Trading with Noise and Doubt: DDQN and Noisy DQN in Financial Markets

Dale Ren Matthew Jeung Shawn Meng
University of Chicago
{daleyren, mkjeung, shawnmeng88}@uchicago.edu

May 29, 2025

Abstract

We explore the application of Double DQN and Noisy DQN reinforcement learning algorithms across three distinct financial domains: equities (AAPL), cryptocurrencies (BTC), and prediction markets (Kalshi). Each market presents unique challenges, including volatility, non-stationarity, and sparse reward signals. We design a unified trading environment and evaluate the robustness, adaptability, and performance of each agent under these conditions. Our findings highlight the importance of state representation, exploration strategies, and market-specific adaptations in deploying RL-based trading agents.

1 Introduction

Reinforcement learning (RL) offers a compelling framework for financial decision-making due to its ability to operate in environments defined by sequential interactions, feedback-based rewards, and evolving dynamics. Unlike traditional supervised learning models that rely on labeled datasets and assume static input-output relationships, RL agents learn policies that adapt over time through trial and error—an approach that mirrors the ongoing decision processes in trading. Financial markets, however, pose unique challenges: they are noisy, partially observable, and often feature delayed or sparse rewards. Moreover, the environment is non-stationary, meaning past patterns may not repeat in the future. These complexities make financial markets a natural but difficult testbed for RL. Despite these challenges, RL methods hold the promise of discovering adaptable, self-improving strategies that can generalize beyond historical regimes. In this study, we focus on applying RL to three distinct market types:

- **Equity markets (AAPL)**: These are relatively stable and exhibit longer-term price structure, making them suitable for initial benchmarking and controlled experimentation.
- **Cryptocurrency markets (BTC)**: Characterized by high volatility and regime shifts, these markets test an agent's ability to learn under extreme uncertainty and adapt to abrupt changes.
- **Event-driven prediction markets (Kalshi)**: These markets are sparse in reward feedback and have a more discrete outcome structure. They introduce challenges like short episode lengths, rapidly shifting probabilities, and multiple concurrent submarkets—mirroring partially observable or multi-task RL problems.

These markets were deliberately chosen to cover a spectrum of *data learnability*: from smooth, trend-driven equity time series to sporadic, sparse, and irregular Kalshi event streams. This enables us to evaluate each algorithm's robustness across a range of information structures, reward sparsity levels, and volatility regimes. To tackle these environments, we examine two deep Q-learning architectures:

- **DDQN** addresses overestimation bias common in volatile environments by decoupling action selection and evaluation—a critical feature when misestimation can lead to costly trades.[12]
- **Noisy DQN** replaces external ε-greedy exploration with parameterized noise in the network itself, enabling the agent to learn structured, state-dependent exploration strategies. This is particularly valuable in markets with sparse rewards (like Kalshi), where random exploration can be ineffective.[4]

Together, these methods allow us to study the interplay between exploration strategies, noise robustness, and state representation under different market conditions. Our goal is to assess the feasibility and effectiveness of RL agents in each domain and offer insight into the algorithmic choices most suited for real-world trading applications.

2 Background and Related Work

Reinforcement learning (RL) has emerged as a powerful framework for sequential decision-making problems, including algorithmic trading. In this section, we briefly review foundational RL algorithms—particularly Q-learning and its deep variants—and highlight existing work applying these methods to financial markets. We then describe how Double DQN and Noisy DQN address specific limitations of traditional deep Q-learning in high-noise, sparse-reward, and non-stationary environments.

2.1 Deep Q-Learning in Finance

Deep Q-Networks (DQN) extend Q-learning by approximating the optimal action-value function $Q^*(s, a)$ using deep neural networks. To address instability from non-linear function approximation, DQN introduces experience replay, target networks, and mini-batch updates. The Q-values are updated to minimize the temporal difference error:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s,a) \right) \tag{1}$$

where θ and θ^- denote the online and target network parameters, respectively [9]. While effective, DQN can struggle with financial environments due to non-stationarity, sparse rewards, and overfitting. Recent work improves robustness via noise filtering [10], sentiment integration [11], and architectural enhancements like wavelet transforms and attention mechanisms [17].

