Portfolio Optimization: A Comparison of Deep Q-Learning and Actor-Critic Methods

Krishang Karir KKARIR@UWO.CA
Western University
London, Ontario, Canada

Suharsh Sandhu SSAND3@UWO.CA
Western University
London, Ontario, Canada

Bassam Syed BSYED5@UWO.CA
Western University
London, Ontario, Canada

Aditya Manickam Arumugam AARUMUG6@UWO.CA
Western University
London, Ontario, Canada

Asif Ihtemadul Haque AHAQUE62@UWO.CA
Western University
London, Ontario, Canada

Abstract

We compare two reinforcement-learning approaches—ensemble Deep Q-Learning (DQL) and an Actor-Critic (AC) policy—for dynamic portfolio allocation on two distinct universes: 5 large-cap tech stocks (AAPL, MSFT, GOOGL, SBUX, TSLA) and 5 high-volatility "meme" stocks (GME, AMC, SPCE, NVAX, NOK) over 2014–2023 and 2017-2014, respectively. States encode portfolio weights plus simple technical indicators; actions are constrained daily rebalancings, between 1-5% in Rewards combine multi-step log returns with rolling Sharpe-ratio and drawdown penalties. On out-of-sample 2023 data, both RL agents outperform a 1/N buy-and-hold benchmark.

1 Introduction

1.1 The Problem

Portfolio optimization—allocating assets to maximize return and control risk—remains challenging in modern, volatile markets. Classical mean–variance methods assume stationary returns and continuous rebalancing, which fails when:

- Market regimes shift abruptly (e.g. COVID-19, 2020), breaking diversification assumptions.
- Trading is subject to discrete lot sizes, minimum trades and liquidity limits.
- Decisions ignored temporal dependencies and compounding in multi-period horizons.

1.2 Limitations of Existing Approaches

Mean–variance optimization is extremely sensitive to input estimates, while stochastic programming becomes intractable beyond small portfolios with realistic constraints. Even ML-based return forecasts typically decouple prediction from portfolio construction, missing the feedback loop between actions and market outcomes.

1.3 Our Solution

We cast portfolio management as a Markov decision process (MDP) and compare two RL paradigms:

- **Deep Q-Learning (DQL)** for discrete 1–5% rebalancing moves.
- **Actor-Critic (PPO)** with continuous weight adjustments and entropy regularization.

States combine discrete allocations and simple technical indicators; actions encode realistic trading constraints; and rewards blend multi-step log returns with rolling Sharpe and drawdown penalties. This framework adapts to changing markets, respects trading limits, and jointly optimizes growth, risk, and feasibility.

2 Previous Work

Recent work in RL-based portfolio optimization highlights three key innovations: adaptive state encoding, realistic action constraints, and multi-objective rewards. (Jiang et al., 2017) first applied Deep Q-Networks to rebalancing but ignored trade-size and liquidity limits; (Deng et al., 2016) introduced Actor-Critic methods for regime adaptation yet omitted concentration risks; (Zhang et al., 2020) enriched state representations with technical and macro factors for better regime detection; and (Almahdi & Yang, 2017) incorporated risk-sensitive objectives balancing return and drawdown. Together, these studies inform our design of a richly featured state, constrained action space, and composite reward function.

3 Model Description: Deep Q-Learning for Portfolio Rebalancing

3.1 Goal & Rationale

We employ **Deep Q-Learning (DQL)** to learn a rebalancing policy that maximises long-horizon *risk-adjusted* return under realistic trading constraints. DQL suits this task because it natively handles discrete action spaces and sequential decision-making, while mature stabilisation techniques (target net, double DQN, replay) curb training instabilities.

3.2 MDP Specification

- **State** : 16-dim vector $\rightarrow$ 6 dims = current portfolio weights (5 assets + cash, scaled to $[0, 1]$) + 10 dims = per-asset market indicators (moving-average ratio & rolling volatility).
- **Action** : 151 discrete rebalances: transfer 1–5 % between any two assets or cash, plus a *no-trade* option. Constrains lot-size realism and forbids leverage.
- **Reward** : n-step log-return $-\ 0.5\sigma\ -\ 0.5$DD, where σ = rolling volatility, DD = drawdown penalty. Promotes growth while penalising risk.
- **Discount** : $\gamma = 0.98 - 0.99$ to capture long-horizon compounding.

