agent, a **fixed quote baseline** is established as the performance benchmark. The baseline agent quotes symmetrically at a fixed offset of zero on both sides, i.e. action = [0, 0], meaning it always posts its sell order at exactly the best ask and its buy order at exactly the best bid without any adaptive adjustment.

This baseline reflects a naive top-level quoting strategy — one that provides liquidity at the tightest possible spread but applies no intelligence to inventory management, order flow signals, or market conditions. It serves as the lower bound against which the learning-based SAC agent and its extensions are evaluated. Any improvement over this baseline can be directly attributed to the agent's learned policy rather than the structure of the environment.

3.2.3 SAC

The SAC agent uses an Actor-Critic architecture comprising three networks: a policy network (Actor), and two Q-value networks (Critics) with a shared target network structure for stable training.

Actor Network:

The Actor takes the current state s_t as input and outputs the parameters of a Gaussian distribution over actions:

$$\pi_\phi(a_t \mid s_t) = \mathcal{N}(\mu_\phi(s_t), \sigma_\phi(s_t))$$

Equation 7: SAC actor Gaussian policy

Actions are sampled via the reparameterisation trick and passed through a tanh squashing function to enforce the [−1, 1] action bounds. The actor is updated by maximising:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}}\left[\alpha \log\, \pi_\phi(a_t \mid s_t) - Q_\theta(s_t, a_t)\right]$$

Equation 8: SAC actor update

Critic Network

Two Q-networks are maintained to mitigate overestimation bias via the clipped double-Q trick. Each critic is updated by minimising the Bellman residual:

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}}[(Q_\theta(s_t, a_t) - y_t)^2]$$

Equation 9: SAC critic loss

where the target is:

$$y_t = r_t + \gamma \left(\min_{\theta'}\, Q_{\theta'}(s_{t+1}, a_{t+1}) - \alpha \log\, \pi(a_{t+1} \mid s_{t+1})\right)$$

Equation 10: SAC critic Bellman target

Implementation and Hyperparameters

The SAC agent is implemented using the Stable Baselines3 library [21] with an MlpPolicy. Target networks are updated via soft updates with coefficient τ:

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$$

Equation 11: SAC target network soft update

Hyperparameter	**Value**
Policy	MlpPolicy
Discount Factor γ	0.90

Batch Size	512
Learning Rate	3×10^{-4}
Soft Update τ (default)	0.005
Replay Buffer Capacity (default)	1,000,000
Warm-up Steps (default)	100
Updates per Step (default)	1

Table 4: SAC Hyperparameter

*All remaining hyperparameters follow Stable Baselines3 defaults.

3.2.4 SAC with LSTM

A fundamental limitation of the baseline SAC agent is that it is **memoryless** — it treats each state observation independently, with no access to the history of market events. In practice, LOB dynamics are strongly sequential: the current order flow imbalance, spread, and inventory position carry more predictive power when interpreted in the context of recent history. A single-step observation is insufficient to capture regime changes such as trending versus mean-reverting market conditions.

To address this, an LSTM network is introduced to wrap the SAC policy, maintaining a hidden state h_t that summarises relevant past observations. The LSTM updates its hidden state according to the standard gating equations:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \text{(forget gate)}$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \text{(input gate)}$$
$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \text{(candidate cell)}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \text{(cell state)}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), h_t = o_t \odot \tanh(c_t) \text{(output)}$$

Equation 12: LSTM

where x_t is the current state observation and h_t is the LSTM hidden state passed to the SAC Actor and Critic networks in place of the raw state.

The LSTM hidden state is the primary input to both the Actor and Critic, allowing the policy to condition quoting decisions on the accumulated history of market observations. This is particularly valuable in the Hawkes process environment, where intensities evolve continuously and the memory of past order arrivals directly informs the current $\lambda_i(t)$.

Training recurrent policies requires **Truncated Backpropagation Through Time (TBPTT)**, where gradients are computed over a fixed sequence window rather than the full episode. A sequence length of 16 steps is used with a burn-in period of 8 steps, during which the LSTM hidden state is initialised without gradient computation. This prevents the initial random hidden

state from corrupting gradient estimates while still allowing the LSTM to build meaningful context before learning begins.

3.2.5 SAC with LSTM with Attention Mechanism

While the LSTM provides the agent with sequential memory, it weights all past timesteps through its internal gating mechanism without explicitly distinguishing which observations are most relevant to the current quoting decision. Financial market data contains significant noise, and not all historical states contribute equally to predicting optimal quote placement.

An **Attention Mechanism** is introduced on top of the LSTM to allow the agent to selectively focus on the most informative parts of the input sequence. Given the sequence of LSTM hidden states $H = \{h_1, h_2, ..., h_t\}$, attention scores are computed as:

$$e_k = \text{score}(h_t, h_k) = \frac{h_t^\top W_a h_k}{\sqrt{d_k}}$$

$$\alpha_k = \frac{\exp(e_k)}{\sum_{j=1}^{t} \exp(e_j)}$$

$$c_t = \sum_{k=1}^{t} \alpha_k \, h_k$$

Equation 13: Attention mechanism

where e_k is the raw attention score between the current hidden state h_t and past hidden state h_k, α_k is the normalised attention weight, d_k is the key dimension for scaling, and c_t is the context vector — a weighted summary of the most relevant historical states. This context vector c_t is concatenated with the current hidden state h_t and passed to the SAC Actor and Critic networks.

The key benefits over plain LSTM are twofold. First, the agent can directly attend to distant but relevant market events — for example, a large aggressive order that occurred several steps ago may still be highly predictive of current market dynamics in the Hawkes process, and attention allows the agent to retrieve this signal explicitly.

Second, the attention weights α_k provide a degree of **interpretability**, revealing which past market states the agent prioritises when making quoting decisions — a valuable property for understanding learned market-making behaviour.

All hyperparameters remain identical to the SAC+LSTM configuration. The attention layer adds minimal computational overhead while providing a meaningful inductive bias suited to the sequential, noisy nature of LOB data.

3.2.6 Summary of Agent Progression

Component	Limitation Addressed	Key Benefit
Baseline [0,0]	—	Benchmark fixed quoting strategy
SAC	Discrete action space of DQN	Continuous adaptive quoting, entropy-driven exploration
SAC + LSTM	Memoryless single-step observation	Sequential memory of order flow history
SAC + LSTM + Attention	Uniform treatment of all past states	Selective focus on most relevant market events

Table 5: Agent Summary

3.3 Difficulties encountered & Mitigations

3.3.1 Difficulties

Modeling the environment represents a challenging endeavor, particularly due to the necessity of ensuring that a flawed or erroneous environment is not constructed, as such a misstep could distort return results. Establishing that the mathematical model meets the required standards of robustness is essential, thereby enabling effective training of the agent and the production of accurate outcomes. As noted by Law and Viens [19], much of the current research on market-making approaches remains inconsistent with respect to direction, timing, and volume, which can result in illusory gains in back testing results.

3.3.2 Mitigations

Utilizing the research conducted by Law and Viens [19], a weakly consistent limit order book model has been developed, capturing the interdependent dynamics between price fluctuations and limit order book behavior, specifically with respect to direction and timing.

The erroneous assumption in the original Avellaneda and Stoikov model [2], which posits that price movements are independent of the arrival of market orders and limit order book dynamics, has been rectified in the weakly consistent pure jump market model proposed by Law and Viens [19]. Consequently, the environment has been constructed based on the proposed model by Law and Viens [19].

4 Experiments and Results

4.1 Experimental Setup

All experiments are conducted within the Hawkes process LOB environment described in Section 3.1. Training and evaluation are performed separately — the agent is trained over a fixed number of timesteps and subsequently evaluated over 1,000 episodes using a fixed, frozen policy.

Two evaluation modes are reported for the SAC-based models: a **deterministic policy**, where the actor outputs the mean action $\mu_\phi(s_t)$ directly without sampling, and a **stochastic policy**, where actions are sampled from the full Gaussian distribution $\pi_\phi(a_t|s_t)$. The deterministic policy reflects the agent's best estimate of the optimal action, while the stochastic policy captures the distribution of outcomes under the learned exploration behaviour.
The following metrics are reported across all models:

Metric	Description
Total Episode Reward	Primary performance metric; cumulative reward over an episode reflecting mark-to-market PnL minus inventory risk penalty
Wealth	Terminal cash position + mid-price*Terminal inventory
Inventory	Mean absolute inventory of each episode

Table 6: Performance Metrics

Total episode reward serves as the headline comparison metric, as it already incorporates inventory valuation via the mid-price throughout the episode. Wealth is reported as a complementary terminal metric, capturing the agent's final mark-to-market position by accounting for both realised cash and the value of any remaining inventory at the closing mid-price. This provides a more complete picture of profitability than cash alone, as an agent ending with high cash, but a large unhedged inventory position would still reflect this exposure in the wealth figure. Mean absolute inventory is treated as a diagnostic metric, measuring the average positional risk carried by the agent throughout each episode rather than at a single terminal snapshot.

Training timesteps differ across models to account for the increased complexity of recurrent architectures:

Model	Training Timesteps
Baseline [0,0]	— (no training)
SAC	100,000
SAC + LSTM	200,000
SAC + LSTM + Attention	200,000

Table 7: Model training timestep

SAC+LSTM and SAC+LSTM+Attention were trained for 200,000 timesteps as preliminary runs at 100,000 timesteps yielded insufficient convergence, likely due to the additional complexity of learning recurrent representations alongside the policy.

