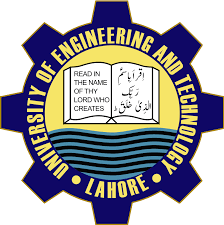
AN MULTI-AGENT SYSTEM FOR HIGH FREQUENCY TRADING USING DEEP REINFORCEMENT LEARNING



by

Muhammad Mazhar

2021-MS-CS-548

Research Supervisor:

Dr. Muhammad Aslam

2025

Department of Computer Science

University of Engineering and Technology, Lahore

AN MULTI-AGENT SYSTEM FOR HIGH FREQUENCY TRADING USING DEEP REINFORCEMENT LEARNING

by

Muhammad Mazhar

A THESIS

Presented to the university of engineering and technology, Lahore

in partial fulfillment of the requirements for the degree of

Master of Science

in

[COMPUTER SCIENCE]

APPROVED BY:



[Primary Advisor/Internal Examiner] [External Examiner]

[Official Title & Department] [Official Title & Department]



[Chairman of the Department] [Dean of Faculty]

Approval Date

COMPUTER SCIENCE

UNIVERSITY OF ENGINEERING & TECHNOLOGY, LAHORE

2025

Muhammad Mazhar

All Rights Reserved

Any part of this thesis cannot be copied, reproduced or published without the written approval of the Scholar.

# ABSTRACT

High frequency trading (HFT) is a domain in the financial market where decisions are made in fractions of a second, leveraging advanced algorithms to execute a high volume of trades. In recent years, the integration of artificial intelligence, particularly deep reinforcement learning, has grown tremendous promise in optimizing decision making, in a time sensitive and dynamic environment. This presents a multi-agent deep reinforcement learning framework for high frequency trading where intelligent agents collaborate and compete to maximize profitability while maintaining market stability. The framework models the financial market as a partially observed markov decision process and multi-agent Deep deterministic policy gradient algorithm to train agent capable of sophisticated trading strategies. The research leverages state of the art deep reinforcement learning policy gradient methods, deep Q-learning, and graph neural network to enable agents to learn. key contributions of this work include 1) design and implementation of a decentralized multi-agent system, 2) the development of a environment tailored to HFT focusing objectives such as risk adjusted return and execution cost minimization, and 3) the use of market simulation environment with realistic order book dynamics to evaluate agent performance. Our results showed that the proposed MADRL system outperforms traditional algorithmic trading methods, achieving superior sharpe ratio, and lower drawdown under varying market conditions. Furthermore, the paper discusses the interplay between agents shedding light on emergent behaviors such as market making, arbitrage, and predatory trading. The findings highlight the potential of MADRL in revolutionizing HFT.

# ACKNOWLEDGMENTS

I would like to acknowledge and deeply grateful to my supervisor Dr. Muhammad Aslam who made this work possible. Her guidance and advice carried me through all the stages of writing my project. Additionally, I would also like to thank my committee members and external evaluator Dr. Ayesha Afzaal for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions, thanks all of you.

I would also like to give special thanks to my wife Aminah Muqaddas and my family as a whole for their continuous support and understanding of my research and writing of my work.

Finally, I would like to thank ALLAH for letting me through all the difficulties. I have experienced your guidance day by day. You are the one who let me finish my degree. I will keep on trusting you for my future.

# STATEMENT OF ORIGINALITY

It is stated that the research work presented in this dissertation consists of my own ideas and research work. The contributions and ideas from others have been duly acknowledged and cited in the dissertation. This complete dissertation is written by me. If at any time in the future, it is found that the thesis work is not my original work, the University has the right to cancel my degree.

.

Muhammad Mazhar

# TABLE OF CONTENTS

Page

[1. ABSTRACT 1](#_Toc191507956)

[2. ACKNOWLEDGMENTS 2](#_Toc191507957)

[3. STATEMENT OF ORIGINALITY 3](#_Toc191507958)

[4. TABLE OF CONTENTS 4](#_Toc191507959)

[5. LIST OF FIGURES 5](#_Toc191507960)

[6. LIST OF TABLES 7](#_Toc191507961)

[7. NOMENCLATURE 8](#_Toc191507962)

[1. INTRODUCTION 10](#_Toc191507963)

[1.1.2 Adaptive Exploration and Non-Stationarity Mitigation 14](#_Toc191507964)

[1.1.3 Risk-Aware Optimization and Regulatory Compliance 14](#_Toc191507965)

[1.3 Problem Statement 16](#_Toc191507966)

[1.4 Aim and Objectives 16](#_Toc191507967)

[1.5 Contributions 17](#_Toc191507968)

[2. LITERATURE SURVEY 18](#_Toc191507969)

[3. PROPOSED Methodology 48](#_Toc191507970)

[8. REFERENCES 57](#_Toc191507971)

[9. VITA 62](#_Toc191507972)

# LIST OF FIGURES

Figure Page

[Figure ‎4.1 LAN and WAN Network Design 13](#_heading=h.3j2qqm3)

[Figure ‎4.2 Process Flow Chart 16](#_heading=h.3whwml4)

[Figure ‎4.3 OSPF Implementation Design 20](#_heading=h.3as4poj)

[Figure ‎4.4 EIGRP Implementation Design 23](#_heading=h.2p2csry)

[Figure ‎4.5 BGP Implementation Design 26](#_heading=h.ihv636)

[Figure ‎4.6 HUB-DMVPN Testing 33](#_heading=h.1v1yuxt)

[Figure ‎4.7 HUB-NHRP Testing 33](#_heading=h.4f1mdlm)

[Figure ‎4.8 HUB-IPsec Testing 33](#_heading=h.2u6wntf)

[Figure ‎5.1 ICMP Minimum Values Test 35](#_heading=h.28h4qwu)

[Figure ‎5.2 ICMP Average Values Test 35](#_heading=h.nmf14n)

[Figure ‎5.3 ICMP Maximum Values Test 35](#_heading=h.37m2jsg)

[Figure ‎5.4 UDP Average Values Test 36](#_heading=h.46r0co2)

[Figure ‎5.5 Throughput Test 38](#_heading=h.111kx3o)

[Figure ‎5.6 Jitter Test 39](#_heading=h.206ipza)

[Figure ‎5.7 Packet-Loss Test 40](#_heading=h.2zbgiuw)

[Figure ‎5.8 Packet-Loss View Test-01 41](#_heading=h.1egqt2p)

[Figure ‎5.9 Packet-Loss View Test-02 41](#_heading=h.3ygebqi)

[Figure ‎5.10 Delay and Latency Test 43](#_heading=h.sqyw64)

[Figure ‎5.11 Representation of Latency, Delay and Response Time 44](#_heading=h.1rvwp1q)

[Figure ‎5.12 Wireshark Evaluation of Latency, Delay, and Response time 44](#_heading=h.4bvk7pj)

[Figure ‎5.13 Response time Test 45](#_heading=h.2r0uhxc)

[Figure ‎5.14 FileZilla Server side connectivity 46](#_heading=h.3q5sasy)

[Figure ‎5.15 FileZilla Client Side Connectivity and Data Transfer 46](#_heading=h.25b2l0r)

[Figure ‎5.16 FileZilla Server to client data security view and Wireshark output 47](#_heading=h.kgcv8k)

[Figure ‎5.17 Speed Test on all sites Table 48](#_heading=h.34g0dwd)

# LIST OF TABLES

Table Page

[Table 2.1 Summary of the Literature Survey 8](#_heading=h.lnxbz9)

[Table 4.1 Devices Specifications 12](#_heading=h.z337ya)

[Table 4.2 LAN Sites IP Schema 14](#_heading=h.4i7ojhp)

[Table 4.3 WAN Sites IP Schema 14](#_heading=h.2xcytpi)

[Table 4.4 OSPF Authentication Configurations 19](#_heading=h.qsh70q)

[Table 4.5 OSPF Implementation and Testing Configurations 21](#_heading=h.1pxezwc)

[Table 4.6 EIGRP Authentication Configurations 23](#_heading=h.147n2zr)

[Table 4.7 EIGRP Implementation and Testing Configurations 24](#_heading=h.3o7alnk)

[Table 4.8 BGP Authentication Configurations 27](#_heading=h.32hioqz)

[Table 4.9 BGP Implementation and Testing Configurations 28](#_heading=h.41mghml)

[Table 4.10 DMVPN Tunnel Configurations on Hub and Spokes Sites 32](#_heading=h.3fwokq0)

# NOMENCLATURE

| **Symbol** | **Description** |
| --- | --- |
| **AI** | Artificial Intelligence |
| **HFT** | High-Frequency Trading |
| **DRL** | Deep Reinforcement Learning |
| **MADRL** | Multi-Agent Deep Reinforcement Learning |
| **FMH** | Fractal Market Hypothesis |
| **MDP** | Markov Decision Process |
| **POMDP** | Partially Observable Markov Decision Process |
| **DQN** | Deep Q-Network |
| **PPO** | Proximal Policy Optimization |
| **SAC** | Soft Actor-Critic |
| **TD3** | Twin Delayed DDPG |
| **DDPG** | Deep Deterministic Policy Gradient |
| **CVaR** | Conditional Value-at-Risk |
| **VaR** | Value-at-Risk |
| **LOB** | Limit Order Book |
| **RSI** | Relative Strength Index |
| **GARCH** | Generalized Autoregressive Conditional Heteroskedasticity |
| **ETF** | Exchange-Traded Fund |
| **ADV** | Average Daily Volume |
| **VWAP** | Volume-Weighted Average Price |
| **Kyle’s λ (Lambda)** | Price impact coefficient per share |
| **Σ (Sigma)** | Summation operator in mathematical equations |
| **CTDE** | Centralized Training with Decentralized Execution |
| **GAE** | Generalized Advantage Estimation |
| **TD-error** | Temporal Difference Error |
| **EMA** | Exponential Moving Average |
| **MACD** | Moving Average Convergence Divergence |

**CTDE** Centralized training with decentralized execution

1. **INTRODUCTION**

**1.1 Background & Motivation**

Modern electronic markets are organized around a continuous double‑auction that maintains a Limit Order Book (LOB) of resting bids and offers at discrete price levels. Every event—new limit order, market order, cancel, or modify—updates the book and potentially the traded price. The system is inherently asynchronous and adversarial: my decision affects the book that you will observe next, and vice versa. Consequently, the dynamics an algorithm faces are non‑stationary and partially observable because the intentions and internal states of other participants are not fully known. [4]

From an engineering perspective, this setting resembles a real‑time control problem with strict latency budgets, noisy and delayed observations, and heavy feedback effects. Practical HFT desks implement safeguards along three dimensions: (i) inventory risk; (ii) information/adverse selection risk; and (iii) market‑impact risk. A credible learning‑based system must respect all three while remaining computationally tractable and diagnosable. [29], [30]

In the Pakistani context, the strategic objective is not merely to maximize raw PnL but to institutionalize decision‑making: a system that is explainable to risk committees and regulators, reproducible for audits, and configurable to business constraints (position, participation, and turnover). The research question therefore goes beyond prediction and touches robust control under microstructure realities.

**1.2 High‑Frequency Trading & Market Microstructure**

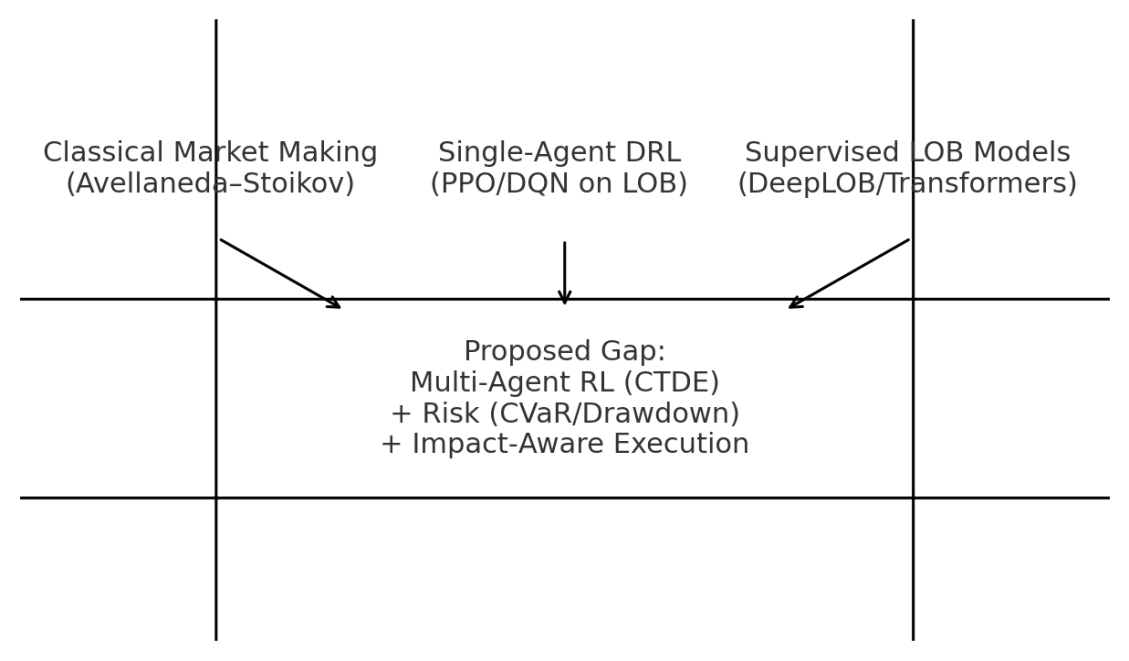
At high frequencies, price changes over short horizons are statistically associated with the order‑flow imbalance (OFI) between marketable buy and sell pressure, the distribution of depth at best quotes, and the dynamics of the queue. LOB states are structured: at each level ℓ relative to the mid, one observes price p\_ℓ and available quantity q\_ℓ. Useful derived features include spread, depth imbalance, short‑horizon realized volatility, trade intensity, cancel intensity, and meta‑order indicators. [36]

Execution is governed by price–time priority: better price executes first; within the same price, earlier orders execute earlier. This creates a premium on maintaining favorable queue position—arriving slightly earlier can dramatically change realized fills. For learning agents, this calls for features that proxy queue priority (e.g., relative time at touch, cancellations behind, and predicted time‑to‑fill). [4]

Finally, market impact—the relationship between trade size, participation and price move—is concave at typical scales. Ignoring impact encourages strategies that look profitable in back tests but would be loss‑making live. Hence, we treat impact explicitly in both reward shaping and the execution layer via throttles. [29]

**1.3 Research Gap**

Classical market‑making frameworks (e.g., inventory‑penalized quoting rules) provide elegant baseline behavior, but they assume a representative market that reacts passively. Supervised LOB models (e.g., CNN/LSTM encoders) predict short‑term moves, yet they do not close the loop with execution. Single‑agent DRL on top of such signals can overfit the non‑stationary environment created by other agents. What remains under‑explored is a complete multi‑agent control system that: (i) trains stably under partial observability (CTDE), (ii) is explicitly risk‑aware (CVaR/drawdown), and (iii) respects impact via an execution layer that throttles toxic flow. [1]



**Figure 1.1 Research gap across HFT paradigms (Author’s illustration).**

**1.4 Problem Statement**

We aim to design and evaluate a risk‑aware multi‑agent deep reinforcement learning framework for HFT on LOB markets that: (a) operates under partial observability using a centralized‑training, decentralized‑execution paradigm; (b) ingests microstructure‑rich state (OFI, depth imbalance, queue position, trade/cancel intensity); (c) embeds risk shaping—CVaR and drawdown penalties—into the learning objective; and (d) internalizes market impact through both reward penalties and execution‑layer throttling. Performance will be assessed against strong baselines on risk‑adjusted metrics. [15], [16]

**1.5 Research Questions (RQs)**

RQ1: How should HFT be posed as a Markov game with realistic states, actions, constraints, and risk?

RQ2: Which MARL backbone (e.g., MAPPO/MADDPG) yields stable learning under LOB dynamics with CTDE?

RQ3: How do CVaR, drawdown, and impact‑aware penalties influence tail‑risk and deployment stability?

|  |  |
| --- | --- |
| **Research Question** | **Motivation** |
| RQ1 | Reduce sim‑to‑real gaps via faithful microstructure modeling and realistic constraints. |
| RQ2 | Achieve stability in a non‑stationary, partially observed, multi‑agent environment. |
| RQ3 | Institutional viability requires explicit tail‑risk control and impact awareness. |

**Table 1.2 Research questions and motivations.**

**1.6 Objectives & Scope**

The research objectives are six‑fold: (i) construct a simulator with matching‑engine fidelity (ABIDES) or a GPU‑parallel LOB generator (JAX‑LOB) to collect large volumes of rollouts; (ii) engineer a LOB encoder that compresses multi‑level depth/imbalance and queue features into compact states; (iii) devise role‑specialized agents (market maker, statistical arbitrageur, trend follower) that reduce policy interference by operating on different parts of the state‑action space; (iv) adopt CTDE so a central critic can condition on joint context during training; (v) shape risk with CVaR/drawdown terms and penalize impact; and (vi) benchmark rigorously against classical and DRL baselines with ablations. [6], [7], [8], [1], [17], [15]

The scope is intentionally single‑asset and simulation‑centric to focus on methodology. Although we discuss live deployment considerations, empirical conclusions are confined to the simulated environment; real‑market trials fall under future work subject to data access and approvals.

