Deep Reinforcement Learning for Cryptocurrency Portfolio Management: A Comparative Study

Jose Luis Marquez Jaramillo, Taylor Hawks
EN.605.741 - Reinforcement Learning
Johns Hopkins University

December 2025

Abstract

Cryptocurrency markets present unique challenges for portfolio management due to their high volatility, 24/7 trading, and evolving asset universe.

We investigate the application of deep reinforcement learning (DRL) to dynamic cryptocurrency portfolio allocation, implementing and comparing four RL agents—Deep Q-Network (DQN), Double DQN (DDQN), REINFORCE, and REINFORCE with learned baseline—against traditional baselines including Equal Weight, Market Cap Weight, and Mean-Variance Optimization.

Our environment processes over seven years of historical data (2018–2025) for 37 cryptocurrencies, using a 60-day lookback window of OHLCV features. We design a discrete action space with 70 delta-based portfolio adjustments for value-based methods and a continuous Dirichlet-sampled simplex output for policy gradient methods, incorporating realistic constraints including transaction costs (0.1%) and turnover limits (30%).

On a held-out test period (January 2024–October 2025), Mean-Variance optimization achieved the highest cumulative return (366.54%) and Sharpe ratio (1.64). Among RL agents, DDQN performed best (190.77%, Sharpe 1.25), outperforming both passive baselines (Equal Weight 168.50%, Market Cap 159.30%) and demonstrating that value-based deep RL can capture exploitable structure in cryptocurrency markets. The policy gradient methods also performed competitively under deterministic evaluation: REINFORCE achieved 161.84% and REINFORCE+Baseline reached 167.37%—nearly matching Equal Weight—with minimal turnover (0.35–0.88%).

Our results demonstrate that deep RL can outperform passive investment strategies in cryptocurrency portfolio management. DDQN's delta-based action space provided the strongest returns, while REINFORCE's Dirichlet policy achieved competitive performance when using deterministic (mean-based) inference. We discuss the importance of inference-time behavior for continuous policies and identify directions for future research.

1 Introduction

The cryptocurrency market has grown from a niche experiment to a multi-trillion-dollar asset class, with total market capitalization exceeding $3.20 trillion as of December 2025 [7], surpass-

ing the previous peak of approximately $3 trillion in November 2021 [19, 25]. This growth brings unique challenges for portfolio management: daily volatility that frequently exceeds traditional equity markets by several times [3, 18], continuous 24/7 trading across fragmented exchanges with varying transaction costs [2, 13], and an evolving asset universe as new tokens emerge and others fail. Furthermore, cross-asset correlations in cryptocurrency markets shift substantially between bull and bear regimes [9], complicating covariance estimation for traditional optimization approaches.

Deep reinforcement learning (DRL) has demonstrated strong performance in sequential decision-making domains from game playing to robotics, prompting researchers to explore its application to financial portfolio management [12, 15]. The portfolio management problem—dynamically allocating capital across multiple assets to maximize risk-adjusted returns—naturally maps to a Markov Decision Process (MDP). At each decision point, the agent observes market conditions and its current holdings, then selects target portfolio weights subject to constraints such as position limits and transaction costs. Unlike supervised learning approaches that predict returns and then optimize, RL directly learns the allocation policy end-to-end, potentially capturing complex temporal dependencies and trading off immediate costs against future opportunities.

Despite promising results in controlled experiments, applying DRL to real-world portfolio management faces several challenges that prior work has incompletely addressed. First, most studies assume a fixed asset universe [12, 26], but cryptocurrency portfolios must handle assets entering and leaving the tradable set as market conditions change. Second, the choice of action space—whether continuous weights, discrete allocation templates, or incremental adjustments—significantly affects learning dynamics but remains underexplored [15]. Third, rigorous comparison against strong traditional baselines is often lacking; the equal-weight (1/N) portfolio, despite its simplicity, has proven remarkably difficult to beat in out-of-sample tests [6].

This paper investigates whether deep reinforcement learning can learn profitable portfolio allocation strategies for cryptocurrency markets that outperform classical approaches. We develop a comprehensive experimental framework that addresses the practical challenges of variable asset universes, realistic transaction costs, and fair comparison across different algorithmic paradigms. Our goal is not merely to achieve positive returns—which depend heavily on the test period—but to understand under what conditions and architectural choices DRL methods succeed or fail relative to traditional baselines.

Our approach combines a realistic portfolio environment with multiple agent architectures. The environment handles monthly universe reconstitution, enforces trading constraints (turnover limits, concentration caps), and applies proportional transaction costs consistent with centralized exchange fee structures [4]. We implement four DRL agents spanning value-based and policy gradient paradigms: Deep Q-Network (DQN) with a novel delta-based action catalog, Double DQN (DDQN) to address overestimation bias [23], REINFORCE for continuous policy optimization [24], and REINFORCE with a learned value baseline to reduce gradient variance. These are benchmarked against three traditional strategies: equal weighting, market-capitalization weighting, and mean-variance optimization with shrinkage estimation [14].

Contributions. The main contributions of this work are:

1. **Variable universe environment**: A portfolio management environment handling dynamic asset membership with monthly reconstitution, cold-start rules, and automatic weight redistribution. We release a frozen dataset export enabling exact reproduction.

2. **Delta-based discrete action space**: A 70-action catalog of relative portfolio adjustments for DQN agents, preserving portfolio continuity and enabling context-dependent feasibility learning.

3. **Canonical padding for variable-size observations**: A scheme mapping variable-size observations to fixed-dimensional representations while preserving asset identity for Q-value learning.

4. **Rigorous out-of-sample evaluation**: A 646-day held-out test demonstrating that DDQN outperforms passive baselines, and that policy gradient methods achieve competitive performance under deterministic evaluation.