4.2 Hyperparameter Study — Discount Factor γ

The discount factor γ controls the agent's effective planning horizon. A value of γ = 0.90 corresponds to a relatively short horizon of approximately 10 steps, whereas γ = 0.99 extends this to roughly 100 steps. In the context of market making, the agent must manage inventory risk over an entire episode, and a short γ causes the agent to underweight future inventory penalties, potentially leading to myopic quoting behaviour that sacrifices long-term profitability for short-term spread capture.

To validate this, the SAC agent was trained under both γ = 0.90 and γ = 0.99, with all other hyperparameters held constant. The results demonstrate that γ = 0.99 yields higher and more stable episode rewards throughout training, confirming that a longer planning horizon is more appropriate for this problem. All subsequent experiments therefore adopt γ = 0.99.

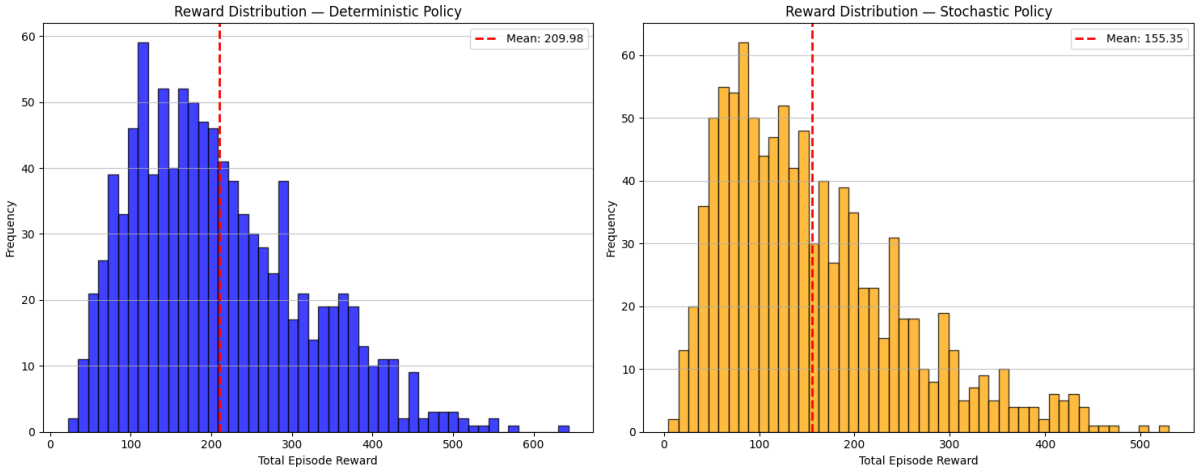

Figure 4: SAC with γ = 0.90

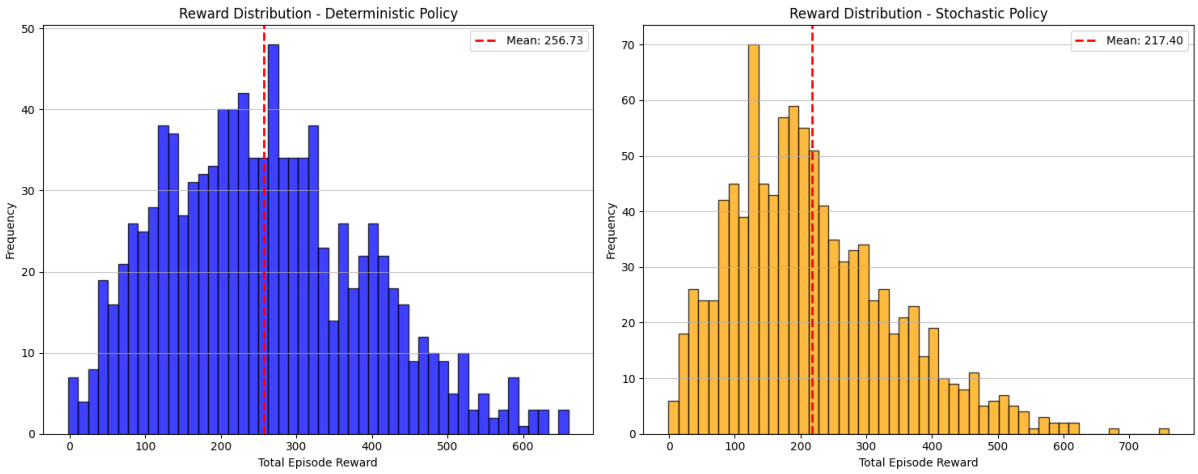

Figure 5: SAC with $\gamma = 0.99$

The results demonstrate that $\gamma = 0.99$ yields higher and more stable episode rewards, confirming that a longer planning horizon is more appropriate for this problem. All subsequent experiments adopt $\gamma = 0.99$.

4.3 Training Convergence

Training curves for all three learned agents are presented in Figures 6, 7, and 8 respectively. Each curve plots the smoothed episode reward (window = 50) alongside critic loss, actor loss, and entropy temperature α over the course of training.

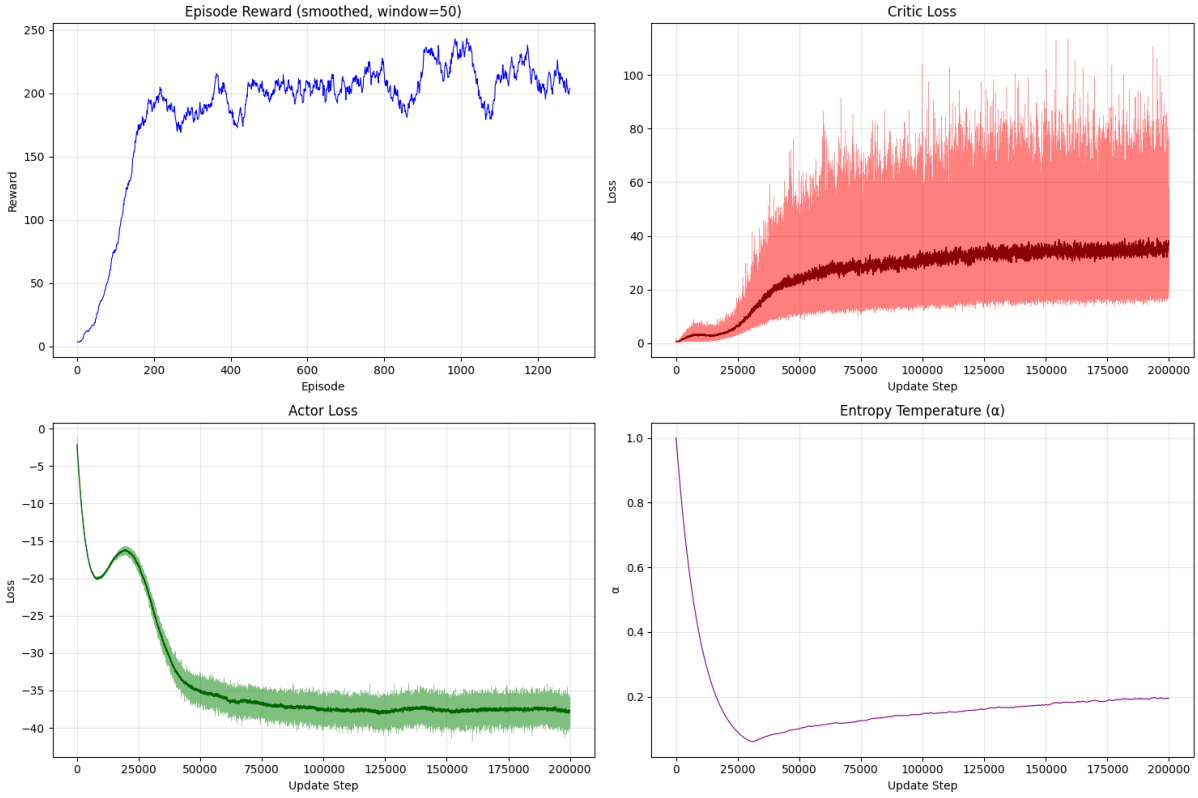

Figure 6: SAC training curves

SAC (Figure 6) demonstrates the strongest and most stable convergence of the three models. The episode reward rises sharply from near zero to approximately 175–200 within the first 200 episodes, before plateauing and oscillating stably around 200 for the remainder of training — reaching a peak of ~230. The critic loss stabilises around 30–35 after 50,000 update steps, the actor loss converges smoothly to approximately −37 by 50,000 steps and remains stable thereafter, and the entropy temperature α decays steadily from 1.0 to ~0.2, indicating the agent settles into a near-deterministic but not fully collapsed policy. Overall, SAC exhibits clean, well-behaved convergence within 200,000 timesteps.

