An Adaptive Deep Reinforcement Learning Framework for Intelligent Transaction Execution in Internet Finance

Yuxuan Qin[1], Jitong Zou[2,*], Zixuan Guo[3] and Bingjie Zi[1]

[1]Northeastern University, Boston, USA

[2]Case Western Reserve University, Cleveland, USA

[3]New York University, New York, USA

[*]Corresponding author: Jitong Zou. jxz1817@case.edu

Abstract

This study treats large-order execution as a short-horizon control problem in which the state can change while the order is still being worked. The proposed Adaptive Regime-Gated Double Dueling Deep Q-Network (Adaptive-RG-DDQN) combines a context-dependent mixture of dueling Q experts with two explicit risk mechanisms: an inventory penalty that increases with observed volatility and spread, and a deterministic shield that can revise a neural action when flow, liquidity, or time-to-deadline makes that action unacceptable. The numerical evidence is intentionally limited to simulation. We use a fully specified four-regime Markov market generator with square-root temporary impact, evaluate three independently trained seeds, and test all policies on 900 matched out-of-sample paths. Adaptive-RG-DDQN produces a mean implementation shortfall of 2.87 bps, compared with 3.83 bps for TWAP and 5.83 bps for static Double Dueling DQN. On the same market paths, the mean reductions are 0.95 bps versus TWAP (p=2.0e-6) and 2.96 bps versus static DDQN (p=2.4e-17). Its difference from an ex-post volume oracle is not significant. The ablation is more informative than the headline comparison: removing the shield eliminates most of the improvement, whereas the present experiment does not establish a separate gain from regime gating. The contribution is therefore a reproducible way to combine learned execution with inspectable risk controls, not a claim of live trading profitability.

Keywords

Algorithmic execution, internet finance, deep reinforcement learning, Double DQN, mixture of experts, market impact, risk control.

1. Introduction

Execution and return prediction solve different problems. A trader working a large sell order already knows the desired position change; the question is how to distribute that order through time without creating unnecessary cost. On digital-asset exchanges and app-mediated securities venues, this decision must be made while liquidity, spreads, volatility, and participant activity can change inside the execution window. The relevant costs include crossing the spread, temporary market impact, adverse selection, and the exposure created by inventory that remains unexecuted. Classical optimal-execution models express these trade-offs under explicit assumptions about impact and liquidity [1]-[6]. Those assumptions make the models interpretable, but a schedule calibrated to one impact coefficient or one expected volume profile may become poorly matched to the market before the parent order is finished.

Reinforcement learning replaces a pre-specified trading schedule with a policy learned from sequential interaction. Earlier work applied RL to order placement and execution decisions [14],

[15], and later studies examined Double DQN, PPO, and policy distillation for execution [24]-[26]. The difficulty is not simply whether a neural agent can select actions. Three design questions matter in practice. A single value network may use one compromise representation for states that have very different risk, fixed reward weights do not reflect the fact that carrying inventory is more expensive when volatility or spreads widen, and empirical comparisons become hard to audit when proprietary data, preprocessing, and simulated fills are intertwined. Results from adaptive network routing provide a related example in which dueling-DQN controllers respond to changing event streams without requiring expensive inference [33]. This motivates testing whether value-based execution can adapt while keeping the source of that adaptation visible.

Our response is to separate the learned decision from the rule that decides whether the decision may be released. The neural component is a Double Dueling DQN with several expert heads whose outputs are mixed by a gate driven by observable market context. Training also changes the inventory penalty with current volatility and spread. At deployment, the proposed action is checked against a deterministic set of execution constraints. The check can increase urgency after adverse imbalance, during a volatility spike, when liquidation has fallen behind the required pace, or close to the deadline. Because this layer sees only contemporaneous variables and is implemented outside the network, its effect can be tested without attributing every improvement to representation learning.

The resulting study has two architectural contributions and two experimental ones. On the modeling side, the Q function is divided among context-sensitive experts, while inventory and clock information are prevented from directly driving the gate; risk is also handled through both a state-dependent reward term and an external validation layer. On the evaluation side, all comparisons are made on a disclosed four-regime simulator with persistent regime transitions, stochastic liquidity, order-flow pressure, and square-root impact, and policies are paired on exactly the same generated paths. The ablation is deliberately treated as part of the main result. The complete controller beats rule-based schedules and an unshielded static DDQN, yet a static DDQN equipped with the same shield is statistically indistinguishable from the full architecture in the ablation sample. That distinction limits what can reasonably be claimed for regime gating.

2. Related Work

A. Optimal execution and market impact

The execution literature provides both the objective used here and the main cautions about impact. Bertsimas and Lo [1] and Almgren and Chriss [2] formulated discrete-time execution as a balance between expected cost and risk, with later work allowing nonlinear impact and adaptive schedules [3]-[6]. Microstructure evidence shows why fixed-impact assumptions are restrictive: price response varies with order-book events, trade information, institutional order splitting, and the available liquidity state [7]-[13]. In particular, the empirical tendency toward concave, approximately square-root impact motivates the participation-based impact term in our simulator, while the evidence on liquidity variation motivates allowing both execution cost and inventory risk to change with market conditions.

B. Reinforcement learning for trading and execution

Execution research has borrowed from several strands of reinforcement learning rather than from one single algorithmic line. Moody and Saffell [14] used direct reinforcement for trading, while Nevmyvaka et al. [15] showed that execution itself could be learned in a high-dimensional market. The later tool kit includes value-function approximation [17], Double Q-learning [18], the dueling value/advantage split [19], replay [20], and stabilization techniques used in deep RL [21]-[23]. Execution studies have drawn on these pieces in different ways: Ning et al. [24]

used Double DQN, Lin and Beling [25] used PPO, Fang et al. [26] distilled an oracle policy, and Macri and Lillo [27] allowed liquidity to vary through time. Our experiment is arranged to answer a narrower question than which architecture is largest. The adaptive reward, expert gate, and execution shield can each be removed or held fixed, so the matched-path tests can assign observed gains to a component. Zhang et al. [34] provide a useful parallel outside finance: their constrained optimization study also separates higher-level adaptation from lower-level control and embeds domain knowledge in the reward rather than leaving every constraint to be learned implicitly.