Paper organization. The remainder of this paper is organized as follows. Section 2 reviews related work in deep reinforcement learning for portfolio management and positions our contributions. Section 3 describes our environment design, agent architectures, and baseline implementations in detail. Section 4 presents the experimental setup, training procedures, and quantitative results. Section 5 interprets the divergent outcomes between value-based and policy gradient methods and discusses limitations. Section 6 concludes with directions for future research.

2 Related Work

We review prior work in four areas: deep reinforcement learning for portfolio management, value-based and policy gradient methods, traditional portfolio optimization, and the unique characteristics of cryptocurrency markets that motivate our design choices.

2.1 Deep Reinforcement Learning for Portfolio Management

The application of deep reinforcement learning to portfolio management was pioneered by Jiang [11], who introduced the Ensemble of Identical Independent Evaluators (EIIE) framework for cryptocurrency trading. This architecture processes each asset's price history through identical neural network modules, producing allocation scores that are normalized via softmax to obtain portfolio weights. The follow-up work [12] extended this framework with the Portfolio Vector Memory (PVM), which feeds the previous portfolio weights back into the policy network. This recurrent structure helps the agent internalize transaction costs by learning to avoid unnecessary rebalancing.

Ye et al. [26] proposed State-Augmented Reinforcement Learning (SARL), which enriches the observation space with market-wide indicators such as index returns and volatility measures. By conditioning on broader market context, SARL agents can potentially adapt their strategies

to different market regimes. The authors demonstrated improved performance on Chinese stock markets compared to baseline methods.

Lucarelli and Borrotti [15] applied Deep Q-Learning to cryptocurrency trading, exploring discrete action spaces where each action corresponds to a complete portfolio allocation. Their work highlighted the challenges of applying DQN to portfolio management, including the need for careful reward shaping and the difficulty of learning in highly non-stationary financial environments.

Limitations of prior DRL approaches. Despite promising backtested results, several limitations persist across this literature. First, most studies assume a fixed asset universe throughout training and evaluation, which does not reflect the reality of cryptocurrency markets where tokens are frequently listed and delisted. Second, evaluation periods are often short or overlap with training data, making it difficult to assess true generalization. Third, comparison against traditional baselines is frequently limited to buy-and-hold strategies, which can underperform in trending markets and provide limited insight into whether DRL adds value over established portfolio optimization methods. Our work addresses these gaps by explicitly modeling variable universes, using strict train/test separation, and benchmarking against strong traditional methods.

2.2 Value-Based and Policy Gradient Methods

Deep Q-Networks (DQN) [17] combine Q-learning with neural networks and experience replay, but suffer from overestimation bias. Double DQN [23] addresses this by decoupling action selection from evaluation. Applying DQN to portfolio management requires discretizing the action space; Gao et al. [8] used fixed allocation templates, but this scales poorly with universe size. Our delta-based action catalog defines actions as relative portfolio changes, enabling smooth transitions through portfolio space.

REINFORCE [24] naturally handles continuous action spaces and has been applied to portfolio management where weights are treated as continuous outputs. Actor-critic methods reduce variance through learned baselines [20, 22]. The EIIE framework [12] uses softmax outputs to satisfy simplex constraints; our REINFORCE implementation follows this paradigm with Dirichlet sampling for exploration.

2.3 Traditional Portfolio Optimization

Mean-variance optimization, introduced by Markowitz [16], remains the foundation of quantitative portfolio management. The framework selects portfolio weights to maximize expected return for a given level of variance, or equivalently, minimize variance for a target return. However, mean-variance optimization is highly sensitive to estimation error in expected returns and covariances.

DeMiguel et al. [6] compared fourteen portfolio strategies across multiple datasets, finding that the simple equal-weight (1/N) portfolio frequently outperformed optimized strategies out-of-sample. Their analysis attributed this to estimation error: the benefits of optimization are

overwhelmed by the noise in estimated parameters, particularly expected returns. This result established equal weighting as a baseline that optimized strategies often fail to beat.

To improve robustness, practitioners employ shrinkage estimators for covariance matrices. Ledoit and Wolf [14] proposed shrinking the sample covariance toward a structured target (e.g., diagonal or constant correlation), reducing estimation error at the cost of some bias. Our mean-variance baseline implements Ledoit-Wolf shrinkage, representing a reasonable best-effort classical approach.

Challenges for traditional methods in cryptocurrency markets. While mean-variance optimization has been extensively studied for equity portfolios, cryptocurrency markets present additional challenges. Return distributions exhibit heavy tails and time-varying volatility that violate the Gaussian assumptions underlying mean-variance theory [3, 18]. Cross-asset correlations shift substantially between bull and bear market regimes [9], meaning covariance estimates from calm periods may be misleading during crises. Furthermore, the short history of most cryptocurrencies limits the statistical reliability of parameter estimates. These characteristics motivate exploring whether data-driven RL methods can learn more adaptive allocation strategies.

2.4 Positioning of This Work

Our work extends prior DRL portfolio research by addressing three key gaps. First, unlike EIIE [12] and SARL [26], we explicitly model variable asset universes with monthly reconstitution and eligibility rules. Second, our delta-based action catalog enables smooth portfolio transitions while maintaining a tractable discrete action space for value-based learning. Third, our canonical padding state encoder preserves asset identity information essential for evaluating delta actions.

We contribute to the broader discussion about the practical value of DRL for portfolio management. While prior work has often reported positive results in backtesting, rigorous out-of-sample evaluation has been limited. Our 646-day test period reveals that both algorithmic paradigm and inference-time behavior matter: value-based methods with discrete actions outperform passive baselines, while continuous policy gradient methods achieve competitive performance when using deterministic evaluation (outputting the distribution mean rather than sampling). These findings suggest that action space design and deployment strategy deserve more systematic attention than they have received in prior work.

3 Methods

This section describes our methodology for applying deep reinforcement learning to cryptocurrency portfolio management. We first formalize the problem as a Markov Decision Process (Section 3.1), then detail our environment design including data processing and constraint handling (Section 3.2). We present four RL agent architectures (Section 3.3) and three baseline strategies (Section 3.4) that operate under identical constraints for fair comparison.