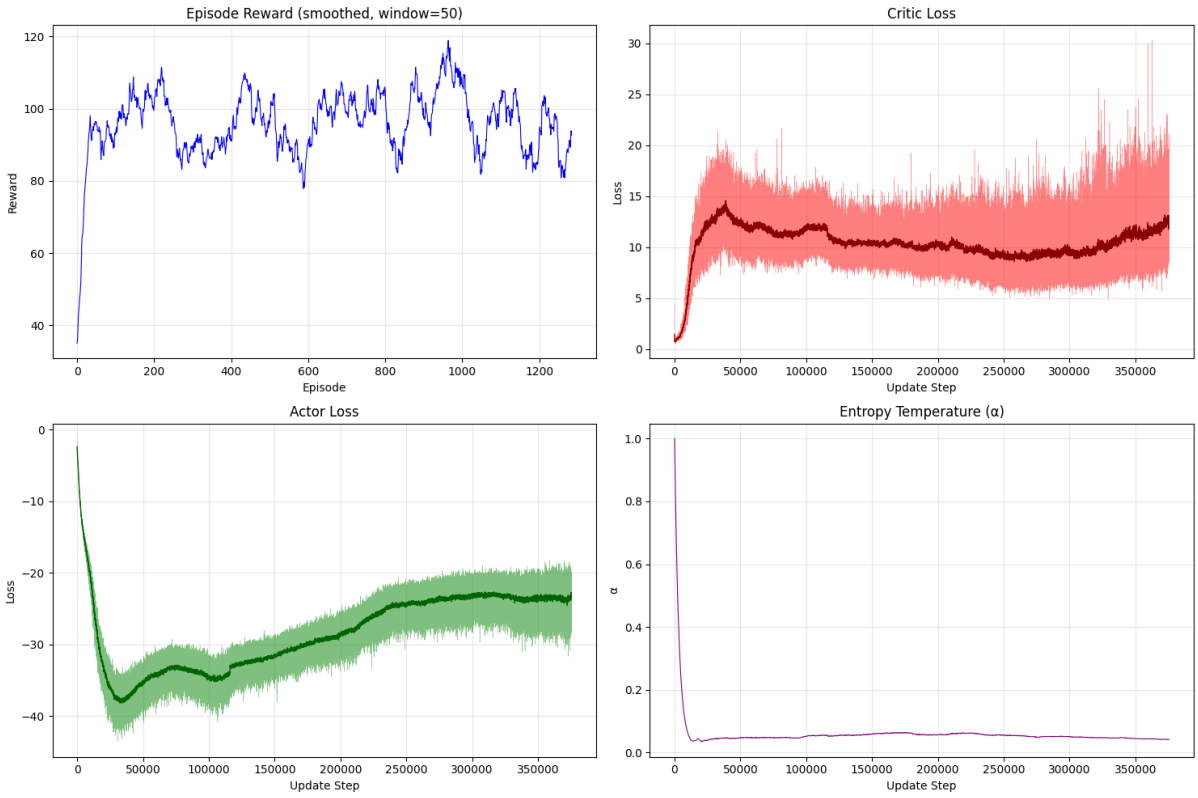

Figure 7: SAC with LSTM training curves

SAC+LSTM (Figure 7) shows a different convergence profile. The episode reward rises quickly to ~90–100 within the first 100 episodes but then oscillates considerably throughout training, ranging between 80 and 120 without a clear plateau. The update steps extend to ~375,000 — longer than the other two models — yet the reward curve does not stabilise, suggesting the recurrent architecture requires more training to converge. The critic loss remains stable around 10–13, and the actor loss gradually recovers from −40 toward −20 over the full training run. The entropy temperature α decays to near zero and remains there, indicating the policy becomes near-deterministic early despite the reward instability.

*The critic and actor loss curves for SAC+LSTM and SAC+LSTM+Attention display different update step ranges (~375,000 and ~200,000 respectively), despite both models training for an equivalent number of episodes. This is attributed to a difference in gradient update frequency between the two architectures, where the additional computational overhead of the attention mechanism results in fewer gradient updates being performed within the same episode budget. The episode reward curve remains the primary convergence indicator and is consistent across both models.

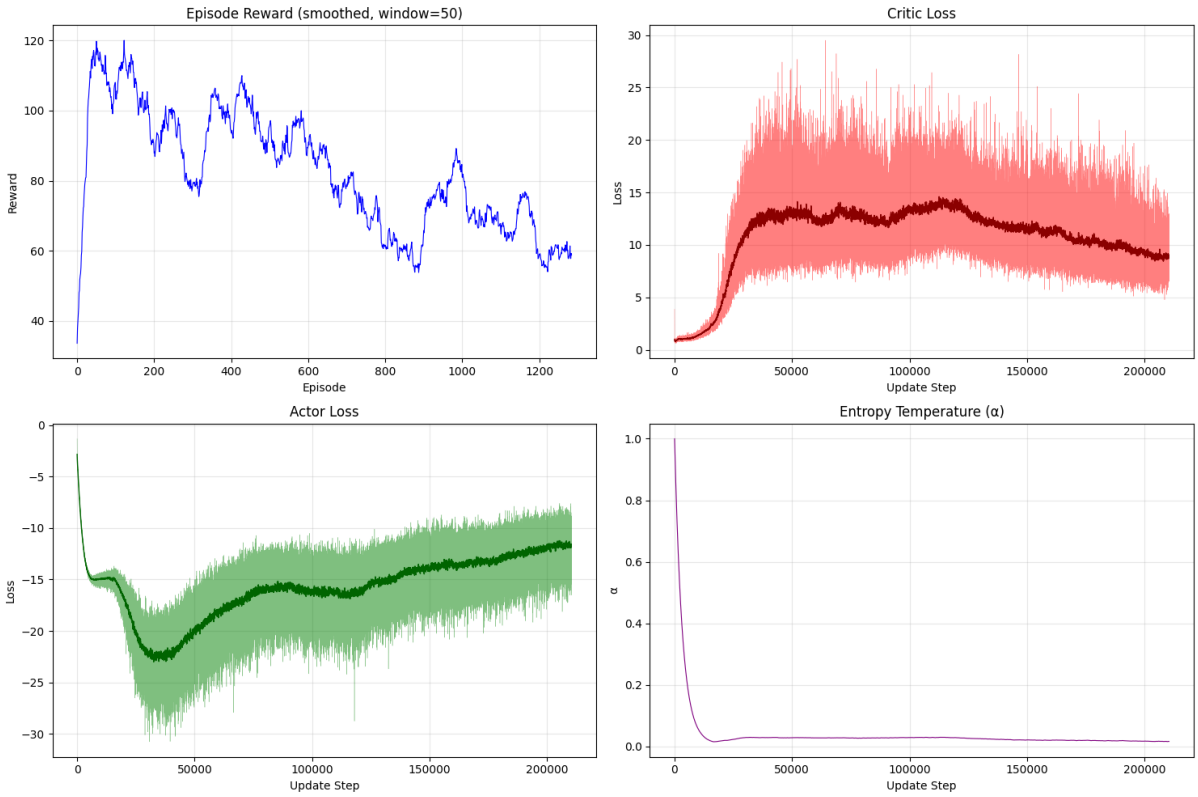

Figure 8: SAC with LSTM (Attention) training curves

SAC+LSTM+Attention (Figure 8) exhibits the most concerning convergence behaviour. The episode reward peaks sharply at ~115 around episode 100 but then undergoes a sustained decline throughout training, settling around 60–80 by the end — significantly below its early peak. This degradation after an initial peak suggests the agent overfits or destabilises as training progresses, a known challenge when training recurrent policies with off-policy methods. The critic loss stabilises around 10–15, and the actor loss recovers from −20 toward −10, but the rising reward trend expected from these loss improvements does not materialise in the episode reward curve. The entropy temperature α decays to near zero rapidly, suggesting the agent commits to an exploitative policy too early before discovering a robust quoting strategy.

Taken together, these training curves reveal a clear pattern: SAC converges most reliably, achieving the highest and most stable episode rewards, while the LSTM-augmented variants exhibit greater training instability and lower peak performance. This provides early evidence that the added recurrent complexity does not translate into improved policy learning within the current observation space, a finding that is further examined in the results section.

4.4 Results

4.4.1 Baseline vs SAC

The fixed baseline agent quotes at action [0, 0] throughout every episode, posting orders at the best bid and ask without any adaptive adjustment. While this ensures maximum fill probability, it provides no mechanism for inventory management or response to changing market conditions.