C. Data, price formation, and internet-finance markets

Public market datasets are useful for learning state representations, but they do not automatically constitute execution evidence. FI-2010 [28] supplies a shared limit-order-book benchmark, and DeepLOB [29] together with the universal price-formation study of Sirignano and Cont [30] demonstrates that order-flow history contains predictive information. Cryptocurrency markets are relevant because they trade continuously and display economically meaningful cross-venue frictions [31]. The missing object, however, is counterfactual execution. A candle archive records neither queue position nor hidden depth, cancellations, or the fill that would have occurred had a different child order been submitted. We therefore use public-data loaders only as extension infrastructure. Every performance number reported in this study is produced by the stated simulator rather than by labeling candle data as realized execution.

3. Execution Problem Formulation

A. Parent order, state, and actions

Each episode begins with a sell parent order of size Q0 and ends after T=60 decisions. Before decision t, the policy observes remaining inventory xt, the current mid-price Mt, market volume Vt, spread st in basis points, the short-horizon volatility estimate sigma_t, and order-flow imbalance It, together with the normalized quantities summarized in Table I. The action itself is one of six multipliers, A={0, 0.5, 1, 1.5, 2, 3}. Let bt=xt/(T-t) denote the equal-time amount implied by the inventory still outstanding. Choosing at requests qt=at bt, subject to a ceiling of 20% of contemporaneous market volume. A common terminal liquidation rule clears whatever remains after the regular horizon; consequently, no policy can look inexpensive simply by leaving part of the parent order unfinished.

Table I. Normalized State Variables Used by All Learned Policies.

State group	Features	Purpose
Schedule	x/Q0; time left; elapsed	Progress/deadline
Price	last return; momentum	Short pressure
Risk	rolling volatility; spread	Risk/crossing cost
Liquidity	log volume; slice/volume	Capacity limit
Flow	order-flow imbalance	Adverse selection

B. Execution-price and objective model

For a sell order, the simulator prices a fill below the current mid. One part of the concession is half the quoted spread. The other is temporary impact, whose size increases with participation rather than linearly with order size. We use the concave behavior documented in [11], [12] and

write rho_t=q_t/V_t, with rho0=0.05 as the reference participation rate. The resulting fill-price rule is

$$P_t^{exec} = M_t [1 - \{0.5s_t + (0.42s_t + 0.18\sigma_t) \sqrt{(\rho_t/\rho_0)}\}/10^4]. \quad (1)$$

Repeated child orders should not be independent of the trading that came before them. We therefore carry a small share of each trade's temporary impact into the next mid-price. If that term were omitted, the agent could submit several child orders while effectively treating its own earlier executions as costless history. Implementation shortfall is measured against the arrival mid-price M0:

$$IS = 10^4 (M_0 - \Sigma_t q_t P_t^{exec}/Q_0)/M_0. \quad (2)$$

Because this is a sell program, smaller IS means cheaper execution. IS can fall below zero on a path if favorable exogenous price movement more than offsets spread and simulated impact. That possibility is also why path-to-path dispersion is much larger than the mean. We therefore avoid relying on the average alone and report four summaries for every policy: mean, median, the 95th percentile, and CVaR above that percentile.

C. Reproducible market dynamics

The simulated market switches among four latent states: calm-liquid, normal, volatile-thin, and directional-stress. Each state has its own return volatility, spread, log-volume level, and imbalance persistence. In the stress state, a directional drift is drawn for the spell and remains in force until the regime changes. That drift, as well as the regime label itself, is hidden from the policy. The agent sees current imbalance, but imbalance is only an imperfect clue about the latent state. Consequently, the environment contains persistent changes that an adaptive policy can exploit while still requiring inference from noisy observations rather than direct regime access.

Table II. Base Parameters of the Four-Regime Market Generator.

Regime	sigma	Spread	Volume	Stay prob.
Calm	2.2	1.2	135	0.965
Normal	4.5	2.4	95	0.955
Volatile	9.0	6.5	46	0.945
Stress	11.5	9.0	58	0.930

4. Adaptive-RG-DDQN Framework

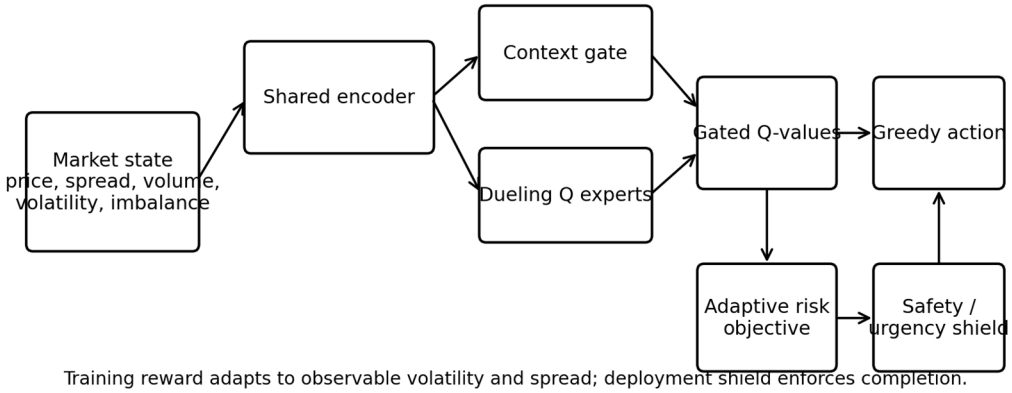

Figure 1. Architecture of the proposed Adaptive-RG-DDQN framework.