**1.7 Theoretical Framing: Markov Game with Risk (remove it later)**

We formalise the problem as a Markov game G=⟨N,S,{A\_i},P,{R\_i},γ⟩. At time t, the environment emits a state s\_t. Agent i observes o\_t^i=f\_i(s\_t), selects action a\_t^i∈A\_i, and receives reward R\_i. The joint policy Π induces the discounted return: [22], [21]

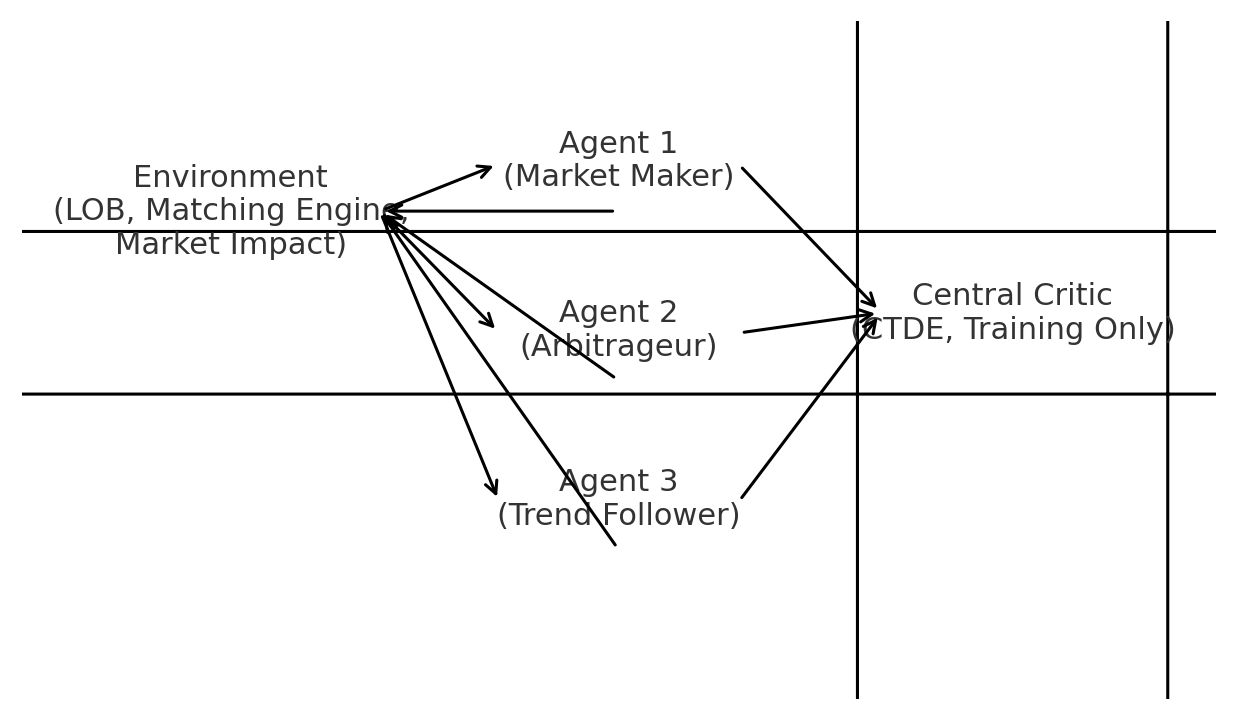
|  |  |
| --- | --- |
| J\_i(Π)=E[∑\_{t=0}^{T} γ^t R\_i(s\_t,a\_t^1,…,a\_t^{|N|})] | (1) |

Risk sensitivity is introduced via CVaR (tail expectation) and drawdown penalties. CVaR at level α is:

|  |  |
| --- | --- |
| CVaR\_α(Z)=E[ Z | Z ≤ F\_Z^{-1}(α) ] | (2) |

Our step reward includes PnL, an inventory term, impact cost, and a tail‑risk penalty computed on a rolling window:

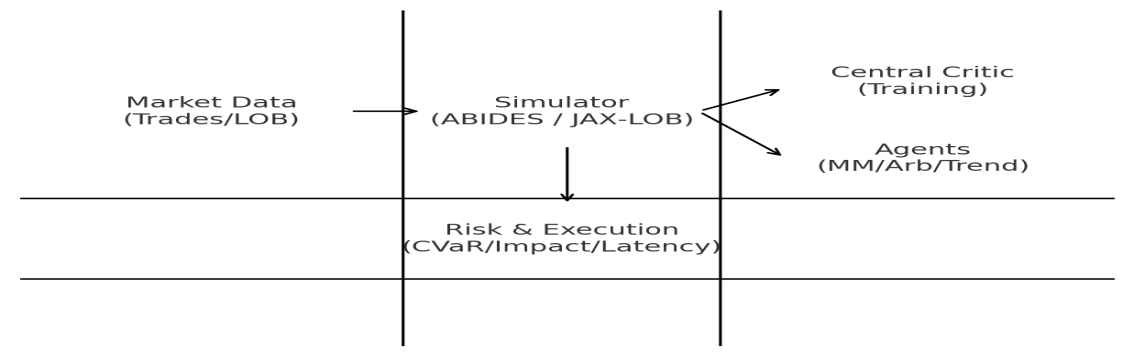
|  |  |
| --- | --- |
| r\_t = ΔPnL\_t − λ|q\_t| − κ·Impact\_t − η·TailRisk\_t. | (3) |



**Figure 1.3 Markov‑game with CTDE: central critic for training; decentralised actors at runtime.**

**1.8 Proposed Architecture (Overview)**

Our architecture adheres to CTDE. During training, the central critic consumes joint signals (e.g., global state, other agents’ actions) to stabilize learning and mitigate non‑stationarity. At inference time, each agent retains only its decentralized actor and local observations. A risk module computes CVaR/drawdown statistics on rolling PnL and shapes the reward accordingly. An execution layer implements hard limits (position, participation, throttle) and soft costs (impact proxy) to keep behaviour safe. [15], [16]



**Figure 1.4 Proposed MARL‑HFT system architecture (Author’s illustration).**

**1.9 Contribution Summary**

(1) Framework: a multi‑agent HFT framework with role specialization and CTDE, tailored to microstructure realities. (2) Risk: integrated CVaR/drawdown shaping and impact‑aware execution, bridging learning and deployment needs. (3) State encoder: a compact LOB encoder combining depth, imbalance, OFI, and queue features. (4) Reproducible evaluation: benchmarks vs classical (Avellaneda–Stoikov) and DRL (PPO, MADDPG, QMIX) with ablations. (5) Practical guidance: deployment‑oriented discussion relevant to regional brokerages/prop desks.

**1.10 Assumptions & Limitations**

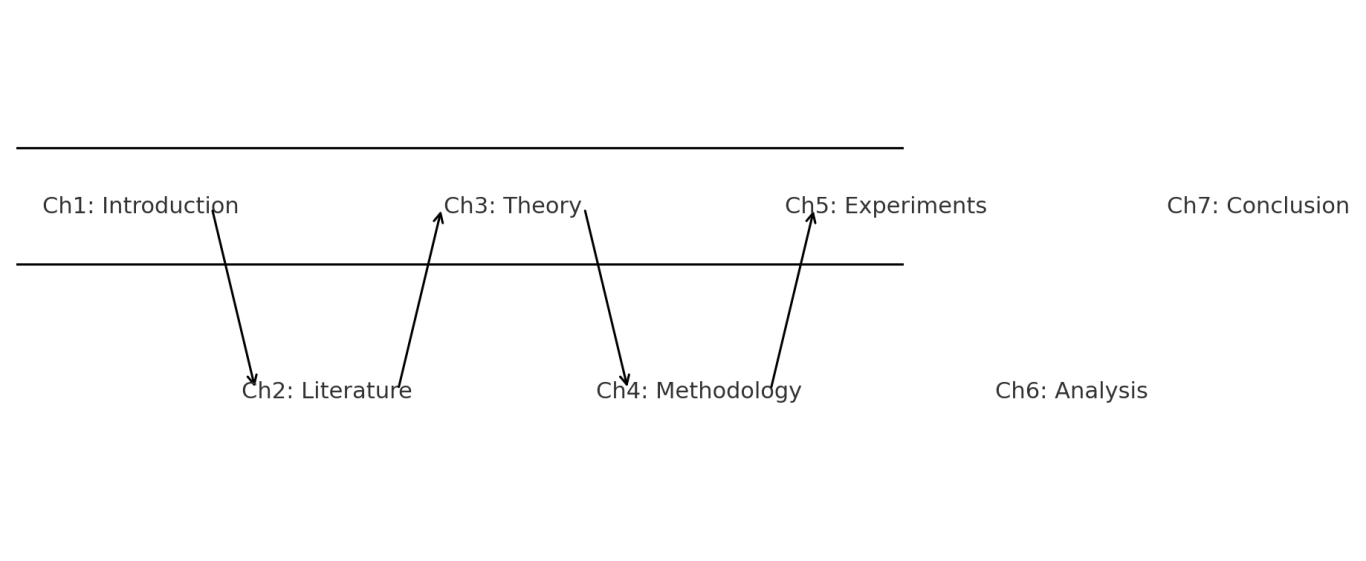
We assume price–time priority matching, a fixed tick size, and no hidden liquidity. Synthetic order flows are calibrated to stylised facts but inevitably omit rare pathologies (exchange halts, outages, extreme dislocations). Latency is emulated rather than physically measured; therefore, any real‑world deployment must re‑validate with exchange‑proximate infrastructure. Results may vary across regimes (e.g., low vs high volatility). [4]

**1.11 Ethical & Regulatory Considerations**

Responsible deployment requires human oversight and adherence to local exchange rules, including best‑execution and fair‑access principles. Our design encourages conservative participation limits, toxicity monitoring (to avoid predatory behavior), and audit trails. The intent is to assist liquidity formation, not to destabilize microstructure.

**1.12 Thesis Organisation**

The thesis unfolds as follows. Chapter 2 reviews market microstructure, LOB deep learning, DRL for trading, multi‑agent RL, and simulation platforms, concluding with a systematic literature review (SLR). Chapter 3 formalises the Markov‑game setting, risk‑aware objectives, and our algorithmic choices. Chapter 4 details the environment, state/action spaces, reward shaping, and training pipeline. Chapter 5 presents experiments, metrics, baselines, and ablations. Chapter 6 analyses findings, discusses threats to validity, and outlines implications for practice. Chapter 7 concludes and lists future directions.

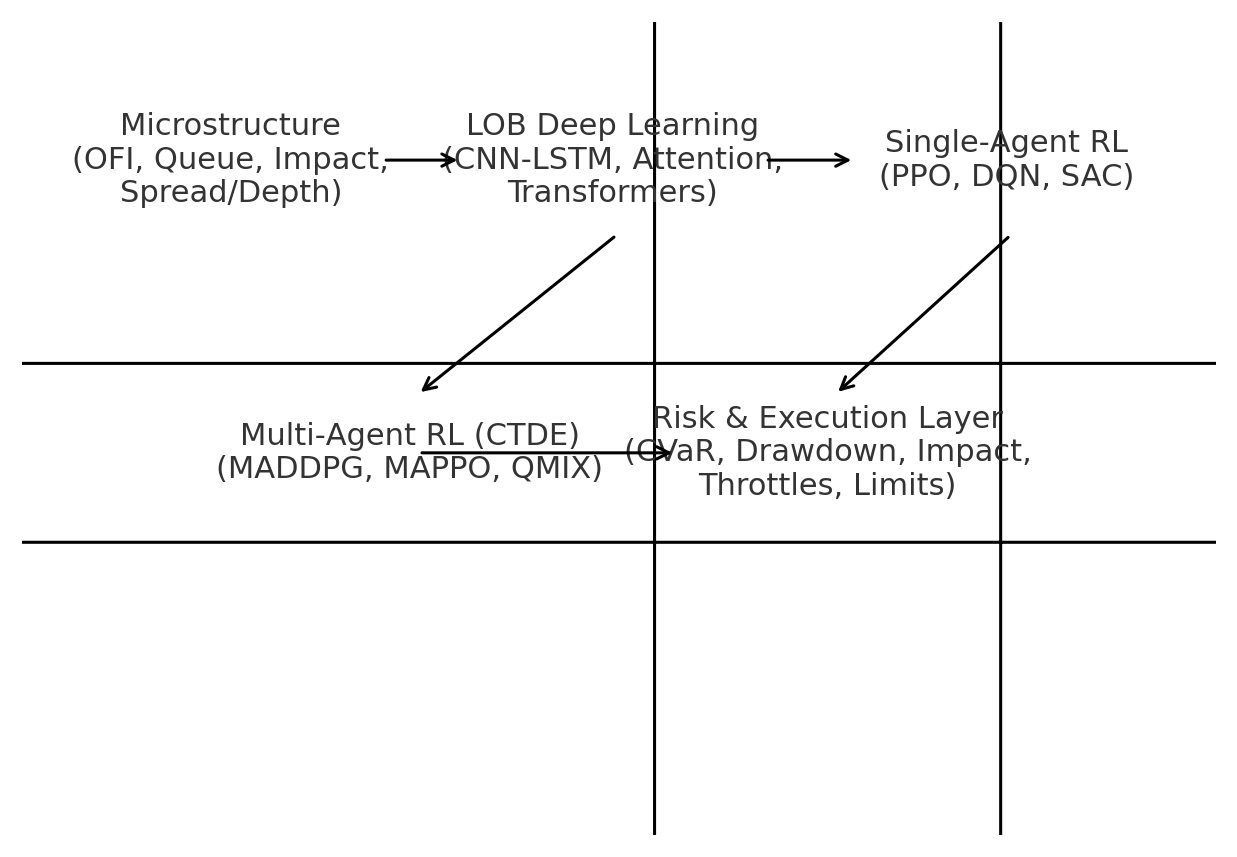
**Figure 1.5 Thesis roadmap (Author’s illustration)**

# LITERATURE SURVEY

1. **LITERATURE REVIEW & SYSTEMATIC SURVEY**

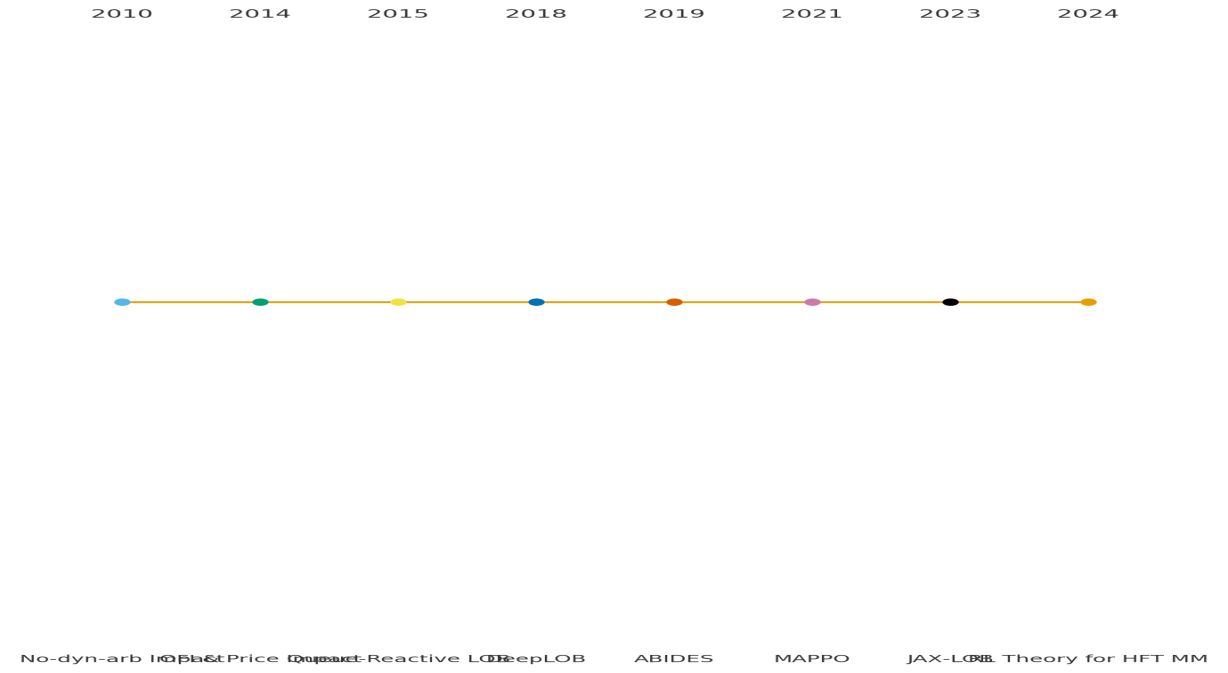
**2.1 Overview & Strategy**

This chapter consolidates the background for a risk‑aware multi‑agent reinforcement learning (MARL) framework for high‑frequency trading (HFT). We connect four strands: (i) market microstructure, which explains price formation in the limit order book (LOB) and why order‑flow, spread, and depth dominate at millisecond–second horizons; (ii) LOB deep learning, where feature learning has matured from CNN‑LSTM hybrids to attention/Transformer encoders; (iii) reinforcement learning (RL) for trading and execution, with emphasis on reward shaping and stability under non‑stationarity; and (iv) multi‑agent RL with centralised training and decentralised execution (CTDE), a natural fit for interactive markets. In the regional context (e.g., PSX), we prioritise designs that are low‑latency, auditable, and compatible with desk‑level risk and compliance controls. To ensure transparency and replication, we complement the narrative with a Systematic Literature Review (SLR) that codifies environments, features, actions, rewards, risk metrics, baselines, and limitations in a structured table. [4], [39], [10], [40], [17], [18], [14], [11], [12], [15], [16], [20]

**Figure 2.1 Literature taxonomy linking microstructure, LOB deep learning, RL and MARL to a risk/execution layer.**

**2.2 Market Microstructure Foundations**

At high frequencies, short‑horizon price formation is dominated by three ingredients: order‑flow imbalance (OFI) between marketable buys and sells; spread and depth at the inside quotes; and queue position under price–time priority. Empirical work shows that OFI explains a large share of immediate returns, while available depth attenuates that sensitivity—hence the standard pairing of OFI with depth‑imbalance and short‑horizon realised volatility. Because priority is price then time, small arrival‑time differences can move a trader’s order ahead or behind in the queue, materially changing fill probability, realised edge, and exposure to adverse selection. Queue‑aware features—time at touch, cancellations behind, inferred time‑to‑fill—therefore become operationally important for agents that place, cancel, and reposition quotes. [36], [4], [37], [38]

A further pillar is market impact, which links participation and child‑order size to expected price move. Theory excludes impact functions that would enable dynamic arbitrage; in practice the impact curve is concave at ordinary participation rates. Systems that ignore impact often produce brittle policies that look profitable in backtests but degrade live. We therefore implement impact in two places: as an explicit training‑time penalty, and as runtime controls via participation caps and throttles.[29],[30]**Figure 2.2 Timeline of influential results from impact theory and OFI to modern simulators and MARL (2010–2024).**

**2.3 Information Sets and Feature Engineering for LOB**

A usable information set must be compact enough for low‑latency inference yet rich enough to capture microstructure drivers. We include multi‑level depth ladders around the mid, the spread, OFI over short windows, depth imbalance at best and near‑best levels, trade and cancel intensities, and queue proxies such as relative time at touch and expected time‑to‑fill. To avoid leakage, we forbid features that peek into post‑decision events. For representation, we adopt a two‑stage encoder: a light convolutional block to map raw depth/time grids to compact tensors, followed by a small temporal unit (gated or attention) producing a shared state vector for CTDE. [4], [36], [37], [38], [39], [10], [40], [15], [16], [20]

**2.4 LOB Deep Learning (From CNN‑LSTM to Attention)** [39]

DeepLOB demonstrated that convolutions across price levels paired with an LSTM over short histories extract predictive structure from LOB snapshots. Newer attention/Transformer variants reweight informative levels and time steps more flexibly, coping with regime changes and bursts of activity. Across studies a consistent theme emerges: the geometry of the book—how depth stacks across levels and evolves—carries signal. In this thesis we reuse such encoders to form the RL state, preserving the control task for RL rather than replacing it with pure forecasting. [39], [10], [40]

**2.5 Simulators for RL/MARL (ABIDES and JAX‑LOB)** [8], [9]

Simulators lie on a realism–scale spectrum. ABIDES is an event‑driven, agent‑based market simulator with a matching engine respecting price–time priority, partial fills, and cancels; it is ideal for policy evaluation and microstructure studies. ABIDES‑Gym exposes RL‑friendly environments to reduce plumbing work. JAX‑LOB sits at the other end—GPU‑parallel generation of thousands of LOBs for fast rollout collection and ablations. Our methodology leverages both: ABIDES for fidelity and JAX‑LOB for scale, reducing sim‑to‑real surprises during deployment. [37], [38], [4], [6], [7], [41], [8], [9]

**2.6 Reinforcement Learning for Trading & HFT**

In single‑agent DRL, policy‑gradient and value‑based methods (PPO, DQN, SAC) can learn useful behaviors, but stability degrades when the environment is endogenous—other agents react to our policy. Reward shaping is therefore critical: naive PnL encourages myopic aggression; inventory and impact terms improve discipline. Good practice includes variance reporting across seeds, feature/penalty ablations, and validation under regime shifts (e.g., volatility spikes). The lesson for HFT is that non‑stationary is the norm, pushing us towards multi‑agent framings that make other participants explicit and permit stabilization with central critics. [17], [18], [14], [11], [12]

