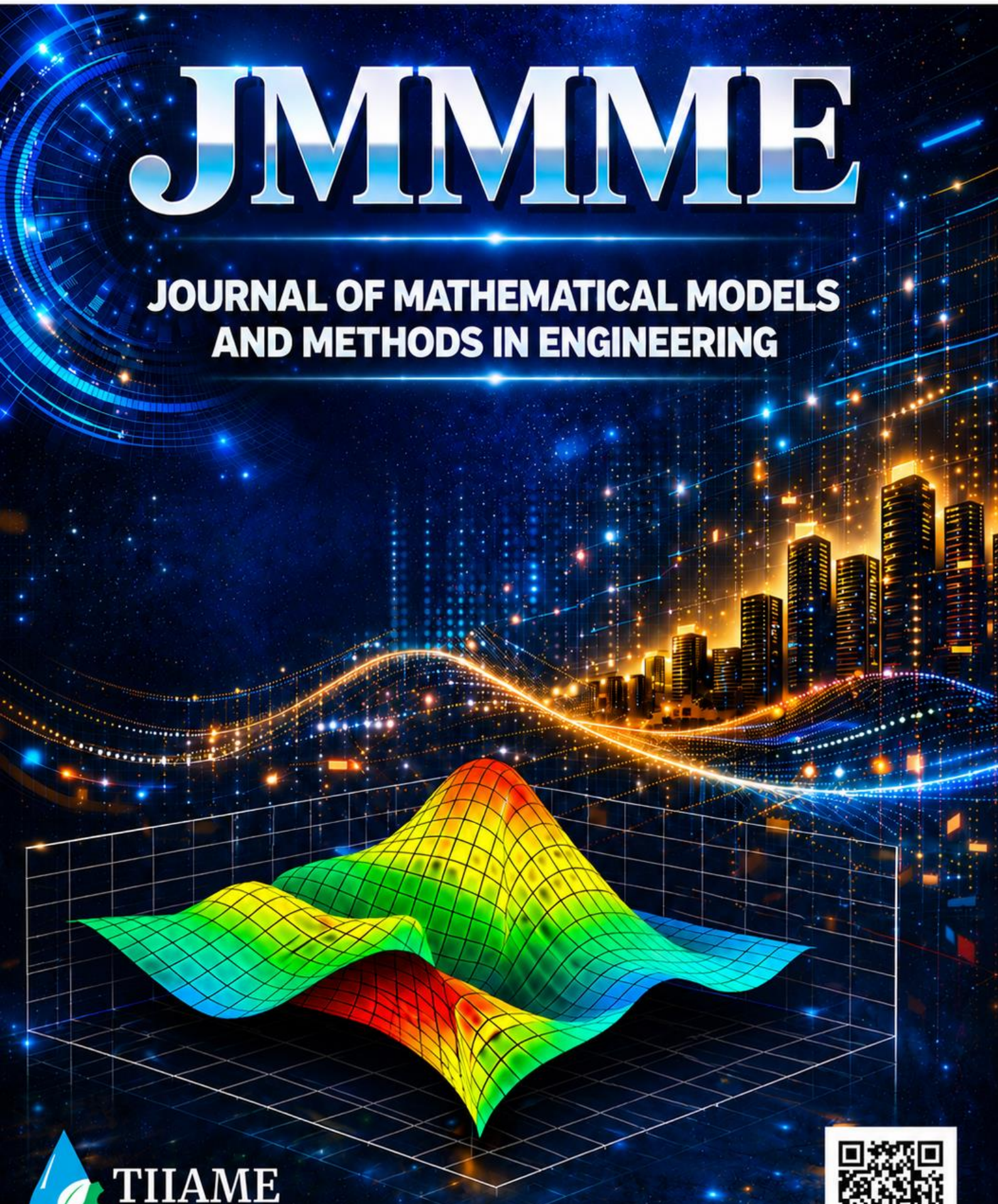
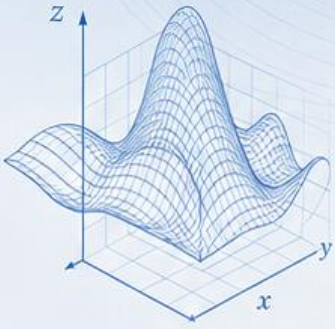
INTERNATIONAL STANDARD SERIAL NUMBER INTERNATIONAL CENTRE
Google Scholar
zenodo
doi
OpenAIRE
OPEN ACCESS

JMMME

JOURNAL OF MATHEMATICAL MODELS AND METHODS IN ENGINEERING



TIIAME
National Research University



JMMME

JOURNAL OF MATHEMATICAL MODELS AND METHODS IN ENGINEERING

$$\frac{dx}{dt} = Ax + b(t)$$

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u + f$$

$$\sum_{i=1}^{n} a_i x_i + b$$

$$f(x) = \int_a^b f(x)\,dx$$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

JOURNAL INFORMATION

"Journal of Mathematical Models and Methods in Engineering" is an international, peer-reviewed, and open-access scientific journal established in the Republic of Uzbekistan by the National Research University "Tashkent Institute of Irrigation and Agricultural Mechanization Engineers" under Certificate No. 1648073 dated June 15, 2026.

$$\sum_{i=1}^{n} w_i x_i$$

The journal specializes in publishing scientific research on mathematical modeling, numerical methods, computational algorithms, optimization, and analytical approaches across various fields of engineering — including civil engineering, mechanical engineering, energy, hydraulic engineering, materials science, robotics, and environmental protection. It serves as a scientific platform for scholars, engineers, educators, and doctoral researchers to exchange theoretical and applied knowledge.



All submitted manuscripts undergo a double-blind peer review process and are published in strict compliance with high academic quality standards and publication ethics.



The journal has been published since June 15, 2026, once per quarter, that is, 4 times per year. Manuscripts are accepted in English and must be prepared in accordance with international IMRAD standards.

TIIAME
National Research University

FOUNDER	TIIAME National Research University
ADDRESS	39 Qori-Niyoziy Street, Mirzo Ulugbek District, Tashkent, Republic of Uzbekistan
PHONE	+998 (93) 397-12-39
WEBSITE	jmmme.uz
CERTIFICATE	No. 1648073, June 15, 2026
PUBLISHED	2026-06-15
LANGUAGE	English
FREQUENCY	Quarterly / 4 times per year
REVIEW	Double-blind peer review
STANDARD	International IMRAD standards

0 1 0 1
1 0 1 0
0 1 0 1
1 0 1 1
1 1 0 1
1 1 0 1
0 1 1 0
0 0 1 1
1 1 0 1
1 1 1 1
2 0 2 2
1 2 1 1

INDEXED AND DISCOVERABLE THROUGH

ISSN INTERNATIONAL STANDARD SERIAL NUMBER | Google Scholar | zenodo | doi | OpenAIRE | OPEN ACCESS

JMMME

TIIAME National Research University

JOURNAL OF MATHEMATICAL MODELS AND METHODS IN ENGINEERING

EDITOR-IN-CHIEF

Xudayarov Baxtiyar Alimovich

Doctor of Technical Sciences, Professor

TIIAME National Research University, Tashkent, Republic of Uzbekistan

ASSOCIATE EDITOR

Abdullayev Akmaljon Abdujalilovich

PhD, Associate Professor, Department of Higher Mathematics

TIIAME National Research University, Tashkent, Republic of Uzbekistan

COMPUTER OPERATOR AND DESIGNER

Sattarov Ulug'bek Umed o'g'li

EDITORIAL BOARD MEMBERS

Turayev Fozil Jurakulovich
Head of the Department of Higher Mathematics, Associate Professor
TIIAME National Research University, Tashkent, Republic of Uzbekistan

Ergashev Tukhtasin Gulomjonovich
Doctor of Physical and Mathematical Sciences, Professor
TIIAME National Research University, Tashkent, Republic of Uzbekistan

Eshkuvatov Zaynidin Karimovich
Doctor of Physical and Mathematical Sciences, Professor
TIIAME National Research University, Tashkent, Republic of Uzbekistan

Eshmatov Baxtiyor Xasanovich
Doctor of Physical and Mathematical Sciences, Associate Professor
TIIAME National Research University, Tashkent, Republic of Uzbekistan

INTERNATIONAL SUBJECT EDITORS

Andriy Verlan
Associate Professor, Department of Manufacturing and Civil Engineering, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Dr. Mohd Kamal Mohd Nawawi
PhD, Associate Professor, School of Quantitative Sciences, Universiti Utara Malaysia, Kedah, Malaysia.