A. Regime-gated dueling value function

The ten-dimensional state is first encoded by two shared layers into a 64-dimensional representation. Three dueling heads then operate on that representation. Each head estimates one scalar state value and a vector of action advantages, so expert k forms its action value as

$$Q_k(s,a) = V_k(s) + A_k(s,a) - |A|^{-1} \Sigma_a' A_k(s,a'). \quad (3)$$

The gate is intentionally denied schedule information. Its six inputs are recent return, momentum, volatility, spread, volume, and imbalance; remaining inventory and the clock stay outside this branch. Otherwise, an apparently specialized expert could simply become an 'early-horizon' or 'late-horizon' head, which would say little about market-state adaptation. If g_k(s) is the softmax weight assigned to expert k, the controller receives the weighted value

$$Q(s,a) = \Sigma_k g_k(s) Q_k(s,a), \quad \Sigma_k g_k(s)=1. \quad (4)$$

A small entropy regularizer is used while the experts are still forming so that the gate does not collapse immediately onto one head. Apart from that gate-specific term, the temporal-difference update follows Double DQN: the online network chooses the next action and the target network supplies the value used in the target.

$$y_t = r_t + \gamma(1-d_t)Q^-(s_{t+1}, \arg\max_a Q(s_{t+1},a)). \quad (5)$$

We minimize a smooth-L1 temporal-difference loss with AdamW and refresh the target parameters every 360 environment steps. Actions remain discrete. This is useful for the execution setting because every multiplier corresponds to a recognizable participation band, and the proposed order can pass through a deterministic rule check before it is released.

B. Adaptive risk-aware reward

The reward is built from the marked change in portfolio wealth at one step. We then charge for two things that the wealth increment does not fully represent: inventory that is still exposed and lateness relative to the schedule. The static baseline uses a fixed inventory coefficient. Adaptive-RG-DDQN instead increases that coefficient only after observed volatility or spread moves beyond its nominal reference:

$$\lambda_t = \lambda_0[1 + 1.8 \max(\sigma_t/5 - 1,0) + 1.2 \max(s_t/3 - 1,0)]. \quad (6)$$

$$r_t = 0.1[\Delta W_t(\text{bps}) - \lambda_t(x_{t+1}/Q_0)^2\sigma_t - u_t]. \quad (7)$$

Neither equation contains the latent regime label. The larger inventory charge is triggered solely by quantities the agent can observe, namely the volatility and spread measures used in the state. Static DDQN is kept comparable on simulator, replay, action set, and depth, but it uses one dueling head and holds lambda_t at lambda_0. Chen et al. [35] study a different financial problem, yet the design principle is related: their uncertainty calibration changes with volatility rather than assuming one risk level for every market condition. Here that same idea is applied to the cost assigned to unexecuted inventory.

C. Observable safety and urgency shield

The shield sits between action selection and order release. It can raise the multiplier for a seller when imbalance is strongly adverse, when high volatility arrives with nonpositive imbalance,

or when the remaining inventory is too large for the time left. Reducing an aggressive proposal is allowed only under a narrower combination: strongly favorable imbalance, a wide spread, and enough remaining time. The final part of the schedule is governed by hard urgency floors. During the last 20% of decisions the released action is at least 1.5 times the equal slice; during the last 10% it is at least twice that slice. These checks use only the current state, never future prices or future volume. Operationally, the shield is therefore a risk-limit layer, not another forecasting network. Liang et al. [36] use a related neuro-symbolic construction in which explicit domain constraints validate the output of an adaptive model.

Putting the shield outside the Q network is also what makes the attribution test meaningful. If performance deteriorates when the same learned policy is run without this layer, the lost performance cannot be credited to the representation alone. Section VI-D therefore compares the full controller, a no-shield version, static DDQN, and static DDQN with the identical shield.

5. Experimental Design

A. Data policy and reproducibility

All headline numbers in this paper come from the simulator, not from claimed exchange execution. Binance makes candles and trades available and FI-2010 contains genuine limit-order-book observations [28], but neither source tells us the counterfactual fills for several large-order policies on exactly the same realized market path. Those missing fills have to be generated by assumptions about matching, available liquidity, and impact. We make those assumptions explicit in the simulator and reuse each generated path across policies. The accompanying Binance adapter is meant for later calibration or representation pretraining; it is not called by the scripts that create the reported tables.

B. Training protocol and baselines

Training is repeated independently for seeds 7, 19, and 31, with 10,000 environment interactions for each Static-DDQN and Adaptive-RG-DDQN run. Experience is sampled uniformly from replay; prioritized replay is intentionally excluded so that sampling strategy does not become another difference between the two agents. Out-of-sample evaluation uses 300 new paths per seed. Thus each policy is evaluated on 900 paths, and the path with a given seed and episode index is identical across policies. The matched construction is what permits paired statistical testing. None of these evaluation paths is used for parameter fitting or hyperparameter choice.

Table III. Training and Evaluation Configuration.

Item	Value	Item	Value
Horizon	60	Train steps	10,000/seed
Actions	6	Seeds	7, 19, 31
Hidden units	64	Replay	120,000
Batch	128	Discount	0.995
Learning rate	3e-4	Target sync	360 steps
Eval paths	900/policy	Max part.	20%

The comparison set contains four policies that can operate without future information and one reference that cannot. TWAP uses the current equal-time slice. POV aims at 5% of observed volume. The front-loaded rule removes more inventory near the start, and static DDQN

provides a learned policy without the gated architecture or the adaptive shield. VWAP-Oracle is different: it allocates with knowledge of volume later in the same episode, so it is included only as an ex-post yardstick. Every method is subject to the same participation cap and the same terminal liquidation rule.

C. Statistical analysis

The evaluation is paired at the path level. For baseline B on path i, let d_i=IS_{B,i}-IS_{A,i}, where A is Adaptive-RG-DDQN; d_i>0 means the adaptive policy paid less on that same realization. Table V reports the sample mean of these paired differences, a two-sided paired t test, and a 95% confidence interval. Pairing removes much of the nuisance variation created by different market paths, but it does not turn a simulator result into a market-wide statement. The p values therefore answer only the mean-cost question under this experimental design. Tail statistics and regime-conditioned averages are reported separately because they describe different aspects of execution risk.

D. Implementation footprint and audit trail

The implementation was kept small enough that the model itself can be inspected. The released PyTorch modules contain 9,255 trainable parameters for static DDQN and 18,360 for the gated network. Runs use one CPU thread and a six-action space; no specialized hardware is needed to reproduce the setup. Each run writes its configuration, seed-specific logs and checkpoints, episode-level policy outcomes, aggregate summaries, and paired-test output. Ablation, shield diagnostics, and impact sensitivity are regenerated by separate runners. Thus a number appearing in a result table can be traced downward to the relevant episode records rather than being accepted from a manually copied plot.