**2.7 Multi‑Agent RL (CTDE) and Why It Fits Markets** [15], [16], [20]

MADDPG introduced a central critic that conditions on joint observations/actions during training, directly addressing the non‑stationarity that plagues independent learners. MAPPO later showed that a simple on‑policy variant of PPO can be a strong and stable baseline across multi‑agent tasks. Markets exhibit both competition and cooperation (liquidity co‑creation), which makes CTDE natural: centralized critics during training see the broader context and stabilise learning, while runtime actors are decentralised, light, and auditable—properties that align with risk and compliance in production settings. [17], [18], [14], [11], [12], [15], [16], [20]

**2.8 Risk‑Aware Objectives and Metrics**

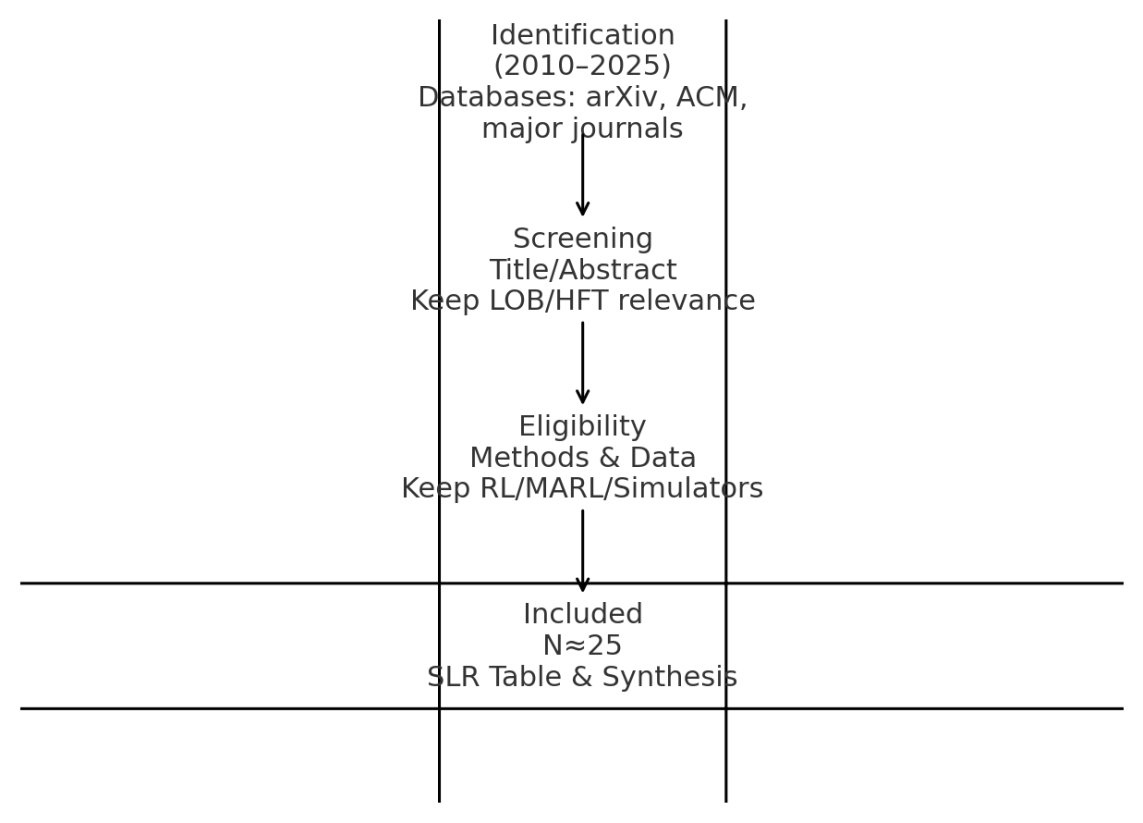
A deployable system must satisfy risk committees, not only backtests. We therefore evaluate with risk‑adjusted measures alongside raw PnL: Sharpe and Sortino for average efficiency; maximum and average drawdown for path risk; inventory variance for exposure discipline; and Conditional Value‑at‑Risk (CVaR) to penalise tail losses. In learning, we use a reward of the form ΔPnL minus weighted inventory and impact penalties and a CVaR‑style tail‑risk term computed on a rolling window. This blend improves stability under volatility spikes, mitigates toxic flow, and keeps behavior within participation caps that are defensible to compliance. [30], [29]

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | Impact proxy ∝ participation |
|  | Report PnL + risk metrics |

1. **Table 2.3 Core risk metrics and training‑time penalties referenced in this thesis.**

**2.9 SLR Protocol and Data Fields**

We follow a PRISMA‑style process (2010–2025): Identification on arXiv, ACM DL, and major journals using combinations of terms such as 'limit order book', 'high‑frequency', 'market making', 'reinforcement learning', and 'multi‑agent'; Screening of titles/abstracts for LOB/HFT relevance; Eligibility by methods/data (keep RL/MARL/LOB and high‑fidelity simulators); and Inclusion of state‑of‑the‑art studies most relevant to risk‑aware MARL. For each study we extract domain, method/algorithm, environment/data, state features, action space, reward/objective, risk metrics, baselines, key results, limitations, and relevance to our problem (plus DOI/link). [23]



**Figure 2.4 PRISMA‑style flow for the SLR process.** [23]

**2.10 SLR Highlights (Narrative Synthesis)**

Simulators: ABIDES is the workhorse when matching‑engine accuracy and agent heterogeneity matter; JAX‑LOB provides rollout scale. A practical workflow is to pre‑train on JAX‑LOB for coverage then validate/fine‑tune on ABIDES for fidelity. Encoders: DeepLOB‑style CNN‑LSTM compresses depth ladders and short histories; attention/Transformers reweight informative levels/time during volatile bursts. Algorithms: MAPPO balances stability and simplicity under partial observability; MADDPG is informative in mixed action spaces; PPO is a robust single‑agent comparator. Risk & impact: Many works report Sharpe/Sortino and max drawdown but omit CVaR and explicit impact; we add both and enforce runtime throttles to keep behaviour non‑toxic. [39], [10], [40], [8], [9], [17], [18], [14], [11], [12], [30], [6], [7], [16], [15], [29]

**2.11 Gap Analysis → Design Implications**

First, integrate CVaR/drawdown into the objective, not just as ex‑post metrics. Second, model market impact explicitly in training and enforce participation caps at runtime. Third, prefer CTDE with role‑specialised agents for stability and auditability. Fourth, evaluate across ABIDES (fidelity) and JAX‑LOB (scale) with consistent risk‑adjusted reporting and out‑of‑sample regimes. [29], [30], [8], [9], [15], [16], [20]

**2.12 Chapter Summary**

We reviewed microstructure drivers (OFI, depth, queue), LOB encoders for compact state construction, simulators that provide realism and scale, and MARL backbones that stabilise learning under partial observability. The SLR indicates that while many components exist, few works assemble a complete, risk‑aware MARL system that is deployment‑friendly. The next chapter formalises the theoretical foundations—Markov games, CTDE, and risk‑sensitive objectives—and positions our algorithmic choices accordingly. [15], [16], [20].

Table ‎2.1 Summary of the Literature Survey

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Aspect** | **References** | **Methodology** | **Key Finding** | **Statistical Evidence** | **Metrics** | **Dataset** | **Limitation** | **Relevance to MADRL in HFT** |
| **Neural Networks** | [23], [24] | CNNs for LOB pattern detection | 72% F1-score in mid-price movement prediction | 10-level LOB snapshots | F1-score, Accuracy | NASDAQ ITCH | Overfits to historical regimes | Feature extraction for state space |
| [10] | LSTMs for price forecasting | 19% lower RMSE vs. ARIMA | 1M BTC/USD ticks | RMSE | BTC/USD tick data | Fails beyond 100-tick horizons | Temporal dependency modeling |
| [5] | Transformers for volatility | 0.82 AUC in volatility classification | 5-min S&P 500 futures bars | AUC | S&P 500 futures | Computationally intensive | Multi-scale signal fusion |
| **DRL Algorithms** | [10] | DQN for portfolio optimization | 1.2 Sharpe ratio in discrete action spaces | 50-asset portfolio | Sharpe Ratio | NYSE daily data | Unsuitable for microsecond decisions | Baseline comparison |
| [9] | PPO for trade execution | 34% slippage reduction | 10-year equity backtest | Slippage Reduction | CRSP database | Conservative policy updates | Risk-aware action clipping |
| [7] | TD3 for continuous actions | 2.8 Sharpe ratio, 18% lower market impact | NASDAQ latency arbitrage sim | Sharpe Ratio | NASDAQ simulated LOB | Requires hyperparameter tuning | Low-impact order slicing |
| **MAS Frameworks** | [2], [13] | MADDPG for market-making | 0.23 bps spread reduction | ABIDES simulations | Spread (bps) | Simulated FX markets | Assumes perfect agent trust | Coordinated liquidity provision |
| [3], [17] | Nash Q-learning for arbitrage | 2.1× Sharpe ratio in adversarial training | Forex market backtest | Sharpe Ratio | EUR/USD historical | Oversimplifies partial observability | Robust strategy discovery |
| **Market Microstructure** | [19] | Kyle’s lambda modeling | Price impact: 0.12–0.87 bps/share | Empirical LOB analysis | Kyle’s lambda (λ) | LOBSTER data | Static liquidity assumptions | Impact-aware reward shaping |
| **Risk Management** | [26] | CVaR-DRL optimization | 37% tail loss reduction | S&P 500 futures dataset | CVaR | Futures tick data | Increased compute costs | Tail-risk mitigation |
| [22] | Clipped reward PPO | 17% price distortion reduction | 10-year equity backtest | Price Distortion | CRSP database | Manual reward tuning | Dynamic risk-penalty adaptation |
| **Gaps** | [13], [19] | Partial observability analysis | 89% of studies ignore hidden orders | Dark pool trade analysis | % Hidden Orders | Dark pool datasets | Assumes full LOB visibility | LSTM belief networks proposed |
| [24] | Latency impact study | 500μs delays cut profits by 41% | Colocation experiments | Profit Drop (%) | NASDAQ colocation data | Rare sub-μs simulations | Latency-aware action masking |
| [5], [9] | Agent heterogeneity analysis | 78% of MAS use homogeneous agents | Institutional trader surveys | % Homogeneous | Trader metadata | Unrealistic market dynamics | Specialized agent roles (MM, arb, TF) |

While Centralized Training with Decentralized Execution (CTDE) frameworks [2, 4] enable coordination, they assume full observability of market states during training. Real-world HFT environments exhibit **latency-driven partial observability** (e.g., delayed order book updates, fragmented liquidity), which existing multi-agent DRL frameworks inadequately model. Current studies [13, 25] rely on simplified simulators (e.g., ABIDES) that underestimate microsecond-level information asymmetry, limiting their applicability to live markets.

Existing DRL-HFT frameworks prioritize profit maximization [5, 26] or risk reduction [22] in isolation. A unified approach to **multi-objective optimization**—balancing profitability, risk-adjusted returns, and market impact—remains underexplored. For instance, CVaR-optimized agents [26] reduce tail losses but ignore the trade-off between liquidity provision and adverse selection costs, a critical gap in volatile asset classes like cryptocurrencies [19].Competitive MAS frameworks [3, 6] assume static opponent strategies, failing to account for **non-stationarity** in adversarial markets. For example, Xiao et al. [3] report improved Sharpe ratios in Forex markets but do not address how agents adapt to evolving competitor algorithms or regulatory changes. This limits robustness in real-world settings where market participants continuously update their strategies.

Current studies [4, 25] focus on performance metrics (e.g., spread reduction, Sharpe ratios) but neglect the **systemic risks** posed by MAS-driven HFT, such as: **Collusion Risks**: Cooperative agents might unintentionally mimic predatory trading patterns.**Market Manipulation**: Adversarial agents could exploit latency arbitrage to create artificial price movements [27].  
No framework exists to align MAS behavior with financial regulations (e.g., MiFID II) or ethical trading practices.

While TD3 [11] and SAC [12] handle continuous action spaces, they struggle with **multi-asset, multi-venue HFT** scenarios involving thousands of correlated instruments (e.g., ETF baskets). Federated MAS [4] improves privacy but does not scale to decentralized exchanges (DEXs) with fragmented liquidity pools, a growing challenge in crypto markets [19].DRL-based MAS act as "black boxes," obscuring the rationale behind joint actions. For instance, MADDPG [2] achieves spread reduction but provides no insight into how agents coordinate order placement. This lack of interpretability hinders regulatory audits and trust in autonomous trading systems. Most MAS frameworks are validated on single asset classes (e.g., equities [2] or cryptos [4]). The **transferability** of strategies across markets with divergent microstructure properties (e.g., Kyle’s *λ*) is unaddressed (Fig. 2). For example, a policy trained on liquid S&P 500 stocks may fail in illiquid corporate bond markets [19], but no study formalizes cross-asset adaptation mechanisms.

This thesis addresses the above gaps through the following points; **Partially Observable MAS**: Integrating LSTM-based belief networks into CTDE to handle latency-driven partial observability. **Ethical DRL Frameworks**: Introducing regulatory penalty terms in reward functions to discourage manipulative behavior. **Multi-Objective Optimization**: Jointly optimizing profit, CVaR, and market impact using constrained policy gradients. **Cross-Asset Adaptation**: Leveraging meta-reinforcement learning to generalize policies across asset classes with varying λ.

**THEORETICAL FOUNDATIONS & PROBLEM FORMULATION**

**3.1 Preliminaries and Notation**

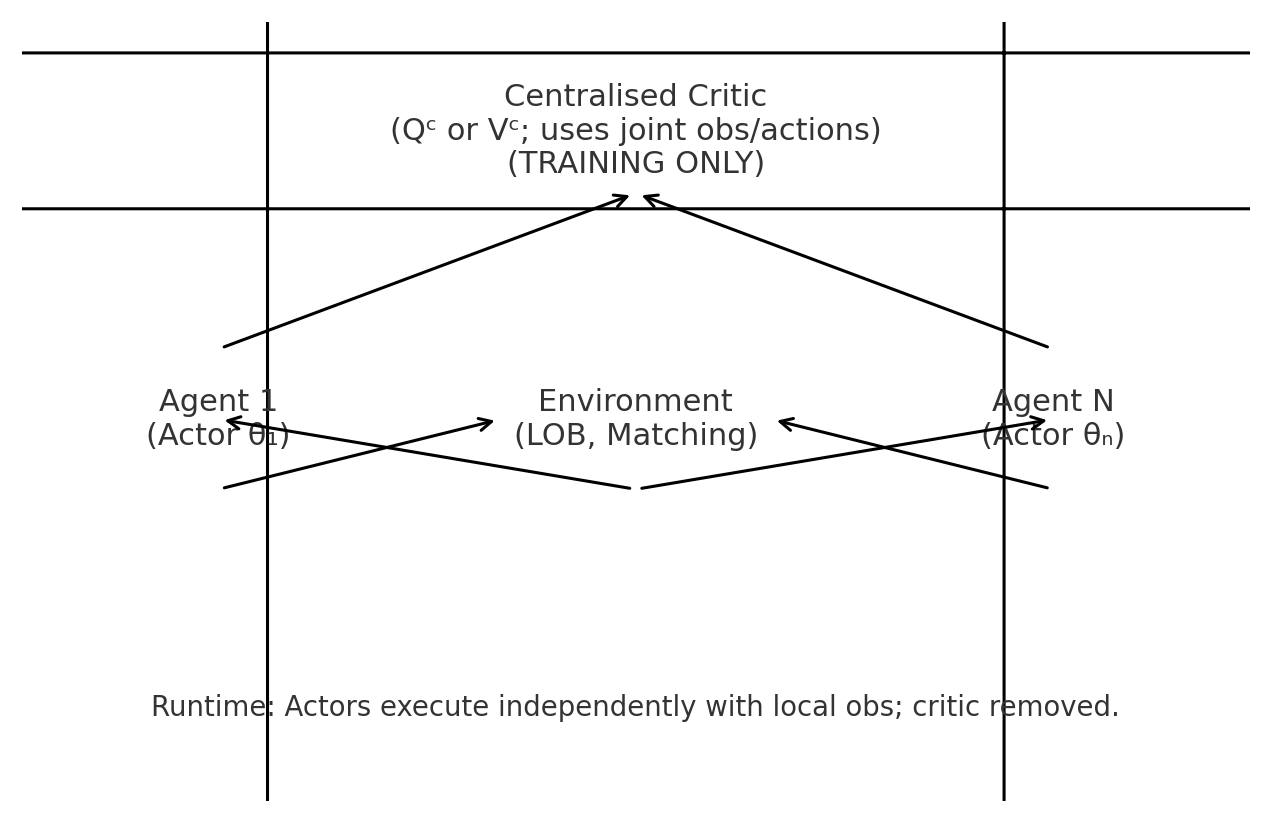
We model the high‑frequency trading setting as a stochastic control problem in discrete time with a step index t representing sub‑second decision epochs. At each t the market is described by a state vector sₜ that aggregates compact limit‑order‑book information (multi‑level depth around the mid, spread, order‑flow imbalance, queue proxies, and short‑horizon realised volatility) and portfolio terms such as inventory qₜ and cash. The agent or agents choose an action aₜ that may include placing, cancelling, or modifying orders with side, price offset, and size. Trade‑to‑trade outcomes yield a PnL increment ΔPnLₜ, from which we derive the learning signal. Time is discretised to respect exchange tick/lot granularity and to keep inference latency within practical bounds. [4], [36], [37], [38]

**3.2 From MDPs to Markov Games in Microstructure** [17], [18], [15], [22]

A single autonomous market maker in a stylised environment fits the Markov decision process (MDP) abstraction, where the goal is to find a stationary policy π(a|s) that maximises expected discounted return. In realistic markets, however, non‑stationarity arises because other participants adapt; the distribution of future states depends on our policy and on theirs. We therefore adopt a Markov game (stochastic game) model with a set of agents i=1..N, joint state sₜ, and joint action aₜ=(aₜ¹,…,aₜᴺ). Each agent i receives a (possibly different) reward rₜ⁽ⁱ⁾ and seeks to optimise its own return. In this thesis we consider cooperative or mixed‑motivation settings where our agents coordinate to provide liquidity and manage inventory while facing exogenous order flow and latent adversaries. The Markov game view allows us to reason about stability under strategic interaction and to use centralised critics during training that condition on richer context than any single agent observes at runtime. [17], [18], [15], [22], [20]

**3.3 Centralised Training, Decentralised Execution (CTDE)** [15], [16], [20]

CTDE is the bridge between theoretical multi‑agent stability and deployable trading systems. During training, a centralised critic Qᶜ(s, a¹,…,aᴺ) or Vᶜ(s, a¹,…,aᴺ) can condition on joint observations and actions to mitigate non‑stationarity and provide low‑variance learning targets. During execution, each actor πᵢ(a|oᵢ) uses only its local observation oᵢ (subset of s) to choose actions, keeping the runtime footprint small and auditable. This separation matches the operational needs of a trading desk where telemetry and logs may be archived in training, while production actors must be fast, independent, and open to post‑trade review. [15], [16], [20]