2.2 Double DQN

While DQN have achieved success in sequential decision-making tasks, they are prone to fundamental flaw: overestimation bias. This issue arises due to the use of the max operator in the Q-learning target, which tends to select high-value estimates even when they result from random noise. Formally, the standard DQN update target is defined as:

$$y^{\text{DQN}} = r + \gamma \max_{a'} Q(s', a'; \theta)$$

Within this framework, the same Q-network is used both to select the next action and to evaluate its value. Since the max operator tends to prefer actions with high estimated values, any random overestimation in

$Q(s', a')$ is reinforced. This leads to a positive bias in the target and consequently causes the value function to drift away from its true expectations:

$$\mathbb{E}\left[\max_{a'} Q(s', a')\right] \geq \max_{a'} \mathbb{E}\left[Q(s', a')\right] \tag{3}$$

To address this, van Hasselt et al. (2016) introduced Double DQN (DDQN), which decouples action selection and evaluation:

$$y^{\text{DDQN}} = r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-) \tag{4}$$

This formulation preserves the stability benefits of a target network while reducing the overoptimistic estimates produced by standard DQN. Since the evaluation is perforce by θ^-, which is held fixed for several iterations, the learned values are less sensitive to transient fluctuations in the action-value estimates [12]. By reducing the upward bias in the Q-values, DDQN produces more accurate targets, stabilizes training, and facilitates more robust convergence in stochastic environments. This is particularly valuable in financial domains, where noisy signals and delayed rewards can easily amplify the drawbacks of standard Q-learning updates.

2.3 Noisy DQN

A key challenge in reinforcement learning is balancing exploration and exploitation. Traditional DQNs typically use an ε-greedy policy to encourage exploration, where the agent chooses a random action with probability ε and the greedy action otherwise. Although simple, this strategy can result in inefficient exploration, especially in environments with few rewards or high uncertainty. Noisy DQN addresses this by replacing fixed exploration schedules with learnable stochasticity [4]. Instead of explicitly choosing random actions, Noisy DQN injects parametric noise into the weights of the neural network, resulting in inherently randomized policies:

$$y = (\mu_w + \sigma_w \odot \varepsilon_w)x + \mu_b + \sigma_b \odot \varepsilon_b \tag{5}$$

μ and σ are learnable parameters, and ε is sampled noise. This formulation allows the agent to learn the scale and structure of its own exploration behavior during training, rather than relying on heuristics such as decaying ε. By integrating noise directly into the function approximator, Noisy DQN enables state- and action-dependent exploration that evolves with training. Unlike ε-greedy methods, where randomness is applied externally and uniformly across the state space, Noisy DQN introduces learnable stochasticity at the parameter level. This allows the agent to explore more effectively in regions of the state space where it is uncertain, while behavior is more deterministic in well-understood situations. The magnitude of exploration is no long government by a manually tuned decay schedule, rather it is optimized jointly with the policy itself. Similar to DDQNs, this mechanism is particularly valuable in domains such as financial markets, where the environment is non-stationary, and rewards are often sparse, noisy, or delayed. In such settings, naive random exploration may fail to uncover profitable policies, especially when suboptimal actions carry significant downside risk. Noisy DQN, by contrast, promotes directed exploration: adapting the agent's behavior to reflect the uncertainty in its value estimates. This helps reduce variance in policy updates, improves convergences stability, and allows the agent to identify more robust strategies.