3.3 Dataset

- **Stable Set**: AAPL, MSFT, GOOGL, SBUX, TSLA (2014–2023).
  Large-cap, highly liquid technology and consumer stocks with historically moderate volatility, serving as a benchmark for performance in relatively predictable market conditions.

- **Volatile Set**: GME, AMC, SPCE, NVAX, NOK (2017–2023).
  Small- to mid-cap or event-driven equities characterized by sharp price swings and speculative trading, chosen to evaluate model robustness under extreme market turbulence.
- **Split:** 80% train / 20% validation; 2023 held-out test (~ 250 trading days).
- **Source:** Yahoo Finance

3.4 Network Architecture

$$16 \;\rightarrow\; 128 \;\rightarrow\; 256 \;\rightarrow\; 128 \;\rightarrow\; 151$$

Three fully-connected layers use `ReLU` activations, LayerNorm, and Dropout ($p = 0.3$) for stabilisation. ReLU is used as the activation layer after every LayerNorm. The output layer returns a Q-value for each discrete action.

3.5 Training Methodology

1. **TD(0) Bootstrapping (Online Net, θ)**
   Instead of expensive Monte-Carlo roll-outs, we update the online network Q_θ with a one-step bootstrap target
   $$y = r + \gamma\, Q_\theta(s', a^\star), \qquad a^\star = \arg\max_{a'} Q_\theta(s', a')$$
   where s' is the deterministic state obtained after executing the chosen rebalancing move. This yields fast, incremental value propagation and keeps the learning signal aligned with the agent's planning horizon.

2. **Target Network** (θ^-)
   Because the online net's parameters change every gradient step, using it for both action-selection and evaluation would create a moving target and unstable feedback loops. A shadow copy is therefore updated only every k episodes, anchoring the TD target and dramatically lowering training oscillations.
   $$y = r + \gamma\, Q_{\theta^-}(s', a^\star), \qquad a^\star = \arg\max_{a'} Q_{\theta^-}(s', a')$$
   $$Loss = mse\_loss(Q_\theta(s, a), y)$$

3. **Double DQN De-biasing**
   Standard DQN often overestimates action values because it uses the same network both to select the greedy action and to evaluate its value. Double DQN remedies this by decoupling these two steps:
   $$a^\star = \arg\max_{a'} Q_\theta(s', a') \quad \Longrightarrow \quad y = r + \gamma\, Q_{\theta^-}\big(s',\, a^\star\big),$$
   - **Action selection** ($\arg\max$) is performed by the *online* network Q_θ.
   - **Value estimation** uses the *target* network Q_{θ^-} evaluated at $a^\star$.
   - This separation prevents the same estimation error from being used twice, substantially reducing the positive bias in Q-value targets.
   - Empirically, it leads to more stable learning and better risk–return trade-offs in noisy financial environments.

4. **Experience Replay Buffer** (capacity = 10 000)
   Transitions $\big(s, a, r, s'\big)$ are stored and later sampled uniformly in mini-batches, destroying temporal correlation, mixing old and new regimes, and improving data-efficiency (each market day is reused many times).

5. **Hybrid Exploration Policy**
   ε-greedy drives broad exploration early on ($\varepsilon_{\text{start}} = 1.0$) and is annealed to 0.2 (mostly exploitative) by episode 200. Inside the "greedy" branch we apply a Boltzmann soft-max over valid Q-values, so the agent still samples high-value *alternatives* instead of committing too early to a single allocation rule.

6. **Two-Network Ensemble**
   We train *two* independent Q-nets and average their predictions. Ensemble voting counters the high variance caused by noisy return sequences and produces more reliable decisions across bullish, bearish, and side-ways regimes.

7. **Early-Stopping on Validation**
   Every 10 episodes the greedy policy is unrolled on a 20 % validation slice (never seen during replay updates). If the log-terminal value fails to improve for 5 consecutive checks we halt, preserving generalization and saving compute.

