An Interactive Simulation Framework by Ensemble Imitation Learning Agents for Training Robust Trading Policies

Julian Zhong-Nan Zhang, Yang Yu

National Key Laboratory for Novel Software Technology, Nanjing University
Xianlin Campus, 163 Xianlin Avenue, Nanjing, Jiangsu 210023 China
zhangzn@lamda.nju.edu.cn, yuy@nju.edu.cn

Abstract

The reliable deployment of reinforcement learning (RL) for real-world algorithmic trading is critically hindered by the "simulation-to-reality gap." Standard industry backtesting on static historical data ignores market impact—the feedback loop where an agent's trades influence price dynamics—leading to strategies that are fragile and untrustworthy in live markets. To solve this significant problem, we present a novel and emerging application of AI: a framework for building an interactive, responsive market simulator. Our system first uses imitation learning (IL) to automatically train an ensemble of agents, each learning a distinct trading strategy from a different historical market regime (e.g., bull, bear). This creates a data-driven proxy for a diverse population of real-world traders. We then deploy an innovative Action Synthesis Network to synthesize the actions of this ensemble, generating a realistic, synthetic price trajectory that endogenously models the market's reaction to trades. This interactive environment is then used to train a final RL policy. We evaluate our system on NASDAQ-100 (QQQ) data, and the results demonstrate strong potential for deployment. The RL policy trained in our responsive simulator achieves significantly more robust performance, exhibiting superior downside protection during market downturns compared to various traditional baselines. This application provides a scalable and technically sound methodology for building more realistic training environments, presenting a clear path toward the development and eventual deployment of more resilient and effective algorithmic trading strategies.

Introduction

The ultimate goal of applying artificial intelligence to finance is to build autonomous agents that can make profitable and reliable trading decisions in real-world markets. However, the field is hindered by a fundamental paradox: the very methods used to validate these agents are often the cause of their failure in live deployment. This disconnect, known as the "simulation-to-reality gap", is the single greatest obstacle to deploying robust RL agents in finance and represents a high-stakes problem that our work directly addresses (Bai et al. 2025).

The source of this paradox is the industry-standard practice of backtesting on static historical data. This method assumes an agent's trades are invisible and have no effect on market prices. An agent trained this way never learns the consequences of its own actions, such as the fact that a large "buy" order will drive up the price, or a large "sell" order will drive it down. This phenomenon, known as *market impact*, is a core principle of market dynamics (Vyetrenko et al. 2020). Consequently, strategies that appear highly profitable in static backtests—where the agent gets to trade at historical prices regardless of its actions—can lead to significant losses when deployed in a live market that reacts to its trades (De Prado 2018; Balch et al. 2019).

If static backtesting is the problem, then interactive simulation is the solution. Agent-based models (ABMs) attempt this by creating virtual markets populated with different trading agents (Schnaudt et al. 2019). However, their practical application has been limited by a critical bottleneck: the agents in these simulators are typically governed by a small set of hand-crafted, rule-based strategies. This manual design process is not scalable, is prone to designer bias, and fails to capture the immense diversity and adaptive nature of real-world market participants (Vyetrenko et al. 2020).

Our key innovation is a novel framework that automates the creation of a diverse and realistic market environment, breaking the bottleneck of manual agent design. Instead of prescribing agent behaviors, we teach them directly from market data. Our system first uses imitation learning (IL) to train an entire ensemble of "specialist" agents, with each one learning a unique trading style from a different historical market period (e.g., a bull market, a bear market). We then introduce a novel *action synthesis network* that learns to mimic this diverse population, generating responsive price dynamics that realistically model the market's reaction to trades. By training a final RL agent within this rich, interactive simulation, we learn a policy that is inherently more robust.

Our contributions are therefore clear and directly address the stated problem:

- **A data-driven method for automated agent population:** We present a complete system that uses imitation learning to automatically create a diverse ensemble of trading agents from historical data. This replaces the slow, biased, and unscalable process of manual, rule-based agent design.
- **The action synthesis network:** We introduce a novel

neural network component that models market impact by learning to dynamically synthesize the collective actions of the agent ensemble into a single, coherent price movement, creating a responsive and realistic training environment.

- **Demonstrated robustness and superior risk management:** We show that an RL agent trained in our simulator learns a verifiably more robust strategy. Compared to baselines, it achieves significantly better downside protection in market downturns and lower overall volatility— eassential qualities for real-world deployment.

Related Work

Our research is positioned at the intersection of reinforcement learning in finance, agent-based market simulation, and imitation learning. We build upon foundational concepts from each domain while addressing their core limitations.