6. Results and Analysis

A. Overall execution quality

Table IV. Implementation Shortfall in Basis Points On 900 Out-Of-Sample Paths.

Policy	Mean	Median	P95	CVaR95
TWAP	3.83	3.27	69.10	101.41
POV	4.57	4.25	69.46	102.74
FrontLoaded	4.69	4.01	57.93	84.79
VWAP-Oracle	2.61	2.74	67.92	103.11
Static-DDQN	5.83	6.06	66.49	98.23
Adaptive-RG-DDQN	2.87	3.71	65.71	93.39

Across the 900 held-out paths, the adaptive controller finishes every parent order and averages 2.87 bps of implementation shortfall. TWAP averages 3.83 bps, POV 4.57 bps, the front-loaded schedule 4.69 bps, and static DDQN 5.83 bps. The largest mean separation is therefore between the two learned policies, not between the adaptive model and a hand-coded schedule. In these simulations, leaving the learned policy unconstrained appears costly when it delays liquidation or reacts poorly to adverse flow. The tail summary is consistent with that reading: Adaptive-RG-DDQN has the lowest CVaR95 among deployable policies, 93.39 bps. Front-loaded execution shows a smaller standard deviation, which is unsurprising because it spends less time carrying market exposure.

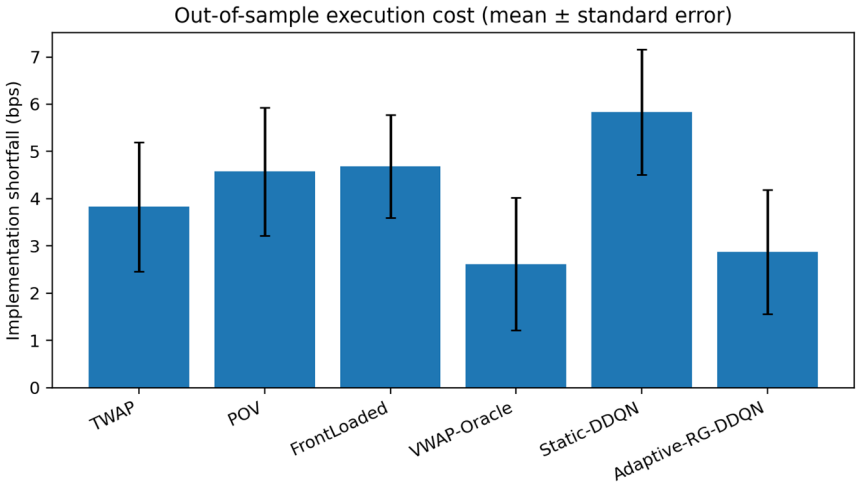

Figure 2. Mean out-of-sample shortfall with standard-error bars.

The oracle provides a useful ceiling on what can be inferred from these numbers. Its mean IS is 2.61 bps, only 0.26 bps below the adaptive policy, and the paired difference is not significant. That does not make the oracle a feasible benchmark: it allocates with future episode volume in hand, while Adaptive-RG-DDQN sees only information available at the decision time. The defensible conclusion is therefore limited to the simulated setting: contemporaneous control reaches an average cost close to the volume-informed reference on these paths.

B. Paired significance and tail behavior

Table V. Paired Cost Reduction of Adaptive-Rg-Ddqn; Positive Is Better.

Baseline	Delta bps	95% CI	p
TWAP	0.95	[0.56, 1.35]	2.0e-06
POV	1.70	[1.04, 2.36]	5.0e-07
Front	1.81	[1.20, 2.42]	7.8e-09
Oracle	-0.26	[-0.74, 0.22]	0.2917
Static	2.96	[2.29, 3.63]	2.4e-17

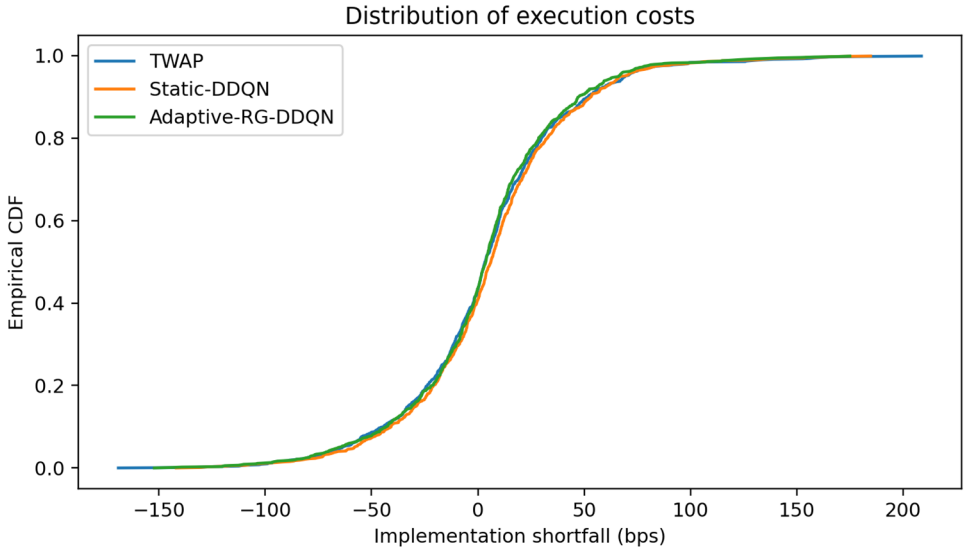

Figure 3. Empirical cost distributions for TWAP and learned policies.

Using the same market realization for both policies makes the cost differences much easier to estimate. The paired means favor Adaptive-RG-DDQN by 0.95 bps over TWAP (p=2.0e-6), 1.70 bps over POV (p=5.0e-7), 1.81 bps over the front-loaded rule (p=7.8e-9), and 2.96 bps over static DDQN (p=2.4e-17). The confidence interval for the oracle comparison crosses zero. These tests establish differences in simulated execution cost for the stated experiment; they do not establish live alpha, nor do they imply that the same margins would persist on another venue.