3.6 Baseline: Equal-Weight Buy-and-Hold

We implement a simple passive benchmark with no rebalancing:

1. **Initial Allocation.** Allocate 100% of capital equally across the N risky assets:

$$w_i = \frac{1}{N}, \quad i = 1, \ldots, N.$$

2. **Initial Shares.** With \$1 of starting capital, purchase

$$\text{shares}_i = \frac{w_i}{P_{0,i}}$$

   of asset i, where $P_{0,i}$ is its price on day 0.

3. **Buy-and-Hold Simulation.** For each day t,

$$V_t \;=\; \sum_{i=1}^{N} \text{shares}_i \times P_{t,i},$$

   where $P_{t,i}$ is the price of asset i on day t.

4. **Final Return.** Compute the total return as

$$\text{Return} = \big(V_T - 1\big) \times 100\%,$$

   where T is the last trading day.

4 Model Description: Actor-Critic for Portfolio Rebalancing

4.1 Policy Gradient Rationale

We implement a Proximal Policy Optimization (PPO) variant of the Actor-Critic framework, which is more stable and sample-efficient for continuous control settings. This method is suitable for adjusting portfolio weights under real-world trading constraints with smooth transitions.

4.2 PPO Actor–Critic Architecture

Our PPO agent is implemented as a single neural module with a shared feature extractor and two "heads"—one for the policy (actor) and one for the value estimate (critic). Concretely:

- **Input:** a normalized 5-dimensional portfolio state vector

$$s = \big(w_1, \ldots, w_5\big), \quad w_i \in [0,1],\ \sum_i w_i = 1,$$

  obtained by dividing each 0–20 integer weight by 20.

- **Shared Encoder:**
  - $\mathrm{Linear}(5 \to 128)$, followed by ReLU.
- **Actor Head:**
  - $\mathrm{Linear}(128 \to A)$, where $A = |\texttt{actions}|$.
  - Softmax over these A logits to produce a valid categorical distribution $\pi(a \mid s)$.
- **Critic Head:**
  - $\mathrm{Linear}(128 \to 1)$, producing a scalar state-value estimate $V(s)$.
- **Key Details:**
  - $\gamma = 0.99$ discounting, PPO-clip $\epsilon = 0.2$, and 4 minibatch epochs per update.
  - On each step, only *valid* actions (those that respect 0–20 bounds and sum–to–20 constraint) are entertained by masking out others before sampling.
  - The combined loss is

$$L = -E\big[\min(r_t(\theta)\,A_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\,A_t)\big] \ + \ \tfrac{1}{2}\,E\big[(G_t - V(s_t))^2\big],$$

    where $r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$.

This "shared-trunk + dual-head" design lets the actor and critic efficiently share feature representations, speeding convergence and stabilizing learning.

5 Evaluation

5.1 Experimental Setup

The models were training using Apple M2 CPU. The training duration spans from 20 minutes to 4 hours.

5.2 Measuring Q-Learning Model's Performance

Stable Dataset

- **Baseline Return On Investment(ROI)/Profit:** 59.39%
- **RL Model ROI (profit):** 70.62%
- **Annualized Sharpe Ratio:** 2.30

The RL agent outperforms the buy-and-hold baseline by 11.23% points, demonstrating its ability to capture persistent market trends while rigorously controlling risk. By integrating moving-average and volatility indicators into the state and shaping rewards with rolling Sharpe and draw-down penalties, the model intentionally adopts a conservative allocation strategy—limiting over-sized positions and avoiding over-exuberant re-balancing to achieve stable, robust performance.

Volatile Dataset

- **Baseline ROI:** –37.37%
- **RL Model ROI:** +17.12%
- **Annualized Sharpe Ratio:** 1.23

In highly volatile market conditions, the baseline strategy suffers significant losses, whereas the RL agent preserves capital and achieves **positive** returns. This highlights the impact of incorporating market indicators and shaping rewards through appropriate risk-sensitive penalties.