Vatin Nikolay Ivanovich
Doctor of Technical Sciences, Professor, Peter the Great St. Petersburg Polytechnic University.

Norihan Md Arifin
Professor of Applied Mathematics, Head of the Department of Mathematics and Statistics, Universiti Putra Malaysia, Selangor, Malaysia.

Sergey Leonov
Doctor of Engineering, Professor at National Technical University "Kharkiv Polytechnic Institute".

Amarjit Singh
PhD, Department of Mechanical Engineering, National Institute of Technology Hamirpur HP, India.

Vishesh Ranjan Kar
Assistant Professor, Department of Mechanical Engineering, National Institute of Technology Jamshedpur, India.

Muhammad Suzuri Hitam
Professor, Faculty of Computer Science and Mathematics, University Malaysia Terengganu, Malaysia.

Tkachenko Oleg Pavlovich
Doctor of Physical and Mathematical Sciences, Associate Professor, Leading Researcher at the Computing Center of the Far Eastern Branch of the Russian Academy of Sciences, Khabarovsk, Russia.

Gobithaasan Rudrusamy
Assoc. Prof. Dr. of School of Mathematical Sciences, University Sains Malaysia, Penang, Malaysia.

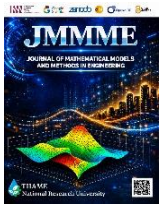
Prof. Dr. Muhammad Zaini Bin Ahmad
Doctor of Industrial Computing, Professor, Department of Mathematical Sciences, Faculty of Intelligent Computing, Universiti Malaysia Perlis, Malaysia.

Mahmoud Mohamed Selim Saleh
Doctor of Applied Mathematics, Professor, College of Science and Humanities, Prince Sattam bin Abdulaziz University, Al-Kharj, Saudi Arabia.

Pradeep G. Siddheshwar
M.Sc., Ph.D. Senior Professor of Mathematics, Bangalore 560 029, India.

Journal of Mathematical Models and Methods in Engineering

journal homepage:
https://jmmme.uz/index.php/journal



Comparing Stochastic Training Environments for Deep Reinforcement Learning with Deep Q-network: Learning Stability and Transfer to Real Data

Yu.V.Yurchenko[1*], O.Yu. Zakovorotnyi[2]

[1,2]*Department of Computer Engineering and Programming, National Technical University "Kharkiv Polytechnic Institute", 2 Kyrpychova St., Kharkiv 61002, Ukraine*

ARTICLE INFO

Article history:
Received 9 September 2026
Received in revised form 10 September 2026
Accepted 2 October 2026
Available online 11 October 2026

Keywords:

neural network; modeling theory; model synthesis; mathematical model; deep reinforcement learning; deep Q-network (DQN); stochastic environment modeling; sim-to-real transfer; training stability

ABSTRACT

The paper investigates how the choice of a generative stochastic model of the training environment affects the numerical stability of the learning and the transfer of the Double DQN policy to real time series. Four environments — GBM, GARCH(1,1)-t, Minimal Neural SDE, and Neural Jump-Diffusion — are compared, created on chronologically previous data of three financial time series: NVDA, SPY, and KO. For each combination of asset and environment, ten training runs are performed, a total of 120, with the architecture, state and action spaces, reward functions, and hyperparameters unchanged. The transfer was evaluated on a 250-day chronologically separated real window. Differences were found between the environments in all groups of indicators studied. Training in the GARCH environment was accompanied by the largest temporal-difference error and the largest gradient norms, but at the same time the smallest discrepancy of state space ranges and the lowest turnover on all three assets. In the Double DQN configuration used with Huber loss function and gradient clipping, no numerical divergence of estimates was observed in 120 runs; at the same time, the positive bias of Q-scores was preserved in all twelve asset–environment combinations. . In 14 of the 120 runs, the trained agent outperformed the corresponding fixed-allocation baseline selected on synthetic validation data. Agents generally held a larger share of capital in the asset when trained in environments with higher drift, although this pattern was not consistent across all assets. The obtained results characterize the engineering interaction between the generative environment and the DQN agent, but do not indicate the superiority of agents over passive deterministic control bases

1. Introduction

The application of Deep Reinforcement Learning (DRL) methods to investment portfolio management problems has evolved over the past decade from a niche research topic to an active, rapidly growing field at the intersection of computer science and quantitative finance: a recent systematic review covers 167 publications on the application of RL to quantitative finance [1]. The reason for this interest is clear: the problem of dynamic capital allocation between assets is naturally formulated as a sequential decision-making problem under uncertainty, precisely the setting for which reinforcement learning is designed.

[1]Corresponding author. Present address: Department of Computer Engineering and Programming, National Technical University "Kharkiv Polytechnic Institute", 2 Kyrpychova St., Kharkiv 61002, Ukraine, Evening Building, Room 309. Tel.: +38057-707-61-65.

E-mail: Yurii.Yurchenko@cs.khpi.edu.ua (Yu.V.Yurchenko), Oleksandr.Zakovorotnyi@khpi.edu.ua (O.Yu. Zakovorotnyi)

The classical apparatus for solving such problems — dynamic programming and the exact Bellman equation — provides a mathematically rigorous, but practically unfeasible path: for a realistic financial state (price history, volatility, current position) the state space is continuous and multidimensional, and the exact tabular solution of the Bellman equation faces the "curse of dimensionality." It is this computational impossibility that motivates the transition from exact methods to approximate ones: a neural network acts as a universal approximator of the value function Q(s,a), and Deep Q-Network (DQN) is a specific, well-studied algorithm for learning such an approximation based on the interaction of the agent with the environment[2]. But here there is a practical obstacle, specific to the financial domain: unlike games or robotics simulators, where the agent can interact with the environment millions of times, direct training of a DQN agent on a live financial market is impossible - it is too expensive, too risky, and historical data of one specific asset over a reasonable period is simply not enough to stably train a neural network. One practical approach is to train the agent in a simulated environment ("sandbox") built on the basis of a stochastic model calibrated to a real asset.

It is here that the central question arises, which shapes the problem of this work: to what extent does the statistical realism of such a simulated environment affect the quality of the resulting agent? Contemporary studies of financial RL emphasize that a simulator that fails to reproduce the properties of the real process provides a distorted representation of the environment. Consequently, the ability of artificially generated data to reproduce stylized market properties is regarded as an important prerequisite for confidence in the resulting policy [3].

The price of a financial asset is a realization of a stochastic process whose future values are unknown, although its statistical regularities can be estimated and reproduced by a model. Such models range from a Wiener process with Gaussian increments, which underlies geometric Brownian motion (GBM), to more complex constructions capable of representing finer statistical effects in price series. Despite its analytical convenience, the Gaussian model does not account for several phenomena that are systematically observed in real return series. Its selection therefore also determines the distributions of states and rewards available to the agent during training.