The distribution plot adds an important qualification to the mean table. Outcomes still overlap substantially because exogenous price movement dominates many individual paths. Relative to static DDQN, the adaptive policy shifts the distribution rather than making it narrow. That is appropriate for an execution algorithm: it can reduce costs caused by its own scheduling decisions, but it cannot remove directional market movement that occurred while the parent order was being worked.

C. Market-regime robustness

Table VI. Mean Shortfall by Dominant Market Regime (Bps).

Regime	TWAP	Static	Adaptive
Calm	2.11	4.04	1.60
Normal	1.34	2.83	0.89
Volatile	8.03	10.12	6.81
Stress	4.19	7.55	1.44

Conditioning on the dominant simulated regime does not produce a uniform-sized benefit. Adaptive-RG-DDQN nevertheless has the lowest mean IS of the three reported policies in calm, normal, volatile-thin, and directional-stress episodes. Costs are highest in volatile-thin paths, where wide spreads, high volatility, and low volume arrive together. Directional stress shows the largest numerical separation: 1.44 bps for the adaptive controller versus 4.19 for TWAP and 7.55 for static DDQN. Earlier acceleration after adverse imbalance or rising deadline pressure explains part of that difference. Only 104 paths are stress-dominant, however, so the stress row is better viewed as a focused stress-test result than as a general guarantee.

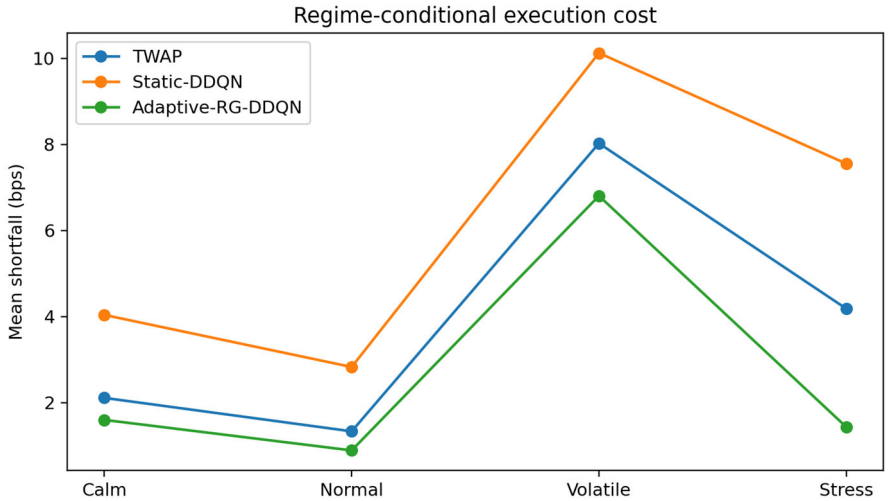

Figure 4. Mean execution cost conditioned on the dominant market regime.

D. Ablation and attribution of gains

Table VII. Ablation Results On 300 Paired Paths Per Variant (Bps).

Variant	Mean	P95	CVaR95
Full framework	2.79	66.26	89.66
No shield	5.39	79.44	116.42
Static + shield	2.62	60.98	85.94
Static	5.62	70.05	94.92

The ablation changes where credit for the main result should go. On the paired sample, removing the shield raises mean IS from 2.79 to 5.39 bps, an increase of 2.59 bps (p=0.0020). Giving the same shield to static DDQN lowers its mean to 2.62 bps, and that value is not significantly different from the complete Adaptive-RG-DDQN result (p=0.536). The full controller still outperforms unshielded static DDQN, but these comparisons do not identify the expert gate as the source of that advantage. What is demonstrated most clearly is the value of the explicit control layer. Establishing an additional contribution from expert specialization will need either more statistical power, different calibration, or both.

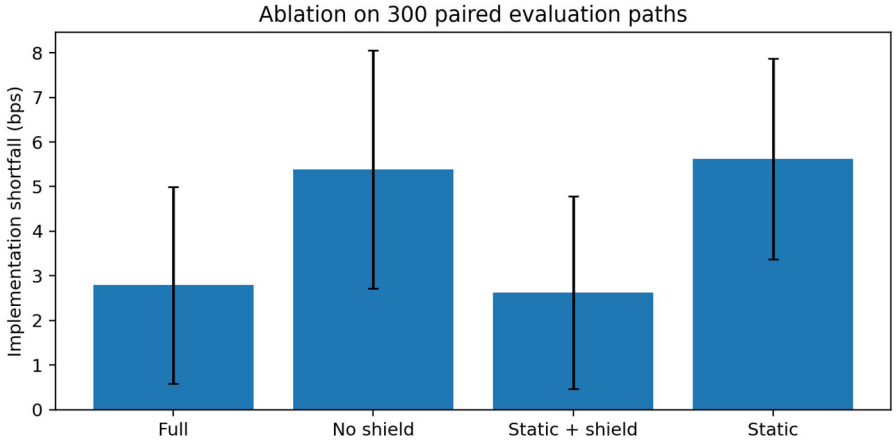

Figure 5. Mean ablation cost with standard-error bars.

That neutral result prevents a misleading architectural story. If the only baseline were static DDQN without a shield, the entire performance gap could be attributed to the gated network even though Table VII does not support that interpretation. The component test instead points to concrete engineering choices that remain relevant regardless of the gate: enforceable risk limits, deadline-aware intervention, and behavior that remains controlled when liquidity deteriorates abruptly.

E. Intervention diagnostics and inventory trajectory

We also record what the shield changes, not just the final cost. Across 300 replayed paths, each decision stores the network proposal and the action actually released. A correction occurs on 41.5% of proposals. The mean multiplier moves from 0.92 before validation to 1.37 after it, so the network is frequently less aggressive than the rule layer allows. Corrections are not evenly distributed: the rates are 31.0% in calm states, 53.4% in volatile states, 52.6% in stress states, and 46.9% during the final third of the schedule. These diagnostics show that the rules are

activating where they were designed to activate. They also reveal that the learned policy has not yet absorbed the completion constraint well enough; a better-aligned policy would need fewer interventions.

Table VIII. Shield Intervention Diagnostics On 18,000 Decisions.