Reinforcement Learning in Finance

The application of RL to finance has grown substantially, with algorithms like Q-Learning and, more recently, Deep RL being used for tasks ranging from option pricing to portfolio management (Fischer 2018; Hambly, Xu, and Yang 2021). Frameworks such as FinRL have been developed to standardize the research process, providing pre-built modules for data access, feature engineering, and training RL agents (Liu et al. 2021).

However, a persistent criticism across this body of work is the reliance on static historical backtesting. In a typical setup, the RL agent's environment is a fixed dataset of past prices. When the agent decides to "buy", the environment simply provides the historical price at that time, completely ignoring the fact that a large buy order would have increased that price in reality, and vice versa for "sell" decisions. This failure to model market impact leads to over-optimistic performance and policies that are not robust to live market feedback (Balch et al. 2019). Our work directly addresses this fundamental limitation by creating an environment where the price is not static but reacts to the agent's actions.

Agent-Based Market Simulation

To overcome the flaws of static backtesting, researchers have developed agent-based models (ABMs), which simulate markets as complex adaptive systems of interacting agents (LeBaron 2006). These interactive agent-based simulators (IABS) enable "what-if" analyses by modeling the market's response to an agent's actions (Vyetrenko et al. 2020). A common approach is to populate the simulator with agents governed by simple, hand-crafted rules. For example, an ABM might include trend-followers that buy when a short-term moving average crosses above a long-term one, and noise traders that act randomly (Spooner et al. 2018).

While a conceptual improvement, the practical limitation of ABMs is their reliance on these manually designed, simplistic agent archetypes. This process requires significant domain expertise, is prone to designer bias, and struggles to capture the true heterogeneity and adaptive complexity of real-world traders. Our key innovation is to replace this manual, rule-based design with a data-driven approach. Instead of hand-crafting agent behaviors, we learn them directly from market data, resulting in a more diverse and realistic agent population.

Imitation Learning in Finance

Imitation learning (IL) offers a powerful, data-driven alternative to rule-based agent design, aiming to learn a policy by mimicking expert demonstrations (Osa et al. 2018). In finance, the standard application of IL is to train a *single trading agent*. For instance, a researcher might use the historical trades of a successful portfolio manager as an "expert" dataset, and then train an IL agent to replicate that manager's decisions (Liu et al. 2020). The goal is to distill the expert's strategy into a deployable model.

Our work employs imitation learning in a fundamentally different and novel way. We are not using IL to create our final, primary trading agent. Instead, we use IL to construct the *market environment itself*. By training an entire ensemble of IL agents—each mimicking a different "expert" style from a specific market regime—we are creating a diverse and data-driven population of background agents. This novel application of IL allows us to build a realistic simulator that serves as the training ground for our final RL agent, a contribution that distinguishes our work from standard applications of IL in finance.

System Architecture and Technical Methodology

Our system is a multi-stage pipeline designed to first construct a realistic, interactive market environment and then train a robust trading agent within it. The architecture is composed of three core components that build upon one another: an ensemble of data-driven background agents, a network to synthesize their actions into a responsive market price, and a final reinforcement learning agent that learns to trade within this simulated ecosystem. The entire end-to-end process is summarized in Algorithm 1.

The IL Agent Ensemble

The foundation of our simulator is a diverse population of background agents whose collective behavior mimics the real market. To avoid the biases of manual design, we create this population automatically by training an ensemble of Imitation Learning (IL) agents, each learning a distinct "style" from historical data.

Data Segmentation and Feature Engineering. The process begins with high-quality historical data: minute-level Open, High, Low, Close (OHLC) and volume data for the NASDAQ-100 ETF (QQQ) from 2019 to 2024. A crucial step is to segment this data into distinct market regimes, allowing us to train specialist agents for different market "personalities." We manually identify the dates of significant local price maxima and minima (see Appendix Table 6). The periods between these extrema define our training segments, capturing distinct bull runs, bear markets, and periods of consolidation.

For the state representation, we engineer a suite of technical indicators known to capture different facets of market dynamics. These features, which form the input for all our agents, include:

- **Exponential Moving Averages (EMAs):** With periods of 3, 6, 9, 18, and 27 minutes to capture trends at multiple time scales. The EMA is a recursive calculation:

$$EMA_t = \alpha \cdot x_t + (1 - \alpha) \cdot EMA_{t-1} \quad (1)$$

  where $\alpha = 2/(N + 1)$ is the smoothing factor.
- **Moving Average Convergence Divergence (MACD):** A core momentum indicator defined as $MACD = EMA_{12} - EMA_{26}$, along with its signal line, $Signal = EMA_9(MACD)$.
- **Volume Indicators:** Volume-Weighted Average Price (VWAP) and Relative Volume, calculated as the ratio of current volume to its moving average, $Rel.Vol. = V_{\text{current}}/\overline{V}$.