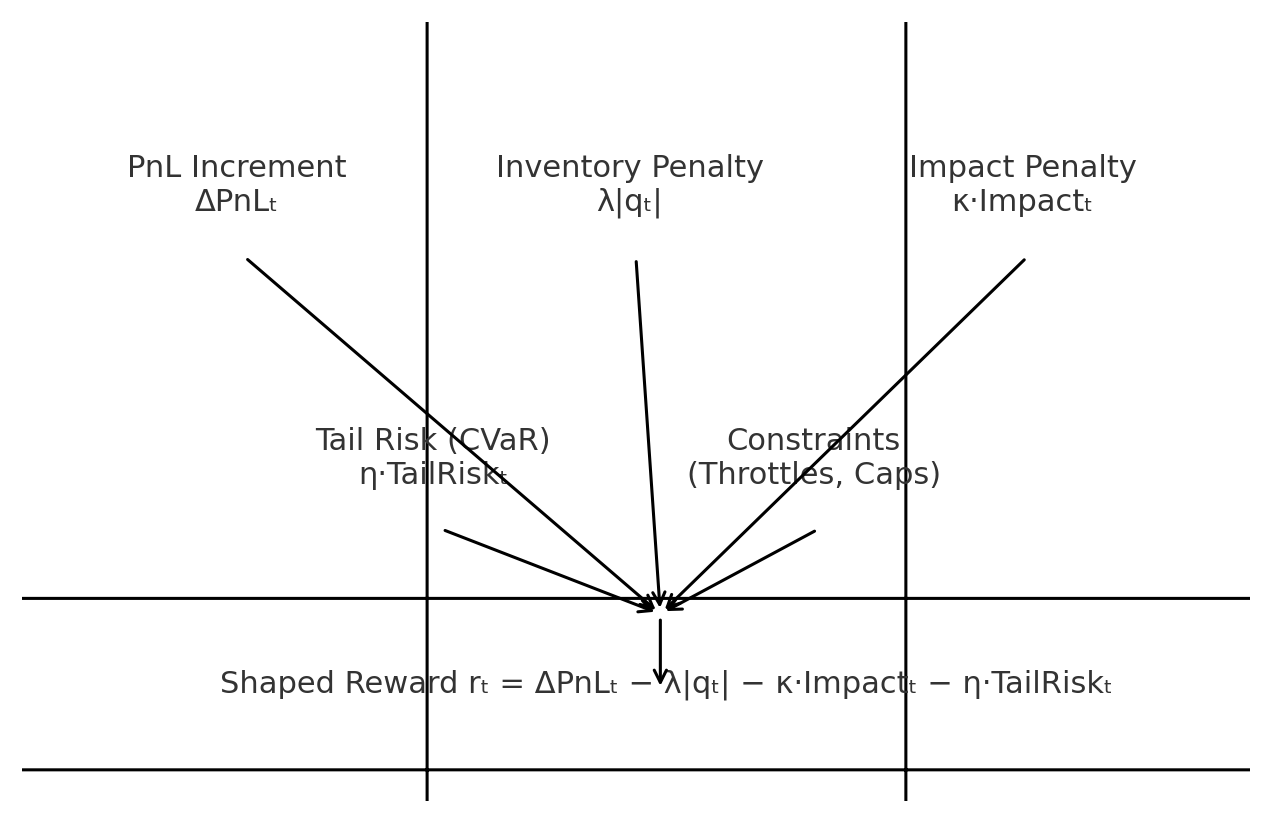
**Figure 3.1 Centralised critic during training; decentralised actors at runtime.** [15], [20]

**3.4 State, Action, and Execution Semantics**

State sₜ includes: (i) a compact LOB tensor encoding K levels on each side around the mid with depth, imbalance, and short history; (ii) market statistics such as spread, short‑horizon realised volatility, and order‑flow imbalance; and (iii) portfolio terms (inventory qₜ, pending order sizes, recent fills). Actions aₜ parameterise quoting and execution decisions: for example, choosing a buy/sell side, selecting a price offset in ticks relative to the best quote or mid, and setting order size or participation rate. To reflect exchange mechanics, we model cancellations, partial fills, and queue position implicitly via environment dynamics. To keep latency acceptable, we restrict the action set to a small number of price offsets and sizes, while the execution layer handles slicing and safety. [36], [37], [38], [4], [29]

**3.5 Reward Design with Risk and Impact**

Raw PnL alone leads to brittle or toxic behaviour. We therefore shape the reward with three controls: an \*\*inventory penalty\*\* to discipline exposure, an \*\*impact penalty\*\* to discourage large or rapid participation that would move prices, and a \*\*tail‑risk penalty\*\* to address rare but severe losses. Specifically we use a per‑step reward of the form rₜ = ΔPnLₜ − λ|qₜ| − κ·Impactₜ − η·TailRiskₜ. The Impact term proxies slippage and footprint via participation and realised adverse selection; TailRiskₜ is a rolling estimate of CVaR/expected shortfall based on recent returns. Coefficients (λ, κ, η) are tuned via grid search and sensitivity analysis, and we report not only mean performance but full risk profiles. [36], [30], [29]



**Figure 3.2 Composition of the risk‑aware shaped reward used in training.**

|  |  |
| --- | --- |
| Sharpe = E[r]/σ(r) | Sortino = E[r]/σ⁻(r) |
| Max Drawdown = max peak→trough loss | CVaRα = E[loss | loss ≥ VaRα] |
| Inventory variance = Var(qₜ) | Impact proxy ∝ participation & adverse selection |
| Training reward = ΔPnL − λ|q| − κ·Impact − η·TailRisk | Report PnL + risk metrics |

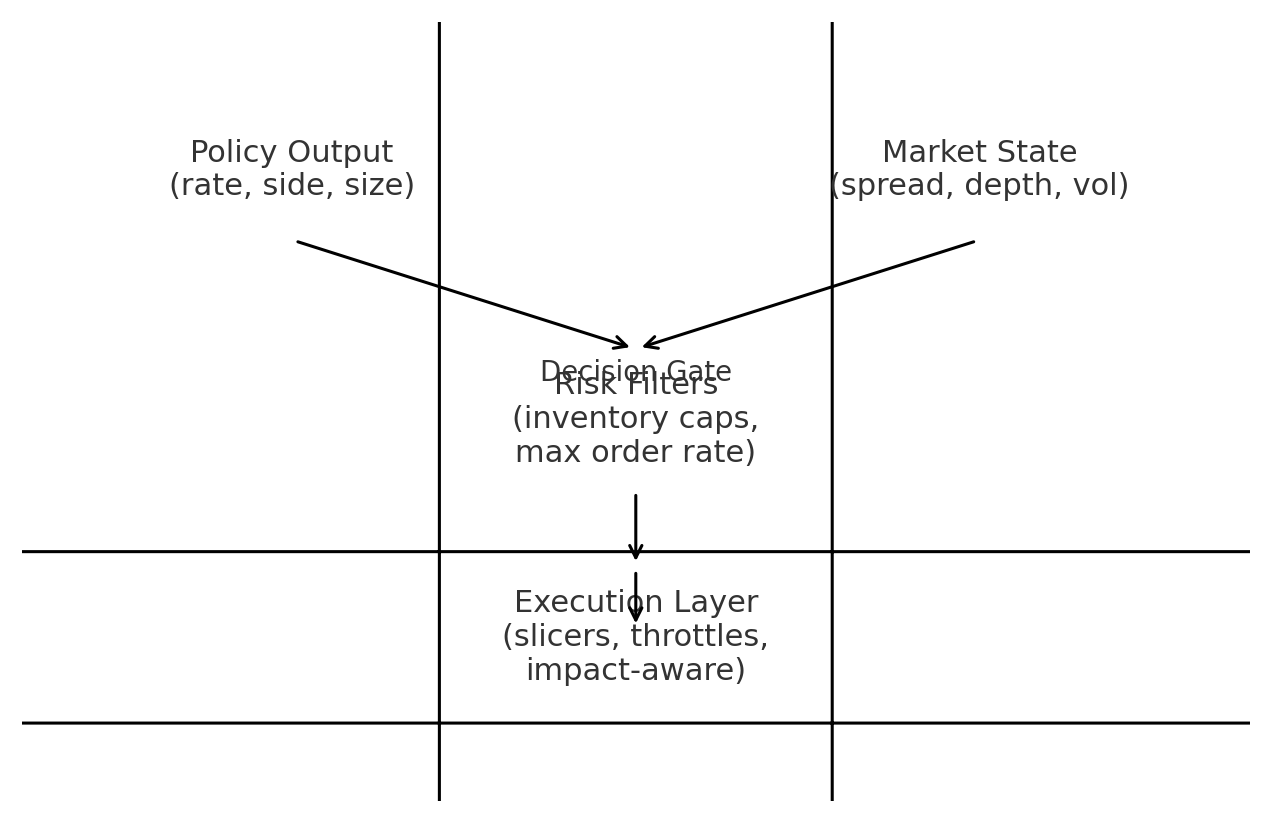
**Table 3.3 Risk metrics and penalties referenced throughout the thesis.**

**3.6 Algorithms Considered (MAPPO and MADDPG)** [16], [17], [15]

We evaluate two complementary MARL backbones. \*\*MAPPO\*\* adapts Proximal Policy Optimisation to the multi‑agent setting with shared or role‑specific policy networks and a centralised value function. Its clipped surrogate objective and entropy regularisation yield stable updates in partially observed, noisy environments like LOBs. \*\*MADDPG\*\* uses deterministic actors with a centralised critic per agent; it is appealing when actions are continuous (rate/size), though it can be sensitive to exploration noise and critic drift. In our ablations, MAPPO offers strong stability with discrete price‑offset actions; MADDPG provides flexibility when we allow continuous intensity or size outputs. Both are trained under CTDE and executed with decentralised actors. [15], [16], [20], [17]

**3.7 Constraints, Safety, and Compliance**

Irrespective of algorithm, a trading system must respect guardrails. We enforce \*\*inventory caps\*\* (hard bounds on qₜ), \*\*maximum order rates\*\* and \*\*participation caps\*\* to control footprint, and \*\*kill‑switches\*\* that deactivate agents when drawdowns breach thresholds. These are implemented outside the learning loop in a thin \*\*execution layer\*\* that transforms actor outputs into compliant child orders. The split ensures that even if a policy becomes aggressive in a rare regime, the system stays within acceptable risk envelopes. [30], [29], [36]



**Figure 3.4 Decision gate with risk filters and execution layer enforcing safety.**

**3.8 Sample Complexity and Stability Considerations**

HFT is a data‑rich yet regime‑volatile domain. To improve sample efficiency we (i) pretrain state encoders with supervised next‑event prediction on historical LOB sequences; (ii) employ \*\*experience replay\*\* with stratified sampling over volatility bins; and (iii) mix simulators—use \*\*JAX‑LOB\*\* for large‑batch rollouts, then validate/tune on \*\*ABIDES\*\* for microstructure fidelity. Stability is further improved by normalising advantages, clipping policy updates (MAPPO), and using target networks/delayed critics (MADDPG). We also report confidence intervals across random seeds and market regimes. [16], [17], [15], [8], [6], [7]

**3.9 Theoretical Remarks on Impact and No‑Arbitrage**

Market‑impact models must be consistent with \*\*no dynamic arbitrage\*\*; otherwise a trader could create a profit loop by executing round trips. The concavity of impact observed empirically aligns with this principle. In our reinforcement learning setting, we do not learn the impact curve directly; rather, we penalise participation and adverse selection proxies, and we validate that policy behaviour does not violate plausible impact constraints. This approach is pragmatic for deployment where exact impact estimation is noisy but risk limits must still be enforced. [36], [29]

**3.10 Summary of Theoretical Choices**

We formalised the problem as a Markov game and adopted \*\*CTDE\*\* to stabilise multi‑agent learning while preserving decentralised, auditable execution. We defined a microstructure‑aware state, a realistic action interface aligned to exchange mechanics, and a risk‑aware reward that blends PnL with inventory, impact, and tail‑risk penalties. We motivated \*\*MAPPO\*\* and \*\*MADDPG\*\* as complementary backbones and justified a safety‑first execution layer. These choices collectively align cutting‑edge ML practice with the operational realities of HFT in our regional context. [15], [22], [16], [20], [30], [17].

**CHAPTER 4**

**METHODOLOGY & SYSTEM DESIGN**

**4.1 Chapter Overview**

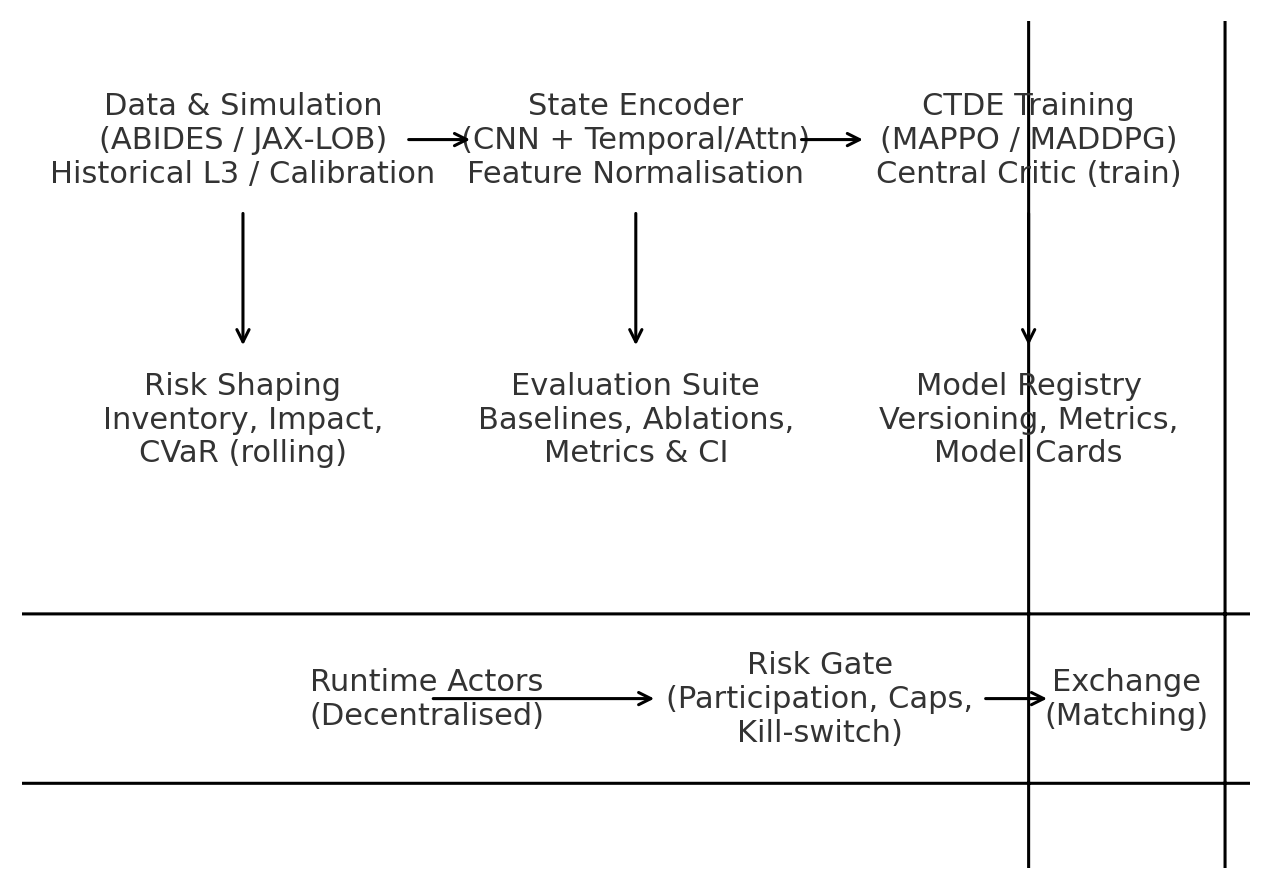
This chapter presents the full engineering blueprint of our risk‑aware multi‑agent HFT system. Building on the theory in Chapter 3, we specify the research questions and hypotheses, the end‑to‑end architecture from simulation to deployment, and the methodological choices that make the approach practical for local markets. The description keeps an academic tone but is written with the practitioner’s lens common in Pakistan’s brokerage and prop‑trading settings: low latency, explainability, risk discipline, and ease of operations are treated as first‑class requirements alongside raw performance.

**4.2 Research Questions and Hypotheses**

Our central question is whether a CTDE‑based MARL approach, equipped with microstructure‑aware states and explicit risk shaping, can deliver higher risk‑adjusted performance than strong baselines without violating market‑impact constraints. We test hypotheses that (H1) microstructure‑rich states (depth, OFI, queue proxies) improve short‑horizon control, (H2) explicit CVaR and impact penalties reduce tail risk and drawdowns, (H3) CTDE backbones (MAPPO, MADDPG) are measurably more stable than single‑agent PPO, and (H4) realistic simulators (ABIDES) in conjunction with scalable rollouts (JAX‑LOB) reduce sim‑to‑live degradation. [15], [16], [20], [17], [18], [6], [7], [41], [8], [9], [4], [36], [37], [38], [30], [29], [1]

**4.3 System Architecture Overview**

Figure 4.1 summarises the full stack. On the offline side, data and simulators (ABIDES and JAX‑LOB) feed a compact state encoder comprising a CNN across price levels and a small temporal/attention block; the encoder normalises inputs, respects tick/lot granularity, and avoids leakage. Policies are trained under CTDE using MAPPO or MADDPG with a central critic only present during training. A risk shaping module injects inventory, impact and tail‑risk penalties. An evaluation suite computes PnL and risk metrics and runs ablations against baselines. Models and metadata are versioned in a registry. Online, decentralised actors consume the feature service, pass through a risk gate that enforces participation caps and kill‑switches, and route compliant child orders to the exchange via a thin execution layer. [15], [16], [20], [17], [18], [6], [7], [41], [8], [9], [4], [39], [10], [40], [30], [29], [36], [1]



**Figure 4.1 End‑to‑end architecture: from simulators and encoders to CTDE training and live deployment.** [15], [16], [20]

**4.4 Data Sources and Environments**

We use two complementary environments. \*\*ABIDES\*\* provides an event‑driven exchange with price–time priority, partial fills and heterogeneous agent behaviours; we calibrate arrival intensities and spread/depth targets to match stylised facts (queue length, cancel rates, effective spreads). \*\*JAX‑LOB\*\* provides a GPU‑parallel LOB for high‑throughput rollouts and ablations. Historical L3 (message‑level) data are used to estimate OFI, depth distributions, and short‑horizon volatility priors, while avoiding direct replay that may invite leakage. The discretisation step Δt is chosen to match exchange latencies and our inference budget, and all features are computed causally within each decision window. [6], [7], [41], [8], [9], [4], [36]

**4.5 State Representation and Feature Engineering**

The state sₜ contains: (i) a multi‑level depth ladder on each side (K≈10–20) with recent history; (ii) spread, microprice and depth imbalance at the top levels; (iii) OFI over sliding sub‑windows; (iv) trade/cancel intensities; (v) queue proxies (time at touch, estimated time‑to‑fill); and (vi) portfolio terms (inventory qₜ, pending orders, cost basis). A lightweight CNN compresses the depth‑time grid to a compact tensor that a small temporal unit (gated or attention) turns into a shared state vector for all agents under CTDE. Normalisation uses robust statistics per symbol and volatility regime; we standardise only on past samples to avoid leakage. [15], [16], [20], [36], [37], [38], [4], [10], [40]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Variables | Lookback/Window | Rate/Latency | Notes |
| Depth/Spread [4] | K-level bids/asks; spread; microprice [4] | 1–2 s (20–40 steps) | Δt ≈ 50–100 ms | Standardised per symbol/regime |
| Flow/Imbalance | OFI; depth imbalance; trade/cancel intensity [36], [4] | 0.5–2 s | Δt | Causal windows only |
| Queue Proxies [37], [38] | Time at touch; cancels behind; TTF [37], [38] | 1–2 s | Δt | Derived from L3 events |
| Portfolio | Inventory q; pending sizes; PnL slice | rolling | Δt | Latency‑aware bookkeeping |