Two empirical deviations from the Gaussian model occur in many financial return series. First, extreme price movements occur substantially more often than predicted by the normal distribution, indicating increased tail mass, or heavy-tailedness. Second, periods of large and small fluctuations are not randomly scattered over time but form clusters: a turbulent trading day is more likely to be followed by another turbulent day than by a calm one, a phenomenon known as volatility clustering. These and related properties are commonly summarized in the literature as the “stylized facts” of financial time series, whose classical list was formulated by Cont [4]. A key question for the present study is whether these properties remain valid after a quarter century of technological and regulatory change. A recent re-examination of Cont’s eleven stylized facts using modern U.S. market data provides a direct answer: eight of the eleven properties, including both phenomena described above, were strongly confirmed [5].

Geometric Brownian motion cannot structurally reproduce these two properties because it assumes constant volatility and Gaussian increments. An agent trained only in a GBM environment therefore receives homoskedastic experience without volatility clustering or substantial tail mass. To investigate the influence of simulator structure, four calibrated generative environments were compared under the same DQN architecture, state and action spaces, reward function, and hyperparameters:

- a baseline GBM, which models neither of the two stylized facts;
- GARCH(1,1)-t, a widely used econometric stochastic model that represents both stylized facts;
- Minimal Neural SDE, a neural diffusion model with state- and time-dependent drift and diffusion coefficients but no explicit jump component. In the pools generated for DQN training, however, the model did not exhibit substantial autocorrelation of squared returns;
- Neural Jump-Diffusion, a model with a diffusion (Gaussian) component augmented by a jump component governed by a Poisson process.

Agents trained in the four environments are evaluated over a chronologically separated one-year interval that was not used to update either the generator parameters or the DQN weights. The central idea of the study is to systematically analyze the relationship between the choice of a generative stochastic environment as an integral source of training experience and the measurable properties of Double DQN. With the network architecture, state and action spaces, reward function, and hyperparameters held fixed, the study compares training dynamics, the numerical behavior of the $Q(s,a)$ approximation, variation across runs, state-space coverage, and policy-transfer characteristics on real data. The hypothesis to research : changing the environment will produce systematic differences in these properties even when the DQN architecture remains unchanged. The comparison treats each environment as a whole and does not isolate the causal contribution of tail mass, volatility memory, jumps, or neural parameterization. Nor is it assumed that a statistically more complex generator automatically yields a better transferred policy. Main focus is the engineering interaction between simulator design and the DRL system: training stability, the behavior of gradients and the action-value function, state coverage between training and deployment, and the organization of the data-generation pipeline. To achieve this goal:

- implemented and statistically characterized four types of generators of stochastic environments of different complexity classes — GBM, Minimal Neural SDE, GARCH(1,1)-t and Neural Jump-Diffusion;
- designed the architecture of the Double DQN agent with a target network, an experience replay buffer, a Huber loss function and gradient norm trimming;
- conducted a series of numerical experiments with learning process telemetry;
- evaluated the characteristics of the transfer of learned policies (Sim-to-Real transfer) on independent arrays of real time series of different volatility — Nvidia Corporation (NVDA), SPDR S&P 500 ETF Trust (SPY) and The Coca-Cola Company (KO).

2. Methods: Mathematical Formulation of the Deep Q-Network Reinforcement Learning Architecture

The functional framework described in this section is subject to three principles of controlled experimental comparison. First, the agent architecture, state and action space, reward function, hyperparameters, and training procedure remain unchanged for all four generative environments; the controlled factor of the experiment is the type of stochastic transition generator. Therefore, systematic differences between groups of agents are considered to be related to the choice of the generative environment in general, while the variability caused by random initialization of the neural network, ε-greedy exploration, and choice of training trajectories is estimated by repeated runs with different initial seeds, which is in line with the recommendations for reproducible evaluation of DRL algorithms [7]. At the same time, such a comparison does not isolate the causal contribution of each individual property of the environment—heavy tails, volatility clustering, jump component, or neural network parameterization—since each generator is considered as a holistic source of learning experience. Second, the standard, well-studied DQN architecture with a experience replay buffer and a separate target network [2] was chosen as the basis of the agent, supplemented by the Double DQN mechanism designed to reduce systematic overestimation of the value function [6]. This choice is because the subject of the work is the interaction of a fixed agent configuration with different generative environments, and not the search for the best DRL algorithm. Third, the Huber loss function, gradient norm trimming, and periodic target network updates are applied equally in all experiments; therefore, the results obtained characterize the behavior of the full Double DQN configuration in each environment but are not a separate assessment of the causal contribution of each stabilization mechanism.

2.1 Formulation of the Sequential Decision System

The dynamic capital-management problem is formalized as an episodic Markov decision process (MDP), represented by the tuple (S, A, P, R, γ, T), whose elements are defined to directly correspond to the functional structure shown in Fig. 1.

The study uses an MDP approximation in which s_t is the seven-dimensional state representation available to the agent. Internal variables of the generative models, including the GARCH conditional variance and the latent variables of the neural stochastic models, are not provided directly to the agent. Consequently, the sufficiency of s_t for the strict Markov property is not guaranteed, and the implemented environment may also be viewed as a partially observable process. Using the same representation of s_t for every generator nevertheless provides a common information interface for the controlled comparison.

State space: the state S_t at step t is defined as the seven-dimensional vector

$$s_t = (\tau_t, w_t, x_{t-2}^{norm}, x_{t-1}^{norm}, x_t^{norm}, \sigma_t^{norm}, I_t), \quad (1)$$

where $\tau_t = (T-t)/T$ is the normalized fraction of time remaining until the episode horizon; w_t is the actual fraction of capital invested in the risky asset immediately before the decision at step t (thus accounting for drift caused by market movement after the previous rebalancing); $x_{t-2}^{norm}, x_{t-1}^{norm}, x_t^{norm}$ are the three most recent price log returns normalized by the volatility of the training period; σ_t^{norm} is the rolling realized volatility; $I_t \in \{0,1\}$ is the jump indicator ($|x_t| > 3\sigma_t$). The absolute capital level $\ln W_t$ is deliberately excluded from the state: under logarithmic utility, the optimal fraction w^* is independent of the scale of wealth, so including it would only introduce a risk of overfitting to the narrow capital range encountered in the training episodes.

Action space: $A=\{0,1,\ldots,10\}$ is defined as a discrete action space, where each integer index is mapped to a corresponding target portfolio fraction $w'_t = a_t/10 \in [0,1]$.

Transition dynamics: the key distinction between this study and a conventional DQN formulation is the systematic variation of the stochastic component of the transition kernel. In each individual training run, the kernel $P(s_{t+1}|s_t,a_t)$ is formed by combining one of the four calibrated return generators with deterministic allocation and capital-update rules, and it remains fixed throughout the run. Within the comparison for a given asset, the type of generative environment changes across experimental groups, while the Double DQN architecture, state representation, action space, reward function, and hyperparameters remain unchanged. This design enables a controlled comparison of GBM, Minimal Neural SDE, GARCH(1,1)-t, and Neural Jump-Diffusion as integral sources of training experience and makes it possible to establish how environment choice is associated with training dynamics, the behavior of the approximated $Q(s,a)$, state-space coverage, and transferred-policy characteristics. However, the design does not separately identify the causal contribution of heavy tails, volatility clustering, the jump component, or neural parameterization; dedicated ablation experiments would be required for that purpose. The rationale for this comparison is supported by studies demonstrating that the properties of financial RL agents depend on the construction of the simulated environment [3], while heavy tails and volatility clustering, which are reproduced with different accuracy by the investigated generators, remain empirically confirmed stylized properties of financial time series [4], [5].