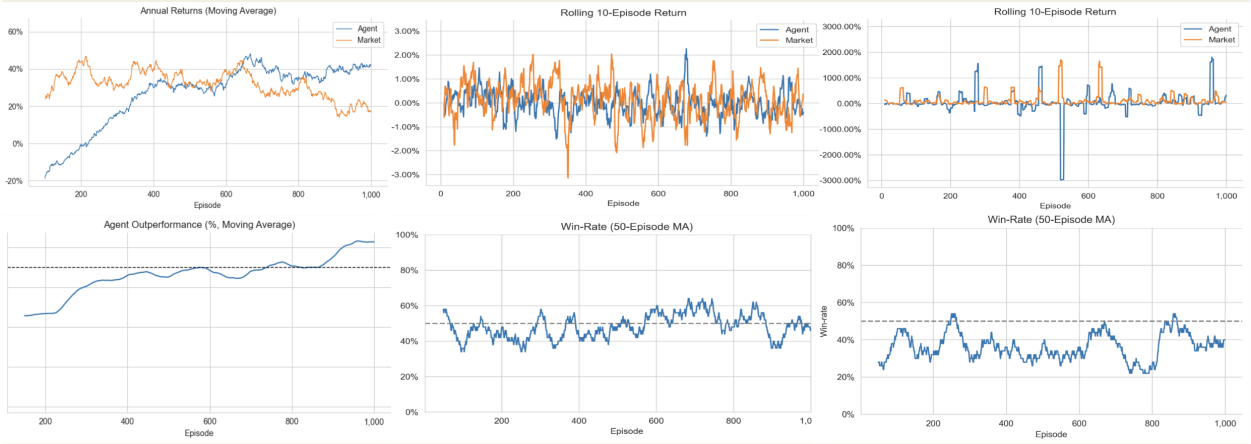

Figure 1: Training of three datasets over 1000 episodes. Graphs display the moving average of annual returns and the agent's performance relative to market movement.

3 Methodology

We evaluate two deep reinforcement learning algorithms—Double DQN and Noisy DQN—designed to address challenges in financial markets. Double DQN reduces overestimation bias through decoupled action selection and evaluation, while Noisy DQN enhances exploration via learnable parameter noise. Below, we describe the network architectures, training procedures, and environment setups.

3.1 Network Architecture and Training

Both the DDQN and Noisy DQN agents use deep neural networks to approximate $Q(s, a)$ for discrete trading actions: Buy, Hold, or Sell. Each network consists of two fully connected hidden layers with 256 units and ReLU activations. The DDQN employs standard dense layers and an ε-greedy exploration strategy, where ε decays linearly and then exponentially. It maintains separate online and target networks, updated via stochastic gradient descent and a soft update mechanism controlled by τ. Noisy DQN replaces dense layers with NoisyLinear layers, introducing trainable stochasticity (see Equation (5)). This enables adaptive exploration without explicit scheduling. Noise is resampled at each forward pass, during both training and action selection. Both agents use an experience replay buffer to decorrelate transitions and improve sample efficiency. The temporal difference (TD) loss is:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(y^{\text{target}} - Q(s, a; \theta)\right)^2\right] \tag{6}$$

where

$$y^{\text{target}} = \begin{cases} r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-) & \text{(Double DQN)} \\ r + \gamma \max_{a'} Q(s', a'; \theta^-) & \text{(Noisy DQN)} \end{cases} \tag{7}$$

Optimization is performed using Adam with ℓ_2 regularization to prevent overfitting in noisy or volatile settings.

3.2 Environment Design

We model a simplified trading environment where the agent can take one of three discrete actions at each time step:

$$A = \{\text{buy},\ \text{sell},\ \text{hold}\}$$

This compact action space allows the agent to incrementally adjust its position by buying or selling one unit of the asset or choosing to hold. Such discrete action frameworks are widely used in reinforcement learning for financial trading [3, 11], as they strike a balance between expressivity and tractability. The state space captures market dynamics using a mix of raw price data and technical indicators, consistent with prior work in financial RL [2, 16, 11]. Each state is represented as a feature vector:

$S = \{\text{open, close, high, low, volume, returns, ret}_2\text{, ret}_5\text{, ret}_{10}\text{, ret}_{21}\text{, RSI, MACD, ATR, Stoch, UltOsc}\}$

This includes:

- **OHLCV**: Core market data reflecting price and volume.
- **Returns (ret$_n$)**: Short- and medium-term momentum (lags of 2, 5, 10, 21).
- **Indicators**: RSI [13], MACD [1], ATR [14], Stoch, and UltOsc [15] provide signals on momentum, trend strength, and volatility.

These features were selected for their widespread use in quantitative trading and empirical support in RL-based financial applications. Studies such as [2] demonstrate that combining raw prices with technical indicators improves both convergence speed and final policy quality. Our reward function computes profit and loss (PnL) at each time step based on the agent's position and market returns:

$$reward = \text{pos}_0 \cdot \text{market\_return} - \text{costs}_{\max(0,\text{step}-1)} \tag{8}$$

This setup rewards profitable trades while penalizing overtrading through execution costs. While effective for training, it simplifies real-world complexities like bid-ask spreads, slippage, and multi-outcome contracts, which future work should address for live deployment.