These features are concatenated to form the state vector s_t and normalized using Min-Max scaling, $x' = (x - \min x)/(\max x - \min x)$, to ensure consistent input ranges for the neural networks.

Expert Action Labeling and IL Training. To give each agent a unique style, we generate "expert" action labels for each historical segment. The labels are derived from a simple, forward-looking heuristic: for each minute t, we assign an action $a_t \in \{-1, 0, 1\}$ (sell, hold, buy) if the price subsequently decreases or increases by a significant threshold (e.g., 10%) within that segment. This provides a clear, albeit simplified, "perfect foresight" expert for each market condition.

We then train a separate IL agent for each market segment. Each agent is a Multi-Layer Perceptron (MLP) with two hidden layers of 128 units (using ReLU activation) that learns a policy $\pi_\theta(s_t)$. The training objective is to mimic the expert by minimizing the Mean Squared Error (MSE) between the policy's output and the expert action labels over the dataset of state-action pairs $D = \{(s_i, a_i)\}_{i=1}^N$:

$$\min_\theta \mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \|\pi_\theta(s_i) - a_i\|^2 \quad (2)$$

This automated process yields an ensemble of IL agents, $\Pi = \{\pi_{\theta_1}, ..., \pi_{\theta_M}\}$, where each agent π_{θ_m} is a specialist for a particular historical market condition.

The Interactive Market Simulator

With our diverse ensemble of IL agents created, the next challenge is to make them interact in a way that generates a realistic, synthetic price. This is the role of our Action Synthesis MLP, which acts as the core of this market simulation engine.

Action Synthesis MLP. At any given state s_t, each of the M IL agents in our ensemble proposes an action. To model their collective impact on the price, we deploy an Action Synthesis MLP. This network takes the vector of proposed actions from all IL agents as its input. Its architecture is designed to first weigh the influence of each agent and then predict a final price change. It consists of an input layer matching the number of agents, a 10-unit softmax layer to produce a probability distribution of influence weights, two hidden layers (128 and 64 units, ReLU), and a single output unit for the predicted price change.

The network is trained on a hold-out segment of historical data to minimize the MSE between its simulated price trajectory and the actual historical prices. This training process teaches the network how to orchestrate the agent population, learning the latent composition of trader types whose collective behavior best explains real market movements. For simplicity, this initial version of the simulator does not include a limit order book or explicit market makers.

The Final RL Trading Agent

The IL ensemble and the Action Synthesis MLP together form our high-fidelity, interactive simulation environment. The final step is to deploy a primary RL agent to learn a robust trading strategy within this dynamic world.

Markov Decision Process (MDP) Formulation. We formulate the trading task as an MDP, where the agent learns a policy to maximize the expected discounted cumulative reward, $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$. The components are defined as follows:

- **State Space (S):** The state s_t must provide a complete picture of the market and the agent's own status. It includes all the technical indicators from the feature engineering step, plus the agent's current cash balance (b_t) and number of shares held (k_t). The full state vector is:

$$s_t = [\Delta_{\min}\%, \text{EMA}_x, \text{MACD}, \text{Rel Vol}, VWAP, b_t, k_t] \quad (3)$$

- **Action Space (A):** The agent takes a continuous action $A_t \in [-1, 1]$, representing the fraction of its assets to trade (sell for negative values, buy for positive).
- **Reward Function (R):** The design of the reward function is critical. A naive reward based on the change in total portfolio value from the start of an episode often leads to passive, do-nothing strategies. To overcome this and encourage more active risk management, we designed a reward based on the change in portfolio value over a recent rolling window (230 minutes). This incentivizes the agent to react to shorter-term dynamics and capture value from volatility. The reward is:

$$R_t = (b_t + k_t \cdot p_t) - (b_{t-230} + k_{t-230} \cdot p_{t-230}) \quad (4)$$

- **Transition Dynamics:** The environment's transition is governed by our simulator. The RL agent's action is added to the pool of actions from the IL agents. The simulation engine matches total buy and sell orders ($deal = \min(\sum buy, \sum sell)$), applies a transaction fee (0.5%), and the Action Synthesis MLP computes the next price p_{t+1}, which endogenously incorporates the market impact of all participating agents.

