Team members

- Navya Priya Nandimandalam, navyapriya_n@tamu.edu, 136004813
- Alisha Raj, alisharaj2607@tamu.edu, 735007776

Project Summary

We'll fork the wassname Gym-style portfolio repo and add a discrete rebalancing wrapper that turns portfolio allocation into a clean MDP. On top, we'll implement tabular Q-learning for the discrete env and DDPG for a matched continuous-action variant. Using identical rewards (returns minus costs, drawdown penalty) and walk-forward splits, we'll compare stability and performance across seeds and transaction-cost regimes. We will extend our comparison by including two additional baseline algorithms (TD3 and PPO) from Stable-Baselines3 to benchmark against modern actor–critic and policy-gradient methods. On the tabular side, we'll integrate algorithmic enhancements such as UCB-ε exploration (to balance exploration and exploitation more effectively) and Double Q-learning (to reduce overestimation bias). We'll ship a reproducible evaluation harness with baselines (equal-weight, momentum) and metrics (Return, Sharpe, MaxDD, Turnover).

Literature Survey

[1] Alidousti, M. et al. (2025): A Novel Data-Efficient Double DQN Framework for Intelligent Financial Portfolio Management

Summary:

This paper proposes a Portfolio Double Deep Q-Network (PDQN) framework for portfolio optimization, designed to improve data efficiency and stability in volatile financial markets. It tackles overestimation bias by decoupling action selection and evaluation steps through Double Q-learning. The framework achieves competitive performance using only 3 years of training data (vs. 6-9 years in prior work) and 30 episodes (vs. 500), demonstrating significant data efficiency. The authors incorporate transaction cost modeling, LSTM-based temporal feature extraction, Leaky ReLU activation, Xavier initialization, Huber loss, and dropout regularization for enhanced learning stability.

Key Findings:

- Implements Double Q-learning with separate online and target networks to reduce overestimation bias

- Achieves strong performance with significantly reduced data requirements (3 years vs. 6-9 years) and fewer training episodes (30 vs. 500)
- Uses LSTM autoencoder to extract compressed temporal features from historical technical indicators (OHLC prices, volume)
- Employs Huber loss for training stability and Leaky ReLU to prevent dying neuron problems
- Includes transaction cost penalties and dropout regularization to prevent overfitting
- Achieves higher Sharpe and Sterling ratios than standard DQN and Buy-and-Hold strategies across multiple test years (2016-2019)

Similarity:

This study aligns closely with our project's motivation to explore overestimation bias reduction and risk-adjusted learning in portfolio management. Both works aim to quantify the benefit of Double Q-learning mechanisms and measure outcomes via risk metrics such as Sharpe Ratio and Maximum Drawdown.

Difference:

While Alidousti et al. use deep neural networks with LSTM-based feature extraction for continuous state spaces, our project applies Double-Q logic at a tabular level with discrete rebalancing actions to isolate algorithmic effects without neural network approximation noise. We extend their setup by: (1) comparing tabular Q-learning variants (standard, Double-Q, UCB-ε) against modern deep RL baselines (DDPG, TD3, PPO) under identical reward functions and transaction costs, (2) testing stability across multiple random seeds and cost regimes (10-50 bps), (3) using a discrete action space ({-Δ, 0, +Δ}) rather than continuous weight allocation, and (4) providing a reproducible evaluation harness for fair algorithmic comparison.

[2] Jiang, Z., Xu, D., & Liang, J. (2017): A Deep Reinforcement Learning Framework for Financial Portfolio Management

Summary:

This foundational work introduces the EIIE (Ensemble of Identical Independent Evaluators) architecture for deep portfolio management, using deterministic policy gradients to learn continuous portfolio weights directly. The model handles transaction costs through a Portfolio-Vector Memory (PVM) and uses Online Stochastic Batch Learning (OSBL). Three variants (CNN, basic RNN, LSTM) were tested on cryptocurrency markets with 30-minute trading periods.