4 Datasets and Market Context

We selected three distinct markets to evaluate the generalizability of our reinforcement learning agents: an equity market (AAPL), a cryptocurrency market (BTC-USD), and an event-based prediction market (Kalshi). Equity markets have been widely studied in deep reinforcement learning literature, with numerous studies validating the effectiveness of DQNs and their variants in capturing price dynamics and generating trading signals [3, 16]. Our use of AAPL allows us to replicate and benchmark against prior work. To extend beyond traditional financial assets, we applied our models to BTC-USD, which introduces additional challenges such as higher volatility, noise, and thinner liquidity during off-peak hours. These characteristics test the stability and robustness of learned trading policies. Finally, we evaluated our agents on Kalshi, a novel event-driven market offering binary contracts on real-world outcomes (e.g., inflation rates, elections). Kalshi presents a structurally different reward landscape—often sparse and discontinuous—making it a meaningful domain for assessing the adaptability of deep RL agents to alternative financial settings. This progression from equities to crypto to prediction markets serves to stress-test exploration strategies and value estimation under varying market regimes.

- **Equity Market: AAPL**. For our equity market, we selected AAPL, which has daily candles and relatively stable trends over time. We selected AAPL as previous DQN and DDQN models have proven to be successful.
- **Cryptocurrency Market: BTC**. Bitcoin features minute-level candles. The BTC market is highly volatile and prone to regime-dependent swings, presenting a harder challenge than AAPL.

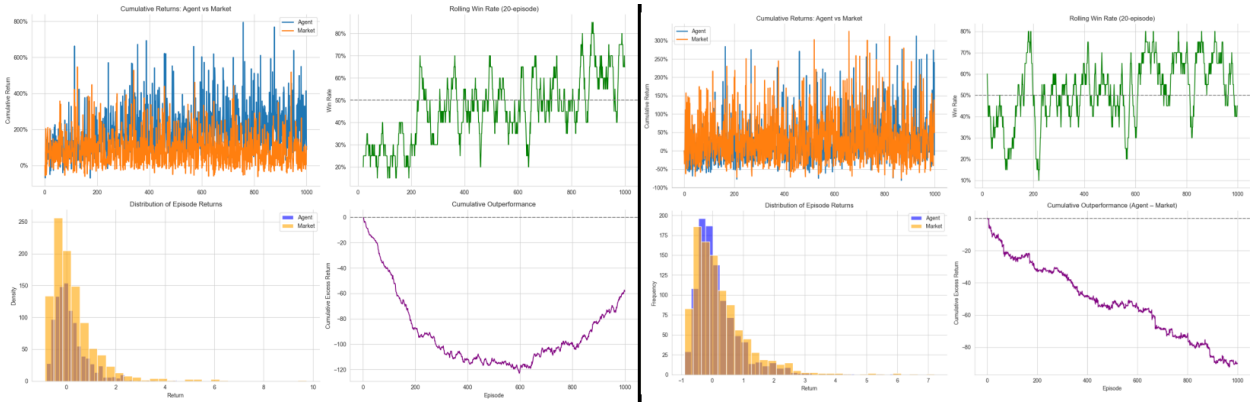

Figure 2: Left: Results from our DDQN agent trading AAPL. Our agent shows positive returns over time. Right: Results from our Noisy DQN agent trading AAPL. Our agent exhibits poorer performance, with significant losses despite a relatively even win rate (per time step) compared to the market.

- **Event-Based Market: Kalshi**. Kalshi is structured differently from AAPL and BTC, as a prediction and event-based market. Bettors can buy or sell YES/NO markets at a price ranging from 1-99 cents, which implies the odds of the YES/NO market. We tested our algorithm on two common Kalshi markets: 1) predicting Bitcoin's range at hourly intervals, and 2) predicting the outcome of NBA games. These markets are highly event-driven and see large swings in buy/sell prices. Further, any given event will feature five or more individual YES/NO markets, creating a more complex environment for our models.