Reward function: the reward at each step is the logarithmic capital increment net of the transaction fee:

$$r_t = \ln\left(\frac{W_{t+1}}{W_t}\right), \quad (2)$$

where capital W_{t+1} is calculated in two consecutive stages. First, a fee proportional to the actual trade size is deducted from current capital. The trade size is the difference between the new target fraction w'_t, determined by the agent's action, and the fraction w_t actually held by the portfolio immediately before rebalancing, after accounting for drift caused by the previous market movement; it is not the

nominal magnitude of the action itself. The remaining capital then increases or decreases according to the risky-asset price movement and the risk-free rate applied to the residual position. The agent's action does not change the law that generates the next risky-asset return: the stochastic market process remains exogenous to the simulator. However, the action determines the target allocation—the fraction of capital assigned to the risky asset—and, through rebalancing, affects transaction costs, the reward, and the portfolio state carried to the next step. The problem therefore has limited but nonzero sequential dependence. Structurally, it is close to a contextual-bandit model, but does not reduce entirely to one because the outcome of the current action depends on the previous position and the cost of changing it.

Horizon and discounting: each episode has a fixed finite length of T=250 trading days. The discount factor is set to $\gamma=1$ because the evaluation metric is the undiscounted sum $\ln(W_t/W_0)$; choosing $\gamma<1$ would create a discrepancy between the objective optimized by DQN and the reported result. For a finite horizon, $\gamma=1$ does not cause the divergence problem associated with an infinite sum in infinite-horizon tasks.

Value function: the optimal action-value function $Q^*(s,a)$ satisfies the Bellman optimality equation:

$$Q^*(s,a) = \mathrm{E}\left[r_t + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a \right], \tag{3}$$

An exact tabular solution is impractical for a seven-dimensional continuous state space, which motivates approximation by a neural network.

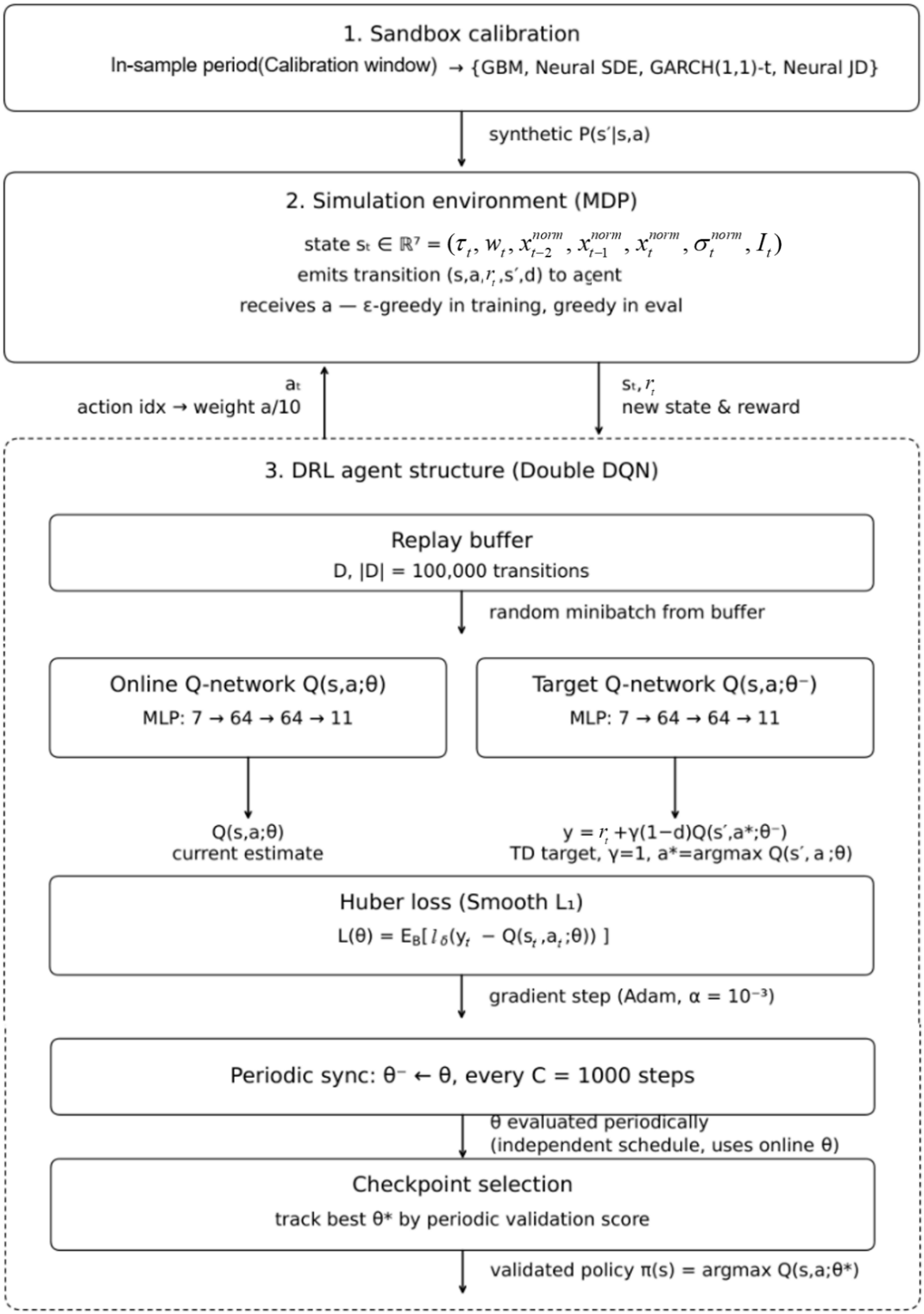


Fig. 1. Architectural diagram of the DRL agent interaction with stochastic transition generators and the Double DQN optimization procedure.

2.2 Double DQN Architecture and Checkpoint Selection

Transitions $(s_t,a_t,r_t,s_{t+1},d_t)$ are stored in the circular replay buffer D ($|D|$=100000). Training uses random minibatches $B\sim D$ ($|B|$=64), which breaks temporal correlations in the sampled experience and stabilizes training of the approximator [2]. The online network $Q(s,a;\theta)$ and target network $Q(s,a;\theta^{-})$ are both MLPs with the architecture 7→64→64→11 and ReLU activations (Fig. 1):

$$h_1 = \mathrm{ReLU}(W_1 s + b_1),\ h_2 = \mathrm{ReLU}(W_2 h_1 + b_2),\ Q(s,\cdot) = W_3 h_2 + b_3, \tag{4}$$

The temporal-difference target y_t is computed according to Double Q-Learning [6]: the online network selects the best action, while the target network only evaluates its value. This separation is intended to reduce the systematic overestimation of action values that characterizes the original DQN algorithm [2]:

$$y_t = r_t + \gamma(1-d_t)\,Q\left(s_{t+1}, \arg\max_{a'} Q(s_{t+1},a';\theta);\ \theta^{-}\right), \tag{5}$$

Huber loss is used instead of quadratic MSE. Beyond the threshold δ, it penalizes the error linearly rather than quadratically, limiting the contribution of large TD errors to the loss and reducing their destabilizing influence:

$$\mathrm{L}(\theta) = \mathrm{E}_B\left[\, l_\delta\left(y_t - Q(s_t,a_t;\theta)\right)\right], \tag{6}$$

where the function is defined piecewise as follows:

$$l_\delta(x) = \begin{cases} \dfrac{1}{2}x^2, & |x| \le \delta \\ \delta\left(|x| - \dfrac{1}{2}\delta\right), & |x| > \delta \end{cases}$$

where δ is the transition threshold between the quadratic and linear penalties (the implementation uses the conventional value δ=1), and y_t is the temporal-difference target defined by Eq. (5). Optimization uses Adam with $\alpha=10^{-3}$; the target-network parameters are synchronized periodically, every C=1000 steps: $\theta^{-}\leftarrow\theta$.

Because DQN learning curves are nonmonotonic and sensitive to the random seed [7], each generative environment is evaluated over multiple independent runs, and the network used for the final evaluation is not the last checkpoint but the checkpoint θ^{*} with the best periodic validation score; peak performance is often attained well before training ends. Since evaluating generalization on data closely related to the training set systematically overestimates performance [8], the synthetic pool is organized into training, validation, and designated Q-diagnostic subsets. Final evaluation is performed on a chronologically separated real-data window of T=250 days as a sim-to-real transfer test.