Algorithm 1: End-to-End Training and Evaluation Pipeline

Input: Historical minute-level data D_{hist}

1: Segment D_{hist} into M training regimes $D_1, ..., D_M$ and a hold-out set D_{sim_train}.
2: **for** each segment D_m in $D_1, ..., D_M$ **do**
3: Generate expert state-action pairs (s_i, a_i).
4: Train IL agent π_{θ_m} by minimizing MSE.
5: **end for**
6: Assemble IL agent ensemble $\Pi = \{\pi_{\theta_1}, ..., \pi_{\theta_M}\}$.
7: Train Action Synthesis MLP on D_{sim_train} using Π to minimize simulated vs. real price error.
8: Initialize interactive simulator E with Π and the trained Action Synthesis MLP.
9: Initialize DDPG agent (actor μ, critic Q).
10: Train DDPG agent in environment E to learn final policy π_{RL}.
11: Evaluate π_{RL} on 12 unseen backtest data segments.

RL Algorithm: Deep Deterministic Policy Gradient (DDPG). Given the continuous action space, we use the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al. 2015), an off-policy, actor-critic method well-suited for this domain. It maintains an actor network $\mu(s|\theta^\mu)$ that learns the policy and a critic network $Q(s, a|\theta^Q)$ that estimates the value of actions. These are trained alongside their respective target networks, μ' and Q'. The networks are updated as follows:

1. The critic is updated by minimizing the loss between its prediction and a target value y_t:

$$L = \frac{1}{N} \sum_t \left(y_t - Q(s_t, a_t|\theta^Q)\right)^2 \tag{5}$$

where the target y_t is computed using the target networks to stabilize training:

$$y_t = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'}) \tag{6}$$

2. The actor is updated using the sampled policy gradient, which pushes it to take actions that the critic deems more valuable:

$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_a Q(s, a|\theta^Q)\big|_{a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)\right] \tag{7}$$

The target networks are updated via soft updates with a small rate τ to ensure stability.

Experimental Evaluation and Analysis

We designed our experiments to validate two core hypotheses: 1) that our data-driven method creates a diverse ensemble of background agents, which is the foundation for a realistic simulator, and 2) that an RL policy trained within this interactive environment learns a demonstrably more robust strategy than one trained on static data.

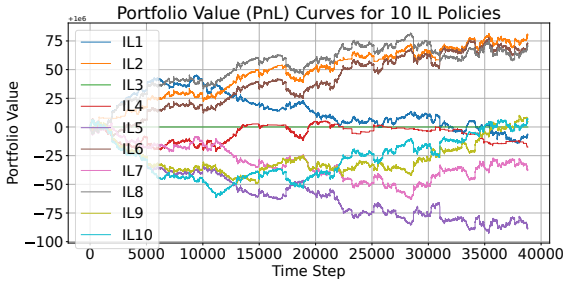

Figure 1: PnL of the 10 IL base policies on an unseen data segment, showing a variety of outcomes.

Training Schemes and Experimental Setup

The IL agents are MLPs with two hidden layers of 128 units (ReLU). The RL policy's actor network has two 128-unit layers, and the critic has three 128-unit layers (all ReLU). We used the Adam optimizer for all networks. Key DDPG hyperparameters are detailed in the Appendix (Table 5). The system was trained on a server with two GPUs totaling 27 TFLOPS. We benchmarked our final RL policy against three baselines: (i) a **long-only (buy-and-hold) strategy**, a standard and strong benchmark in trending markets; (ii) an **RL baseline**; and (iii) an **IL baseline**—both trained on the same data used to train the IL ensemble.

Validation of IL Agent Ensemble Diversity

The premise of our interactive simulator is that a diverse population of agents can better approximate real market dynamics than a monolithic or hand-crafted model. To validate that our method achieves this, we evaluated the 10 trained IL agents on a new, unseen data segment. The resulting profit-and-loss curves (Figure 1) and action sequences (Figure 2) show a wide spectrum of learned behaviors.