5 Experimental Design

We split each dataset into an 80-20 train/test split, reserving the most recent 20% of the time series for out-of-sample evaluation. This temporal holdout ensures no future information leaks into training. For each market, we tuned hyperparameters using a coarse grid search over learning rates, batch sizes, and training iterations, selecting parameters that offered both learning stability and strong validation returns. Each model was evaluated across 100 trading episodes, using fixed initial capital and market data windows. We report three key metrics: (1) cumulative return, measured as net asset value (NAV) relative to starting capital and (2) win rate, defined as the percentage of episodes where the agent outperformed the market benchmark. Noisy DQN was excluded from Kalshi experiments due to the market's binary payoff structure and sparse feedback, which hindered stable training. Future work may explore reward shaping or alternative architectures to address these challenges.

6 Results and Analysis

6.1 Performance on AAPL

Figure 2 shows that the DDQN agent outperformed Noisy DQN on the AAPL trading task, achieving higher and more consistent returns. This likely stems from DDQN's decoupled action selection and evaluation, which reduces overestimation bias [12], and from hyperparameters tuned based on prior work [6]. In contrast, Noisy DQN adds trainable noise for exploration [4], removing the need for ε-decay but increasing gradient variance. Without careful tuning of noise scale and learning rate, this can hinder convergence.

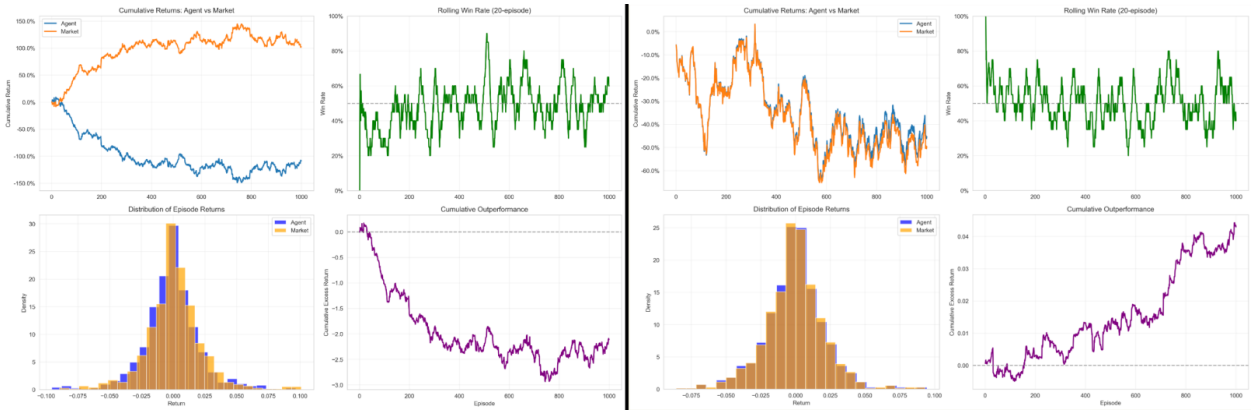

Figure 3: Left: Results from our DDQN agent trading Bitcoin after 1k iterations of training. Our agent shows negative returns over time. Right: Results from our DDQN agent trading Bitcoin after 2k iterations of trading. Our agent exhibits much better performance, learning to mirror the market.

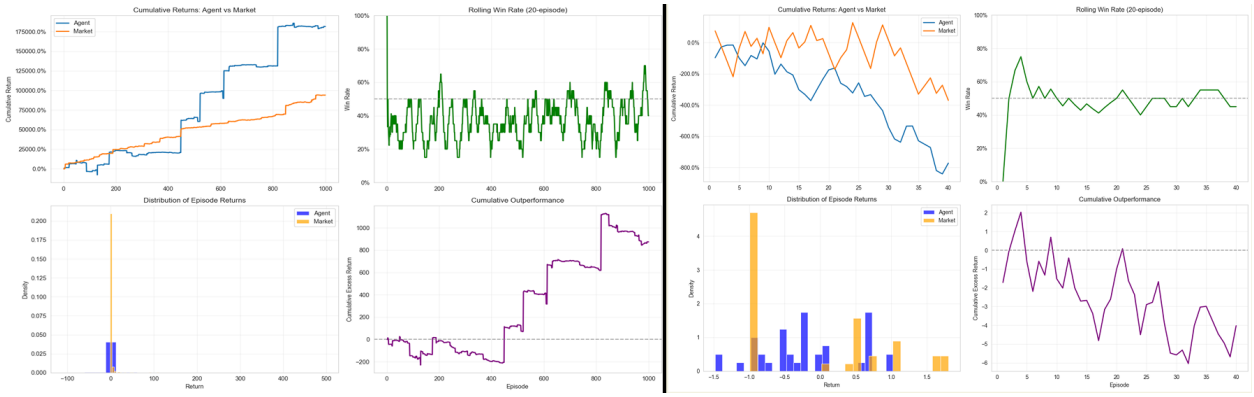

Figure 4: Left: Early results from predicting hourly Bitcoin price. Positive results are misleading due to lookahead bias. Right: Results from predicting the NBA game markets. Our agent consistently lags behind the market in our test set, likely due to a lack of information on real-time events.