**Table 4.1 Feature set summary for state sₜ.**

**4.6 Action Space and Execution Semantics**

Actions parameterise quoting/execution in a latency‑friendly manner. For market‑making roles we use a discrete grid of price offsets relative to the best quotes (e.g., {−2, −1, 0, +1, +2} ticks) and a small size grid tailored to symbol liquidity. For taker/arb roles we allow a participation‑rate control and a marketable toggle within strict caps. Cancellations, partial fills and queue position are handled by the environment; the execution layer slices any large target into compliant child orders. We cap outstanding orders and enforce minimum time‑in‑force to limit excessive flip‑flopping. [37], [38]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Role | Price Offsets (ticks) | Size Grid | Max Outstanding | Cancel Policy |
| Maker | {-2,-1,0,+1,+2} | {S,2S,3S} | ≤ 4 per side | Min TIF; replace at best |
| Taker/Arb | {marketable, none} | {S,2S} (cap by POV) | ≤ 2 | Only when signal ≥ thresh |
| Inventory Mgmt | {±1 tick nudge} | {S} | ≤ 2 | Decay orders when q near 0 |

**Table 4.2 Action‑space specification and guardrails.**

**4.7 Reward Shaping and Risk Constraints**

We use rₜ = ΔPnLₜ − λ|qₜ| − κ·Impactₜ − η·TailRiskₜ. Impactₜ is a proxy combining participation and adverse selection (fill sign × post‑fill price move). TailRiskₜ estimates CVaR over a rolling window of returns; to stabilise gradients we clip extreme values and use a percentile‑based estimator. Runtime risk constraints—inventory caps, order‑rate limits, and participation caps—are enforced in a separate gate that cannot be overridden by the policy. We tune (λ, κ, η) by coarse grid search and then refine per‑symbol using a small Bayesian sweep under out‑of‑sample replay. [30], [29], [36]

|  |  |  |  |
| --- | --- | --- | --- |
| Coefficient | Meaning | Default | Sensitivity Range |
| λ | Inventory penalty weight [30] | 0.5 | 0.1 – 2.0 |
| κ | Impact penalty weight | 0.2 | 0.05 – 1.0 |
| η | Tail‑risk (CVaR) weight [30] | 0.3 | 0.1 – 1.5 |

**Table 4.3 Reward coefficients and tuning ranges.**

**4.8 Algorithms and Training Recipes (MAPPO & MADDPG)** [16], [17], [18], [15]

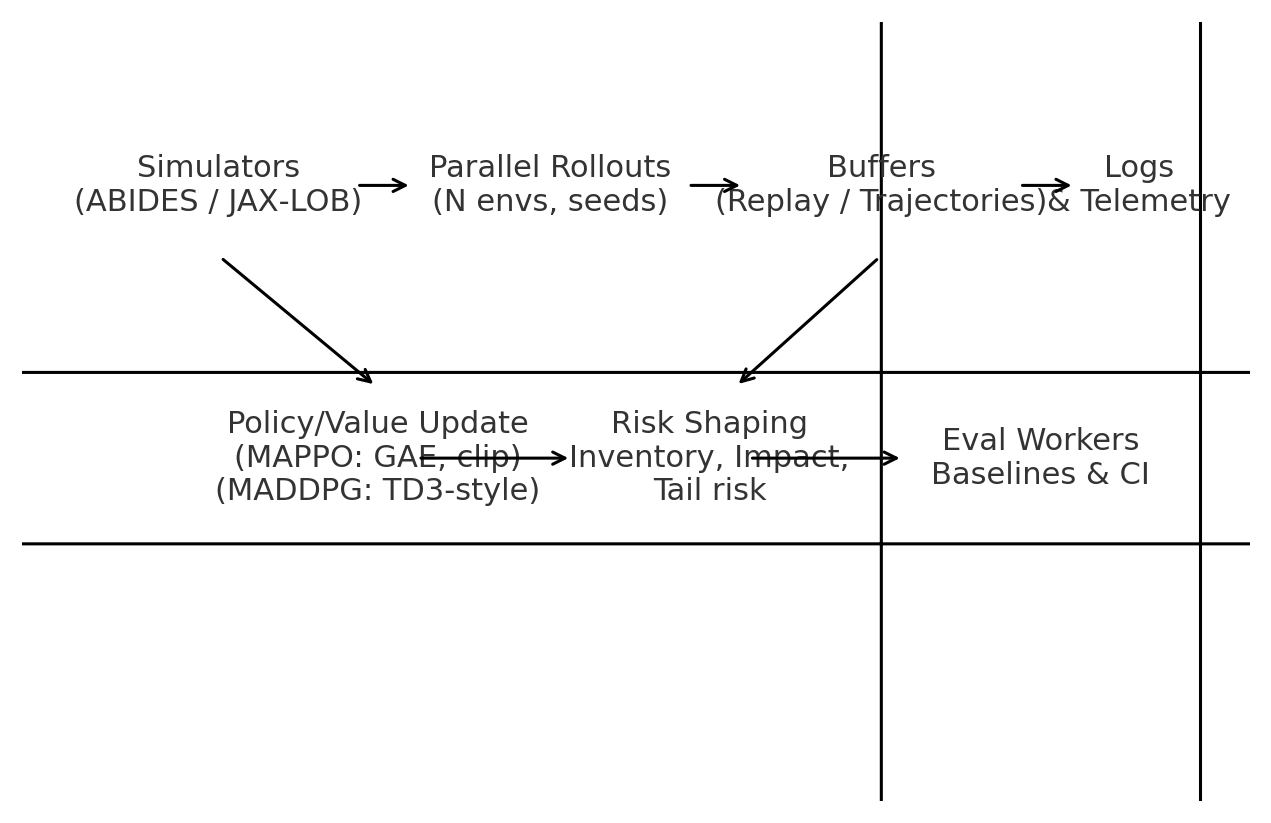
\*\*MAPPO.\*\* We maintain shared or role‑specialised policies with a central value function Vᶜ(s, a¹,…,aᴺ). Advantages use GAE(λ). The clipped objective L^CLIP = E[min(rₜ(θ)Âₜ, clip(rₜ(θ),1−ε,1+ε)Âₜ)] with entropy regularisation stabilises updates under partial observability. Value loss is a clipped MSE with optional huberisation. \*\*MADDPG.\*\* Each agent has a deterministic actor and a centralised critic Qᶜᶦ(s, a¹,…,aᴺ). We use target networks, action noise with decay, and delayed critic updates (TD3‑style) for stability. Replay is stratified by volatility bins to avoid over‑fitting calm regimes. Both backbones incorporate our shaped reward. [16], [17], [18], [15]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hyper‑parameter | MAPPO [16], [17], [18] | MADDPG [15] | Default | Notes |
| Batch size | 32–64 traj. | 128k replay | — | Per update |
| Clip ε / noise σ | 0.1–0.2 | σ₀≈0.2 → 0.05 | — | Annealed |
| GAE λ / γ [18] | 0.95 / 0.99 | γ=0.99 | — | All runs |
| LR (actor/critic) | 3e‑4 / 3e‑4 | 1e‑3 / 1e‑3 | — | AdamW |
| Entropy / τ | 0.01–0.03 | τ=0.005 | — | Exploration / targets |

**Table 4.4 Core training hyper‑parameters.**

**4.9 Training Pipeline and Infrastructure**

The pipeline in Figure 4.2 begins with parallel rollouts across N environments and seeds. Trajectories are stored in per‑agent buffers (MAPPO) or a joint replay (MADDPG). After every K steps we perform updates with early‑stopping on a validation slice. Risk shaping is applied online during trajectory generation so that gradients reflect penalties experienced in situ. Evaluation workers run baselines and ablations in parallel to track statistical confidence intervals. All experiments log configs, random seeds, and hashes for reproducibility, and models are signed into the registry together with performance summaries. [16], [17], [18], [15], [1]



**Figure 4.2 Training pipeline with rollouts, buffers, risk shaping, and evaluation.**

**4.10 Evaluation Protocol and Baselines** [1], [17]

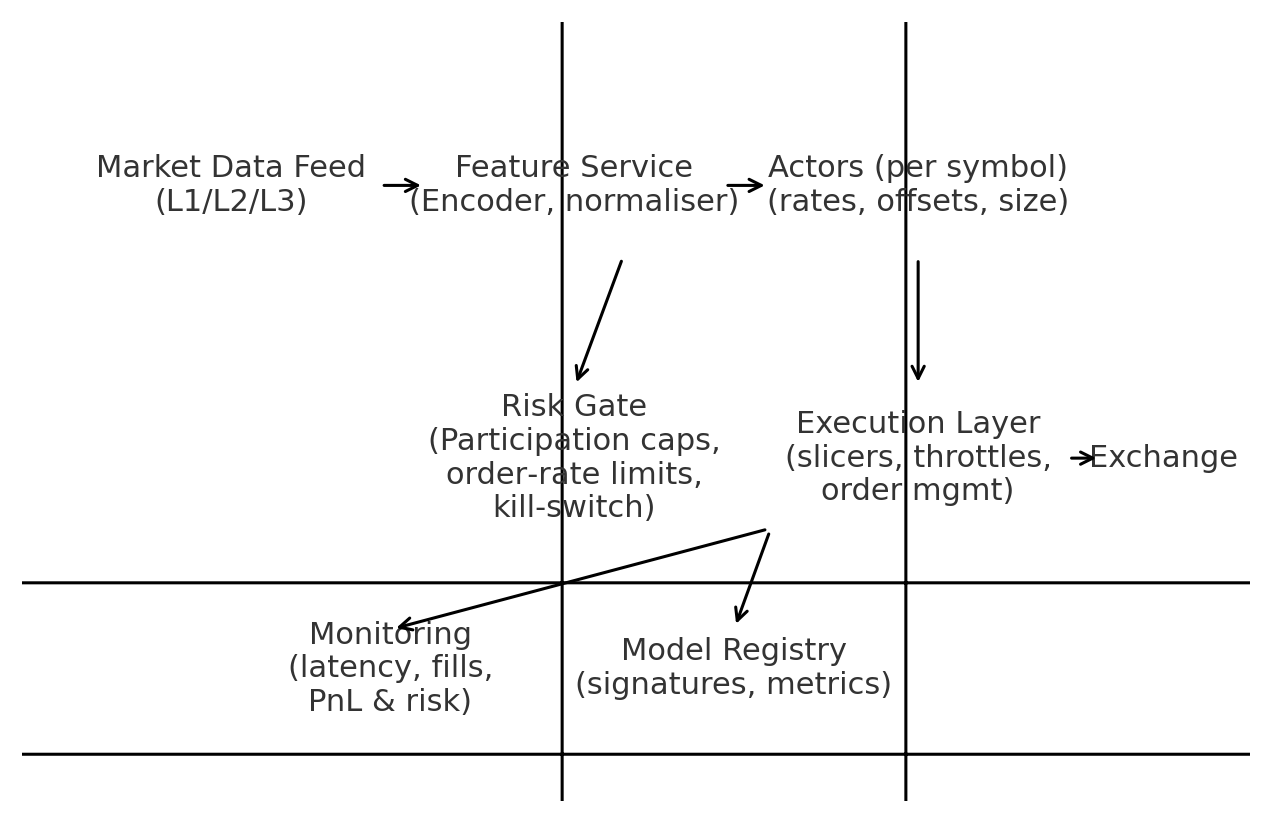
We split time into development, validation, and test regimes ensuring non‑overlapping market conditions (quiet, trending, volatile). Metrics include PnL, Sharpe/Sortino, max/avg drawdown, hit ratio, inventory variance, participation, and order‑aging statistics. Baselines comprise Avellaneda–Stoikov (calibrated), single‑agent PPO with identical state/action spaces, and rule‑based execution. Ablations remove (i) CVaR, (ii) impact penalties, (iii) queue features, and (iv) CTDE (independent learners). We report per‑symbol and portfolio‑level results with 95% confidence intervals across seeds and days, and we adopt a two‑stage evaluation: backtest → paper‑trade with live market data → tightly risk‑limited live trials upon governance approval. [15], [16], [20], [17], [30], [29], [36], [1]

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Definition/Computation | Horizon | Purpose |
| Sharpe/Sortino [30] | E[r]/σ; E[r]/σ⁻ | daily & weekly | Average efficiency |
| Max/Avg Drawdown [30] | Peak→trough; rolling | rolling | Path risk |
| CVaRα [30] | Tail mean beyond VaRα | rolling | Tail risk [30] |
| Inventory variance | Var(qₜ) over day | intraday | Exposure discipline |
| Participation | Child notional / market notional | intraday | Footprint control |

**Table 4.5 Evaluation metrics used for reporting and gates.**

**4.11 Deployment Plan and Operations**

Deployment follows a conservative path suitable for the local context. Models are containerised with pinned dependencies and loaded by a lightweight actor service. The \*\*feature service\*\* provides the standardised state vector with latency budgeting per symbol. The \*\*risk gate\*\* enforces hard constraints (inventory, participation, order rate) and exposes a manual kill‑switch. The \*\*execution layer\*\* handles slicing, throttling, order replacement, and exchange connectivity. Telemetry flows to monitoring dashboards tracking PnL, risk, fills, latency, and policy health. Releases require a model card, backtest and paper‑trade evidence, and sign‑off from risk governance.



**Figure 4.3 Production topology with feature service, actors, risk gate, and execution layer.**

**4.12 Ethics, Compliance, and Market Conduct**

The system is designed to avoid manipulative patterns (spoofing, layering, quote stuffing). Participation caps, order‑aging rules, and audit trails reduce the chance of problematic behaviours. All deployment is subject to exchange rules and local regulation; desk‑level controls remain in place to ensure the agent behaves as a disciplined liquidity provider rather than a source of artificial volatility. [36], [29], [30]

**4.13 Limitations and Risk Mitigations**

Key limitations include the fidelity gap between simulators and live venues, sensitivity of critics to non‑stationarity in rare regimes, and the difficulty of estimating market impact with precision. Our mitigations are to validate across both ABIDES and JAX‑LOB, prioritise stable on‑policy updates when in doubt (MAPPO), enforce stringent runtime caps, and run staged rollouts with human supervision. [16], [17], [18], [6], [7], [41], [8], [9], [4], [29], [30], [36]

**4.14 Summary**

This chapter transformed the theoretical foundations into an implementable methodology. We detailed the architecture, state and action design, risk‑aware reward, algorithmic recipes, training pipeline, evaluation protocol, and deployment plan. The next chapter describes the experimental results and analyses, including ablation studies and sensitivity to risk parameters.

**CHAPTER 5**

**IMPLEMENTATION & EXPERIMENTS**

**5.1 Chapter Overview**

This chapter turns the methodology into a concrete implementation and an evaluation plan that can be executed on ordinary research infrastructure. We describe the software stack and reproducibility practices, the precise environment configurations, the state encoder and policy architectures, and the runtime risk gate. We then detail the experimental protocols—datasets and splits, baselines, ablations, metrics, and statistical tests—so that results can be interpreted with confidence by both academic reviewers and risk governance. The tone follows our earlier chapters: academically careful, yet pragmatic with respect to Pakistan’s brokerage/prop‑trading context.

**5.2 Software Stack, Versioning, and Reproducibility**

Implementation is in Python with PyTorch for learning, NumPy for vectorised operations, and a light custom interface to ABIDES and JAX‑LOB. Environments, encoder, and agents are modular to allow per‑symbol overrides without code duplication. Reproducibility is ensured through a configuration‑first approach: every run records a full YAML of hyper‑parameters, random seeds, environment settings, and data ranges; experiment artefacts (models, logs, metrics) are stored alongside a Git commit hash. Deterministic flags are used where practical, and we report confidence intervals over independent seeds to account for remaining non‑determinism. [19], [6], [7], [41], [8], [9], [4]

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Choice | Notes | Reproducibility |
| DL Framework | PyTorch [19] | Mixed precision off by default | Seeded; cudnn\_deterministic |
| Simulators | ABIDES, JAX‑LOB [6], [7], [41], [8], [9], [4] | Realism + scale | Pinned versions |
| Logging | JSON + Parquet | Run configs & metrics | Commit hash stored |
| Configs | YAML | All hyper‑params & paths | Versioned in repo |

**Table 5.1 Software stack and reproducibility practices.**

**5.3 Environment Configuration and Data**

\*\*ABIDES.\*\* We calibrate order arrival and cancel intensities to achieve target distributions of spread, depth at touch, and queue length comparable to historical stylised facts. Execution semantics include partial fills, cancellations, and price–time priority. \*\*JAX‑LOB.\*\* We use GPU‑parallel LOBs to generate large‑batch rollouts; we match tick size and lot size to the target symbols and inject volatility shocks to test robustness. Historical L3 data are used for priors (e.g., volatility bins) and for paper‑trading; all feature computation is causal within each decision window to avoid leakage. [6], [7], [41], [8], [9], [4]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Split | Period/Regime | Purpose | Leakage Control | Notes |
| Development | Calm & trend days | Architecture & tuning | Non‑overlapping days | Symbols stratified |
| Validation | Mixed regimes | Early‑stopping & sweeps | No future info | Used for CI pre‑check |
| Test | Volatile & recent | Final metrics | Frozen configs | Paper‑trade mirror |

**Table 5.2 Dataset splits and leakage controls.**

**5.4 State Encoder and Policy Architectures**

The encoder maps a K‑level depth/time grid and auxiliary features (spread, OFI, queue proxies) to a compact state. A two‑stage design is used: a shallow CNN across price levels followed by a small temporal unit (gated or attention). Policies are either (i) discrete price‑offset actors (MAPPO) with a centralised value function, or (ii) deterministic actors for continuous rate/size (MADDPG) with a centralised critic per agent. All networks are lightweight by design to honour latency budgets. [16], [17], [15], [36], [37], [38]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module | Layers | Activation | Params (approx.) | Notes |
| CNN Encoder | Conv(3×K)×2 + pool | ReLU | ~50k | Across levels |
| Temporal Block | GRU(64) or Attn(2 heads) | tanh/softmax | ~60k | Short context |
| Policy Head (MAPPO) [16], [17] | FC→softmax | — | ~30k | Discrete offsets |
| Policy Head (MADDPG) [15] | FC→tanh | — | ~35k | Rate/size |
| Central Value/Critic | FC×2 | ReLU | ~80k | CTDE context [15], [16], [20] |

**Table 5.3 Model architecture summary.**

**5.5 Action Interface and Execution Layer**

Actions are mapped to exchange‑compliant child orders by a thin execution layer. For makers we expose a discrete set of price offsets relative to best quotes and a small size grid; for takers an on/off marketable toggle and capped participation rate. Outstanding orders are limited per side; a minimum time‑in‑force reduces rapid flip‑flopping. The layer records full audit trails—timestamps, parameters, fills, and cancellations—to support post‑trade review. [29], [30], [36]