To test the stated hypothesis, the experimental results are analyzed using several interrelated groups of metrics. Training dynamics and numerical stability are characterized by reward curves computed on a separate validation subset of simulated trajectories, TD loss, gradient norm, clipping frequency, and the behavior of the mean maximum action-value estimate (mean max-Q). The properties of the approximated action-value function are additionally examined using *q_bias* and *mc_q_rmse*. The correspondence of the state space between training and deployment is assessed by the share of test states in which at least one feature lies outside the range observed in the training data. Transferred-policy characteristics are described by the mean fraction of capital invested in the risky asset, portfolio turnover, logarithmic capital growth, and comparison with a fixed action selected in advance on validation data. Taken together, these metrics trace how the choice of generative environment is reflected in DQN training and in the behavior of the resulting policy, without interpreting the results as evidence of a universally profitable trading strategy.

3. Numerical Results

The experimental study comprises 120 complete Double DQN training and validation cycles: 3 classes of numerical time series × 4 types of stochastic transition generators × 10 independent random initialization seeds. The datasets were divided into a chronologically preceding interval for simulator calibration and training (19 years) and a fixed, isolated test window comprising the final 250 trading days, on which policy transfer was evaluated without adaptation.

The experiments were grouped into three regimes of input-series dynamics, distinguished by characteristics measured on the training portion. NVDA had the highest drift (approximately 32% annually) and the highest volatility (0.494) of the three series, with moderate persistence in fluctuation magnitude. SPY had moderate drift (10.3%) and low volatility (0.197), but the strongest and most persistent autocorrelation of squared increments, which remained present even at a lag of twenty steps. KO had the lowest drift (8.9%) and volatility (0.186), and its persistence in fluctuation magnitude decayed markedly faster than that of SPY.

3.1 Statistical Characterization of the Transition Space in the Generative Environments

Table 1 summarizes the statistical properties of the synthetic increments in the pools used for DQN training. Three complementary characteristics are considered: median trajectory-wise excess kurtosis over 250-day windows, lag-one autocorrelation of squared increments, and mean pool drift. These indicators describe distributional shape, temporal dependence in increment magnitudes, and average drift, respectively. The reference excess kurtosis of the real data is calculated over the same window length, providing a comparison at the agent's operating horizon. Together, these descriptors characterize the training experience supplied by each generative environment.

The differences between the environments extend beyond excess kurtosis and temporal dependence. For NVDA, Table 1 reports mean pool drift values of +33.3% for GBM, −2.1% for Minimal Neural SDE, +28.2% for GARCH(1,1)-t, and +7.0% for Neural Jump-Diffusion. Consequently, differences in agent behavior cannot be attributed exclusively to a single statistical property of the simulator. Each generator supplies a distinct distribution of experience to the replay buffer, potentially affecting both the rewards observed during training and the temporal-difference targets used to update the action-value function. The subsequent analysis therefore compares the four generators as complete sources of training experience under a common DQN configuration, rather than isolating the effect of any individual distributional feature.

Table 1. Statistical Properties of the Generated Transition Space in the Training Pools

Asset	Stochastic model (data generator)	Median trajectory-wise excess kurtosis (250-day windows)	Variance memory, ACF(x^2) at lag 1	Mean pool drift, %
NVDA	Reference (real data)	3.12	—	—
	GBM	−0.06	−0.006	+33.3
	Minimal Neural SDE	−0.04	−0.008	−2.1
	GARCH(1,1)-t	+2.46	+0.045	+28.2
	Neural Jump-Diffusion	+2.77	−0.011	+7.0
SPY	Reference (real data)	1.68	—	—
	GBM	−0.06	−0.006	+10.9
	Minimal Neural SDE	+0.10	+0.013	+1.4
	GARCH(1,1)-t	+3.29	+0.150	+10.5
	Neural Jump-Diffusion	+8.65	−0.011	+3.6
KO	Reference (real data)	3.81	—	—
	GBM	−0.06	−0.005	+9.4
	Minimal Neural SDE	−0.01	−0.003	+2.3
	GARCH(1,1)-t	+2.50	+0.062	+8.2
	Neural Jump-Diffusion	+4.94	−0.011	+1.9

GBM and Minimal Neural SDE generate nearly homoskedastic transitions with no memory in squared increments (ACF(x^2)≈0 for all assets) and virtually zero excess kurtosis; thus, they reproduce neither of the two stylized facts of the real series. GARCH(1,1)-t is the only model with pronounced autocorrelation of squared increments, whose strength differs across assets (+0.150 for SPY versus +0.045 for NVDA).

No model exactly reproduces the reference excess kurtosis at the same 250-day horizon, and the direction of the deviation depends on the asset. Neural Jump-Diffusion produced the largest median trajectory-wise excess kurtosis among the four generators for every asset. For SPY and KO, it exceeded the corresponding real-data reference, whereas for NVDA it remained slightly lower: 2.77 versus 3.12.

The reference itself depends on the horizon. Over the full training sample (~4780 days), excess kurtosis is 9.06, 14.00, and 11.55—three to eight times larger than the median over 250-day windows—because rare extreme events may not occur within a short window. Comparison with the full-sample reference would create the misleading impression that all four generators are inadequate, whereas at the agent's operating horizon two models on selected assets exceed, rather than fall short of, the real-data excess kurtosis.

3.2 Training Dynamics, Numerical Stability, and Q-Function Diagnostics

The relative ordering of diagnostic metrics across the generative environments was consistent for all three assets. GARCH(1,1)-t produced the largest temporal-difference loss (TD loss), gradient norms, and gradient-clipping frequencies. By contrast, GBM exhibited the largest mean maximum approximated action value on probe states (mean *max-Q*) and the largest mean positive bias in the internal Q diagnostics (*q_bias*). This repeated pattern indicates persistent differences among the optimization regimes.