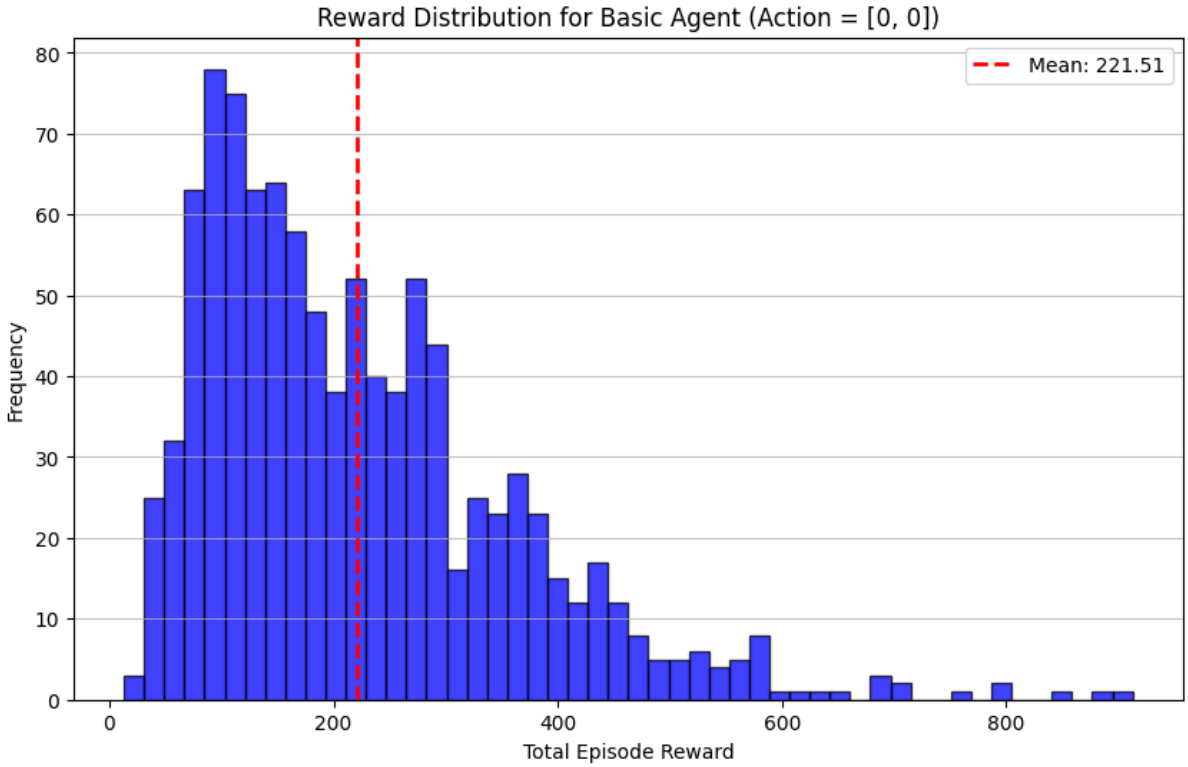

Figure 9: Baseline reward distribution

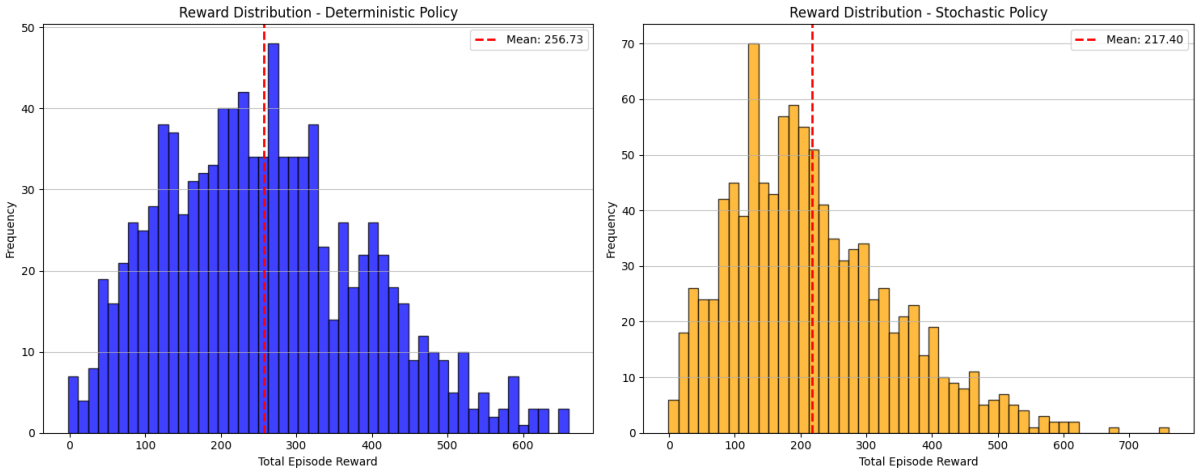

Figure 10: SAC deterministic and stochastic reward distribution

The baseline agent achieves a mean total episode reward of 221.51, with a right-skewed distribution spanning roughly 0 to 900. While the mean appears reasonable, the high variance

and long right tail suggest the baseline performance is largely driven by favourable stochastic market conditions rather than intelligent quoting decisions — the agent has no mechanism to consistently exploit market structure.

The SAC deterministic policy achieves a mean reward of 256.73, representing a 16% improvement over the baseline, with a more concentrated distribution ranging from 0 to ~650. The deterministic policy distribution is notably more bell-shaped and symmetric compared to the baseline, indicating more consistent episode-to-episode performance. The SAC stochastic policy achieves a mean reward of 217.40, slightly below the baseline mean, which is expected — the stochastic policy introduces additional randomness through action sampling that can occasionally lead to suboptimal quotes.

The key distinction between the SAC deterministic policy and the baseline lies not just in the higher mean but in the distributional shape. The baseline exhibits a bimodal-like structure with a large mass of low-reward episodes and a separate cluster of high-reward episodes, suggesting its performance is highly environment-dependent. The SAC deterministic policy, by contrast, shows a smoother unimodal distribution centred around 250–300, indicating the learned policy consistently extracts value across a wider range of market conditions. This validates that reinforcement learning can discover non-trivial quoting strategies that outperform passive top-level quoting in a realistic Hawkes process environment.

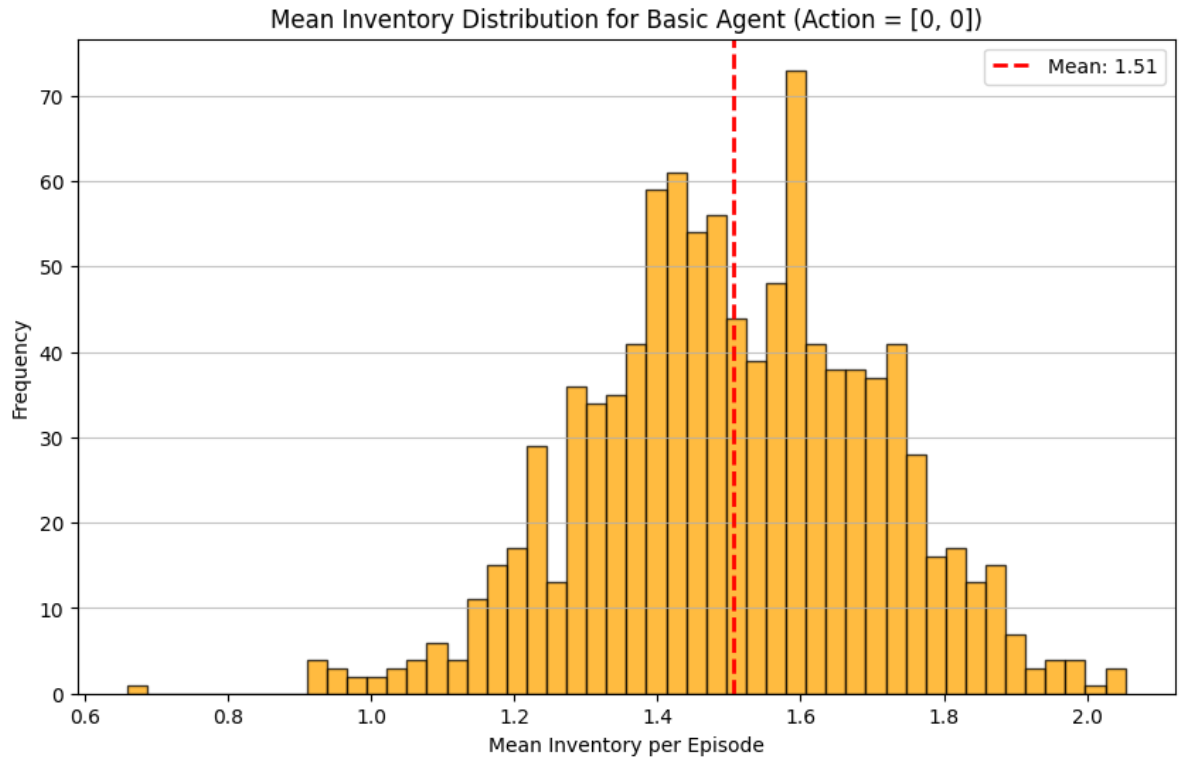

Figure 11: Baseline inventory distribution

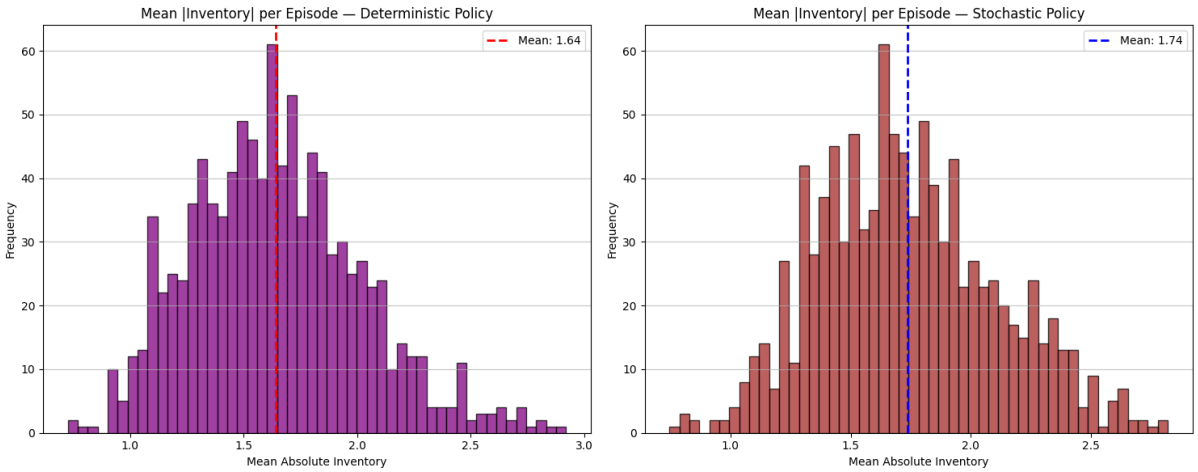

Figure 12: SAC deterministic and stochastic inventory distribution

The mean absolute inventory distributions provide a measure of intra-episode inventory risk carried by each agent throughout trading. The baseline agent carries a mean absolute inventory of 1.51 per episode, with a broad distribution spanning 0.6 to 2.1, reflecting the absence of any inventory management mechanism — the agent quotes symmetrically at [0, 0] regardless of its current position, resulting in largely uncontrolled inventory accumulation throughout each episode.