3.1 Problem Formulation

We formulate cryptocurrency portfolio management as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ defines transition dynamics, $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor.

Transition Dynamics. The transition $\mathcal{P}(s_{t+1}|s_t, a_t)$ is determined by two components: (1) exogenous market price movements, which we do not model or predict, and (2) endogenous portfolio mechanics, where the new weights $\mathbf{w}_t$ become the "previous weights" in the next state. We adopt a model-free approach—agents learn directly from environment interactions without estimating $\mathcal{P}$.

State Space. At each trading day t, the state s_t consists of (1) a per-asset historical observation tensor $\mathbf{X}_t \in \mathbb{R}^{A_t \times 4 \times 60}$ containing normalized OHLCV features over a 60-day lookback window, and (2) the previous portfolio weights $\mathbf{w}_{t-1} \in \Delta^{A_{t-1}}$, where A_t denotes the number of tradable assets at time t. The inclusion of previous weights, analogous to the Portfolio Vector Memory (PVM) mechanism [12], enables the agent to internalize turnover costs.

Action Space. The action a_t specifies a target portfolio allocation $\mathbf{w}_t \in \Delta^{A_t}$ over the current tradable assets. For value-based methods, we discretize this space into 70 delta-based rebalancing actions (Section 3.3.1); for policy gradient methods, we output continuous weights directly (Section 3.3.2).

Constraints. All portfolios must satisfy:

$$w_t^{(i)} \geq 0 \quad \forall i \quad \text{(long-only)} \tag{1}$$

$$\sum_{i=1}^{A_t} w_t^{(i)} = 1 \quad \text{(fully invested, no cash)} \tag{2}$$

$$\|\mathbf{w}_t - \mathbf{w}_{t-1}\|_1 \leq \tau \quad \text{(turnover limit, } \tau = 0.30\text{)} \tag{3}$$

The fully-invested constraint (2) forces the agent to maintain exposure to cryptocurrency risk rather than trivially holding cash, ensuring fair comparison against crypto benchmarks [12, 15].

Reward Function. The agent receives a reward based on net portfolio return after transaction costs:

$$r_t = \log\left(1 + \mathbf{w}_t^\top \mathbf{R}_{t+1}\right) - c \cdot \|\mathbf{w}_t - \mathbf{w}_{t-1}\|_1 \tag{4}$$

where $\mathbf{R}_{t+1} \in \mathbb{R}^{A_t}$ is the vector of simple returns from day t to $t+1$, and $c = 0.001$ is the transaction cost coefficient (0.1% per unit turnover). Critically, the forward returns $\mathbf{R}_{t+1}$ are never observed by the agent at decision time—they are used only by the environment to compute realized rewards after the action is taken.

Objective. The agent seeks to maximize expected cumulative discounted rewards:

$$\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{T} \gamma^t r_t \right] \tag{5}$$

We use $\gamma = 0.93$ for production training, selected via Bayesian hyperparameter optimization (Section 4.3). This relatively high discount factor indicates that longer-horizon planning is beneficial even with daily transaction costs, though the search space included values as low as 0.5.

3.2 Environment Design

Our environment processes over seven years of cryptocurrency market data and enforces realistic trading constraints. We describe the data pipeline, state representation, and constraint enforcement mechanisms.

3.2.1 Data Pipeline

We use daily OHLCV (Open, High, Low, Close, Volume) data for liquid cryptocurrencies from 2018-07-01 through 2025-10-31. The tradable universe is defined by a market-cap-weighted index with monthly constituent rebalancing: at the end of month $m - 1$, we freeze the index membership for all days in month m, preventing intramonth look-ahead bias.

Data Quality. We apply tiered gap repair rules to handle missing data. Single-day gaps are addressed through forward-filling, propagating the previous day's bar. Gaps of two to five days receive linear interpolation for price channels and log-space interpolation for volume to preserve multiplicative scaling properties. Gaps exceeding five days trigger asset ineligibility—the asset is excluded from the tradable universe until 60 consecutive days of clean data become available, ensuring sufficient history for feature construction.

Cold-Start Eligibility. An asset is tradable on day t only if it (1) is included in the current month's index membership, and (2) has at least 60 consecutive days of clean OHLCV data immediately prior to t. This prevents allocation to newly listed or illiquid assets with insufficient trading history [12, 26].

3.2.2 State Representation

For each tradable asset i at time t, we construct a feature tensor from the trailing 60-day window:

Price Normalization. Close, high, and low prices are divided by the asset's closing price on day t, so the most recent close equals 1.0 and prior prices are relative. This improves stationarity and has been shown to stabilize training in crypto portfolio agents [12].

Volume Normalization. We apply $\log(1 + \text{volume})$, then z-score normalize within each asset's 60-day window, clipping extreme values to $[-5, 5]$.

The resulting observation tensor $\mathbf{X}_t \in \mathbb{R}^{A_t \times 4 \times 60}$ has dimensions: A_t assets $\times$ 4 channels (close, high, low, volume) $\times$ 60 days. Because A_t varies over time (due to index rebalancing and cold-start rules), we employ canonical padding for value-based methods (Section 3.3.1).

Universe Changes. When assets exit the tradable universe (due to index rebalancing or data quality issues), the environment forces liquidation at the last available price and redistributes the weight proportionally across remaining assets, incurring transaction costs. New entrants begin with zero weight and become available for allocation only after satisfying the cold-start requirement.

3.2.3 Constraint Enforcement

The environment supports two constraint handling approaches:

Projection Mode (used by REINFORCE): If the proposed allocation violates constraints, the environment projects it onto the feasible set using quadratic programming. The agent is trained on the projected allocation without explicit feedback about violations.