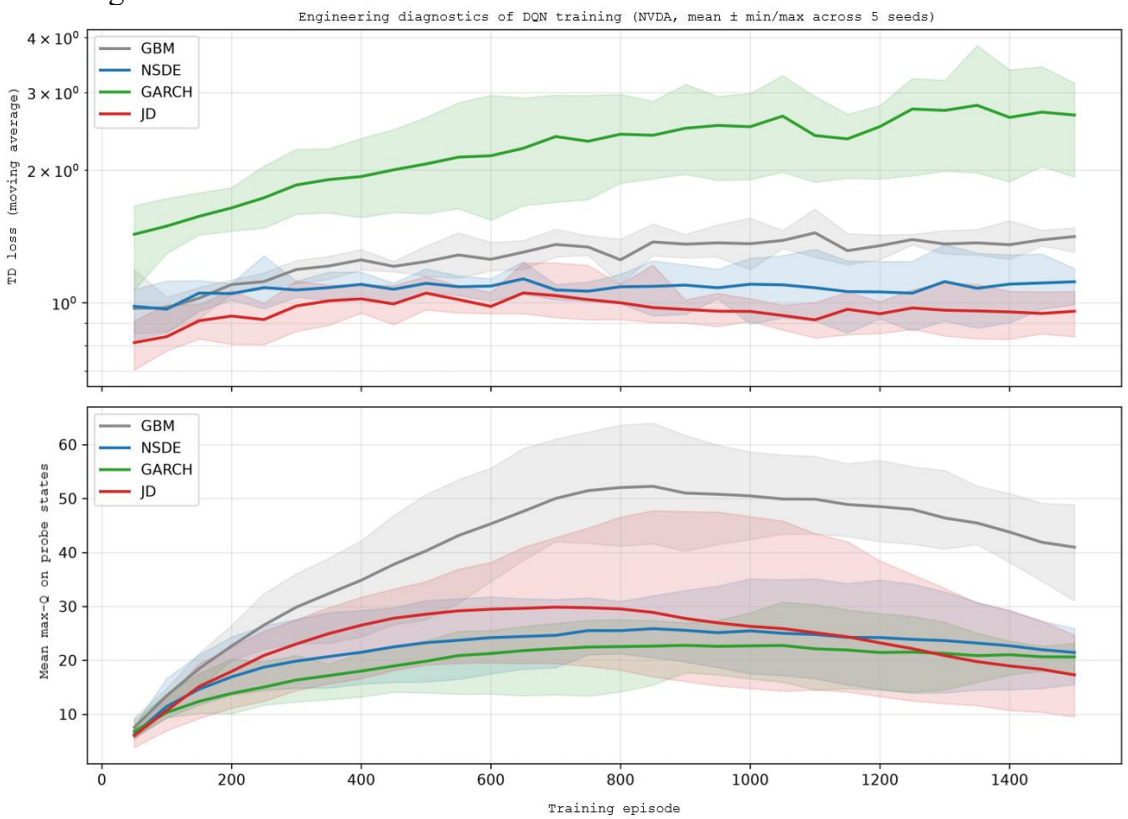


Fig. 2. Training telemetry of the Double DQN optimization process: TD-loss and mean max-Q (NVDA).

Table 2. Telemetry Metrics of Double DQN Training Stability

Asset: generative environment	TD loss (Bellman)[1]	Gradient norm ‖∇θL‖	mean max-Q on probe states	Share of updates with gradient clipping, %	Action-value estimate bias (q_bias)[2]	Monte Carlo action-value RMSE (mc_q_rmse)[2]
NVDA: GBM	1.41 ± 0.11	8.81 ± 1.90	44.0 ± 7.2	14.0	+37.21	49.12
NVDA: Minimal SDE	1.12 ± 0.10	5.28 ± 2.68	22.2 ± 7.0	3.0	+27.72	33.26
NVDA: GARCH(1,1)-t	3.28 ± 1.38	28.65 ± 16.62	24.6 ± 7.6	40.9	+8.49	38.50
NVDA: Jump-Diffusion	0.98 ± 0.08	4.10 ± 1.34	22.1 ± 9.6	0.2	+23.11	29.41
SPY: GBM	0.40 ± 0.03	2.60 ± 0.70	11.5 ± 1.9	0.0	+14.38	18.72
SPY: Minimal SDE	0.17 ± 0.05	1.33 ± 0.65	−0.1 ± 0.3	0.3	+3.85	12.37
SPY: GARCH(1,1)-t	3.32 ± 4.32	44.23 ± 51.66	5.3 ± 1.5	35.3	+3.00	41.36
SPY: Jump-Diffusion	0.22 ± 0.07	1.61 ± 1.07	3.6 ± 0.6	0.4	+2.58	7.74
KO: GBM	0.37 ± 0.06	2.38 ± 1.80	10.3 ± 3.3	0.2	+11.29	16.39
KO: Minimal SDE	0.17 ± 0.01	0.80 ± 0.28	1.8 ± 0.6	0.0	+5.12	7.99
KO: GARCH(1,1)-t	1.67 ± 1.26	13.23 ± 9.71	4.6 ± 1.2	11.8	+6.17	51.02
KO: Jump-Diffusion	0.21 ± 0.04	1.42 ± 0.70	2.1 ± 1.0	0.05	+3.29	7.20

[1]-measured on the network after completion of the 1500th and final training episode. [2]-measured on the restored best checkpoint, i.e., the same network used for the final evaluation.

Table 2 shows that the scale of the temporal-difference error depends on the variance structure of the environment. GARCH(1,1)-t has the highest TD loss for all three assets; for SPY and KO, it exceeds the corresponding GBM value by approximately 4.6–8.3 times. This does not necessarily indicate poorer learning quality, but rather reflects a more complex target-signal distribution. The TD error is the difference between the current estimate $Q(s_t,a_t;\theta)$ and the target y_t recursively constructed by Eq. (5) using the target network. This recursive quantity also depends on the current, not yet fully converged state of the target network, so its variability propagates into optimization. In GARCH, the variance of return x_t is heteroskedastic; through the capital dynamics W_{t+1}, this property is transferred to reward r_t and changes the scale of y_t across volatility regimes. In GBM, reward variance is constant, yielding a statistically more homogeneous target signal. A lower TD loss may therefore characterize an easier approximation problem rather than a more accurate action-value function; its absolute value should not be used as an independent quality measure across environments with different target-signal distributions.

The gradient norm complements TD loss by characterizing the sensitivity of the loss to changes in network parameters. Neural Jump-Diffusion produced some of the lowest mean values: 4.10 for NVDA, 1.61 for SPY, and 1.42 for KO, substantially below the GARCH values (Table 2). This finding is consistent with the expected effect of Huber loss: beyond threshold δ, the error is penalized linearly, limiting the influence of isolated extreme TD targets that are likely in a jump environment. Poisson events may sharply shift the target away from the current network estimate, while the linear branch of Huber loss attenuates their contribution. In GARCH, by contrast, large gradients occurred more systematically because of persistent high-conditional-variance regimes; the corresponding mean gradient norms were 28.65, 44.23, and 13.23 for the three series. The large variation across runs, particularly for SPY, further indicates sensitivity of optimization to the realization of the training process. The norm ‖∇θL‖ is the Euclidean length of the vector of partial derivatives of the loss with respect to network parameters, computed before the Adam step. Two networks may have similar TD losses but different gradient norms because of different local geometry of the loss surface: the first metric describes error magnitude, whereas the second describes sensitivity to weight changes. The implementation clips gradients at $\|\nabla_\theta L\| \le 10$, while telemetry records the pre-clipping norm. For Neural

Jump-Diffusion, only 0.22%, 0.39%, and 0.05% of updates exceeded the threshold, compared with 40.9%, 35.3%, and 11.8% for GARCH. Thus, GARCH generated large-gradient updates much more frequently. Clipping does not alter the pre-clipping norm recorded in telemetry, but reduces the magnitude of the vector passed to the optimizer and thereby prevents an excessive single weight-update step. Numerical stability nevertheless characterizes the complete configuration of Huber loss plus gradient clipping; the isolated contribution of Huber loss cannot be determined without a control experiment using MSE.

The mean maximum approximated action value on probe states (mean max-Q, Table 2) characterizes the numerical scale of the estimates but not their full calibration. Aggregated means ranged from −0.1 to 44.0; none of the 120 training runs exhibited divergence of $\max_a Q(s,a)$ to hundreds or thousands. Numerical boundedness indicates only the absence of divergence and does not guarantee accuracy. The internal Q diagnostic compared $Q(s_0,a)$ with a Monte Carlo estimate of the realized cumulative reward and found positive *q_bias* in all 12 aggregated combinations, ranging from +2.58 for SPY/Jump-Diffusion to +37.21 for NVDA/GBM; at the individual-seed level, the bias was positive in 9–10 of 10 runs. Because the diagnostic subset was not guaranteed to be excluded from subsequent training after validation calls, *q_bias* and *mc_q_rmse* should be interpreted as internal engineering diagnostics rather than independent calibration estimates. The larger GBM bias is generally consistent with its narrower coverage of training states, but does not establish a causal relationship between distribution shift and Q-function overestimation. The conclusion is therefore limited to the presence of systematic positive bias in the current Double DQN implementation, without a claim of out-of-sample calibration.