The SAC deterministic policy carries a mean absolute inventory of 1.64, and the stochastic policy 1.74 — both slightly higher than the baseline. This suggests that the SAC agent does not prioritise inventory minimisation, instead favouring a more aggressive quoting strategy that captures greater spread at the cost of higher positional exposure. Despite carrying higher average inventory, the SAC agent achieves a meaningfully higher mean reward of 256.73 compared to the baseline mean of 221.51, indicating that the additional inventory risk taken is compensated by greater profitability through more active quoting.

4.4.2 SAC vs SAC+LSTM vs SAC+LSTM+Attention

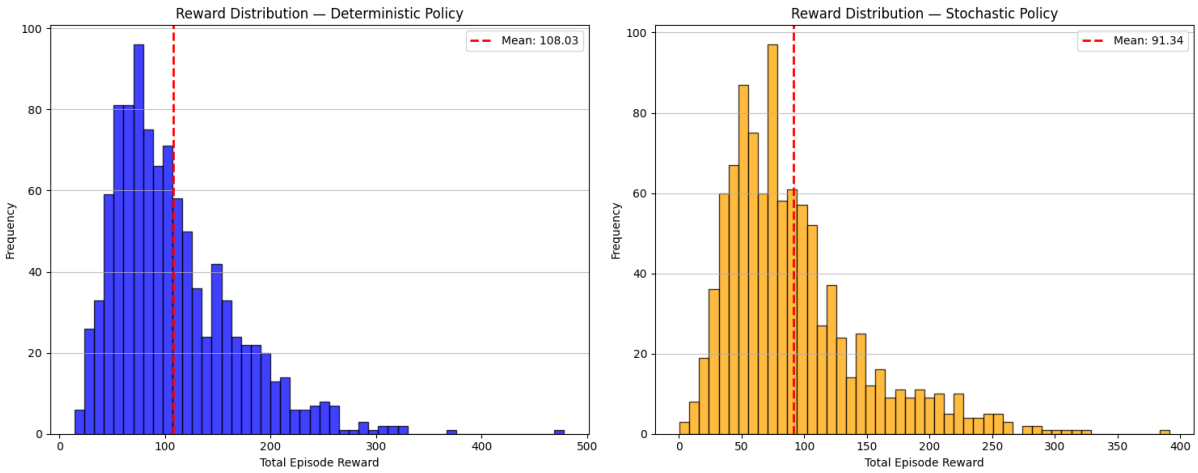

Figure 13: SAC with LSTM reward distribution

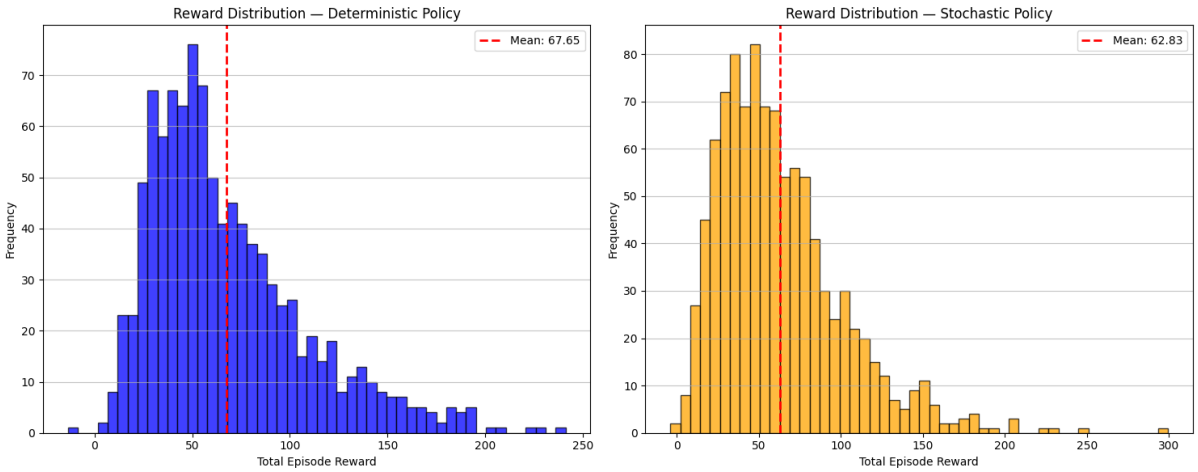

Figure 14: SAC with LSTM (Attention) reward distribution

The results across the three learned agents are summarised below:

Model	Deterministic Mean Reward	Stochastic Mean Reward
SAC	256.73	217.40
SAC + LSTM	108.03	91.34
SAC + LSTM + Attention	67.65	62.83

Contrary to the intuitive expectation that architectural complexity should yield monotonically improving performance, the results reveal a clear **inverse relationship** between model complexity and performance. The standalone SAC agent achieves the highest deterministic mean reward of 256.73, while SAC+LSTM drops significantly to 108.03, a reduction of approximately 58% and SAC+LSTM+Attention further declines to 67.65, representing a 74% reduction relative to SAC. This pattern is consistent across both deterministic and stochastic evaluation modes, confirming it is not an artefact of policy sampling noise.

Examining the distributional shapes, the SAC reward distribution spans a wide range from 0 to ~650, reflecting a learned policy capable of achieving high rewards in favourable market conditions. The SAC+LSTM distribution is considerably more compressed, concentrated between 20 and 300 with a sharp right-skewed peak around 60–100. The SAC+LSTM+Attention distribution is similarly compressed and shifted further left, with the majority of episodes falling between 20 and 150, suggesting convergence to a more conservative but less profitable quoting strategy.

This outcome is primarily attributed to the **limited dimensionality of the observation space**. The agent observes only three features — normalised inventory, spread, and order flow imbalance. While these features are informative for single-step decision making, they may not contain sufficient sequential richness for the LSTM to exploit meaningfully across a window of past observations. The historical sequence of these three features may carry limited additional predictive signal beyond what is already captured in the current state. Under these conditions, the additional parameters and training complexity introduced by the LSTM and attention layers constitute a form of over-parameterisation relative to the available state

information — making optimisation considerably harder without a corresponding performance benefit. This interpretation is further supported by the training curves presented in Section 4.3, where the SAC+LSTM+Attention model exhibited a declining reward trajectory after its early peak, consistent with a model that struggles to maintain a stable policy under increased architectural complexity.

This finding represents an important empirical result, the benefits of recurrent architectures in market-making DRL are **conditional on a sufficiently rich state representation**, a constraint that has not always been made explicit in prior work. Enriching the observation space with additional LOB features such as order book depth, recent trade volume, and a longer history of Hawkes intensities would be a natural and necessary step toward realising the potential of LSTM-based agents in this setting.

The final wealth distributions, defined as terminal cash plus inventory valued at the closing mid-price, are provided in the Appendix (Figures 17–20) as a complementary metric. The wealth results are broadly consistent with the reward findings: the baseline achieves a mean final wealth of 237.14, the SAC deterministic policy achieves 256.15, while SAC+LSTM and SAC+LSTM+Attention achieve 107.57 and 71.22 respectively. The alignment between wealth and reward rankings confirms that the total episode reward is a reliable proxy for overall profitability and that the performance ordering across models is robust across both metrics.