Penalty Mode (used by DQN/DDQN): If the proposed allocation violates constraints, the environment rejects execution and assigns a penalty reward of -10.0. The portfolio remains at its previous state, but time advances normally. The penalty magnitude was chosen to be substantially larger than typical single-step rewards (which range from approximately -0.05 to $+0.05$), ensuring constraint violations are strongly discouraged. This enables the agent to learn context-dependent feasibility through experience—the same delta action (e.g., "increase BTC by 10%") can be safe or risky depending on current holdings [15].

3.3 Agent Architectures

We implement two reinforcement learning agent families: value-based methods (DQN, DDQN) and policy gradient methods (REINFORCE, REINFORCE with learned baseline).

3.3.1 Deep Q-Network (DQN) and Double DQN (DDQN)

Our value-based agents address two key challenges: handling variable universe sizes with fixed-dimensional networks, and designing a discrete action space for portfolio rebalancing.

Canonical Padding StateEncoder. We assign each of the 37 unique assets a fixed canonical position. For each observation, we populate zero-padded arrays $\mathbf{X}_{\text{canonical}} \in \mathbb{R}^{37 \times 4 \times 60}$ and $\mathbf{w}_{\text{canonical}} \in \mathbb{R}^{37}$ at assigned positions, then flatten to 8,917 dimensions and project through a learned linear layer to a 256-dimensional embedding. This preserves asset identity essential for context-dependent Q-values.

Delta-Based Action Catalog. We implement 70 discrete portfolio adjustment actions: a hold action; adjust actions that increase/decrease exposure to top-K assets by $\pm 5\%$, $\pm 10\%$, or $\pm 15\%$; rotation actions transferring weight between asset pairs; and rebalancing actions resetting to equal weight or market-cap weight. Each action applies a delta adjustment and renormalizes to the simplex.

Network Architecture and Training. The Q-network is a 3-layer MLP (512→256→70) with ReLU and dropout. We use experience replay (capacity 10,000), ϵ-greedy exploration decaying from 1.0 to 0.1, and target network updates every 100 steps. DDQN differs only in the TD target: $Q_{\text{target}} = r + \gamma\, Q_{\text{target}}(s', \arg\max_{a'} Q_{\text{online}}(s', a'))$, decoupling action selection from evaluation to reduce overestimation bias [23].

3.3.2 REINFORCE Policy Gradient Agents

Our policy gradient agents directly parameterize portfolio weights as continuous outputs.

Policy Network Architecture. A GRU layer (input size $4 \times N$, sequence length=60) processes the combined assets' 60-day sequence. The GRU layer outputs to a fully connected layer where it is concatenated with the previous weights and a binary initialization flag. This is then fed to another hidden layer then to a final output layer which produces per-asset logits, converted via masked softmax to portfolio weights. We use canonical indexing across 37 positions with a boolean mask for tradable assets.

Training with Dirichlet Sampling. Actions are sampled from a Dirichlet distribution with concentration parameters $\boldsymbol{\alpha} = \text{softplus}(\text{logits})/\tau$, where temperature τ is a parameter introduced to flatten or spike the distribution. When sampled allocations violate constraints, the environment projects them onto the feasible set. Policy gradients use Monte Carlo returns, and the network uses an Adam optimizer.

Learned Value Baseline. To reduce gradient variance, REINFORCE+Baseline adds a value head $V_\phi(s_t)$, which shares the GRU encoder for learning efficiency. The advantage $A_t = G_t - V_\phi(s_t)$ replaces raw returns in the policy gradient. The combined loss is $\mathcal{L} = -\sum_t A_t \log \pi_\theta(a_t|s_t) + 0.5 \sum_t (G_t - V_\phi(s_t))^2$.

3.4 Baselines

We implement three baselines operating under identical constraints. **Equal Weight (EW)** allocates uniformly: $w_t^{(i)} = 1/A_t$. Despite its simplicity, equal weighting has proven competitive by avoiding estimation error [6]. **Market Cap Weight (MCW)** allocates proportionally to market capitalization, mimicking index funds with minimal turnover. **Mean-Variance Optimization (MVO)** implements Markowitz optimization [16]:

$$\mathbf{w}_t^* = \arg\max_{\mathbf{w} \in \Delta^{A_t}} \left\{ \mathbf{w}^\top \hat{\boldsymbol{\mu}}_t - \frac{\lambda}{2} \mathbf{w}^\top \hat{\boldsymbol{\Sigma}}_t \mathbf{w} \right\} \tag{6}$$

using Ledoit-Wolf shrinkage [14] for covariance estimation.

4 Experiments

This section describes our experimental setup and presents the results of our comparative evaluation. We detail the dataset construction (Section 4.1), evaluation metrics (Section 4.2), hyperparameter selection process (Section 4.3), and present the main results (Section 4.4).

4.1 Dataset

We construct a comprehensive cryptocurrency dataset spanning over seven years of market data.

Data Source. Daily OHLCV (Open, High, Low, Close, Volume) data is collected from the CoinGecko Analyst API [5]. The dataset covers 37 unique cryptocurrencies, with daily availability varying between 8 and 35 assets depending on index membership and cold-start eligibility requirements.

Temporal Splits. We partition the data into non-overlapping periods following walk-forward evaluation principles [15]:

Table 1: Dataset temporal splits. The warmup period builds initial lookback windows. Development data is used for training and hyperparameter selection. Test data is strictly held out for final evaluation.

Split	Period	Days	Purpose
Warmup	2018-07-01 → 2018-08-31	62	Build 60-day lookback
Train (train_core)	2018-09-01 → 2023-06-30	1,848	Agent training
Validation	5 regime windows	∼100	Hyperparameter selection
Test	2024-01-01 → 2025-10-31	646	Out-of-sample evaluation

Regime-Based Validation. Rather than using a single contiguous validation period, we carve out five approximately 20-day windows corresponding to qualitatively distinct market regimes. These include the late 2018 cryptocurrency winter (val_2018_crash), the March 2020 COVID-induced market crash (val_covid), the 2021 bull market (val_bull), the 2022 deleveraging period (val_bear), and the 2023 low-volatility consolidation (val_chop). This regime-diverse validation strategy prevents overfitting hyperparameters to a single market condition [11, 12], ensuring that selected configurations generalize across bull markets, bear markets, and high-volatility crash periods alike.