These results highlight DDQN's robustness in stable, well-conditioned markets, though further tuning may improve Noisy DQN's performance.

6.2 Performance on BTC

Figure 3 shows that the DDQN agent achieved steady improvement in cumulative return on the BTC-USD dataset, indicating increasing alignment with market trends. However, the agent displayed a strong preference for the *buy* action, suggesting overfitting or exploitation of short-term trends. This imbalance is common in value-based RL when Q-values are biased due to temporal correlations or unregularized rewards [5]. While DDQN reduces overestimation bias, it does not fully address volatility or regime shifts—key challenges in crypto markets. Future work should explore volatility-aware features, regime-switching models, or ensemble methods [7]. Nonetheless, the agent's positive test performance supports DDQN's viability for crypto trading, with robustness and action balance as priorities for improvement.

6.3 Performance on Kalshi

Several challenges emerged from our attempts to apply DDQNs to Kalshi. Our initial assumption was to trade on the most liquid markets. We initially encountered positive results, outperforming the market. However, we quickly realized that training on only the top markets led to look-ahead bias, as the agent learned to only buy. When we randomly sampled the market from each event, we found much more realistic results. Our agent consistently lagged behind the market, occasionally losing at a massive scale due to large swings in prices. We theorize that the very noisy data, short episode length, and lack of supplementary information contribute to the agent's struggles. Future work should explore how to handle event-driven spikes and sparse signals.

6.4 Cross-Market Comparison

Both Noisy DQN and DDQN models performed significantly better on datasets characterized by consistent directional trends, lower volatility, and longer time horizons. These conditions provided more stable reward signals and reduced the likelihood of Q-value overestimation due to short-term noise. In contrast, environments with high-frequency data, such as minute-level candlesticks, proved substantially more challenging. This was particularly evident in the Kalshi dataset, where large price swings could occur within a single hour, making it difficult for agents to generalize across regimes or anticipate reversals. Across our experiments, DDQN consistently outperformed Noisy DQN on both AAPL and Bitcoin markets. While this may partially reflect suboptimal hyperparameter settings for Noisy DQN, it also suggests that deterministic value updates and decoupled action evaluation may offer more reliable learning under the types of reward structures seen in equity and cryptocurrency markets. Nevertheless, Noisy DQN demonstrated reasonable performance on longer-horizon, trend-following tasks, suggesting that its exploratory advantages may be more beneficial in environments with sparse or delayed rewards. Overall, our results indicate that conventional markets with smoother dynamics remain the most tractable for value-based deep RL agents, though we did observe some promising results from applying these methods to cryptocurrency trading.

7 Discussion

In this section, we highlight key challenges and design considerations uncovered during our experiments. These insights reflect both the strengths and limitations of deep reinforcement learning in financial domains, and serve to guide future efforts in environment design, agent architecture, and evaluation methodology.

Effectiveness of Noise-Based Exploration vs. Epsilon Decay

In our experiments, epsilon-greedy exploration with linear or exponential decay converged to a more effective policy than noise-based exploration. Given the relatively short training horizon (1,000–2,000 iterations), Noisy DQN likely lacked sufficient time to fully leverage its stochastic exploration. This aligns with prior observations that Noisy DQN benefits from longer training regimes due to its reliance on parameter noise convergence.

Overfitting in Low-Data Markets (Kalshi)

A key cautionary result emerged from training on Kalshi data: overfitting to high-liquidity market conditions. The agent learned to overbuy, which yielded favorable returns on a narrow test set but would likely underperform or fail catastrophically in live deployment. This illustrates the danger of lookahead bias and the importance of robust generalization, especially when training on event-based markets with sparse or irregular historical data.