DQL vs Baseline | Daily Weights | Asset Contribution | Stacked Asset Allocation

DQL vs Baseline | Daily Weights | Asset Contribution | Stacked Asset Allocation

Figure 1: Sample visualization of portfolio behavior across various episodes on 2 different datasets (Stable & Volatile)

5.3 Measuring Actor-Critic (PPO) Model's Performance

Stable Dataset

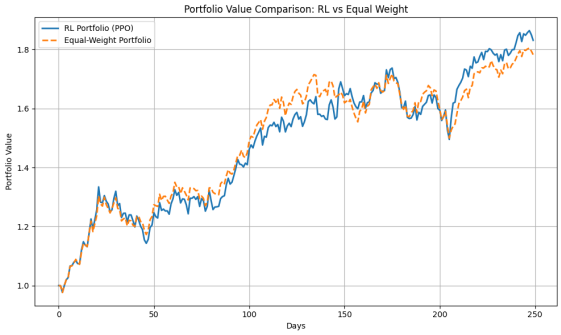
- **Baseline Return On Investment(ROI)/Profit:** 59.39%
- **RL Model ROI:** 83.12%
- **Annualized Sharpe Ratio:** 2.25

On the stable market, the Actor-Critic agent outperforms Q-Learning (70.6% ROI, Sharpe=2.30) by adapting continuously to gradual price trends. Its policy-gradient updates and entropy regularization drive more aggressive growth while maintaining similar risk control.

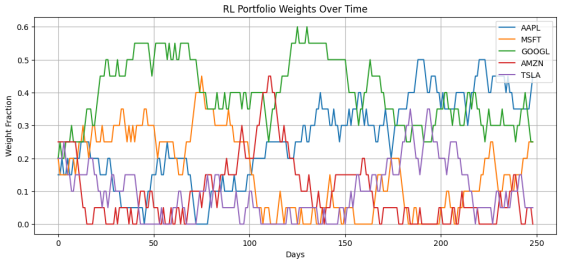
Volatile Dataset

- **Baseline ROI:** –37.37%
- **RL Model ROI:** –20.35%
- **Annualized Sharpe Ratio:** –0.28

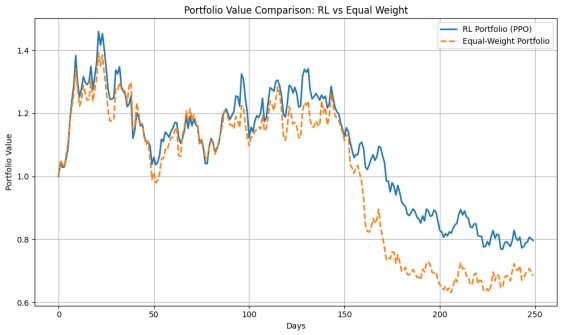
Under high volatility, the PPO agent suffers significant losses and negative risk-adjusted returns. Its continuous action adjustments overreact to noise, and without strong drawdown penalties, it fails to preserve capital.



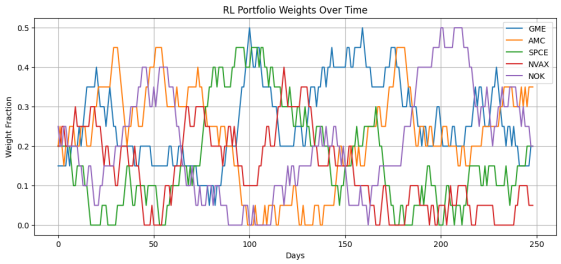
(a) PPO vs Baseline — Stable Dataset



(b) Daily Weights — Stable Dataset



(c) PPO vs Baseline — Volatile Dataset



(d) Daily Weights — Volatile Dataset

Figure 2: Comparison of portfolio behavior under Q-Learning and Actor-Critic (PPO) on both datasets.

5.4 Comparison of Q-Learning vs. Actor-Critic

- **Stable Market:**
  - **Baseline:** 59.39% ROI
  - **PPO:** 83.1% ROI, Sharpe 2.25
  - **DQL:** 70.6% ROI, Sharpe 2.30

  The Actor-Critic framework's continuous, entropy-regularized updates capture smooth trends more aggressively, yielding higher ROI at a comparable Sharpe ratio.

- **Volatile Market:**
  - **Baseline:** –37.37% ROI
  - **PPO:** –20.4% ROI, Sharpe –0.28
  - **DQL:** +17.1% ROI, Sharpe 1.23

  Deep Q-Learning's discrete, constraint-driven actions plus Sharpe-based reward shaping impart conservatism, enabling better capital preservation and positive returns when markets swing violently.