4.2 Evaluation Metrics

We evaluate portfolio performance using standard metrics from quantitative finance, encompassing return measures, risk metrics, and trading activity indicators.

Return Metrics. Cumulative return measures the total percentage gain over the evaluation period, providing an absolute performance benchmark. We also report the Compound Annual Growth Rate (CAGR), which annualizes returns while properly accounting for compounding effects, enabling comparison across periods of different lengths.

Risk-Adjusted Performance. The Sharpe ratio [21] quantifies risk-adjusted return as the ratio of annualized excess return to annualized volatility, assuming a zero risk-free rate given the cryptocurrency context. We complement this with the Sortino ratio, which penalizes only downside volatility by using only negative returns in the denominator, providing a measure more aligned with investor preferences. Maximum drawdown captures the largest peak-to-trough decline observed during the evaluation period, representing worst-case loss exposure. The Calmar ratio, computed as CAGR divided by maximum drawdown, combines these perspectives into a single efficiency metric.

Trading Activity. Portfolio turnover, defined as the mean daily L_1 norm of weight changes $\|\mathbf{w}_t - \mathbf{w}_{t-1}\|_1$, quantifies trading activity. High turnover strategies incur greater transaction costs, making this metric essential for assessing practical viability in markets with non-trivial trading frictions.

4.3 Hyperparameter Selection

We employ Bayesian hyperparameter optimization using Optuna [1] to efficiently search the high-dimensional configuration space.

Search Protocol. For the value-based DQN and DDQN agents, as well as the REINFORCE agents, we use the Tree-structured Parzen Estimator (TPE) sampler to guide the search over 50 candidate configurations.

Each configuration is evaluated by training for 50 episodes using 100-day sliding windows sampled from the training period, followed by validation across five episodes on the regime-diverse validation windows with a fixed random seed for reproducibility. The optimization objective maximizes mean validation return, with MedianPruner providing early termination for unpromising trials. For the DQNs, we additionally implement Q-value explosion detection, terminating any trial where mean Q-values exceed 10,000 to prevent numerical instability from wasting computational resources.

Search Space. For the DQNs, we search over discount factor $\gamma \in [0.5, 0.99]$, learning rate $\in [10^{-5}, 10^{-3}]$ (log-uniform), batch sizes $\{32, 64, 128\}$, replay buffer sizes $\{10{,}000, 50{,}000\}$, ϵ decay schedules, and network architectures up to three layers. For the REINFORCE agents, optimizable hyperparameters are limited to architecture configurations, learning rate, and ϵ decay schedules.

Training Protocol. Using the best hyperparameters identified through the search, we train the final models with extended episode budgets. DQN trains for 400 episodes and DDQN

for 300 episodes, both using 100-day sliding windows sampled from the training period. The REINFORCE agent, contributed as a separate implementation with independent tuning, trains for 10,000 episodes. All agents employ early stopping with patience of 5 validation checks (performed every 50 episodes for value-based methods) to prevent overfitting, and we checkpoint the best model based on mean validation return for final evaluation.

4.4 Results

We evaluate all seven strategies on the held-out test period (January 2024 – October 2025, 646 trading days). Results are summarized in Table 2 and visualized in Figures 1–3.

Table 2: Test set performance comparison across all agents and baselines. Best values in **bold**, second-best underlined. Test period: January 2024 – October 2025 (646 trading days).

Agent	Return (%)	CAGR (%)	Sharpe	Sortino	Max DD (%)	Turnover (%)
Baselines						
Equal Weight	168.50	74.88	1.217	1.250	48.87	**0.25**
Market Cap	159.30	71.46	1.240	1.298	44.17	0.34
Mean-Variance	**366.54**	**139.07**	**1.640**	**1.691**	**39.96**	13.50
RL Agents						
DQN	135.05	62.20	1.092	1.114	48.99	5.67
DDQN	190.77	82.95	1.247	1.298	49.93	6.28
REINFORCE	161.84	72.41	1.195	1.220	48.83	0.88
REINFORCE+Baseline	167.37	74.46	1.202	1.235	49.26	0.35