Group	Interv. %	Proposed	Final
Calm	31.0	1.16	1.44
Normal	35.5	1.00	1.38
Volatile	53.4	0.72	1.34
Stress	52.6	0.70	1.29
Early	37.8	0.87	1.22
Middle	39.8	0.81	1.20
Late	46.9	1.09	1.69

The inventory trace tells the same story from the schedule side. Mean normalized inventory is 1.00 at arrival and then 0.80, 0.63, 0.45, 0.28, and 0.12 at decisions 10, 20, 30, 40, and 50 before reaching zero at the terminal decision. The 10th-90th percentile band is broad early in the horizon, meaning that different market paths genuinely produce different liquidation schedules rather than one fixed TWAP curve. Near the deadline the band contracts because the shield removes progressively more freedom to stay behind the required completion pace.

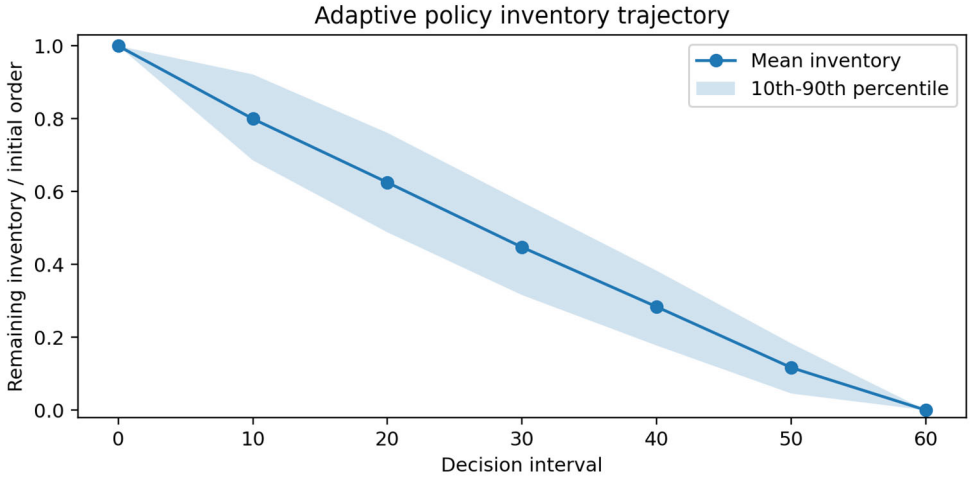

Figure 6. Mean remaining inventory and the 10th-90th percentile band.

F. Sensitivity to impact-model misspecification

Impact sensitivity is evaluated after training rather than by fitting three new agents. We multiply the temporary-impact coefficient by 0.75, 1.00, or 1.25 and run 150 fresh paired paths at each setting. These samples are independent of the paths in Table IV, so only within-experiment comparisons should be read from Table IX. Adaptive-RG-DDQN remains cheaper on average than TWAP and static DDQN at all three multipliers. As impact becomes stronger, its paired gain over TWAP moves from 1.64 to 1.32 bps, whereas the gain over static DDQN increases from 2.52 to 2.94 bps. This experiment shows tolerance to moderate coefficient error. It does not test structural misspecification such as queue priority, strategic responses from other traders, or other mechanisms absent from the simulator.

Table IX. Mean Cost Under Temporary-Impact Scaling (Bps; 150 Paths Per Scale).

Scale	Adaptive	TWAP	Static	Gain/TWAP
0.75	-0.75	0.88	1.76	1.64
1.00	0.25	1.73	2.98	1.48
1.25	1.25	2.57	4.20	1.32

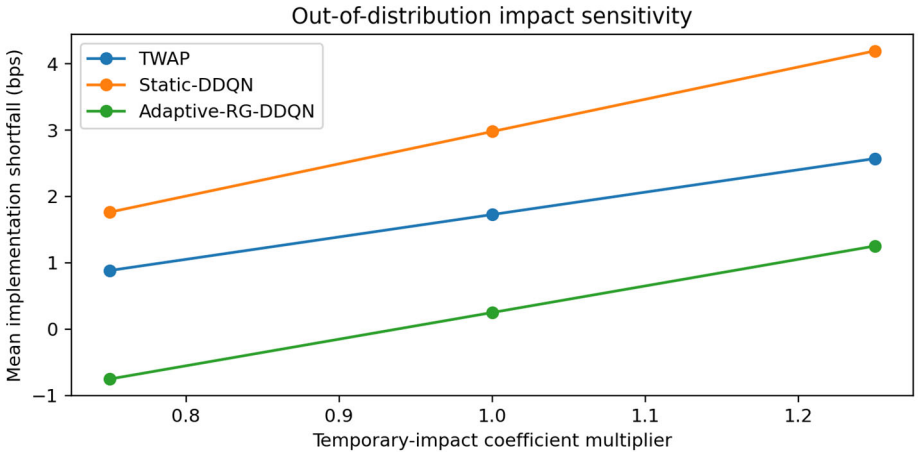

Figure 7. Mean shortfall under out-of-distribution impact scaling.

7. Limitations and Deployment Considerations

External validity remains the main unresolved limitation. The simulator is explicit, but it leaves out several features that matter in a live matching engine: queue priority, passive orders, partial fills, cancellations, fees, latency, hidden liquidity, self-exciting flow, and strategic reactions by other participants. Its impact coefficients are stylized rather than estimated for one venue. There is also an internal warning sign: the shield changes 41.5% of neural proposals, so the policy and the safety layer are still far from aligned. For these reasons, the basis-point levels in the tables should remain simulator-specific and should not be carried directly into a production execution-cost forecast.

A deployment study should reduce modeling assumptions in stages rather than jump from this simulator to live orders. First, spread, volatility, volume, and impact distributions would need to be calibrated on legally usable venue data with chronological train, validation, and test periods. The frozen policy could then be replayed in a richer simulator that includes queue position and latency and compared against the desk benchmark on matched parent orders. If that stage remains acceptable, shadow execution can observe the live market without releasing orders; only after that would a small-notional canary be justified. The shield should have its own version history, and every intervention should retain the state variables that triggered it. Independent limits on order size, participation, price bands, loss, and emergency shutdown should remain outside the learned policy throughout.