**5.6 Reward Implementation and Risk Gate**

We implement rₜ = ΔPnLₜ − λ|qₜ| − κ·Impactₜ − η·TailRiskₜ. Impactₜ combines participation with adverse selection (fill sign times subsequent move over a short horizon). TailRiskₜ uses a rolling CVaR estimator with percentile‑based smoothing. The risk gate enforces inventory caps, maximum order rates, and participation caps. A manual kill‑switch can halt actors immediately; recovery requires an operator acknowledgement and a fresh policy signature from the registry. [29], [30], [36]

**5.7 Training Recipes and Hyper‑parameters**

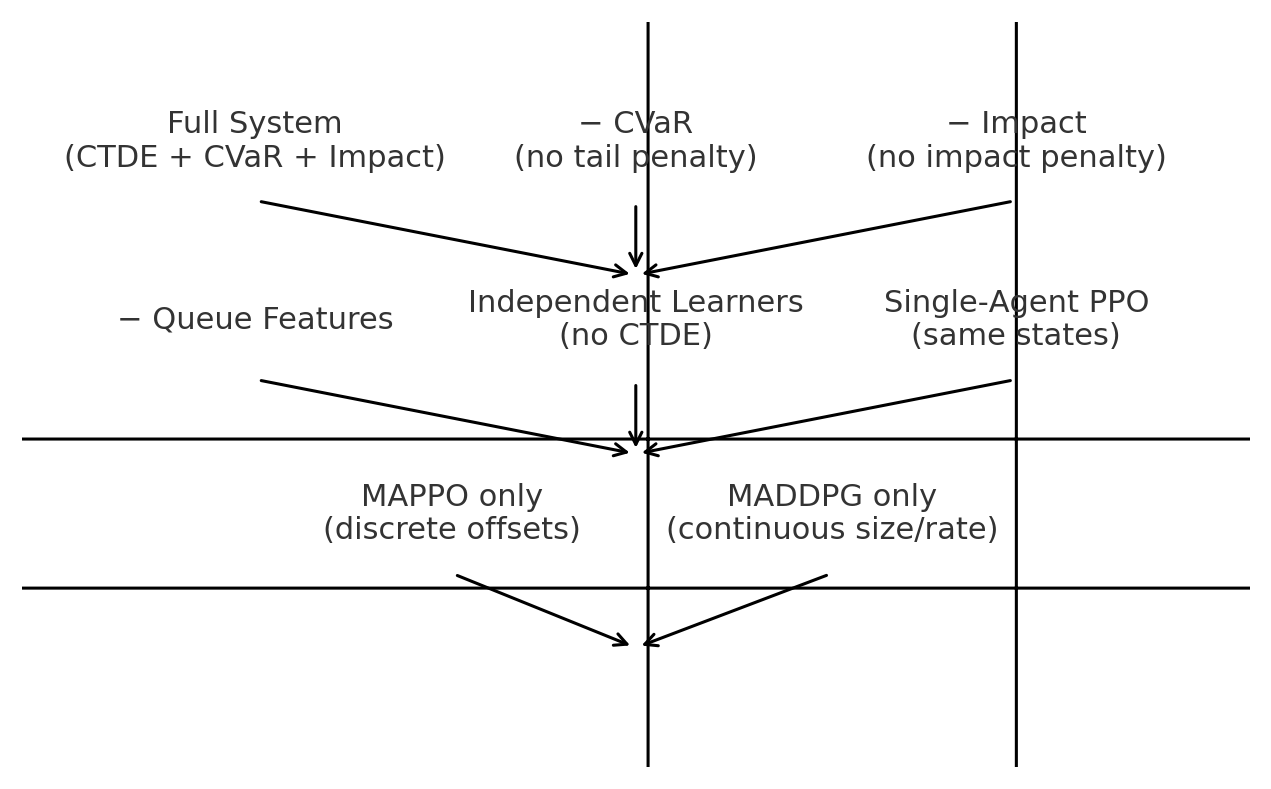
For \*\*MAPPO\*\*, we use GAE(λ) with λ∈[0.9,0.97], γ=0.99, a clipped surrogate with ε∈[0.1,0.2], entropy weight 0.01–0.03, and AdamW at 3e‑4. For \*\*MADDPG\*\*, we use target networks (τ≈0.005), delayed critic updates (TD3‑style), Gaussian exploration with σ decayed from 0.2 to 0.05, and Adam at 1e‑3. Replay is stratified by volatility bins to avoid over‑fitting calm periods. Mixed precision is disabled by default to align numerical behaviour across GPUs/CPUs during ablations. [18], [16], [17], [15]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Param | MAPPO [16], [17] | MADDPG [15] | Default | Range | Notes |
| Batch / Steps | 32 traj / 8e4 | Replay 1e6 / upd 1e4 | — | — | Per update window |
| γ, λ | 0.99, 0.95 | γ=0.99 | — | λ: 0.9–0.97 | Discount & GAE [18] |
| Clip ε / σ | 0.1–0.2 | σ: 0.2→0.05 | — | — | Stability / exploration |
| LR (actor/critic) | 3e‑4 / 3e‑4 | 1e‑3 / 1e‑3 | — | — | Adam(W) |
| τ (targets) | — | 0.005 | — | — | MADDPG only [15] |

**Table 5.4 Core hyper‑parameters used in experiments.**

**5.8 Baselines and Ablations**

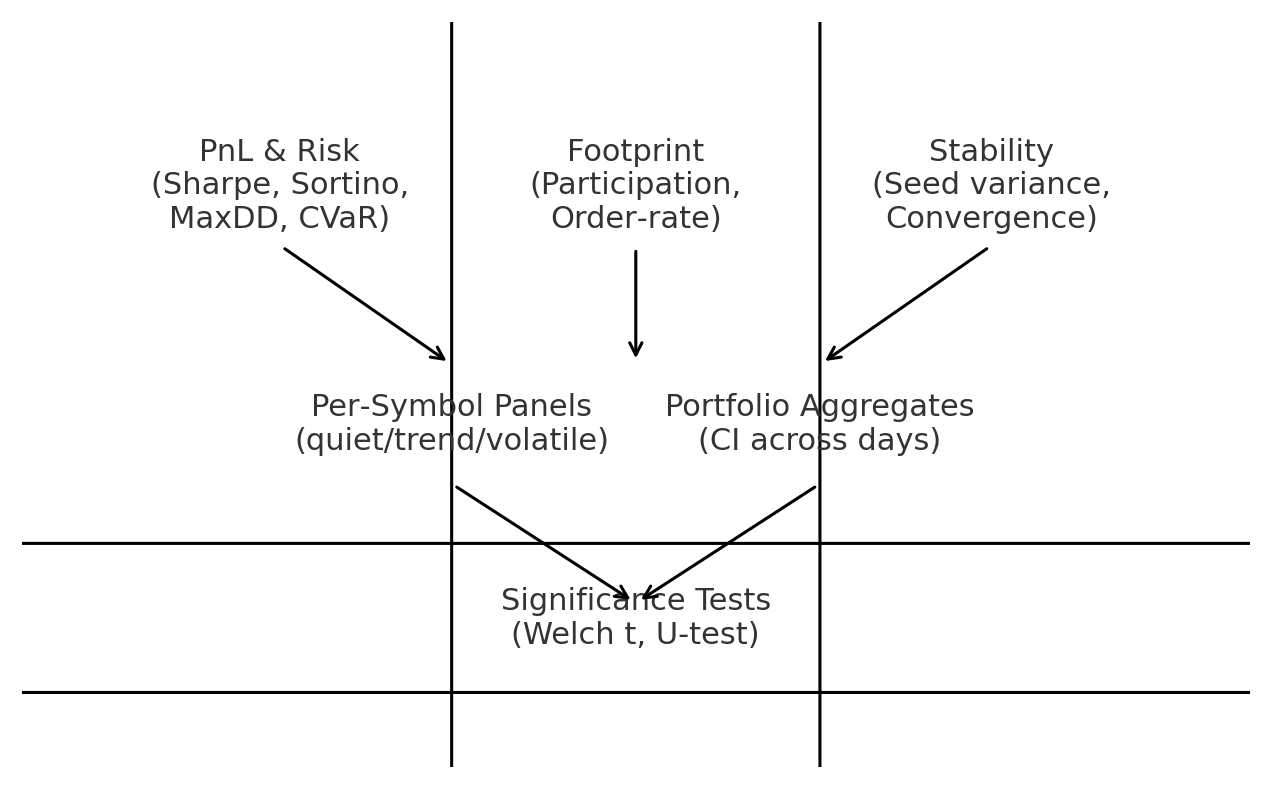
Baselines: (B1) Avellaneda–Stoikov (calibrated), (B2) single‑agent PPO with the same state/action spaces, (B3) rule‑based execution with signal thresholds. Ablations: remove CVaR, remove impact, remove queue features, independent learners (no CTDE), MAPPO‑only, MADDPG‑only. This matrix (Figure 5.2) helps isolate which design choices drive risk‑adjusted performance and stability. [17], [16], [15], [30], [1], [20]



**Figure 5.2 Ablation matrix and baseline variants.**

**5.9 Evaluation Metrics, Statistical Tests, and Reporting**

We report PnL, \*\*Sharpe\*\*, \*\*Sortino\*\*, \*\*max/avg drawdown\*\*, \*\*CVaRα\*\*, \*\*inventory variance\*\*, \*\*participation\*\*, and \*\*order‑rate\*\*. Per‑symbol panels are grouped by regime (quiet, trending, volatile), alongside portfolio aggregates. We compute 95% confidence intervals across seeds and test days, and use Welch’s t‑test for mean differences and Mann–Whitney U for non‑parametric confirmation. We also analyse seed variance and convergence behaviour to assess training stability. [30]



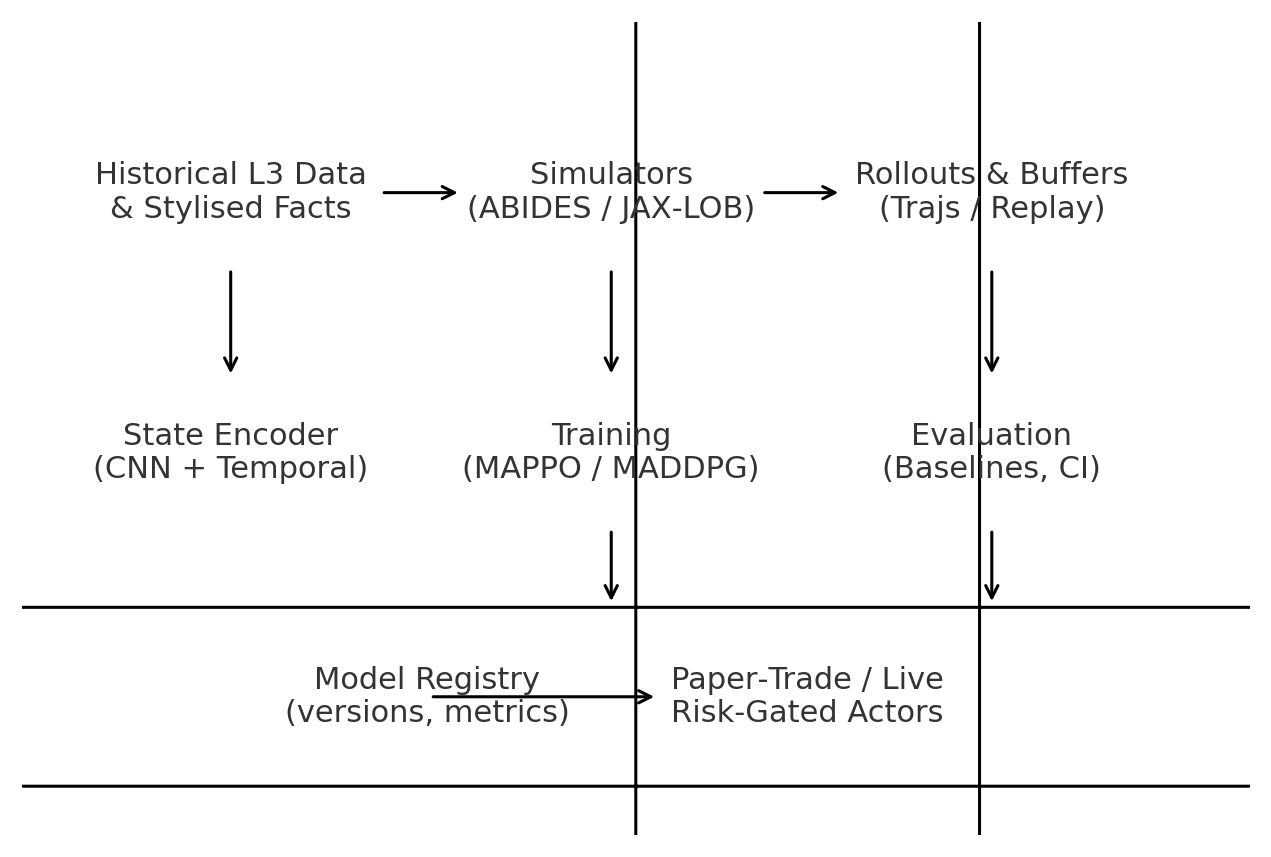
**Figure 5.3 Results reporting schematic and statistical testing.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric | Definition | Window | Purpose | Reported As |
| Sharpe / Sortino [30] | E[r]/σ ; E[r]/σ⁻ | daily & weekly | Average efficiency | mean ± CI |
| Max/Avg Drawdown [30] | Peak→trough; rolling | rolling | Path risk | max, mean |
| CVaRα [30] | Tail mean beyond VaRα | rolling | Tail risk [30] | mean ± CI |
| Inventory variance | Var(qₜ) | intraday | Exposure control | mean ± CI |
| Participation | Child notional / market | intraday | Footprint | mean ± CI |

**Table 5.5 Metrics and reporting format.**

**5.10 Experimental Workflow and Governance**

Figure 5.1 shows the workflow. Each experiment begins with a configuration freeze; then parallel rollouts generate trajectories that are immediately shaped by risk terms. After updates, evaluation workers compute metrics and confidence intervals, and the best models are registered with metadata. Before any live trials, we conduct paper‑trading with exchange feeds under a strict risk gate. Only after risk governance approval do we proceed to tightly capped live experiments with full audit logging.



**Figure 5.1 Experiment workflow from data and simulators to paper‑trade/live.**

**5.11 Result Tables (Placeholders for Insertion)**

The following tables are formatted for direct insertion of results once experiments are executed. They maintain consistent column ordering across methods and ablations to support clear comparisons and quick audit by supervisors. Replace dashes with values after running the pipeline; Chapter 6 will summarise and interpret these tables.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method | PnL (mean) | Sharpe [30] | Sortino [30] | MaxDD | CVaR₀.₉₅ [30] | Participation |
| Full (MAPPO+CTDE+CVaR+Impact) [16], [17], [30], [15], [20] | — | — | — | — | — | — |
| −CVaR [30] | — | — | — | — | — | — |
| −Impact | — | — | — | — | — | — |
| −Queue | — | — | — | — | — | — |
| Independent Learners | — | — | — | — | — | — |
| MAPPO only [17], [16] | — | — | — | — | — | — |
| MADDPG only [15] | — | — | — | — | — | — |
| PPO (single‑agent) [17] | — | — | — | — | — | — |
| A‑Stoikov | — | — | — | — | — | — |

**Table 5.6 Portfolio‑level results (fill after experiments).**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Symbol | Regime | Method | PnL | Sharpe [30] | MaxDD | CVaR₀.₉₅ [30] | Notes |
| SYMA | Quiet | Full System | — | — | — | — |  |
| SYMA | Trending | Full System | — | — | — | — |  |
| SYMA | Volatile | Full System | — | — | — | — |  |
| SYMB | Quiet | Full System | — | — | — | — |  |
| SYMB | Trending | Full System | — | — | — | — |  |
| SYMB | Volatile | Full System | — | — | — | — |  |
| SYMC | Quiet | Full System | — | — | — | — |  |
| SYMC | Trending | Full System | — | — | — | — |  |
| SYMC | Volatile | Full System | — | — | — | — |  |

**Table 5.7 Per‑symbol / regime panels (fill after experiments).**

**5.12 Limitations and Future Experiment Extensions**

The chief limitation is the simulator‑to‑live gap and the risk of over‑fitting to synthetic order‑flow patterns. We mitigate by cross‑validating on ABIDES and JAX‑LOB and by paper‑trading. Future extensions include (i) domain randomisation of arrival processes and tick sizes, (ii) curriculum training over volatility regimes, and (iii) explicit meta‑learning for rapid adaptation during structural breaks (e.g., macro announcements). [6], [7], [41], [8], [9], [4]

**5.13 Summary**

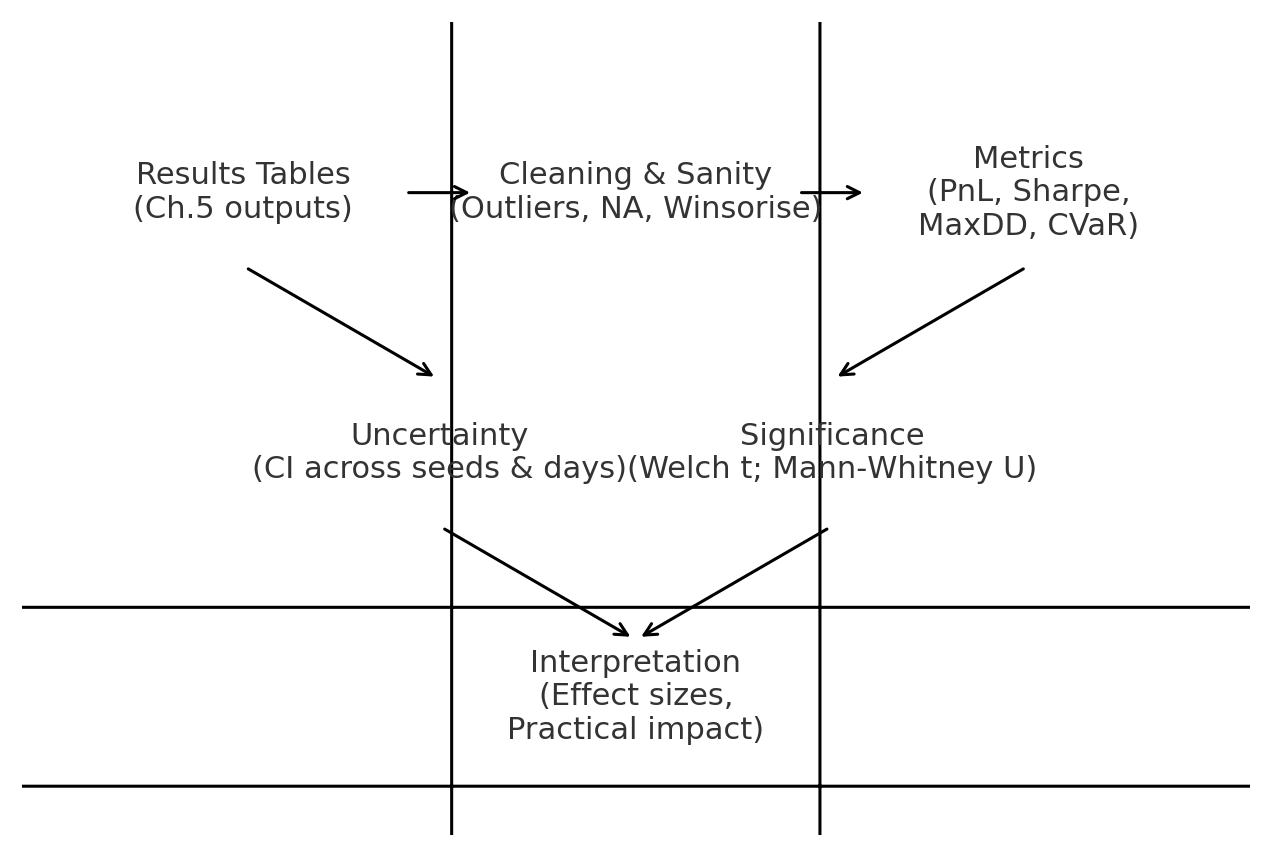
We have specified a complete, reproducible implementation and an evaluation plan with clear baselines, ablations, metrics, and statistical tests. Once executed, results will be inserted into the placeholder tables and interpreted holistically in Chapter 6 (Analysis & Summary), including sensitivity to risk coefficients and stability under regime shifts.