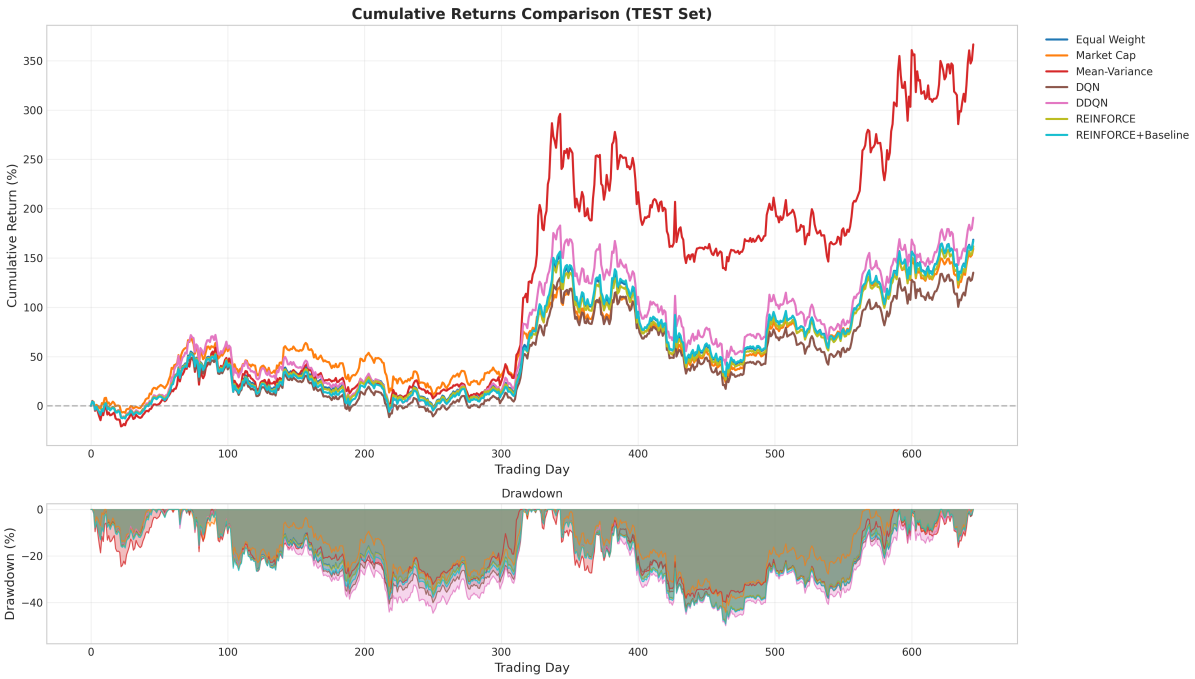

Figure 1: Cumulative returns (top) and drawdown (bottom) over the test period. Mean-Variance achieves 366% return with lowest drawdown; DDQN (190.77%) outperforms passive baselines.

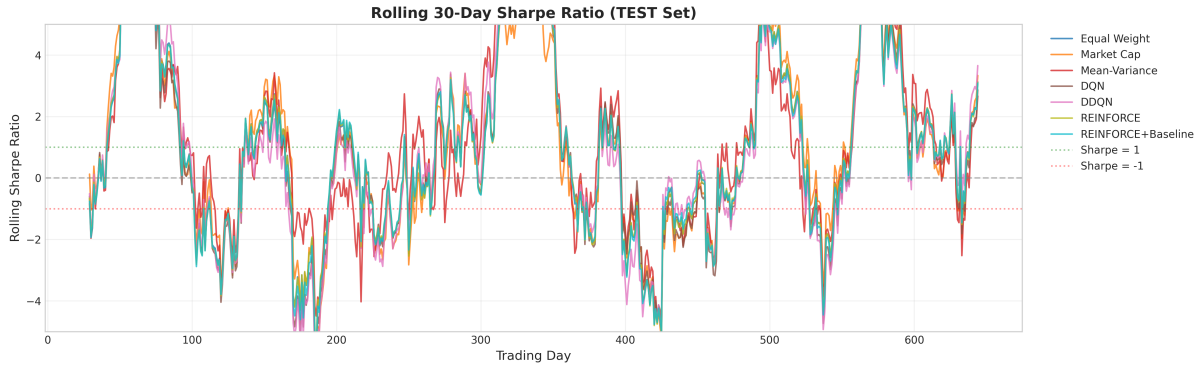

Figure 2: Rolling 60-day Sharpe ratio. Mean-Variance maintains highest risk-adjusted performance throughout most of the period. DDQN and REINFORCE variants track passive baselines with comparable stability.

Figure 3: Portfolio allocation comparison across strategies over the test period. Mean-Variance exhibits dynamic reallocation with concentrated positions, while Equal Weight maintains uniform distribution. DDQN shows moderate rebalancing. REINFORCE variants display stable allocations under deterministic evaluation, with turnover comparable to passive baselines.

Key Observations. Mean-Variance achieved the highest returns (366.54%) and Sharpe ratio (1.640), suggesting historical estimates provided useful predictive signal during this bull market period. Among RL agents, DDQN achieved 190.77% return, outperforming both passive baselines (Equal Weight: 168.50%, Market Cap: 159.30%) with moderate turnover (6.28%). This success aligns with Double Q-learning's theoretical motivation: decoupling action selection from evaluation reduces optimistic bias. Standard DQN achieved only 135.05% despite similar architecture.

The policy gradient agents performed competitively under deterministic evaluation. REINFORCE achieved 161.84% return with only 0.88% daily turnover, while REINFORCE+Baseline reached 167.37%—nearly matching Equal Weight (168.50%)—with turnover of just 0.35%. The learned baseline provided modest improvement (+5.5 percentage points), suggesting some variance reduction benefit. Deterministic evaluation uses the Dirichlet mean ($\alpha_i / \sum_j \alpha_j$) rather than sampling, producing smooth weight proposals that avoid triggering the turnover constraint. However, both policy gradient methods still underperformed DDQN (190.77%), indicating that the delta-based discrete action space may provide structural advantages for portfolio optimization.

5 Discussion

The experimental results presented in Section 4.4 reveal important patterns in how different algorithmic paradigms approach cryptocurrency portfolio optimization. This section interprets these findings, situates them within the broader literature, acknowledges limitations, and identifies directions for future research.

5.1 Interpreting Agent Performance

Mean-Variance optimization's dominance reflects both favorable market conditions and sample efficiency advantages. The test period (January 2024–October 2025) coincided with a bull market where momentum signals in rolling return estimates carried predictive information [10]. MVO directly estimates sufficient statistics from historical data and solves a closed-form optimization, using data efficiently compared to model-free RL that must rediscover statistical relationships through gradient descent. However, MVO's dominance may not generalize to mean-reverting or crash regimes where historical estimates lose validity [9].

The dichotomy between value-based and policy gradient methods reveals important lessons about action space design. DDQN's success (190.77% return, 6.28% turnover) demonstrates that value-based RL can learn meaningful allocation policies. The delta-based action space enabled incremental adjustments without dramatic position changes, while double Q-learning provided stability by reducing overestimation bias [23].