- **Key Insight:**
  - *PPO excels* in steady regimes by leveraging smooth policy updates and a continuous action space, which allows fine-grained allocation adjustments and lower transaction costs.

- *DQL shines* under turbulence through its discrete rebalancing steps, ensemble-based Q-value averaging, and explicit drawdown penalties, which enforce conservative trades and robust downside control.
- *Exploration vs. Stability:* PPO's entropy regularization maintains diversified exploration throughout training, preventing premature convergence, whereas DQL's ϵ-greedy + Boltzmann sampling rapidly focuses on high-reward actions once risk metrics stabilize.
- *Sample Efficiency:* DQL benefits from off-policy replay buffers and Target/Double-DQN updates to reuse past experience efficiently. PPO, being on-policy, needs fresh rollouts each update but gains from lower variance in policy gradient estimates.
- *Implementation Trade-Offs:* PPO's shared actor-critic architecture yields compact code and smoother learning curves, while DQL's separate target network and discrete action masking simplify integration of hard trading constraints.
- *Practical Deployment:* In live settings, discrete DQL policies map directly to order execution frameworks, whereas PPO's probabilistic outputs may require additional projection layers to enforce exact weight sums and leverage limits.

6 Discussion

Our experiments show that neither algorithm uniformly dominates—rather, each excels under different market regimes. In the steadier "tech" portfolio, PPO's smooth, entropy-regularized updates captured gradual trends and produced the highest risk-adjusted returns, whereas in the shock-driven "meme" universe DQL's built-in drawdown penalties and discrete rebalancing steps preserved capital and generated positive P&L when PPO faltered. This contrast underscores the importance of matching learning dynamics—on-policy versus off-policy, continuous versus discrete controls—to prevailing market behavior.

From an implementation standpoint, PPO's reliance on fresh on-policy rollouts can incur higher sample complexity, but yields lower gradient variance through clipped objectives, which proves advantageous when price movements follow gradual drifts. By contrast, DQL leverages an experience replay buffer and a separate, slowly updated target network to aggressively reuse past transitions; its ensemble averaging and Double-DQN update further combat Q-value overestimation, making it particularly robust when abrupt volatility spikes threaten to derail a purely gradient-based policy.

Despite these strengths, both approaches would benefit from richer market models. Neither currently accounts for dynamic transaction costs, liquidity constraints, or market impact—features that often determine real-world strategy viability. Likewise, our regime adaptation relies solely on short-term technical indicators; integrating longer-horizon signals (e.g. macroeconomic, sentiment, or volatility-regime classifiers) could allow the agent to anticipate and prepare for regime shifts. Finally, the pronounced drawdowns exhibited by PPO in the volatile portfolio suggest that actor-critic objectives should incorporate stronger risk-sensitive terms (e.g. CVaR or maximum drawdown penalties) to better guard against extreme losses.

By combining the two paradigms, we can unlock further gains. DQL modules can enforce capital preservation during turbulent regimes, while PPO's continuous policy can exploit trending markets. Multi-objective extensions can optimize turnover, tail risk, drawdown, and return. Meta-learning techniques can enable agents to adapt rapidly to novel assets or unseen market cycles. These innovations will enable future systems to achieve the reliability and flexibility needed for real-world portfolio management.

References

Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1), 77–91.

Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*.

Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2016). Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3), 653–664.

Zhang, Y., Zhao, P., Rong, Y., Xu, C., & Huang, J. (2020). Relation-aware transformer for portfolio policy learning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 4641–4647).

Almahdi, S., & Yang, S. Y. (2017). An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, 87, 267–279.

Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem.

Ganaie, M. A., Hu, M., Malik, A. K., Tanveer, M., & Suganthan, P. N. (2022). Ensemble deep learning: A review. Engineering Applications of Artificial Intelligence, 115, 105151.

Zhou, Z. (2012). Ensemble Methods: Foundations and Algorithms. Chapman and Hall/CRC.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1), 119–139.

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research, 11, 169–198.

Kuncheva, L. I. (2004). Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience.

Rokach, L. (2010). Ensemble-based classifiers. Artificial Intelligence Review, 33(1–2), 1–39.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In Multiple Classifier Systems (pp. 1–15). Springer.