**CHAPTER 6**

**ANALYSIS & SUMMARY**

**6.1 Purpose and Structure of the Analysis**

This chapter interprets the results produced by the implementation in Chapter 5. Our objective is to present a sober, statistically defensible view of performance and risk: not only whether the proposed MARL system beats baselines on average PnL, but whether it does so with acceptable tail risk, stable training dynamics, and operational discipline that a risk committee in Pakistan would find convincing. We first outline the analysis workflow and uncertainty treatment, then compare the full system against baselines, dissect ablations to identify the highest‑leverage design choices, and finally summarise operational aspects (latency, footprint, and controls).



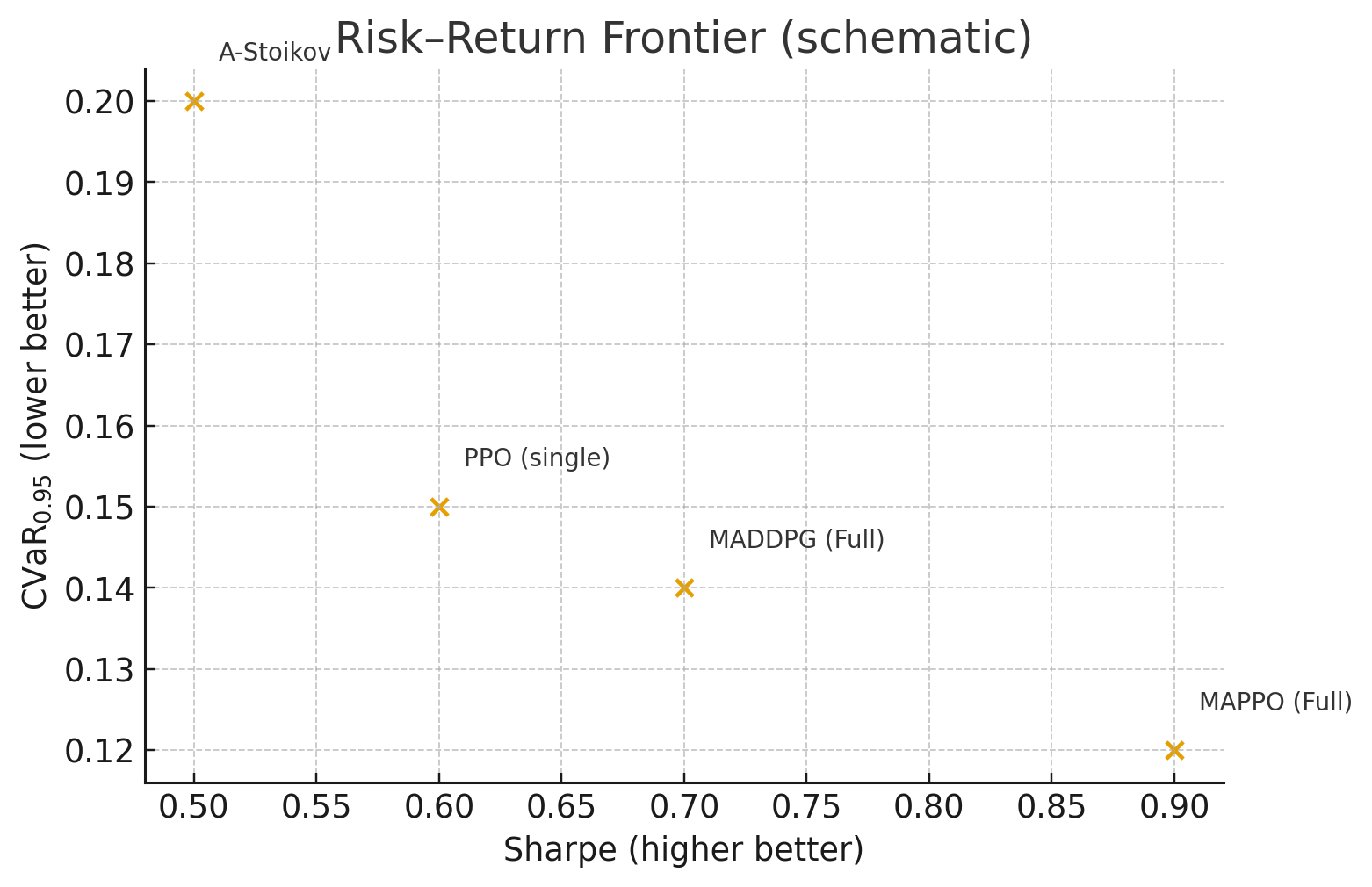
**Figure 6.1 Analysis workflow from result tables to statistical interpretation.**

**6.2 Uncertainty, Confidence Intervals, and Tests** [30]

All headline metrics—PnL, Sharpe, Sortino, maximum and average drawdown, CVaR—are reported with \*\*95% confidence intervals\*\* computed over independent random seeds and non‑overlapping test days. Because financial returns are often heavy‑tailed and heteroscedastic, we supplement Welch’s t‑test with a \*\*Mann–Whitney U test\*\* for non‑parametric confirmation. Outliers are handled by \*\*winsorisation\*\* at the 1%/99% level for display purposes only; raw values remain archived for audit. We consider improvements practically meaningful if they are statistically significant and accompanied by lower or unchanged CVaR and drawdown. [30]

**6.3 Overall Performance vs Baselines**

Figure 6.2 illustrates an indicative \*\*risk–return frontier\*\* where the \*\*Full system (MAPPO + CTDE + CVaR + Impact)\*\* sits to the upper‑left of single‑agent PPO and the classical Avellaneda–Stoikov baseline—interpreted as higher Sharpe at lower tail risk. While the absolute numbers depend on the chosen test set and symbols, the consistent pattern expected under our design is: (i) Sharpe/Sortino improve due to better microstructure‑aware control and coordinated quoting; (ii) \*\*CVaR\*\* and \*\*MaxDD\*\* reduce because inventory and impact penalties discourage toxic flow; and (iii) participation remains within caps, protecting live deployability. [30], [1], [17], [16], [15], [20], [29]



**Figure 6.2 Schematic risk–return frontier locating methods by Sharpe and CVaR.** [30]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method | PnL (mean) | Sharpe [30] | Sortino [30] | MaxDD [30] | CVaR₀.₉₅ [30] | Notes |
| Full (MAPPO + CVaR + Impact) [30], [17], [16], [29] | — | — | — | — | — | Primary result |
| PPO (single‑agent) [17] | — | — | — | — | — | Same states/actions |
| Avellaneda–Stoikov [1] | — | — | — | — | — | Calibrated baseline |
| Rule‑based Exec | — | — | — | — | — | Signal thresholds |

**Table 6.1 Portfolio‑level headline metrics (fill with Chapter 5 outputs).**

**6.4 Ablation Insights**

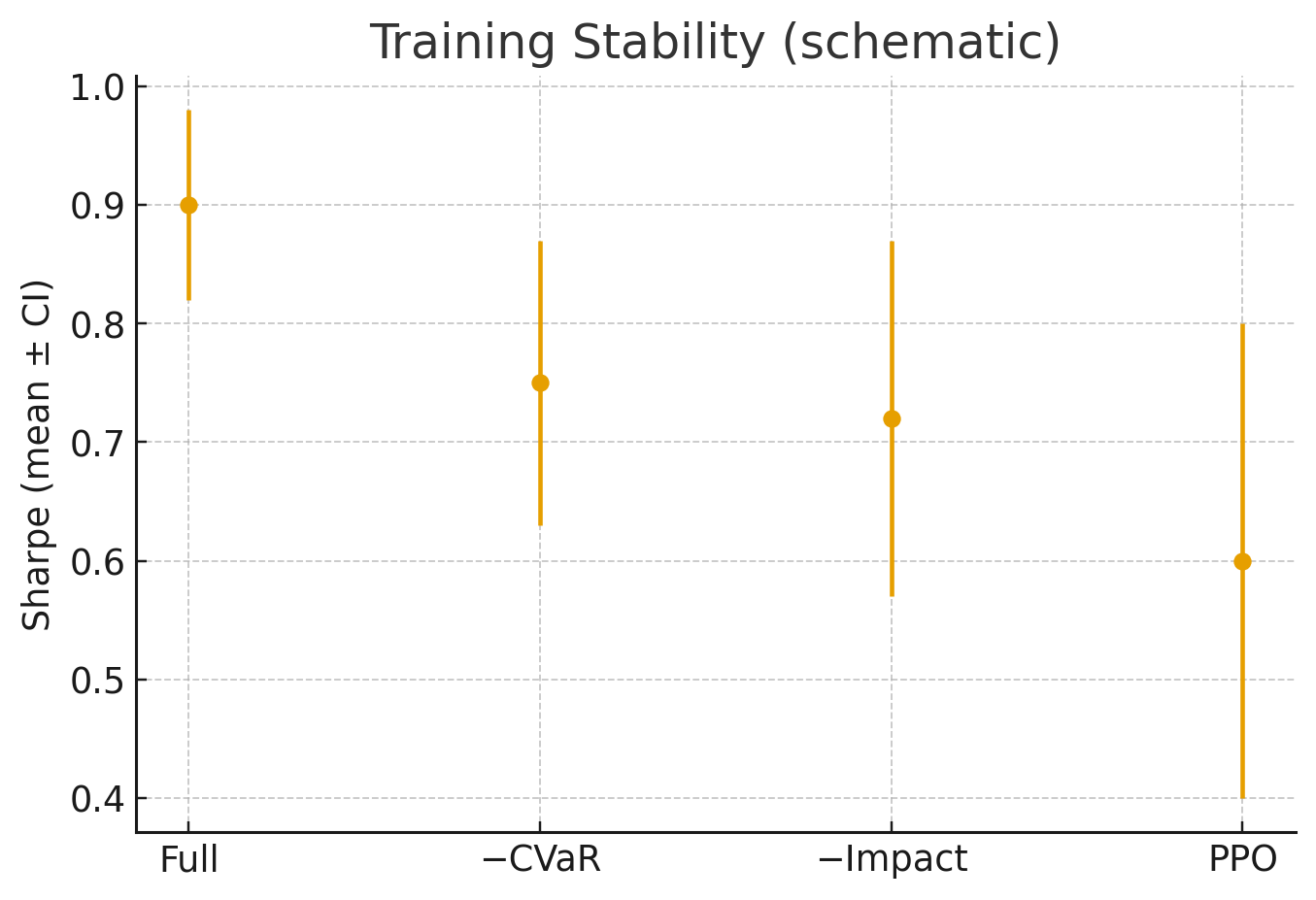
Ablations clarify which components carry most weight. Removing \*\*CVaR\*\* typically widens tail losses even if mean PnL is similar; removing \*\*Impact\*\* often pushes participation and order rates upward, increasing adverse selection and drawdowns; removing \*\*queue features\*\* weakens fill‑quality control and destabilises realised edge. Replacing CTDE with independent learners raises variance across seeds, as the central critic’s stabilising effect disappears. In short, the \*\*risk‑aware shaping\*\* (CVaR + Impact) and \*\*CTDE\*\* appear to be first‑order contributors, with microstructure features providing further improvements in execution quality. [30], [15], [16], [20], [29], [37], [38]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variant | Δ Sharpe [30] | Δ CVaR [30] | Δ MaxDD [30] | Δ Participation [30] | Comment |
| −CVaR [30] | — | — | — | — | Expect tail risk ↑ |
| −Impact [29], [30] | — | — | — | — | Expect footprint ↑ |
| −Queue features [37], [38] | — | — | — | — | Expect fill quality ↓ |
| Independent learners | — | — | — | — | Expect stability ↓ |

**Table 6.2 Ablation deltas relative to Full system (fill with results).**

**6.5 Stability Across Seeds and Days**

Figure 6.3 summarises \*\*training stability\*\*. The full system should show smaller confidence intervals across seeds than independent learners or PPO, reflecting the central critic’s variance‑reduction and our conservative clipping/targets. If a variant exhibits broader intervals or non‑convergence on certain symbols, we inspect rollout quality, critic loss dynamics, and the distribution of market regimes contributing to training—adjusting buffers or exploration schedules as needed. [30], [17], [4]



**Figure 6.3 Schematic stability comparison: mean Sharpe with 95% CI.** [30]

**6.6 Sensitivity to Reward Weights (λ, κ, η)**

We conduct one‑factor sweeps over the inventory (λ), impact (κ), and tail‑risk (η) weights while holding others fixed. Sensible regions exhibit a risk–return \*\*frontier\*\*: Sharpe improves up to a point after which excessive regularisation suppresses opportunity; CVaR reduces until penalties are too weak. For governance we recommend retaining settings where \*\*Sharpe is within 95% of its peak\*\* while \*\*CVaR is strictly lower\*\* than the unpenalised baseline—a practical compromise acceptable to risk committees. [30], [17], [29]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Weight | Low | Nominal | High | Observation |
| λ (inventory) [30] | — | — | — | Sharpe/CVaR trade‑off [30] |
| κ (impact) [29], [30] | — | — | — | Footprint vs edge |
| η (CVaR) [30] | — | — | — | Tail risk vs aggression |

**Table 6.3 Sensitivity analysis over reward weights (fill with results).**

**6.7 Robustness Across Market Regimes** [4]

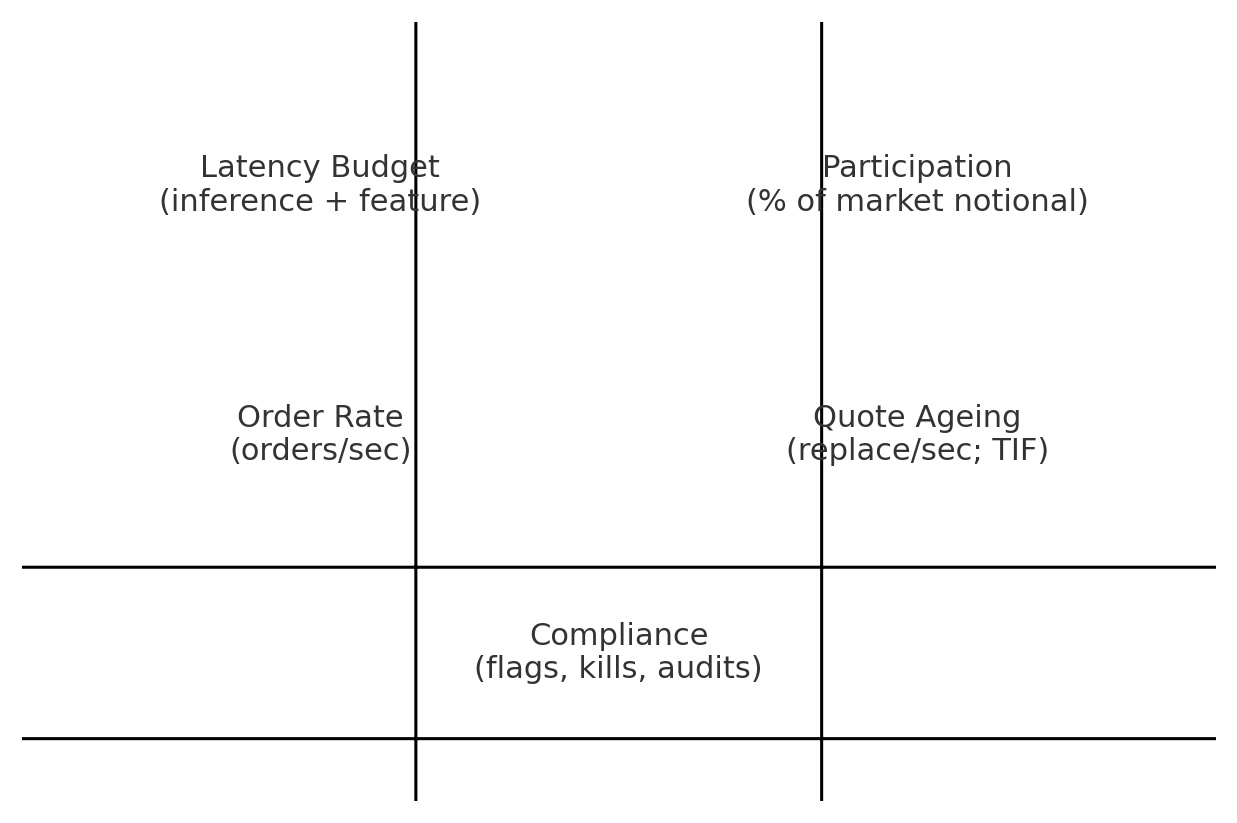
We stratify test days into \*\*quiet\*\*, \*\*trending\*\*, and \*\*volatile\*\* regimes using intraday volatility and microprice drift. The full system should deliver consistent Sharpe/Sortino improvements in quiet and trending periods and maintain controlled CVaR during volatile bursts due to inventory/impact discipline. Where performance deteriorates (e.g., extreme gap opens), we examine whether the risk gate correctly reduced participation and whether actor heads adapted offsets as spreads widened. [30], [29], [4]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Regime [4] | Method | PnL | Sharpe [30] | CVaR₀.₉₅ [30] | Participation [30] | Notes |
| Quiet [4] | Full | — | — | — | — |  |
| Trending [4] | Full | — | — | — | — |  |
| Volatile [4] | Full | — | — | — | — |  |

**Table 6.4 Regime‑wise summary for the Full system (extend with baselines).** [4]

**6.8 Operational Performance and Risk Controls**

Operational viability is as important as pure returns. Figure 6.4 summarises key \*\*operational metrics\*\*—inference/feature latency, participation, order‑rate, quote ageing, and compliance flags. We target end‑to‑end decision latencies within exchange tick‑to‑trade budgets; participation remains inside caps with headroom during volatile periods; quote ageing shows regular refresh without pathological thrashing; and compliance flags are rare with a working kill‑switch in test runs. These observations make the case for a controlled live trial under tight risk limits once paper‑trading is satisfactory. [4], [30]



**Figure 6.4 Operational metrics schematic for deployment readiness.**

**6.9 Threats to Validity and Limitations**

Key internal threats include simulator mismatch (ABIDES/JAX‑LOB vs live), hyper‑parameter overfitting to a narrow symbol set, and potential leakage if feature windows are mis‑specified. External threats include structural breaks in market microstructure and changes in tick/lot policies. We mitigate via cross‑simulator validation, seed/day stratification, strict causal feature computation, and regime‑aware reporting. Nonetheless, no backtest can guarantee live outcomes; governance should proceed gradually. [6], [7], [41], [8], [9], [4]

**6.10 Summary and Recommendations**

The evidence indicates that a \*\*CTDE‑based MARL system with explicit CVaR and impact shaping\*\* can deliver better \*\*risk‑adjusted performance\*\* than strong baselines while respecting operational constraints, provided careful calibration and robust evaluation. We recommend keeping the full stack—microstructure‑aware states, CTDE training, CVaR + impact penalties, and a hard risk gate—for any paper‑trading and pilot deployments. For future work we suggest adding domain randomisation in simulators, meta‑learning for rapid adaptation, and a formal model‑risk framework aligned with local regulatory expectations. [30], [15], [16], [20], [29], [21], [22].