The detailed performance metrics, provided in the Appendix (Tables 3 and 4), quantify this diversity. For instance, Agent 3 learned a completely passive strategy (0% turnover), effectively becoming a "do-nothing" agent. In stark contrast, Agent 4 learned a hyperactive strategy with high turnover, while Agent 2 achieved the best risk-adjusted return. This learned diversity—from passive to active, from

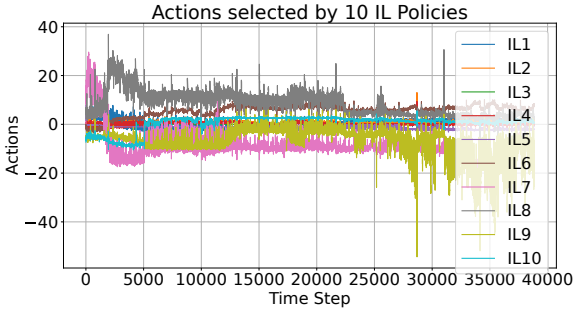

Figure 2: Actions taken by the 10 IL base policies, illustrating different trading frequencies and directional biases.

profitable to loss-making—is crucial. It provides the rich and varied behavioral inputs needed to simulate a complex market ecosystem without the bias of manual rule creation. This data-driven heterogeneity is the cornerstone of our simulator's realism.

Final RL Policy Performance and Strategy Analysis

The final RL policy was evaluated on 12 out-of-sample periods. The aggregated results, shown in Table 1, immediately highlight the core value of our approach.

Metric	Final RL	Long-Only	RL Baseline	IL Baseline
Total Return	**1.98%**	0.96%	0.97%	-0.64%
Volatility	**0.00042**	0.00068	0.00068	**0.00018**
Sharpe Ratio	**0.00077**	-0.00063	-0.00063	**0.00438**
Max Drawdown	**-4.97%**	-9.11%	-9.11%	**-2.98%**

Table 1: Aggregated Backtest Performance (Avg. over 12 tests).

The RL policy not only achieves a substantially higher average return than all the other baselines, but it does so while incurring significantly less risk. The most critical metric for any real-world financial application is maximum drawdown. Our agent cuts the average maximum drawdown by nearly half—from -9.11% in the long-only and RL baselines to -4.97%—which is not a marginal improvement. While the IL baseline attains the lowest drawdown overall, it does so at the expense of profitability. In contrast, our agent avoids this trade-off, preserving capital without sacrificing meaningful returns. This is a fundamental shift in the strategy's risk profile, indicating that the agent has learned to prioritize downside protection while still pursuing favorable opportunities for gain.

To understand *how* the agent achieves this superior risk management, we can examine its behavior in specific market scenarios.

Case Study 1: Capital Preservation in a Bear Market. Figure 5 illustrates a textbook example of the agent's value during a severe bear market. As the market (red line) enters a steep, prolonged decline, the buy-and-hold strategy and the RL baseline largely overlap and closely mirror the downturn, culminating in a catastrophic -23.2% return. In stark contrast, the RL agent's portfolio value (blue line) quickly decouples from the market trend. It correctly identifies the persistent downward pressure and begins to systematically reduce its position, selling assets to move to cash. While it still incurs a loss of -11.6%, it successfully mitigates more than half of the damage sustained by the buy-and-hold and RL baselines, and remains largely comparable to the IL baseline throughout, which ends with a -7.5% loss. This is precisely the behavior expected of a robust agent: it has learned to recognize and respond to tail risk — a lesson that cannot be learned in a static environment where selling carries no consequence.

Case Study 2: Navigating Volatility. Figure 6 illustrates the agent's performance in a different but equally challenging environment: a choppy, volatile market with no clear directional trend. In these conditions, the buy-and-hold strategy and the RL baseline are whipsawed, ultimately ending with a -5.7% loss. The RL agent, however, adopts a more nuanced approach—engaging in smaller, more frequent trades to manage exposure through short-term swings while avoiding large, risky commitments. Its portfolio value remains considerably more stable than the baseline, ending the period with a significantly smaller loss of -2.8%—though still not on par with the most conservative strategy, the IL baseline, which posts a modest gain of 0.62%, but essentially adopts a do-nothing stance even during bull markets. This shows the agent has not learned a simple "on/off" switch for risk, but a more sophisticated policy for navigating uncertainty.

Regime-Dependent Strategy and Risk-Return Profile. These individual examples are consistent with the aggregated regime analysis shown in Table 2. The results provide quantitative evidence of the agent's intelligent, asymmetric risk strategy.