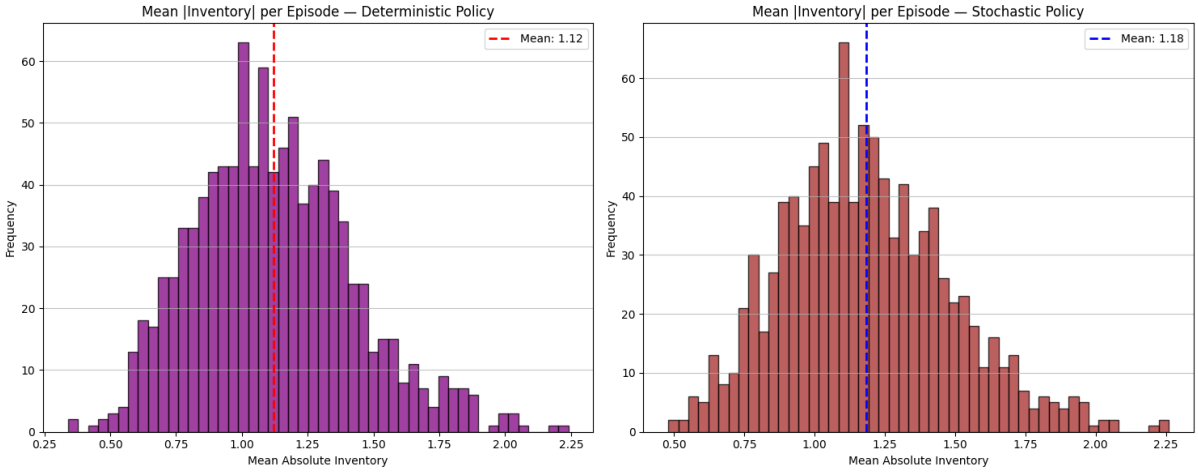

Figure 15: SAC with LSTM inventory distribution

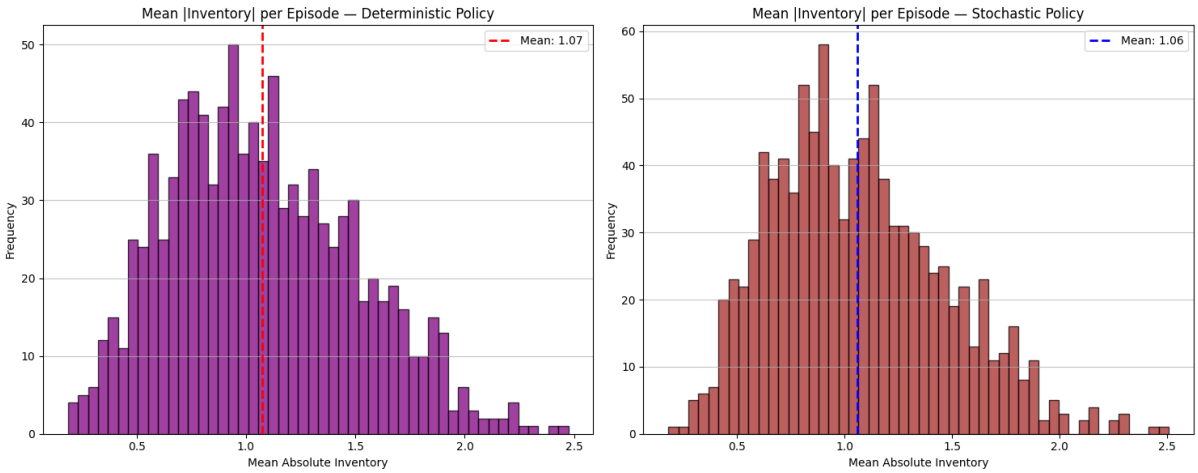

Figure 16: SAC with LSTM (Attention) inventory distribution

The mean absolute inventory results reveal a clear and consistent pattern across the three models, summarised as follows:

Model	Deterministic Mean |Inventory|	Stochastic Mean |Inventory|
SAC	1.64	1.74
SAC with LSTM	1.12	1.18
SAC with LSTM (Attention)	1.07	1.06

Table 8: Inventory summary statistics

A consistent **inverse relationship** is observed between reward performance and inventory risk management across all three models. As architectural complexity increases, mean absolute inventory decreases monotonically, the SAC agent carries the highest average inventory of 1.64 (deterministic), SAC with LSTM reduces this to 1.12, and SAC with LSTM (Attention) achieves the lowest at 1.07. This pattern suggests that the recurrent architectures converge to more conservative and inventory-aware quoting strategies, systematically carrying lower positional exposure throughout each episode. However, this improved inventory control comes at the cost of significantly lower rewards, as demonstrated in Section 4.4.2. This trade-off between reward maximisation and inventory risk management represents an important empirical finding — richer state representations and longer training may be necessary to allow recurrent agents to simultaneously achieve both objectives.

4.5 Discussion

The experiments yield four principal findings. First, a longer planning horizon ($\gamma = 0.99$) is necessary for effective inventory risk management, as the shorter horizon of $\gamma = 0.90$ leads to myopic quoting behaviour that underweights future inventory exposure. Second, the SAC agent successfully learns a quoting policy that outperforms the fixed baseline in terms of mean

reward (256.73 vs 221.51), demonstrating the viability of continuous-action DRL in a Hawkes process LOB. Third, the LSTM and attention extensions do not improve on the base SAC agent in terms of reward performance, achieving mean deterministic rewards of 108.03 and 67.65 respectively, with the limited observation space dimensionality identified as the likely limiting factor. Fourth, an inverse relationship is observed between reward performance and inventory risk management: as model complexity increases, mean absolute inventory decreases monotonically from 1.64 (SAC) to 1.12 (SAC+LSTM) to 1.07 (SAC+LSTM+Attention), suggesting that recurrent architecture converges to more conservative but less profitable quoting strategies.

The inventory risk analysis reveals that the SAC agent prioritises spread capture over inventory minimisation, carrying higher average inventory than both the baseline (1.51) and its recurrent variants despite achieving superior rewards. The recurrent models, by contrast, appear to internalise the inventory risk penalty more strongly, gravitating toward lower positional exposure throughout each episode at the expense of profitability. This suggests a fundamental tension between reward maximisation and inventory control that the current three-dimensional observation space, comprising only inventory ratio, spread, and order flow imbalance, is insufficient to resolve simultaneously. A richer state representation providing more market context may allow future agents to better balance these competing objectives.

These findings collectively point to three clear directions for future work: enriching the observation space with additional LOB features such as order book depth, recent trade volume, and a longer history of Hawkes intensities; revisiting the LSTM and attention architectures under a richer state representation to properly evaluate their potential contribution to both reward performance and inventory management; and investigating whether explicit inventory targeting mechanisms in the reward function could help agents simultaneously achieve profitability and risk control.

5 Conclusion

The objective of this research is to integrate the fields of market making and deep reinforcement learning, with the goal of facilitating intelligent market-making through machine learning. This study challenges traditional rule-based methods of market making by employing deep reinforcement learning techniques within a realistic, mathematically consistent limit order book simulation environment.

Key achievements include the mathematical modelling of limit order book dynamics, specifically relating to price movements and the arrival of orders, utilising jump processes and Hawkes processes respectively. The validity of the mathematical environment has been assessed and the development of the environment using the OpenAI Gymnasium library has been successfully completed. Building upon this foundation, three learning-based agents were developed and evaluated: a Soft Actor-Critic (SAC) agent, an SAC agent augmented with a

Long Short-Term Memory (LSTM) network, and an SAC agent further enhanced with an Attention Mechanism.

The SAC agent demonstrates clear improvement over the fixed baseline quoting strategy, achieving a mean deterministic reward of 256.73 compared to the baseline mean of 221.51, validating that a continuous-action deep reinforcement learning agent can discover non-trivial quoting policies within a realistic Hawkes process environment. A hyperparameter study further confirmed that a longer planning horizon ($\gamma = 0.99$) is essential for effective inventory risk management over the course of an episode. The environment accommodates various real-world conditions, including the arrival of up to eight mutually dependent order types, price jumps, inventory management, and market fees, ensuring the simulation closely reflects real trading dynamics.

However, the LSTM and attention-augmented models did not yield consistent improvement over the standalone SAC agent in terms of reward performance, achieving mean deterministic rewards of 108.03 and 67.65 respectively. This is attributed to the limited dimensionality of the observation space, comprising only inventory ratio, spread, and order flow imbalance, which may not contain sufficient sequential richness for recurrent architectures to exploit meaningfully. An important secondary finding is the inverse relationship observed between reward performance and inventory risk management: as model complexity increases, mean absolute inventory decreases monotonically from 1.64 (SAC) to 1.12 (SAC+LSTM) to 1.07 (SAC+LSTM+Attention), suggesting that recurrent architectures converge to more conservative but less profitable quoting strategies. This highlights a fundamental tension between profitability and inventory control that the current observation space is insufficient to resolve simultaneously.

Although the model proposed by Law and Viens [19] demonstrates the ability to yield reliable results, it does not account for order book depth beyond the best bid and ask, which could hinder a more comprehensive study of LOB dynamics. Furthermore, the agent operates within a single-agent framework, whereas real markets involve multiple competing market makers, a zero-sum dynamic that this research does not fully capture.

Acknowledging these limitations opens clear avenues for future research. Enriching the observation space with additional features such as order book depth, recent trade volume, and a longer history of Hawkes intensities would provide the sequential signal necessary to fully realise the potential of LSTM-based agents and potentially resolve the observed tension between profitability and inventory control. Developing more advanced LOB models that account for full order book depth would further enhance the realism of the training environment. Finally, adversarial multi-agent reinforcement learning could be investigated to model the competitive dynamics between market makers, offering a path toward greater generalisation and robustness under real-world market conditions.