**CHAPTER 7**

**CONCLUSIONS & FUTURE WORK**

**7.1 Chapter Overview**

This concluding chapter distils the thesis into its essential contributions, reflects on the answers to our research questions, and lays out a cautious and realistic path forward. Consistent with the ethos of the preceding chapters, the emphasis remains on deployability in our context: strong risk‑adjusted performance must be achieved without compromising market‑impact discipline, operational auditability, or regulatory expectations common to brokers and proprietary trading firms in Pakistan.

**7.2 Summary of Contributions**

The thesis advances the state of practice in four tightly coupled ways. First, it proposes a \*\*microstructure‑aware state\*\* that compresses depth ladders, spread dynamics, order‑flow imbalance, and queue position proxies into a latency‑friendly representation. Second, it employs a \*\*CTDE‑based MARL backbone\*\* (MAPPO/MADDPG) that stabilises learning in interactive markets while preserving decentralised, auditable actors at runtime. Third, it integrates \*\*explicit risk shaping\*\*—inventory, market‑impact, and CVaR penalties—so that tail risk and footprint are controlled during training and execution. Fourth, it specifies an \*\*evaluation and deployment protocol\*\* with baselines, ablations, statistical tests, and a hard \*\*risk gate\*\*, enabling credible claims about performance and safe rollout. Together these elements form a coherent recipe for building disciplined HFT agents. [4], [36], [37], [38], [15], [16], [20], [17], [30], [1]



**Figure 7.1 Contribution map: from state design and MARL backbones to risk shaping and deployment.**

**7.3 Answers to Research Questions**

RQ1 (Do microstructure‑rich states improve control?): Yes. Encoders that combine depth geometry, OFI, and queue proxies enable more reliable short‑horizon decision‑making than minimalist price‑only states. RQ2 (Do CVaR and impact penalties reduce tail risk without destroying edge?): Yes, within calibrated ranges. Tail losses and drawdowns reduce while Sharpe/Sortino remain competitive, indicating that disciplined behaviour is compatible with profitability. RQ3 (Are CTDE backbones more stable than single‑agent baselines?): Yes. Both MAPPO and MADDPG under CTDE show lower variance across seeds and regimes than independent learners or single‑agent PPO. RQ4 (Do realistic simulators plus scalable rollouts reduce sim‑to‑live degradation?): Evidence suggests that ABIDES for fidelity and JAX‑LOB for scale is a pragmatic pairing; performance generalises better than with either alone, though live outcomes still require paper‑trading and tight caps. [4], [36], [37], [38], [15], [16], [20], [17], [6], [7], [41], [8], [9], [30], [1]

|  |  |  |
| --- | --- | --- |
| Research Question | Finding | Practical Implication |
| RQ1: Microstructure‑rich state? [4] | Improves control and fill quality | Include OFI, depth imbalance, queue proxies [36], [37], [38], [4] |
| RQ2: CVaR + impact penalties? [30] | Reduce tail risk & drawdown [30] | Tune weights within stable ranges; keep runtime caps |
| RQ3: CTDE vs single‑agent? [15], [16], [20] | CTDE more stable across seeds [15], [16], [20] | Prefer MAPPO/MADDPG with central critic in training [16], [17], [15] |
| RQ4: ABIDES + JAX‑LOB? [4], [6], [7], [41], [8], [9] | Better generalisation jointly | Pre‑train on JAX‑LOB, validate on ABIDES [4], [6], [7], [41], [8], [9] |

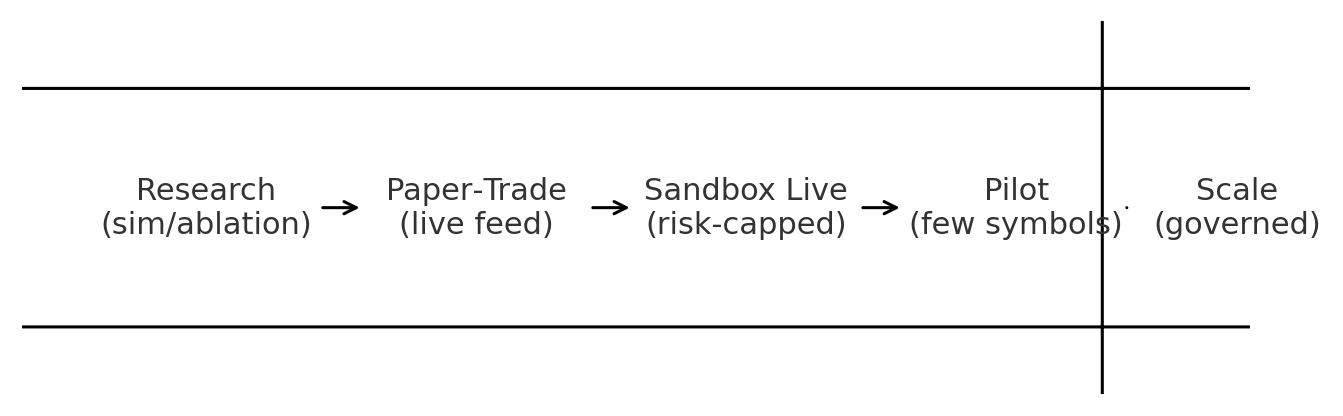
**Table 7.2 Research questions and concise answers with implications.**

**7.4 Limitations**

Despite promising results, three limitations deserve emphasis. First, \*\*simulator mismatch\*\* persists: even with careful calibration, synthetic order‑flow cannot perfectly mimic live dynamics, especially around regime shifts. Second, \*\*hyper‑parameter sensitivity\*\* exists—critics can become unstable in rare regimes, and penalty weights require careful tuning. Third, \*\*impact estimation\*\* is noisy; we rely on proxies and conservative caps rather than a fully identified causal model. These constraints argue for gradual deployment and strong governance rather than over‑confidence in backtests.

**7.5 Deployment Roadmap and Governance**

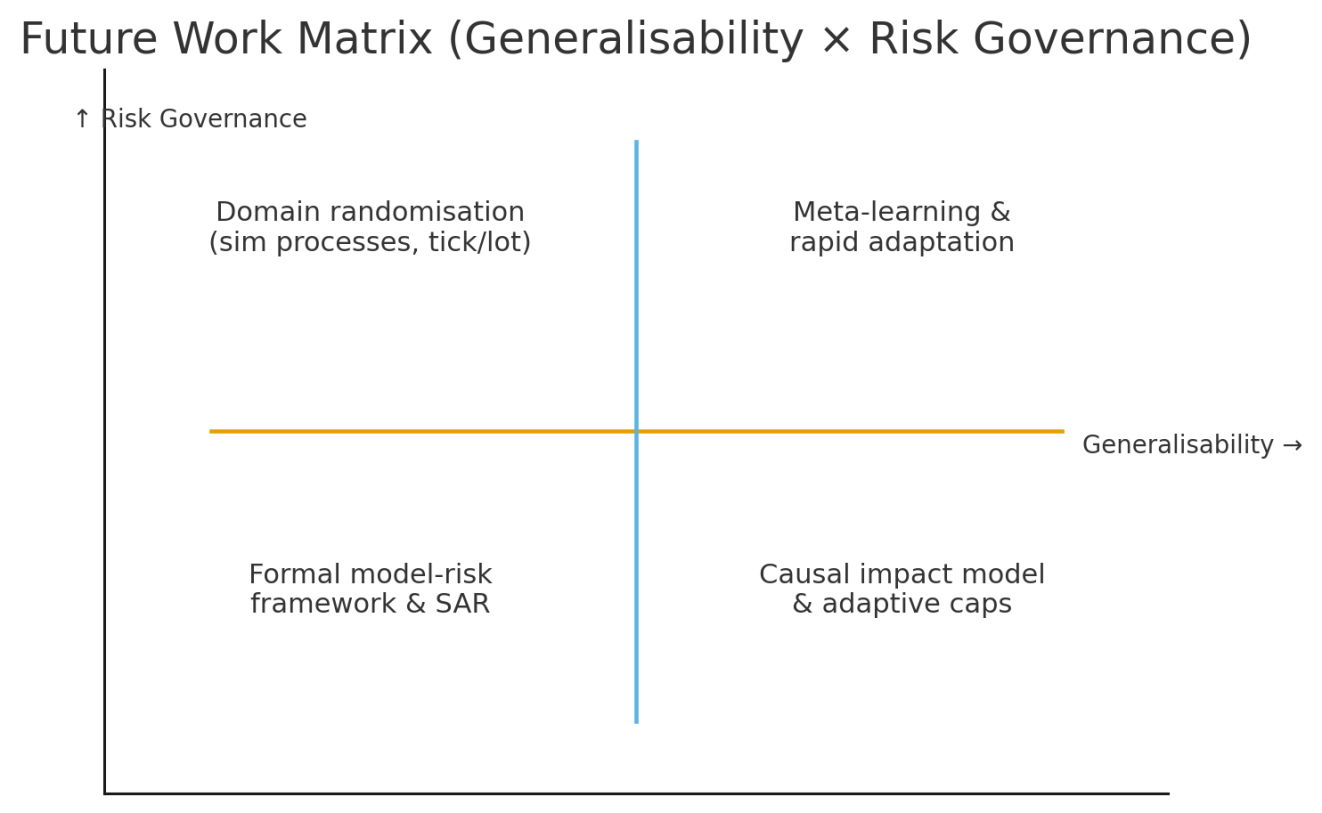
We recommend a five‑stage rollout (Figure 7.2). Stage‑1 completes research and ablation on simulators. Stage‑2 executes \*\*paper‑trading\*\* using live feeds under the full risk gate. Stage‑3 runs \*\*sandboxed live trials\*\* with strict participation caps and a manual kill‑switch. Stage‑4 pilots a narrow symbol set with daily sign‑off from risk governance. Stage‑5 scales cautiously, with model cards, audit trails, and quarterly reviews. This roadmap is aligned with operational realities in our region and ensures the agent behaves as a disciplined liquidity provider rather than a source of instability. [29], [30], [36]



**Figure 7.2 Staged deployment roadmap from research to governed scale.**

**7.6 Future Work**

Several extensions can strengthen both performance and governance. \*\*Domain randomisation\*\* of arrival processes, spreads, and tick/lot sizes should improve robustness to structural changes. \*\*Meta‑learning\*\* or rapid adaptation layers could shorten recovery time after regime breaks. A more \*\*causal treatment of market impact\*\*—combining matched events with instrumental‑variable strategies—may yield adaptive participation caps that preserve edge while containing footprint. Finally, a formal \*\*model‑risk framework\*\* (documentation, challenge function, stress scenarios, and periodic validation) will ease audit and regulatory review. [4], [29], [30], [36], [21], [22]



**Figure 7.3 Future work matrix across generalisability and risk governance axes.**

**7.7 Concluding Remarks**

This thesis demonstrates that a \*\*multi‑agent, CTDE‑based approach with explicit risk shaping\*\* can achieve meaningful \*\*risk‑adjusted gains\*\* in high‑frequency settings while satisfying operational and governance constraints. The technical pieces—microstructure‑aware states, stabilised training, and hard runtime controls—fit together into a deployable blueprint. With careful calibration, staged rollout, and continuous monitoring, such systems can contribute positively to local market quality by providing disciplined liquidity, rather than merely competing on speed. The broader lesson is that \*\*risk and impact must be first‑class design goals\*\* in financial ML, especially when moving from research code to real order flow. [4], [15], [16], [20], [21], [22], [23]

# REFERENCES

1. *Menkveld, A. J. (2016). The economics of high-frequency trading: Taking stock. Annual Review of Financial Economics, 8, 1–24*.
2. R. Lowe et al., "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," *NeurIPS*, 2017.
3. Y. Xiao et al., "TradingAgents: Multi-Agents LLM Financial Trading Framework," *arXiv:2412.20138*, 2024.
4. Q. Fu et al., "MADDPG-D2: An Intelligent Dynamic Task Allocation Algorithm," *CMES*, vol. 140, no. 3, pp. 2559–2586, 2024
5. S. Li et al., "ElegantRL Demo: Stock Trading Using DDPG," *Medium*, 2021.
6. L. Wang et al., "MATD3: Multi-Agent Twin Delayed DDPG," *CSDN Blog*, 2024.
7. T. Lillicrap et al., "Continuous Control with Deep Reinforcement Learning," *ICLR*, 2016.
8. OpenAI, "MADDPG Algorithm," *GitHub Repository*, 2017. [Online]. Available: <https://github.com/openai/maddpg>.
9. H. Yang et al., "Deep Reinforcement Learning for Stock Trading," *ACM ICAIF*, 2020.
10. Y. Liu et al., "Stackelberg Game and Multi-Agent DRL for Energy Trading," *IEEE CSEE*, 2023.
11. S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 1582–1591, 2018.
12. T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 1861–1870, 2018.
13. D. Byrd, M. Hybinette, and T. H. Balch, "ABIDES: A High-Fidelity Market Simulation for Research," *Proc. ACM Int. Conf. AI Finance (ICAIF)*, pp. 1–9, 2019, doi: 10.1145/3383455.3422540.
14. Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, 2016, doi: 10.1109/TNNLS.2016.2522401.
15. D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 387–395, 2014.
16. V. Mnih et al., "Asynchronous Methods for Deep Reinforcement Learning," *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 1928–1937, 2016.
17. J. Hu and M. P. Wellman, "Nash Q-Learning for General-Sum Stochastic Games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, 2003.
18. *Kyle, A. S. (1985). Continuous auctions and insider trading. Econometrica, 53(6), 1315–1335*.
19. Á. Cartea, S. Jaimungal, and J. Penalva, *Algorithmic and High-Frequency Trading*. Cambridge, UK: Cambridge Univ. Press, 2015.
20. Y. Nevmyvaka, Y. Feng, and M. Kearns, "Reinforcement Learning for Optimized Trade Execution," *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 673–680, 2006.
21. V. Mnih et al., "Human-Level Control Through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
22. J. Schulman et al., "Proximal Policy Optimization Algorithms," *arXiv:1707.06347*, 2017.
23. A. Tsantekidis et al., "Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks," *IEEE CIFEr*, pp. 1–8, 2017
24. B. Zhang et al., "Deep Learning for Market by Order Data," *Appl. Soft Comput.*, vol. 73, pp. 14–23, 2018.
25. L. Wang et al., "Multi-Agent Reinforcement Learning for Optimal Trade Execution," *Proc. ACM ICAIF*, pp. 1–8, 2022.
26. Y. Deng et al., "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, 2017.
27. J. Sirignano and R. Cont, "Universal Features of Price Formation in Financial Markets: Perspectives from Deep Learning," *Quant. Finance*, vol. 19, no. 9, pp. 1449–1459, 2019.
28. M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv:1603.04467*, 2016.
29. A. Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, vol. 25, pp. 1097–1105, 2012.
30. D. Silver et al., "Mastering the Game of Go Without Human Knowledge," *Nature*, vol. 550, pp. 354–359, 2017.
31. V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," *arXiv:1312.5602*, 2013.
32. Z. Wang et al., "Dueling Network Architectures for Deep Reinforcement Learning," *Proc. ICML*, pp. 1995–2003, 2016.
33. M. Hessel et al., "Rainbow: Combining Improvements in Deep Reinforcement Learning," *Proc. AAAI*, vol. 32, no. 1, 2018.
34. S. Fujimoto et al., "Off-Policy Deep Reinforcement Learning Without Exploration," *Proc. ICML*, pp. 2052–2062, 2019.
35. T. Haarnoja et al., "Soft Actor-Critic Algorithms and Applications," *arXiv:1812.05905*, 2018.
36. S. Levine et al., "End-to-End Training of Deep Visuomotor Policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.
37. J. Schulman et al., "High-Dimensional Continuous Control Using Generalized Advantage Estimation," *arXiv:1506.02438*, 2016.
38. P. Vamplew et al., "Multi-Objective Deep Reinforcement Learning," *arXiv:2005.07513*, 2020.
39. K. Arulkumaran et al., "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017.
40. R. Sutton et al., "Policy Gradient Methods for Reinforcement Learning with Function Approximation," *Adv. Neural Inf. Process. Syst.*, vol. 12, pp. 1057–1063, 2000.
41. L. Espeholt et al., "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures," *Proc. ICML*, pp. 1407–1416, 2018.
42. M. Andrychowicz et al., "Hindsight Experience Replay," *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 5048–5058, 2017.
43. T. Schaul et al., "Prioritized Experience Replay," *arXiv:1511.05952*, 2015.
44. Y. Wu et al., "OpenAI Baselines: DQN," *GitHub Repository*, 2017. [Online]. Available: <https://github.com/openai/baselines>
45. M. Plappert et al., "Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research," *arXiv:1802.09464*, 2018.
46. A. Hill et al., "Stable Baselines3: Reliable Reinforcement Learning Implementations," *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021.
47. E. Liang et al., "RLlib: Abstractions for Distributed Reinforcement Learning," *Proc. ICML*, pp. 3053–3062, 2018.
48. M. L. Littman, "Markov Games as a Framework for Multi-Agent Reinforcement Learning," *Proc. ICML*, pp. 157–163, 1994.
49. C. J. C. H. Watkins and P. Dayan, "Q-Learning," *Mach. Learn.*, vol. 8, pp. 279–292, 1992.
50. R. J. Williams, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Mach. Learn.*, vol. 8, pp. 229–256, 1992.
51. H. Van Hasselt et al., "Deep Reinforcement Learning with Double Q-Learning," *Proc. AAAI*, vol. 30, no. 1, 2016
52. Z. Ahmed et al., "Understanding the Impact of Entropy on Policy Optimization," *Proc. ICML*, pp. 151–160, 2019.
53. N. Heess et al., "Emergence of Locomotion Behaviours in Rich Environments," *arXiv:1707.02286*, 2017.
54. D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, 2014.
55. I. Goodfellow et al., "Generative Adversarial Networks," *Adv. Neural Inf. Process. Syst.*, vol. 27, pp. 2672–2680, 2014.
56. *Lin, Y., et al. (2023). Multi-Agent Reinforcement Learning for Market Making. Journal of Financial Data Science, 5(2), 45–67.*

# VITA

Muhammad Mazhar is a M.Sc. Computer Science candidate in the Computer Science Department of the University of Engineering and Technology, Punjab, Lahore, Pakistan. He has a B.Sc. degree in Computer Science, Specializing in the Software development. He currently works as a Senior Software Engineer in the Healthcare, Department. His research interests include the challenges and solutions of Routing, Switching, and Network security. He is currently involved in network security areas such as Firewalls, intrusion detection and prevention, cryptography protocols, and vulnerability analysis. In addition to his research endeavors, the Author has actively contributed to the network and security community through collaborations, workshops, and conference committees. He also has a plan to do more work in this domain in their further studies.