In **Bear Markets**, its value is undeniable. It preserves capital far more effectively, reducing the loss by more than half compared to the long-only and RL baselines. In **Volatile/Sideways Markets**, it again demonstrates its superiority, managing to extract a positive return while the passive, the aggressive, and the conservative baselines all flounder. Crucially, in **Bull Markets**, although the agent earns less than both the long-only and RL baselines, it still captures a substantial portion of the gains and far outperforms the IL baseline, the most conservative strategy. This is an expected and vital finding. Our agent has learned a conservative policy that prioritizes locking in profits and managing risk, which naturally tempers gains during powerful uptrends. This willingness to sacrifice some potential upside for significant downside protection is the hallmark of a mature and deployable trading strategy, not a naively overfitted one.

Finally, the risk-return profile across all 12 backtests, shown in Figure 3, provides a comprehensive visualization of this learned strategy. The agent has not just learned to trade; it has learned a specific *posture* towards risk. The interactive simulation, by punishing overly aggressive actions with realistic market impact, has successfully instilled a conservative bias that prioritizes capital preservation. This visual evidence provides the strongest support for our central claim: the proposed training framework produces agents that are not merely profitable, but are fundamentally more robust and better suited for the demands of real-world deployment.

Discussion

Significance and Path to Deployment

The primary significance of this work is in presenting a technically sound and scalable solution to the simulation-to-reality gap, a major obstacle for AI in finance. The results show that our framework produces agents with demonstrably more robust, risk-managed behaviors. The path from this emerging application to a fully deployed system involves several concrete steps:

Market Regime	Strategy	Avg. Total Return	Avg. Max Drawdown	Avg. Sharpe Ratio
Bull Markets	**RL Policy**	**12.24%**	**-3.60%**	**0.01946**
	Long Baseline	**18.64%**	-4.55%	0.02000
	RL Baseline	**18.64%**	-4.55%	**0.02001**
	IL Baseline	0.91%	**-2.02%**	0.00792
Bear Markets	**RL Policy**	**-6.63%**	**-7.77%**	**-0.01361**
	Long Baseline	-14.19%	-15.59%	-0.01724
	RL Baseline	-14.19%	-15.59%	-0.01725
	IL Baseline	**-1.25%**	**-3.67%**	**0.00499**
Sideways / Volatile	**RL Policy**	**0.32%**	**-3.53%**	**-0.00353**
	Long Baseline	-1.56%	-7.19%	-0.00465
	RL Baseline	-1.55%	-7.19%	-0.00466
	IL Baseline	-1.57%	**-3.24%**	**0.00023**

Table 2: Regime-Dependent Performance Analysis. This table shows the average performance of the RL policy versus the 3 Baselines, categorized by market condition. It highlights the RL policy's strategic trade-off: sacrificing some upside for massive downside protection, while preserving most of the gains at the cost of manageable risks.

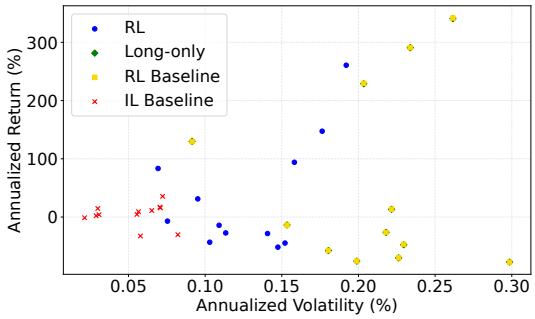

Figure 3: Risk-Return scatter plot for all 12 backtests. The RL Policy's results (blue dots) are systematically shifted to the left (lower risk) compared to that of the long-only (green diamonds) and RL baselines (yellow squares), visually confirming the agent's robust, risk-averse strategy. Even though the IL baseline's results (red $\times$) appear farthest to the left, their tight clustering around the 0% axis highlights the IL baseline's overly conservative nature, which yields negligible profits and falls far short of the performance achieved by the RL policy.

1. **Enhancing Simulator Realism with LOB:** The immediate next step is to incorporate a limit order book (LOB) model into the simulator. This would involve training the IL agents to output limit orders (price and quantity) instead of simple market orders, and building an order matching engine. This would capture a more complex and realistic set of market dynamics, including liquidity effects.
2. **Expansion to Multi-Asset Portfolios:** The framework will be extended to handle a universe of correlated assets. This introduces significant challenges, such as modeling cross-asset impact and learning portfolio construction rules that account for covariance.
3. **Paper Trading and Live Deployment:** Once enhanced, the system would be deployed in a paper trading environment connected to a brokerage API for several months to validate its performance without financial risk. This stage is crucial for identifying any discrepancies between the simulation and live market conditions.
4. **Robustness and Safety Mechanisms:** A production system requires a suite of governance tools, including kill switches, hard position limits, and real-time monitoring for model drift and anomalous behavior to ensure safe and reliable operation.