The policy gradient agents performed competitively under deterministic evaluation. REINFORCE achieved 161.84% return with 0.88% turnover, while REINFORCE+Baseline reached 167.37%—nearly matching Equal Weight (168.50%)—with turnover of just 0.35%. The learned baseline provided modest improvement (+5.5 percentage points), consistent with variance reduction theory. Critically, deterministic evaluation uses the Dirichlet mean ($\alpha_i / \sum_j \alpha_j$) rather

than sampling, producing smooth weight proposals that naturally satisfy turnover constraints. This demonstrates that the policy gradient agents learned meaningful allocation preferences, but realizing this potential required appropriate inference-time behavior. The gap between DDQN (190.77%) and REINFORCE+Baseline (167.37%) suggests that delta-based discrete actions may still provide structural advantages for constrained portfolio optimization.

5.2 Comparison with Prior Work

Our findings partially align with prior RL portfolio work. Jiang et al. [12] and Ye et al. [26] reported strong RL outperformance, though with smaller asset universes (12 vs. our 37), shorter evaluation periods, and often without realistic transaction costs. Our DDQN's outperformance of passive baselines supports the thesis that deep RL can capture exploitable structure, though the margin (190.77% vs. 168.50%) is more modest than some claims.

The contrasting outcomes between value-based and policy gradient methods highlight that action space design substantially influences learning dynamics—a dimension often overlooked in prior work. Our 30% turnover constraint and fully-invested requirement create a more conservative evaluation environment than studies permitting cash allocations or unlimited trading.

5.3 Limitations

Several limitations temper our conclusions. Our test period represents a single market regime (post-bear recovery to bull market); DDQN's success may not generalize to sustained bear markets or crash periods. The transaction cost model (0.1% per unit turnover) ignores slippage and market impact. The policy gradient results depend on deterministic evaluation—stochastic sampling during deployment could yield different behavior and potentially trigger turnover constraints. Our frozen agents lack online adaptation that could improve performance relative to MVO's natural incorporation of recent data. Finally, 50 Optuna trials may not have found optimal configurations, particularly for policy gradient methods.

5.4 Implications and Future Directions

For practitioners, DDQN's outperformance of passive baselines demonstrates that value-based deep RL can add value, though the margin (190.77% vs. 168.50%) warrants weighing implementation complexity against returns. REINFORCE+Baseline's competitive performance (167.37%) with minimal turnover (0.35%) offers an alternative for practitioners preferring continuous action spaces. MVO's 366% return reinforces that classical methods remain formidable when market conditions favor momentum.

For researchers, the comparison between value-based and policy gradient methods reveals nuanced tradeoffs. DDQN's delta-based actions provide built-in turnover control, while REINFORCE's Dirichlet policy requires deterministic evaluation for stable deployment. Deterministic policy methods (DDPG, TD3) represent a natural middle ground worth exploring. Transformer architectures with cross-asset attention could combine MVO's correlation reasoning with deep learning's flexibility. Online learning approaches that adapt to evolving conditions represent a promising frontier.

6 Conclusion

This paper investigated the application of deep reinforcement learning to cryptocurrency portfolio management, comparing four RL agents against classical baselines on a rigorous 646-day out-of-sample test. Our evaluation employed a realistic environment with monthly universe reconstitution, transaction costs, and turnover constraints, providing a more stringent benchmark than typical in this literature.

The central finding is that algorithmic paradigm and inference-time behavior both matter: DDQN's value-based approach with discrete delta actions outperformed all methods except MVO, while REINFORCE's continuous Dirichlet policy achieved competitive performance (167% with baseline) when using deterministic evaluation that outputs the distribution mean rather than sampling. This demonstrates that policy gradient agents can learn meaningful allocation preferences, but realizing this potential requires appropriate inference-time behavior. Mean-Variance optimization's dominance (366% return) further demonstrates that classical estimation-based methods remain formidable competitors when market conditions favor momentum-based signals.

Our methodological contributions—the variable-universe environment with canonical padding, delta-based discrete action space, and frozen dataset export—address practical challenges that prior work largely ignored and enable exact reproduction of our experiments.

Several directions merit future investigation: deterministic policy gradient methods (DDPG, TD3) that provide a natural middle ground between discrete and sampled continuous actions; transformer architectures with cross-asset attention for implicit correlation modeling; extended evaluation across bear markets and crash periods; investigation of stochastic versus deterministic deployment tradeoffs; and online learning approaches that adapt to evolving market dynamics. Our results suggest that the gap between research benchmarks and practical deployment is narrower than sometimes claimed—careful implementation of established RL algorithms can yield portfolio strategies that outperform passive investment approaches.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[2] Andrea Barbon and Angelo Ranaldo. On the quality of cryptocurrency markets: Centralized versus decentralized exchanges. *arXiv preprint arXiv:2112.07386*, 2021.

[3] Lykke Øverby Bergsli, Andrea Fidje Lind, Peter Molnár, and Michal Polasik. Forecasting volatility of bitcoin. *Research in International Business and Finance*, 59:101540, 2022.

[4] Alexander Brauneis, Roland Mestel, Ryan Riordan, and Erik Theissen. The anatomy of a fee change—evidence from cryptocurrency markets. *Journal of Empirical Finance*, 67: 152–167, 2022.

[5] CoinGecko. CoinGecko Cryptocurrency Data API. https://www.coingecko.com/en/api, 2025. Accessed: December 2025.

[6] Victor DeMiguel, Lorenzo Garlappi, and Raman Uppal. Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy? *The Review of Financial Studies*, 22 (5):1915–1953, 2009.

[7] Forbes Digital Assets. Cryptocurrency prices. https://www.forbes.com/digital-assets/crypto-prices/, 2025. Accessed: 8 December 2025.

[8] Ziming Gao, Yuan Gao, Yi Hu, Zhengyong Jiang, and Jionglong Su. Application of deep q-network in portfolio management. In *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*, pages 268–275, 2020. doi: 10.1109/ICBDA49040.2020.9101333.

[9] Nick James and Max Menzies. Collective correlations, dynamics, and behavioural inconsistencies of the cryptocurrency market over time. *Nonlinear Dynamics*, 107(4):4001–4017, 2022.