Some deployment failures would have little to do with statistical accuracy. Internet-finance venues can suffer API outages, symbol changes, fee revisions, distorted token liquidity, and changing custody or routing constraints. A policy calibrated before a matching-engine or fee-rule update can therefore become stale even if its weights have not moved. Operational monitoring should watch the incoming-state distribution, completion probability, realized slippage, and the frequency with which the shield intervenes. The released code is research

software intended to reproduce the experiments; it is not presented as a production trading stack or investment advice.

The ablation leaves one empirical question worth isolating next: after both agents receive the same shield, does the gated expert structure still improve execution? A preregistered study using public or licensed order-book data would make that comparison more informative. It should increase the number of seeds, report uncertainty across dates and assets, and lock the final test period before choosing thresholds or hyperparameters. Keeping that incremental hypothesis fixed would separate any genuine benefit from market-state specialization from the benefit already delivered by explicit risk control.

8. Reproducibility, Governance, and Claim Scope

A. Artifact-level reproducibility

Reproducibility is organized around the experiment artifacts, not just the summary tables. Market generation, training, evaluation, diagnostics, and manuscript output are handled by separate scripts. The main runner records the full configuration and emits seed-indexed learning curves, checkpoints, episode-level outcomes, aggregate statistics, and paired tests; other runners reproduce the ablation, shield audit, and impact-sensitivity study. Values in Tables IV-IX can therefore be traced back to machine-readable episode data. Bit-for-bit equality is not required across hardware, but the paired path identities, documented software setup, and analysis procedure underlying the confidence-interval conclusions are fixed.

B. Synthetic-data disclosure and evidence boundary

The simulator is necessary here because the comparison itself is counterfactual: we want to know what several policies would have paid on the same underlying realization. Historical candles cannot supply those alternative fills, and an observed order book still needs a matching and queue model before unexecuted policies can be evaluated. We keep that boundary explicit by describing the numbers as simulated outcomes and making no inference to realized exchange performance. Negative implementation shortfall is possible when favorable exogenous movement exceeds simulated spread and impact. The Binance adapter is included for follow-on work only and is not used by result generation.

C. Reproduction acceptance tests

A clean-room reproduction begins with an empty output directory. Success means that the run recreates the configuration, six seed-specific checkpoints, training logs, episode-level outcomes for every policy, aggregate summaries, and all manuscript figures. The acceptance test is structural rather than bitwise: policies must share the same path identifiers, every parent order must satisfy the common completion rule, and the paired statistics must be recomputed from episode rows. Small hardware-dependent floating-point differences are acceptable only when they leave the stated confidence-interval conclusions unchanged.

Bibliographic auditing is kept separate from numerical auditing. A machine-readable registry stores a DOI, ISBN, arXiv identifier, or publisher page for each reference together with its verification status. A valid link may still disappear over time, but the registry makes placeholder or unsupported entries easier to detect. This lets a reviewer check where the numbers came from and where the citations came from as two distinct reproducibility tasks.

D. Operational and ethical governance

Model accuracy alone is not sufficient governance for a production study. Venue data must be used under the relevant license and policy, the system must pass model-risk review, and research evaluation must remain separate from capital deployment. Neither the learned policy nor the shield may circumvent exchange limits, market-abuse controls, or human ownership of risk decisions. An audit record should retain the parent-order objective, the proposed action,

the released action, the shield rule that fired, the model version, and the contemporaneous market state. Because execution difficulty changes with liquidity and client urgency, performance should be reported by meaningful cohorts, and a deterministic fallback should remain available when inputs are missing or a distribution-shift alarm is active.

E. Real-data validation protocol

A real-data study should preserve time ordering from the start. Only the training period may be used to calibrate the simulator; hyperparameters and shield thresholds should be fixed before the final test window opens. Competing policies should receive the same parent orders and the same market snapshots. Uncertainty also needs to respect dependence within a trading day and within an asset, rather than treating every child-order decision as an independent observation. Fees, latency, queue assumptions, rejected orders, and missing-data rules should be reported explicitly because any one of them can be larger than a one- or two-basis-point performance difference.

Progression beyond offline tests should be governed by operational conditions, not one backtest number. Shadow execution should require reliable completion, bounded participation, an intervention rate acceptable to the desk, and no material loss of performance when spread, volume, and latency are stressed. If a canary phase is approved, its notional should remain below an independently set limit and it should revert automatically to a deterministic schedule after stale inputs, model exceptions, abnormal rejection rates, or distribution-shift alarms. Those promotion rules keep the live-evidence boundary visible and make the decision to advance the system auditable.

9. Conclusion

Adaptive-RG-DDQN was evaluated as a modular execution controller rather than as a stand-alone forecasting model. It combines a gated Double Dueling DQN, an inventory penalty that changes with observable risk, and an external shield that enforces urgency and safety constraints. Across 900 matched paths from the disclosed four-regime simulator, the full controller lowers mean implementation shortfall relative to TWAP, POV, a front-loaded schedule, and unshielded static DDQN, and every order is completed. Its average gap to an ex-post volume oracle is not significant. The component test determines how this result should be interpreted: most of the measured gain disappears when the shield is removed, while the gated architecture has not shown an independent advantage once the same shield is given to the static model. The evidence therefore supports transparent risk-constrained learning more strongly than it supports the mixture-of-experts hypothesis. Future work should test that remaining hypothesis directly on better calibrated execution environments.

References

[1] Bertsimas, D., & Lo, A. W. (1998). Optimal control of execution costs. Journal of Financial Markets, 1(1), 1–50. https://doi.org/10.1016/S1386-4181(97)00012-8.