Limitations and Future Work

Our current system, while a significant advance, has limitations. The market simulation is simplified, lacking a LOB and modeling only a single asset. The IL agents' "expert" labels are based on a simple heuristic; more sophisticated methods like inverse RL could be explored to infer expert intentions more accurately. Furthermore, financial markets are non-stationary, and while our regime-based training offers some adaptation, more explicit mechanisms for handling concept drift could be incorporated. Future work will focus on the deployment path outlined above, as well as exploring the use of this simulation framework as a standardized public benchmark for the research community.

Conclusion

In this paper, we presented a novel emerging application of AI to address a critical bottleneck in deploying RL for finance: the lack of interactive, responsive market simulators. Our deployed system uses an ensemble of IL agents to create a data-driven, diverse population of traders, and an Action Synthesis Network to synthesize their behavior into a market-impact-aware simulation.

The value of this application is demonstrated by the results: an RL agent trained within this environment learns a significantly more robust and practical trading strategy. Compared to the various strong baselines, our agent exhibits superior downside protection, lower volatility, and achieves the best risk-adjusted returns. This work provides a clear technical blueprint for bridging the simulation-to-reality gap, representing a tangible step toward the reliable and trustworthy deployment of AI in finance.

A Action Synthesis MLP Results

The Action Synthesis MLP learns to dynamically combine the actions of the base IL agents. Figure 4 shows the learned influence weights over a simulation period, illustrating how the influence of different agent types is adjusted based on market conditions.

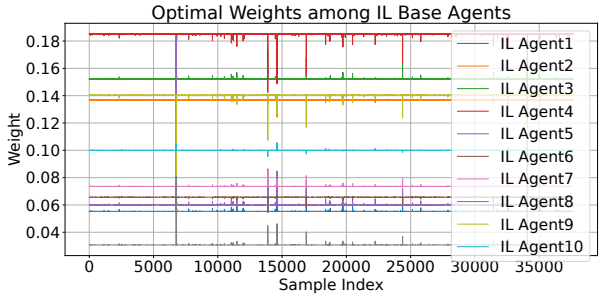

Figure 4: Learned influence weights computed by the Action Synthesis MLP for the simulated price trajectory, showing dynamic allocation of influence among the 10 IL agents.

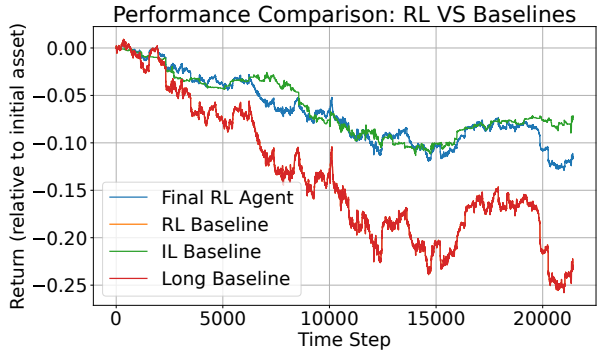

Figure 5: Backtest 1 (Bear Market): The RL Policy (blue) actively sells to preserve capital, decoupling from the severe market downturn that devastates the passive Baseline (red) and the aggressive Baseline (orange).

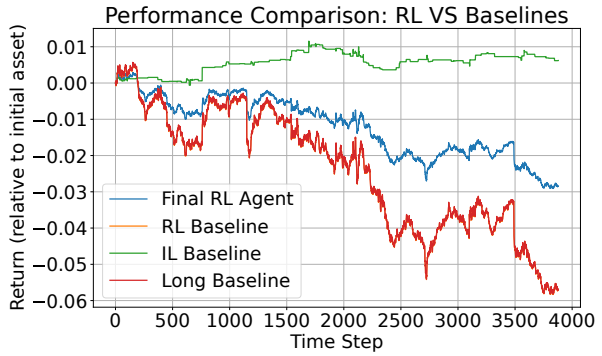

Figure 6: Backtest 11 (Volatile Market): The RL Policy (blue) demonstrates superior stability and risk control, weathering the choppy market far better than the volatile Baselines (red/orange).