[10] Narasimhan Jegadeesh and Sheridan Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, 48(1):65–91, 1993.

[11] Zhengyao Jiang. Cryptocurrency portfolio management with deep reinforcement learning. *arXiv preprint arXiv:1612.01277*, 2016.

[12] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.

[13] Thomas Kim. On the transaction cost of bitcoin. *Finance Research Letters*, 23:300–305, 2017.

[14] Olivier Ledoit and Michael Wolf. Honey, i shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 30(4):110–119, 2004.

[15] Giorgio Lucarelli and Matteo Borrotti. A deep q-learning portfolio management framework for the cryptocurrency market. *Neural Computing and Applications*, 32:17229–17244, 2020.

[16] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[18] Viviane Naimy, Omar Haddad, Gema Fernández-Avilés, and Rim El Khoury. Modelling and predicting the bitcoin volatility using garch models. *PLoS ONE*, 16(7):e0254720, 2021.

[19] Arthur AB Pessa, Matjaž Perc, and Haroldo V Ribeiro. Age and market capitalization drive large price variations of cryptocurrencies. *Scientific Reports*, 13(1):3351, 2023.

[20] Jonathan Sadighian. Deep reinforcement learning in cryptocurrency market making, 2019. URL https://arxiv.org/abs/1911.08647.

[21] William F Sharpe. Mutual fund performance. *The Journal of Business*, 39(1):119–138, 1966.

[22] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[23] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[24] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[25] Marcin Watorek, Jarosław Kwapień, and Stanislaw Drożdż. Cryptocurrencies are becoming part of the world global financial market. *Entropy*, 25(2):377, 2023.

[26] Yifeng Ye, Xiying Zhang, Li Zhang, Hao Wang, and Defu Wang. State augmented reinforcement learning for portfolio management. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. arXiv:2002.05780.

A Hyperparameter Settings

Table 3: DQN and DDQN hyperparameters.

Hyperparameter	**DQN**	**DDQN**
Learning rate	1×10^{-4}	1×10^{-4}
Batch size	64	64
Replay buffer size	10,000	10,000
Target update frequency	100 steps	100 steps
Discount factor (γ)	0.93	0.93
Epsilon start	1.0	1.0
Epsilon end	0.05	0.05
Epsilon decay episodes	500	500
Hidden dimensions	[256, 128]	[256, 128]
Dropout	0.0	0.0
Training episodes	1,200	1,600

Table 4: REINFORCE hyperparameter search space. Best configuration selected via Optuna.

Hyperparameter	**Search Range**	**Selected**
Learning rate	$[10^{-6}, 10^{-4}]$	1×10^{-4}
Recurrent layer	{GRU, LSTM, RNN}	GRU
Hidden dimension	{128, 256, 512}	128
Temperature (τ)	{1, 2, 5}	1
Discount factor (γ)	1.0	1.0
Epsilon start	1.0	1.0
Epsilon end	{0.01, 0.05, 0.1}	0.05
Epsilon decay episodes	{300, 500, 1000}	500
Max allocation constraint	0.35	0.35
Value coefficient (baseline)	—	0.5
Canonical assets	37	37

B Action Catalog

The DQN and DDQN agents use a delta-based action catalog with 70 discrete actions. Each action represents a portfolio adjustment relative to current weights, following the approach of Lucarelli and Borrotti [**?**]. Table 5 summarizes the action categories.

C Training Details

All models were trained on 60-day rolling windows with daily rebalancing. The DQN and DDQN agents used experience replay with uniform sampling from a buffer of 10,000 transitions. The target network was updated every 100 gradient steps via hard copy.

REINFORCE agents used episodic updates with undiscounted ($\gamma = 1$) Monte Carlo returns. The Dirichlet policy with temperature scaling enabled smooth portfolio weight sampling while

Table 5: Delta-based action catalog (70 actions total).

Category	Description	Count	Delta Range
Hold	No portfolio change	1	—
Adjust Top-1	Modify top asset weight	7	±5%, ±10%, ±15%
Adjust Top-2	Modify top-2 equally	7	±5%, ±10%, ±15%
Adjust Top-3	Modify top-3 equally	7	±5%, ±10%, ±15%
Adjust Top-4	Modify top-4 equally	7	±5%, ±10%, ±15%
Adjust Top-5	Modify top-5 equally	5	±5%, ±10%
Rotate 1↔2	Transfer between #1 and #2	10	5%–25%
Rotate 1↔3	Transfer between #1 and #3	8	10%–25%
Rotate 2↔3	Transfer between #2 and #3	4	10%, 15%
Diversify	Spread from concentrated	4	5%–20%
Concentrate	Move to top asset	4	5%–20%
Rebalance Equal	Reset to equal weight	3	All, Top-5, Top-10
Shift to Top-K	Reallocate to top-K	3	K=3, 5, 7
Total		**70**	

respecting the simplex constraint. The baseline variant shared the GRU encoder between policy and value heads, reducing variance in gradient estimates.

Training data covered September 2018 to June 2023 (1,848 days), with January 2024 to October 2025 (646 days) reserved for out-of-sample testing. Five regime-diverse validation windows were carved from the training period for hyperparameter selection. Early stopping based on mean validation return prevented overfitting.

D Code and Data Availability

All code, data-processing scripts, trained model artifacts, and notebooks required to reproduce the experiments in this paper are publicly available at the project's GitHub repository: https://github.com/josemarquezjaramillo/crypto-rl-portfolio.

The experiments reported here were produced from the repository in the main branch. The repository contains:

- Data loader and preprocessing scripts that ingest the CoinGecko Analyst API extracts used in our experiments.
- Training and evaluation scripts for DQN, DDQN, and REINFORCE agents, with exact hyperparameter configurations.
- Notebooks demonstrating dataset construction, training runs, and evaluation visualizations.
- Checkpoints for the best-performing models (see 'checkpoints/'), plus a README with reproduction instructions.