3.3 Policy Characteristics on the Test Window and Transfer Analysis

Table 3. Characteristics of Transferred DRL Policies on Real Data (Mean over 10 Seeds ± Standard Deviation)

Asset: generative environment	Mean target fraction of capital invested in the asset $\bar{w}$	Mean daily portfolio turnover (magnitude of position change)	Logarithmic capital growth	Annualized Sharpe ratio based on daily log capital increments	Number of wins over the fixed action selected on validation
NVDA: GBM	0.92±0.05	0.060±0.025	+0.187±0.037	0.54±0.09	1/10
NVDA: Minimal SDE	0.52±0.17	0.178±0.075	+0.096±0.082	0.40±0.31	5/10
NVDA: GARCH(1,1)-t	0.92±0.07	0.043±0.033	+0.238±0.049	0.69±0.15	6/10
NVDA: Jump-Diffusion	0.69±0.09	0.153±0.079	+0.140±0.072	0.51±0.24	1/10
SPY: GBM	0.90±0.06	0.064±0.032	+0.149±0.021	1.28±0.14	0/10
SPY: Minimal SDE	0.51±0.23	0.053±0.028	+0.112±0.028	1.65±0.50	0/10
SPY: GARCH(1,1)-t	0.89±0.07	0.028±0.023	+0.156±0.017	1.35±0.10	0/10
SPY: Jump-Diffusion	0.74±0.13	0.084±0.021	+0.134±0.022	1.35±0.31	0/10
KO: GBM	0.87±0.04	0.086±0.035	+0.248±0.025	1.52±0.12	0/10
KO: Minimal SDE	0.55±0.08	0.166±0.035	+0.139±0.028	1.18±0.22	0/10
KO: GARCH(1,1)-t	0.88±0.10	0.040±0.028	+0.245±0.043	1.46±0.19	0/10
KO: Jump-Diffusion	0.65±0.12	0.120±0.048	+0.189±0.057	1.48±0.35	1/10

* The last column reports the number of DQN runs in which logarithmic capital growth over the real-data test window exceeded that of the corresponding baseline strategy with a constant target fraction w of capital invested in the asset. This fraction was selected in advance by maximizing the mean reward over synthetic validation trajectories from the corresponding environment. The selected values of w and the resulting logarithmic capital growth over the real-data test window were as follows : NVDA / GBM i GARCH 1.0+0.2410, NVDA / Minimal SDE 0.4+0.1124, NVDA / JD 0.8+0.2035, SPY / all generative environment 1.0+0.1825, KO all generative environment 1.0+ 0.2890.

The final column of Table 3 compares the learned policy with a simple alternative: a fixed action held constant throughout the evaluation window. The key methodological condition is that this fixed action was not selected retrospectively, but by the same procedure and at the same stage as the network checkpoint: it was the best of the eleven actions $a \in \{0,\ldots,10\}$ on synthetic validation trajectories (Section 2.2) and was subsequently applied unchanged to the real-data evaluation. Thus, both sides of the comparison use the same amount of information available before the beginning of the test window.

Of the 120 independent training cycles, the learned policy outperformed the corresponding fixed action in 14 (11.7%), and the wins were distributed very unevenly. For SPY and KO, validation selected w=1.0 in all four generative environments, and the learned policy won in only one of 80 runs. Both test windows had positive cumulative growth, making full exposure a strong fixed-action baseline. For NVDA, by contrast, the wins were concentrated in two environments: 6 of 10 runs for GARCH(1,1)-t and 5 of 10 for Minimal Neural SDE.

These two cases have different interpretations. For GARCH(1,1)-t, validation selected the same action w=1.0 as in the other environments, so the agent outperformed the highest possible fixed-action threshold (+0.2410). This is consistent with the findings of Sections 3.1 and 3.2: the smallest state-space distribution shift and the lowest systematic action-value bias among the four environments. For Minimal Neural SDE, the threshold was substantially lower (+0.1124) because the nearly zero calibrated drift of this environment (Table 1) caused validation to select a conservative fixed action w≈0.4. The learned policy inherited the same conservatism but was able to compensate for it at least partly through flexibility, changing exposure with the state rather than keeping it constant.

The overall policy-transfer results show that the advantage of the learned approximation over a fixed action is not systematic. In ten of the twelve asset×environment combinations, it is either absent or occurs only in isolated runs, and it becomes persistent only in two combinations for NVDA. This result is consistent with the structure of the implemented environment: the agent's action does not change the exogenous return-generating process, but it affects capital allocation, transaction costs, and the next portfolio state. The advantage of a dynamic policy over a fixed action is therefore possible, but is structurally constrained by weaker sequential dependence than in environments where actions directly alter the law of the next transition. An additional factor is the documented difficulty of financial environments for reinforcement learning, including a low signal-to-noise ratio and a high risk of implicit overfitting to a particular historical period [9].

3.4 Summary of the Numerical Analysis

The experiments show that, with the Double DQN configuration held fixed, the choice of generative environment is associated with substantial differences in TD-error dynamics, gradient norms, clipping frequency, the scale of action-value estimates, state-space coverage, and the behavior of transferred policies. The diagnostic metrics do not induce a single ordering of the environments: GARCH(1,1)-t had the largest TD loss and gradient norms, whereas GBM exhibited the largest mean max-Q. The complexity or instability of a generative environment must therefore be regarded as multidimensional.

State-space diagnostics identified a mismatch between training and real data whose magnitude depended on the asset and generative model. Positive bias in the action-value estimates was observed in all twelve aggregated asset–environment combinations. The association between distribution shift and Q-function bias was broadly consistent in direction, but the exact ordering of the models differed across assets. The results are therefore compatible with an explanation based on Q-network extrapolation beyond the training range, but do not independently establish this causal mechanism.

No numerical divergence was observed in the employed Double DQN configuration with Huber loss and gradient clipping. Because variants without these mechanisms were not examined, the individual contribution of each stabilizer cannot be determined from the available results. The working hypothesis thus received partial empirical support within the investigated metrics: the choice of generative environment is associated with measurable differences in training dynamics, internal Q diagnostics, state coverage, and transferred-policy characteristics. However, no monotonic

relationship of the form “a more complex or statistically realistic generator yields a better transferred policy” was established. The results characterize the engineering interaction between the simulator and the DQN agent rather than the agent’s superiority as a trading system.

4. Discussion

The results show that the generative environment in a deep reinforcement learning task is not a neutral provider of training data, but a component that determines the statistical structure of the replay buffer and, through it, the numerical behavior of Double DQN optimization. The largest TD loss, gradient norms, and clipping frequency were observed in GARCH(1,1)-t, whereas GBM produced lower TD loss but larger mean max-Q and mean positive bias in the internal Q diagnostics. A small TD loss therefore cannot by itself be treated as a universal indicator of better learning; its scale depends on the distribution of TD targets produced by the environment. This finding complements the conclusions of FinRL-Meta [9], which identifies financial RL environment design, data quality, and the low signal-to-noise ratio as independent components of a sound experimental pipeline.

The considerable variation in learning curves and policy characteristics across repeated runs is consistent with Henderson et al. [7] and Agarwal et al. [11], who caution against evaluating DRL algorithms using single runs or point estimates alone. Ten training runs for each asset–environment combination made it possible to characterize both the mean behavior and the dispersion of the results. However, these runs used shared or repeatedly generated pools for the corresponding environment; they therefore primarily characterize the stochasticity of DQN optimization for a given generator realization rather than the full uncertainty associated with simulator calibration. The results should consequently be interpreted as a controlled engineering comparison, not as a definitive universal ranking of the models.

The observed mismatch between state ranges in synthetic training and real deployment confirms the importance of separating training and test environments emphasized by Cobbe et al. [8]. GARCH produced the smallest share of features outside the training ranges for all three assets, but this did not ensure a universal advantage of its policies. State-space coverage is therefore important but insufficient for transfer; the reward distribution, the behavior of the Q(s,a) approximator, partial observability, and the checkpoint-selection procedure also matter. The persistent positive mean q_bias does not contradict the purpose of Double DQN, because the algorithm reduces but does not guarantee the complete elimination of action-value overestimation [6]. In the current implementation, however, the Q-diagnostic subset was not guaranteed to remain excluded from subsequent training after validation calls; *q_bias* and *mc_q_rmse* should therefore be treated as internal research diagnostics rather than independent calibration estimates. The importance of distribution shift and partial observability for transferring RL systems to real-world settings is also consistent with the analysis by Dulac-Arnold et al. [12].