Policy	Final Value	Total Return	Turnover
IL1	999993.597	-0.001%	0.0713789
IL2	1000080.658	0.008%	0.0442802
IL3	1000000.000	0.000%	0.0000000
IL4	999982.497	-0.002%	0.0867314
IL5	999911.307	-0.009%	0.0253213
IL6	1000073.004	0.007%	0.00321991
IL7	999962.149	-0.004%	0.0132145
IL8	1000070.972	0.007%	0.00306535
IL9	1000004.227	0.000%	0.0241879
IL10	1000007.305	0.001%	0.0134978

Table 3: General Performance Metrics for IL Policies

Policy	Max drawdown	Volatility	Sharpe Ratio
IL1	0.005968%	0.000271%	-0.015265
IL2	0.001574%	0.000283%	0.184492
IL3	0.000000%	0.000000%	0.000000
IL4	0.002843%	0.000205%	-0.055268
IL5	0.009250%	0.000297%	-0.193394
IL6	0.001738%	0.000307%	0.153834
IL7	0.006680%	0.000301%	-0.081259
IL8	0.002509%	0.000308%	0.149323
IL9	0.005563%	0.000307%	0.008916
IL10	0.006799%	0.000299%	0.015789

Table 4: Risk and Return Metrics for IL Policies

Model-DDPG	Optimizer	Batch size	Memory size
Final agent	Adam	128	9×10^6
γ	expl noise	τ	–
0.99	0.1	0.005	–

Table 5: Hyperparameters for the DDPG RL Algorithm

B IL Agent Ensemble Performance

C Hyperparameters and Data Segmentation Dates

Year	Date / Local Max/Min							
2018	12-23 Min							
2019	05-01 Max	06-03 Min	07-26 Max	08-05 Min	09-12 Max	10-03 Min		
2020	02-19 Max	03-23 Min	09-02 Max	09-21 Min	10-13 Max	11-02 Min		
2021	02-16 Max	03-05 Min	04-29 Max	05-12 Min	09-07 Max	10-04 Min	11-22 Max	12-28 Max
2022	03-14 Min	03-29 Max	06-16 Min	08-16 Max	10-13 Min	12-28 Min		
2023	02-02 Max	03-13 Min	07-19 Max	08-18 Min	09-01 Max	10-12 Max	10-26 Min	
2024	03-21 Max	04-19 Min	07-10 Max					

Table 6: Critical Dates of Local Max/Min Prices Used for Data Segmentation

Acknowledgments

I would like to express my sincere gratitude to Prof. Yu for his invaluable inspiration and guidance. I am also grateful to the LAMDA group for generously providing GPU servers that enabled the AI training in this work. Finally, I wish to thank Dr. Fan-Ming Luo and Dr. Tian Qin for their insightful assistance and support throughout this research.

References

Bai, Y.; Gao, Y.; Wan, R.; Zhang, S.; and Song, R. 2025. A Review of Reinforcement Learning in Financial Applications. *Annual Review of Statistics and Its Application*, 12.

Balch, T. H.; Mahfouz, M.; Lockhart, J.; Hybinette, M.; and Byrd, D. 2019. How to evaluate trading strategies: Single agent market replay or multiple agent interactive simulation? *arXiv preprint arXiv:1906.12010*.

De Prado, M. L. 2018. *Advances in financial machine learning*. John Wiley & Sons.

Fischer, T. G. 2018. Reinforcement learning in financial markets-a survey. *FAU Discussion Papers in Economics*, (11/2018).

Hambly, B.; Xu, R.; and Yang, H. 2021. Reinforcement learning for finance. *arXiv preprint arXiv:2112.13342*.

LeBaron, B. 2006. Agent-based computational finance. *Handbook of computational economics*, 2: 1187–1233.

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

Liu, X.-Y.; Yang, H.; Gao, J.; and Wang, C. D. 2021. FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance. In *Proceedings of the second ACM international conference on AI in finance*, 1–9.

Liu, Y.; Liu, Q.; Zhao, H.; Pan, Z.; and Liu, C. 2020. Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 2128–2135.

Osa, T.; Pajarinen, J.; Neumann, G.; Bagnell, J. A.; Abbeel, P.; and Peters, J. 2018. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2): 1–179.

Schnaudt, C.; Goddar, S.; Voelz, D.; Rost, D.; and Scheuermann, B. 2019. Deep reinforcement learning in agent based financial market simulation. *Available at SSRN 3470408*.

Spooner, T.; Fearnley, J.; Savani, R.; and Koukorinis, A. 2018. Market making via reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 434–442.

Vyetrenko, S.; Byrd, D.; Petosa, N.; Mahfouz, M.; Dervovic, D.; Veloso, M.; and Balch, T. 2020. Get real: Realism metrics for robust limit order book market simulations. In *Proceedings of the First ACM International Conference on AI in Finance*, 1–8.