Covariate Shift Between Training and Testing Periods

When evaluating performance on the Bitcoin dataset, we noticed a covariate shift between training and testing sets. The 80-20 split unintentionally placed a major bullish breakout in the testing set, creating a mismatch in return distributions and compromising performance comparability. Future work should consider time-based cross-validation or stratified temporal splits to mitigate such risks.

Informational Gaps in the State Space

In event-driven markets such as Kalshi NBA predictions and in high-frequency Bitcoin environments, our agents often lagged behind price movements. This may be attributed to informational gaps in the state representation. Future implementations should explore incorporating external features such as live news, odds, or sentiment indicators into the state vector [8]. Feature engineering and dynamic state augmentation may significantly improve decision latency and reaction precision.

Augmented Action/State Spaces and Reward Shaping

Our simplified action space—limited to buying or selling one unit—facilitated early experimentation but may have constrained long-term performance. A more expressive action space should allow for dynamic volume control, portfolio-level decisions, and multi-market selection. For Kalshi, the agent should choose between YES/NO contracts and span across multiple markets under a single event. In parallel, reward shaping mechanisms beyond instantaneous profit/loss—such as penalizing volatility or rewarding drawdown minimization—should be explored to better align learning signals with long-term trading objectives.

8 Conclusion

Through our case study of DDQN and Noisy DQN agents in equity, cryptocurrency, and event-based markets, we identified key challenges in applying deep reinforcement learning to real-world financial data. Our experiments revealed the importance of model architecture, training horizon, data frequency, and state representation in shaping agent performance. In particular, we propose several directions for future research:

- **Expanded state spaces** incorporating contextual signals such as news sentiment, macroeconomic indicators, or order book depth.
- **Portfolio-aware action layers** enabling dynamic position sizing, risk management, and multi-asset decision-making.
- **Online learning and long-horizon backtesting** to evaluate generalization under regime shifts and market non-stationarity.

More broadly, our findings further reveal the difficulty of training deep RL agents under noisy, sparse, or non-stationary conditions—especially in emerging markets like Kalshi. Nonetheless, with appropriate modeling choices and sufficient data, value-based agents such as DDQN and Noisy DQN show strong promise for learning profitable trading strategies. We hope this work contributes to the growing dialogue on robust reinforcement learning in financial systems.

References

[1] Gerald Appel. *Technical Analysis: Power Tools for Active Investors*. FT Press, 2005.

[2] Subhadeep Chakraborty and Alexander Borowiec. Feature engineering in deep reinforcement learning for financial trading. In *ICAIF*, 2022.

[3] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2016.

[4] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Rémi Munos, Demis Hassabis, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2018.

[5] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[6] Stefan Jansen. *Machine Learning for Algorithmic Trading*. Packt Publishing, Birmingham, UK, second edition, 2020. Expert Insight Series.

[7] Xiaoqian Lan, Yan Liu, and Zhenyu Zheng. Ensemble q-learning for robust financial trading under market regimes. *Expert Systems with Applications*, 212:118682, 2023.

[8] Yan Li, Chen Chen, Jun Zhang, Fangxing Yang, Jianhui Wang, and Fangxing Qiu. Reinforcement learning for smart energy management: A review of challenges and opportunities. *Energy Systems*, 13:3–49, 2022.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[10] Jiwon Park, Sanghyun Lee, and Yuna Kim. A deep q-network action instance selection (dais) system for stock trading in noisy environments. *Expert Systems with Applications*, 237:120321, 2024.

[11] Bahodir Sattarov and Jae-Young Choi. Multi-level deep q-networks for bitcoin trading with sentiment-aware signal integration. pages 13875–13882, 2024.

[12] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. 30(1), 2016.

[13] J. Wells. *Relative Strength Index*. Technical Analysis of Stocks & Commodities, 1978.

[14] Welles Wilder. *New concepts in technical trading systems*. Trend Research, 1978.

[15] Larry Williams. Ultimate oscillator. *Technical Analysis of Stocks & Commodities*, 1985.

[16] Yiming Zhang et al. Deeptrader: A reinforcement learning framework for financial portfolio management. *Expert Systems with Applications*, 157:113477, 2020.

[17] Yifan Zhao, Jing Chen, and Xiaodong Wang. Composite multi-role deep q-networks for adaptive stock trading. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):1852–1865, 2024.