Comparison with fixed actions further showed that the simulator’s influence on training does not imply an automatic advantage of a dynamic policy after transfer. The learned policy outperformed the corresponding fixed-action baseline in 14 of 120 runs, with 11 of these cases occurring for NVDA in the GARCH and Minimal Neural SDE environments. For SPY and KO, the positive trend in the real test window made the maximum constant risky-asset fraction a strong baseline that was rarely exceeded. The working hypothesis therefore received partial empirical support: generative-environment choice was systematically reflected in training telemetry, state coverage, and the structure of the resulting policies, but no monotonic relationship between generator complexity and transfer quality was established. The result characterizes the engineering interaction between the simulator and Double DQN rather than the effectiveness of a ready-to-deploy trading system.

5. Conclusion

This study conducted a controlled investigation of how a generative stochastic environment affects Double DQN training and policy transfer. Four environments—GBM, Minimal Neural SDE,

GARCH(1,1)-t, and Neural Jump-Diffusion—were compared for three financial time series under identical agent architecture, state and action spaces, reward function, optimization procedure, and hyperparameters. A total of 120 training runs were analyzed, allowing the transition generator to be treated as an independent component of the DRL pipeline and its association with training telemetry, state-space coverage, and transferred-policy behavior to be traced.

The results show that the environments produce distinct optimization regimes. GARCH(1,1)-t generated the largest TD loss, gradient norms, and clipping frequency, but also the smallest share of real states outside the training ranges and the lowest mean portfolio turnover for all three assets. GBM produced a simpler transition structure and lower TD loss, but exhibited the largest mean max-Q and the highest mean positive bias in the internal Q diagnostics. No single ordering of the generators was found across all metrics; hence, training-environment complexity should be treated as a multidimensional property.

On the chronologically separated real-data windows, the learned policy outperformed the corresponding fixed action in 14 of 120 runs. This does not demonstrate a systematic advantage of DQN over simple baseline policies, nor was such an advantage the primary objective of the study. The principal conclusion is that the choice of generative environment is associated with reproducible differences in the numerical behavior of Double DQN, the distribution of visited states, and the structure of actions after transfer. At the same time, a generator that is statistically more complex or more realistic according to individual metrics does not necessarily yield a better policy on real data.

Future work should use a fully disjoint Q-diagnostic sample, multiple sequential rolling-origin test windows, independent regeneration of training pools, and ablation experiments that separately evaluate the effects of tail mass, volatility memory, and the jump component. It would also be useful to compare Double DQN with recurrent or distributional DQN variants that can better account for partial observability and uncertainty in the action-value function. These extensions would move the analysis from observed associations between simulator design and agent behavior toward a more precise identification of the mechanisms underlying that interaction.

References

[1] N. Pippas, E.A. Ludvig, C. Turkay, The evolution of reinforcement learning in quantitative finance: a survey, ACM Computing Surveys (2025). https://doi.org/10.1145/3733714.

[2] Mnih V., Kavukcuoglu K., Silver D. et al. Human-level control through deep reinforcement learning. Nature. 2015. Vol. 518, № 7540. P. 529–533. doi: 10.1038/nature14236.

[3] Yao, Z., Li, Z., Thomas, M., & Florescu, I. (2024). Reinforcement Learning in Agent-Based Market Simulation: Unveiling Realistic Stylized Facts and Behavior. 2024 International Joint Conference on Neural Networks (IJCNN). https://doi.org/10.1109/IJCNN60899.2024.10650035

[4] R. Cont, Empirical properties of asset returns: stylized facts and statistical issues, Quantitative Finance 1(2) (2001) 223–236. https://doi.org/10.1080/713665670

[5] E. Ratliff-Crain, C.M. Van Oort, M.T.K. Koehler, B.F. Tivnan, Revisiting Cont's stylized facts for modern stock markets, Quantitative Finance 25(9) (2025) 1343–1373. https://doi.org/10.1080/14697688.2025.2530637.

[6] 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

[7] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep Reinforcement Learning That Matters. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1), 3207–3214. https://doi.org/10.1609/aaai.v32i1.11694

[8] Cobbe, K., Klimov, O., Hesse, C., Kim, T., & Schulman, J. (2019). Quantifying Generalization in Reinforcement Learning. Proceedings of the 36th International Conference on Machine Learning, PMLR 97, 1282–1289. https://proceedings.mlr.press/v97/cobbe19a.html

[9] Liu, X.-Y., Xia, Z., Rui, J., Gao, J., Yang, H., Zhu, M., Wang, C.D., Wang, Z., & Guo, J. (2022). FinRL-Meta: Market Environments and Benchmarks for Data-Driven Financial Reinforcement Learning. Advances in Neural Information Processing Systems, 35, 1835–1849. https://doi.org/10.52202/068431-0134

[10] Yurchenko, Yu. V., & Zakovorotnyi, O. Yu. (2026). Stokhastychne modeliuvannia finansovykh chasovykh riadiv: vid heometrychnoho brounivskoho rukhu do neironnoi dyfuziino-strybkovoi modeli [Stochastic modeling of

financial time series: from geometric Brownian motion to a neural diffusion-jump model]. In M. Hladka, O. Bondarenko, M. Kostikov, & I. Borysenko (Eds.), Informatsiini systemy ta tekhnolohii: rezultaty i perspektyvy (IST 2026): materialy 3-i Mizhnar. nauk.-prakt. konf. (pp. 397–401). FIT KNUTSH. https://repository.kpi.kharkov.ua/entities/publication/0c1dc1ed-9217-42f6-b8a8-64e417d910ef

[11] Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., & Bellemare, M. G. (2021). Deep Reinforcement Learning at the Edge of the Statistical Precipice. Advances in Neural Information Processing Systems, 34, 29304–29320. Official NeurIPS publication. https://doi.org/10.48550/arXiv.2108.13264

[12] Dulac-Arnold, G., Levine, N., Mankowitz, D. J., et al. (2021). Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning, 110, 2419–2468. https://doi.org/10.1007/s10994-021-05961-4

JMMME
JOURNAL OF MATHEMATICAL MODELS AND METHODS IN ENGINEERING

CONTENTS

00010	**Comparing Stochastic Training Environments for Deep Reinforcement Learning with Deep Q-network: Learning Stability and Transfer to Real Data** Yuriy Yurchenko, Dr. Olexandr Zakovorotnyi (Author)	4–18
00011	**Optimization of Irrigation Systems and Ecological Assessment: A Case Study of Andijan Region, with a Forecast to 2032** Ibrohimjon Abdurahmonov (Author)	19–28
00012	**Evaluation of Cement Production for Continuous and Discrete Flow Systems by Applying a Discrete Event Simulation Approach** Wan Laailatul, Mohd Kamal Nawawi, Nur Anis Hannan Md Shahid (Author)	29–35
00013	**Optimization of Heat Transfer in MHD Mixed Convection Hybrid Nanofluid Flow Over a Shrinking Plate Using Response Surface Methodology** NORIHAN MD ARIFIN, Nor Ain Azeany Mohd Nasir, Nur Fatin Irwani Muhammad Fadzil, Rusya Iryanti Yahya, Nur Syahirah Wahid, Mohd Syazwan Mohamad Anuar (Author)	36–65
00014	**An Automatic Quadrature Scheme for Bounded and Unbounded Weighted Hypersingular Integrals on the Variable Interval** Zainidin Eshkuvatov (Author)	66–80
00015	**A boundary value problem with a conormal derivative and a nonlocal integral condition for a degenerate elliptic equation of the second kind** Akmaljon Abdullaev (Author)	81–97