References

[1] L. R. Glosten and P. R. Milgrom, “Bid, ask and transaction prices in a specialist market with heterogeneously informed traders,” *Journal of Financial Economics*, vol. 14, no. 1, pp. 71–100, Mar. 1985. [Online]. Available: https://milgrom.people.stanford.edu/wp-content/uploads/1984/09/Bid-Ask-and-Transaction-Prices.pdf

[2] M. Avellaneda and S. Stoikov, “High-frequency trading in a limit order book,” *Quantitative Finance*, vol. 8, no. 3, pp. 217–224, Apr. 2008. [Online]. Available: https://people.orie.cornell.edu/sfs33/LimitOrderBook.pdf

[3] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6795963

[4] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, May. 1992. [Online]. Available: https://link.springer.com/article/10.1007/BF00992698

[5] D. Silver et al., “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016. [Online]. Available: https://www.nature.com/articles/nature16961

[6] L. Fridman et al., “DeepTraffic: Crowdsourced hyperparameter tuning of deep reinforcement learning systems for multi-agent dense traffic navigation,” *arXiv preprint*, arXiv:1801.02805, Jan. 2018. [Online]. Available: https://arxiv.org/abs/1801.02805

[7] S. Sang and L. Li, “A novel variant of LSTM stock prediction method incorporating attention mechanism,” *Mathematics*, vol. 12, no. 7, p. 945, Mar. 2024. [Online]. Available: https://doi.org/10.3390/math12070945

[8] M. Hausknecht and P. Stone, “Deep recurrent Q-learning for partially observable MDPs,” *arXiv preprint*, arXiv:1507.06527, Jul. 2015. [Online]. Available: https://arxiv.org/abs/1507.06527

[9] R. Cont, S. Stoikov, and R. Talreja, “A Stochastic Model for Order Book Dynamics” *SSRN Electronic Journal*, Sep. 2008. [Online]. Available:https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1273160

[10] J. Hasbrouck, “Limit Order Market,” in *Empirical Market Microstructure: The Institutions, Economics, and Econometrics of Securities Trading*, Oxford University Press, Jan. 2007. [Online]. Available: https://books.google.com.hk/books?id=aaReNv846eMC

[11] F. Xie, Y. Liu, C. Hu, and S. Liang, “Dynamic modeling of limit order book and market maker strategy optimization,” *Mathematics*, vol. 13, no. 5, Feb. 2025. [Online]. Available: https://www.mdpi.com/2227-7390/13/5/778

[12] A. Oyewola, O. Akinwunmi, and O. Omotehinwa, “Deep LSTM Q-learning for price movement prediction in the oil and gas sector,” *Knowledge-Based Systems*, vol. 281, Jan. 2024. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0950705123010389

[13] C. Maglaras, C. C. Moallemi, and M. Wang, "A deep learning approach to estimating fill

probabilities in a limit order book," Quantitative Finance, vol. 22, no. 11, pp. 1989-2003, Nov. 2022. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/14697688.2022.2124189

[14] Y.-S. Lim and D. Gorse, "Reinforcement learning for market making in a limit order book," presented at the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2018), Apr. 2018. [Online]. Available: https://www.esann.org/sites/default/files/proceedings/legacy/es2018-50.pdf

[15] M. Gasperov and S. Kostanjčar, "Market making with Hawkes process limit orderbook," *arXiv preprint*, arXiv:2207.09951, Jul. 2022. [Online]. Available: https://arxiv.org/abs/2207.09951

[16] P. Kumar, "Deep Hawkes Process for High-Frequency Market Making," *arXiv preprint*, arXiv:2109.15110, Sep. 2021. [Online]. Available: https://arxiv.org/abs/2109.15110

[17] J. Spooner, D. Savani, and J. Zohren, "Deep reinforcement learning for market making in high-frequency trading," *HAL open archive*, Jan. 2018. [Online]. Available: https://hal.science/hal-01686122v1/document

[18] X. Lu and F. Abergel, "High dimensional Hawkes processes for limit order books: Modelling, empirical analysis and numerical calibration," *Quantitative Finance*, vol. 18, no. 8, pp. 1315–1330, Jan. 2018. [Online]. Available: https://hal.science/hal-01686122v1/document

[19] Law, B and Viens, F. "Market Making under a Weakly Consistent Limit Order Book," *High Frequency,* vol. 2, no.4, pp. 215-238, Aug. 2019 [Online]. Available: https://arxiv.org/pdf/1903.07222

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, Jul. 2018, pp. 1861–1870. [Online]. Available: https://arxiv.org/abs/1801.01290

[21] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: https://jmlr.org/papers/v22/20-1364.html

Appendix:

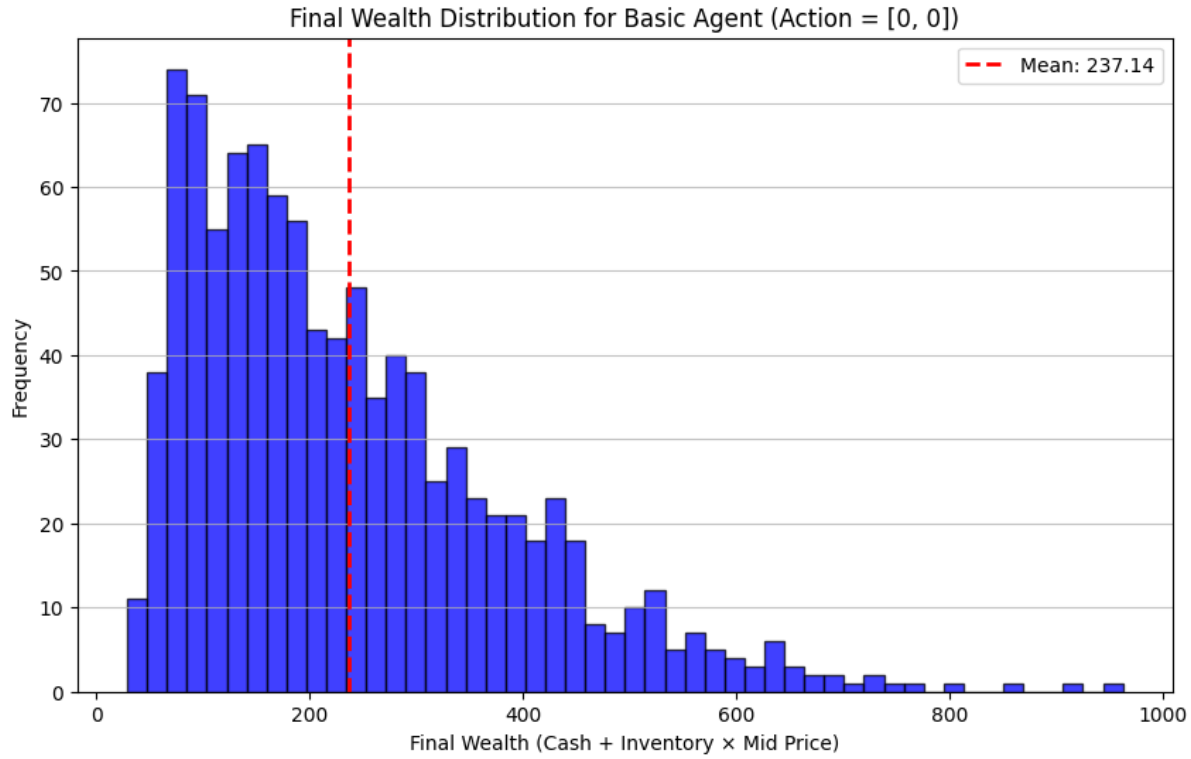

Figure 17: Baseline Model Wealth Distribution

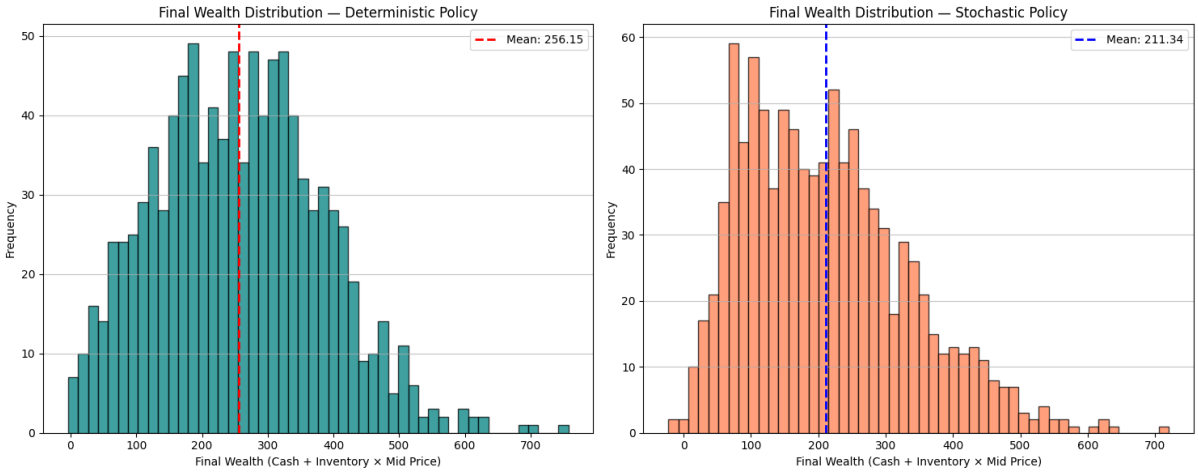

Figure 18: SAC Model Wealth Distribution

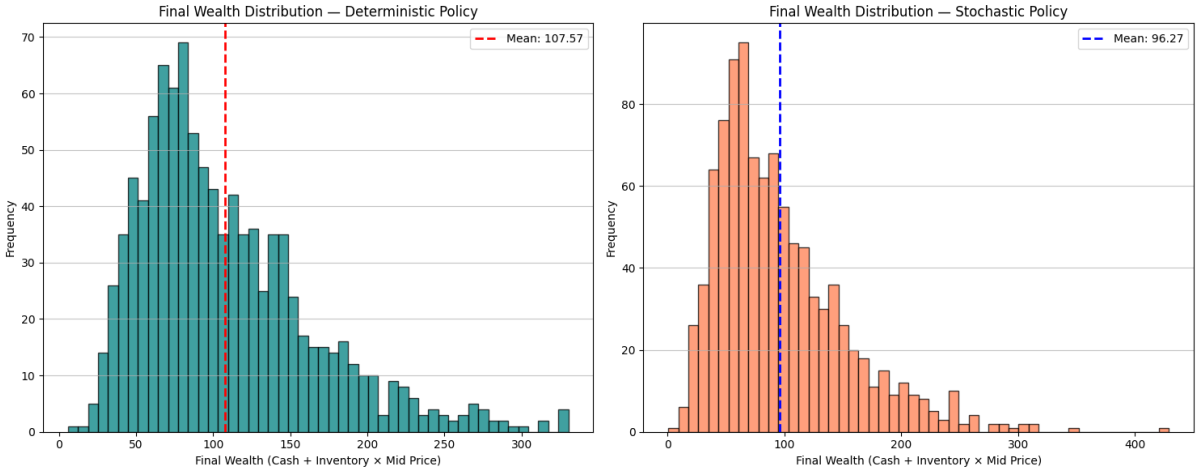

Figure 19: SAC with LSTM Model Wealth Distribution

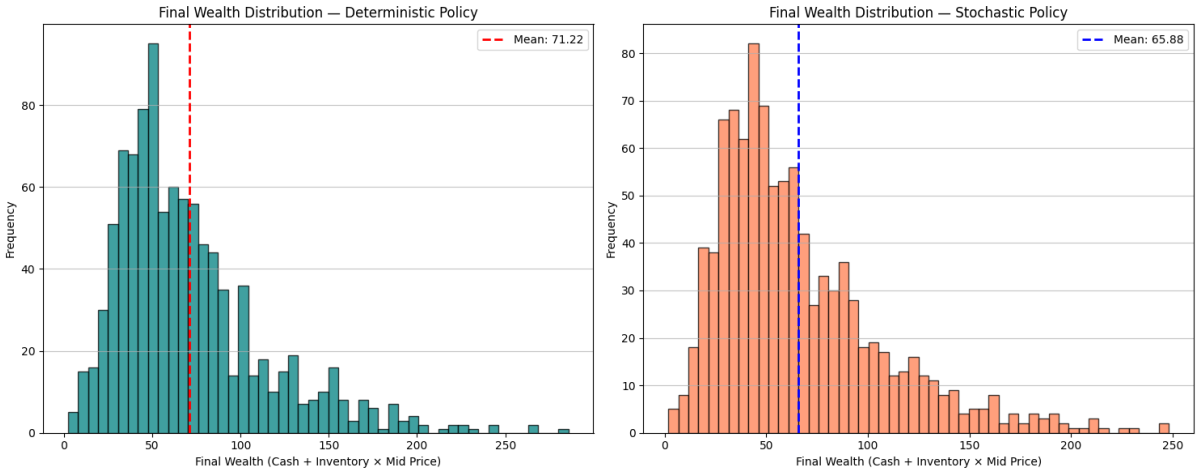

Figure 20: SAC with LSTM (Attention) Model Wealth Distribution

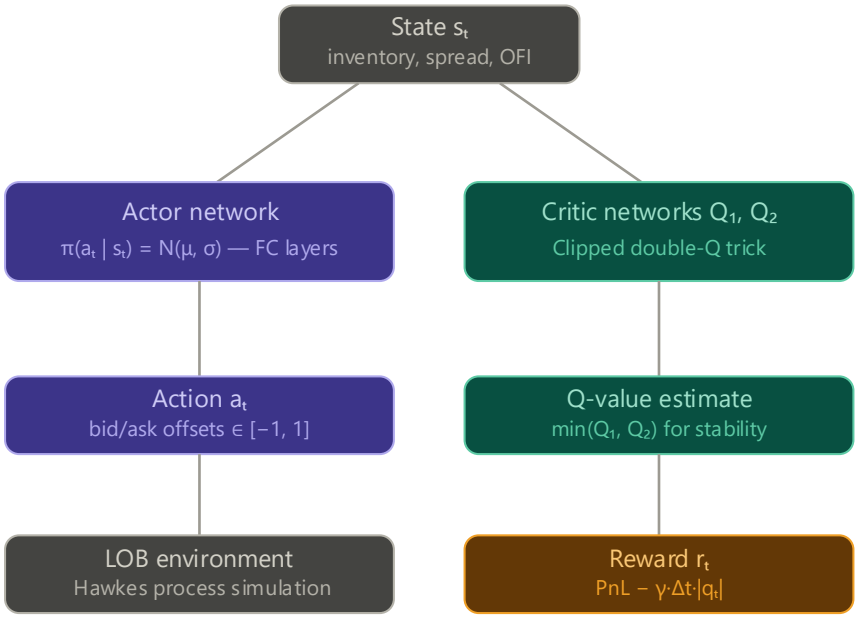

Figure 21: SAC Architecture

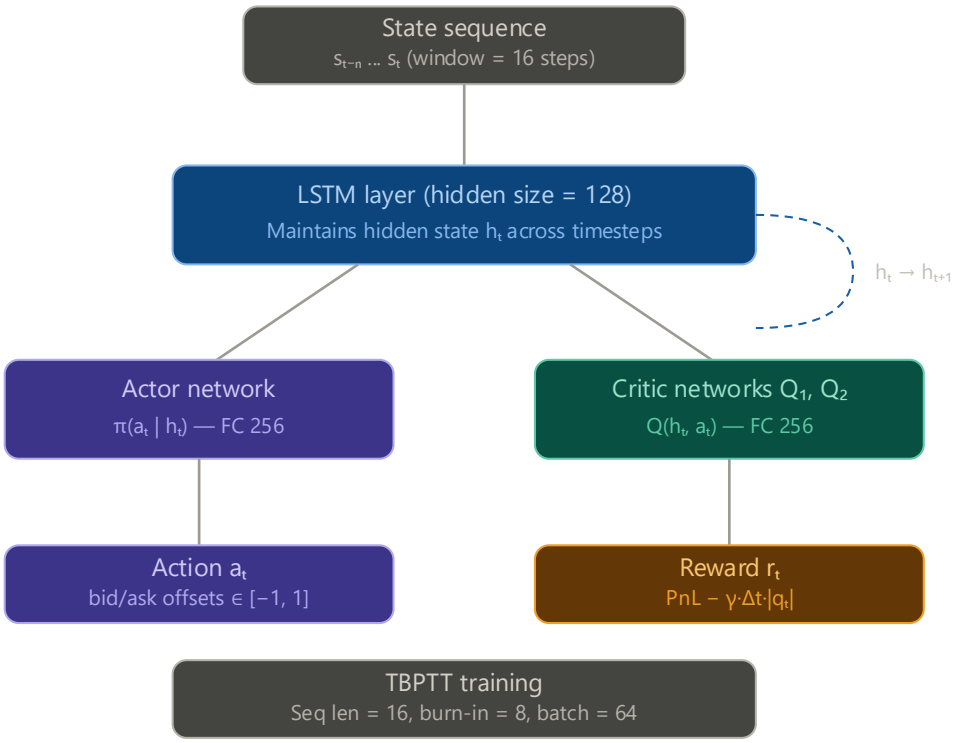

Figure 22: SAC with LSTM Architecture

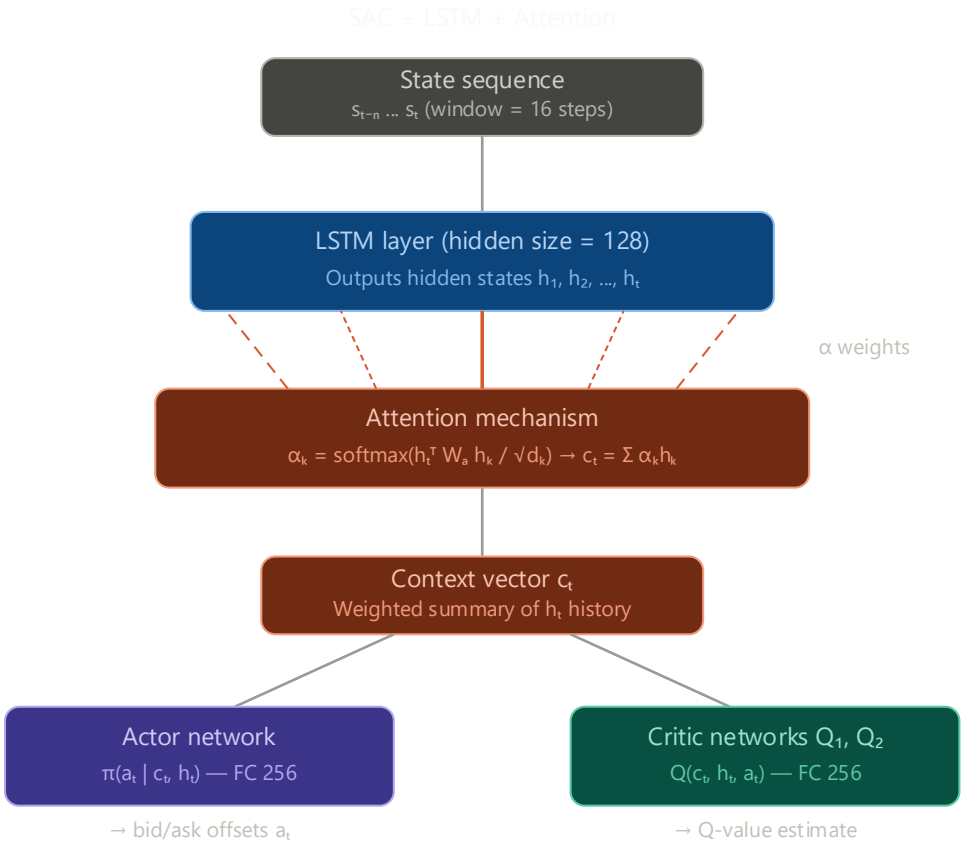

Figure 23: SAC with LSTM (Attention) Architecture