[2] Almgren, R., & Chriss, N. (2001). Optimal execution of portfolio transactions. Journal of Risk, 3(2), 5–39. https://doi.org/10.21314/JOR.2001.041

[3] Almgren, R. F. (2003). Optimal execution with nonlinear impact functions and trading-enhanced risk. Applied Mathematical Finance, 10(1), 1–18. https://doi.org/10.1080/135048602100056

[4] Gatheral, J. (2010). No-dynamic-arbitrage and market impact. Quantitative Finance, 10(7), 749–759. https://doi.org/10.1080/14697680903373692

[5] Almgren, R., & Lorenz, J. (2011). Mean-variance optimal adaptive execution. Applied Mathematical Finance, 18(5), 395–422. https://doi.org/10.1080/1350486X.2011.560707

[6] Gatheral, J., Schied, A., & Slynko, A. (2012). Transient linear price impact and Fredholm integral equations. Mathematical Finance, 22(3), 445–474. https://doi.org/10.1111/j.1467-9965.2011.00478.x

[7] Cont, R., Kukanov, A., & Stoikov, S. (2014). The price impact of order book events. Journal of Financial Econometrics, 12(1), 47–88. https://doi.org/10.1093/jjfinec/nbt003

[8] Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., & Howison, S. D. (2013). Limit order books. Quantitative Finance, 13(11), 1709–1742. https://doi.org/10.1080/14697688.2013.803148

[9] Kyle, A. S. (1985). Continuous auctions and insider trading. Econometrica, 53(6), 1315–1335. https://doi.org/10.2307/1913210

[10] Hasbrouck, J. (1991). Measuring the information content of stock trades. The Journal of Finance, 46(1), 179–207. https://doi.org/10.1111/j.1540-6261.1991.tb03749.x

[11] Gabaix, X., Gopikrishnan, P., Plerou, V., & Stanley, H. E. (2006). Institutional investors and stock market volatility. The Quarterly Journal of Economics, 121(2), 461–504. https://doi.org/10.1162/qjec.2006.121.2.461

[12] Toth, B., Lemperiere, Y., Deremble, C., de Lataillade, J., Kockelkoren, J., & Bouchaud, J.-P. (2011). Anomalous price impact and the critical nature of liquidity in financial markets. Physical Review X, 1(2), 021006. https://doi.org/10.1103/PhysRevX.1.021006

[13] Eisler, Z., Bouchaud, J.-P., & Kockelkoren, J. (2012). The price impact of order book events: market orders, limit orders and cancellations. Quantitative Finance, 12(9), 1395–1419. https://doi.org/10.1080/14697688.2010.528444

[14] Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. IEEE Transactions on Neural Networks, 12(4), 875–889. https://doi.org/10.1109/72.935097

[15] Nevmyvaka, Y., Feng, Y., & Kearns, M. (2006). Reinforcement learning for optimized trade execution. Proceedings of the 23rd International Conference on Machine Learning, 673–680. https://doi.org/10.1145/1143844.1143929

[16] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

[17] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518, 529–533. https://doi.org/10.1038/nature14236

[18] van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with Double Q-learning. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1), 2094–2100. https://doi.org/10.1609/aaai.v30i1.10295

[19] Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. Proceedings of the 33rd International Conference on Machine Learning, 48, 1995–2003. https://arxiv.org/abs/1511.06581

[20] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. International Conference on Learning Representations. https://arxiv.org/abs/1511.05952

[21] Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1), 3215–3222. https://doi.org/10.1609/aaai.v32i1.11796

[22] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. https://arxiv.org/abs/1707.06347

[23] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations. https://arxiv.org/abs/1412.6980

[24] Ning, B., Lin, F. H. T., & Jaimungal, S. (2021). Double Deep Q-Learning for optimal execution. Applied Mathematical Finance, 28(4), 361–380. https://doi.org/10.1080/1350486X.2022.2077783

[25] Lin, S., & Beling, P. A. (2020). An end-to-end optimal trade execution framework based on proximal policy optimization. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, 4548–4554. https://doi.org/10.24963/ijcai.2020/627

[26] Fang, Y., Ren, K., Liu, W., Zhou, D., Zhang, W., Bian, J., Yu, Y., & Liu, T.-Y. (2021). Universal trading for order execution with oracle policy distillation. Proceedings of the AAAI Conference on Artificial Intelligence, 35(1), 107–115. https://doi.org/10.1609/aaai.v35i1.16083

[27] Macrì, A., & Lillo, F. (2024). Reinforcement learning for optimal execution when liquidity is time-varying. Applied Mathematical Finance, 31(5), 312–342. https://doi.org/10.1080/1350486X.2025.2490157

[28] Ntakaris, A., Magris, M., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2018). Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. Journal of Forecasting, 37(8), 852–866. https://doi.org/10.1002/for.2543

[29] Zhang, Z., Zohren, S., & Roberts, S. (2019). DeepLOB: Deep convolutional neural networks for limit order books. IEEE Transactions on Signal Processing, 67(11), 3001–3012. https://doi.org/10.1109/TSP.2019.2907260

[30] Sirignano, J., & Cont, R. (2019). Universal features of price formation in financial markets: perspectives from deep learning. Quantitative Finance, 19(9), 1449–1459. https://doi.org/10.1080/14697688.2019.1622295

[31] Makarov, I., & Schoar, A. (2020). Trading and arbitrage in cryptocurrency markets. Journal of Financial Economics, 135(2), 293–319. https://doi.org/10.1016/j.jfineco.2019.07.001

[32] Cartea, Á., Jaimungal, S., & Penalva, J. (2015). Algorithmic and high-frequency trading. Cambridge University Press. https://doi.org/10.1017/CBO9781139340716

[33] Wang, B., Wang, Z., Zhao, W., Zhang, F., & Shang, W. (2026). DRL-Adapt: Deep reinforcement learning for adaptive routing convergence optimization in large-scale networks. IEEE Open Journal of the Computer Society, 7, 849–863. https://doi.org/10.1109/OJCS.2026.3687441

[34] Zhang, H., Ge, Y., Zhao, X., & Wang, J. (2025). Hierarchical deep reinforcement learning for multi-objective integrated circuit physical layout optimization with congestion-aware reward shaping. IEEE Access, 13, 162533–162551. https://doi.org/10.1109/ACCESS.2025.3610615

[35] Chen, Z., Wang, M., Zeng, Z., & Ping, W. (2026). Uncertainty-aware financial forecasting: Leveraging conformal prediction for risk-adjusted models. IEEE Access, 14, 90053–90077. https://doi.org/10.1109/ACCESS.2026.3702666

[36] Liang, Y., Jiao, Y., Ping, W., Fan, H., & Han, X. (2026). Adaptive event-driven labeling: A neuro-symbolic multiagent framework for causal inference in non-stationary time series. IEEE Access, 14, 104468–104482. https://doi.org/10.1109/ACCESS.2026.3709267