Key Findings:

- Proposes an end-to-end deterministic policy gradient framework to optimize logarithmic portfolio returns

- Introduces the Portfolio Vector Memory (PVM) to maintain continuity between trading periods and account for transaction costs without gradient vanishing
- **Employs Online Stochastic Batch Learning (OSBL) for continuous online learning as new data arrives**
- Compares CNN, basic RNN, and LSTM variants on Poloniex cryptocurrency exchange (12 assets including Bitcoin)
- Achieves superior final accumulated portfolio value (fAPV) and Sharpe ratios compared to traditional strategies and Buy-and-Hold benchmarks despite 0.25% commission rates
- **All three EIIE networks monopolized top 3 positions across all experiments**
- **Demonstrates scalability: training time scales linearly with number of assets due to weight-sharing architecture**

Similarity:

Both Jiang et al. (2017) and our project share a sequential decision-making formulation for portfolio management, emphasizing temporal dependency of wealth evolution and transaction cost modeling. **Both recognize that today's portfolio allocation affects tomorrow's feasible actions due to transaction costs.** Our approach inherits their insight that portfolio actions must account for compounding returns and transaction friction. **Both works use similar reward formulations based on portfolio returns.**

Difference:

While Jiang et al. operate entirely in a **continuous action space** (portfolio weights as continuous vectors), we introduce a **discrete rebalancing MDP wrapper** with actions {-Δ, 0, +Δ} per asset, enabling direct comparison between tabular Q-learning and deep RL methods. Their work focuses on **neural feature extraction from high-dimensional price tensors** using CNN/RNN/LSTM architectures, while ours **isolates algorithmic performance differences** by comparing:

1. **Tabular Q-learning variants** (standard ε-greedy, Double Q-learning, UCB-ε exploration)
2. **Deep RL baselines** (DQN, DDPG, TD3, PPO)

[3] Lu Chung I (2025): Evaluation of Deep Reinforcement Learning Algorithms for Portfolio Optimisation

Summary:

This paper evaluates five benchmark DRL algorithms (A2C, PPO, DDPG, TD3, SAC) on portfolio optimization using simulated data based on geometric Brownian motion with market impact. It uses the Kelly criterion (log utility) as the objective and derives analytical optimal policies to benchmark performance. The study focuses on algorithmic robustness to noisy rewards, sample complexity, and performance under varying market impact conditions.

Key Findings:

- **Off-policy methods (DDPG, TD3, SAC) completely failed** due to inability to learn correct Q-functions with noisy rewards, even with architectural modifications
- **On-policy methods (PPO, A2C) succeeded**, with PPO achieving near-optimal policies (growth rate 0.100 vs. optimal 0.114) using Generalized Advantage Estimation (GAE)
- **PPO's clipping mechanism is critical** for preventing policy drift after convergence and A2C without clipping showed instability
- **Sample complexity is prohibitively high**: requires >2 million steps (equivalent to ~8,000 years of daily price data) even in the simplest setting
- **GAE parameter tuning is essential**: optimal λ values between 0.8-0.9 balance bias-variance tradeoff under noisy rewards
- **Market impact significantly affects strategies**: higher wealth scenarios push agents toward fractional Kelly strategies and performance degrades with increasing transaction costs
- **Regime-switching environments**: PPO combined with Hidden Markov Models successfully learns distinct policies per regime under low wealth, but struggles with high wealth due to increased noise

Similarity:

We adopt the same experimental philosophy i.e., holding reward functions, transaction cost models, and evaluation metrics constant across all methods to ensure fair comparison. Like this study, we use financial performance metrics (Return, Sharpe, Max Drawdown, Turnover) and walk-forward validation splits. Both projects explicitly model transaction costs (we use 10-50 bps, similar to their range) and their impact on optimal strategies. Both studies evaluate PPO, DDPG, and TD3 under identical experimental conditions.

Difference:

1. **Action space**: We introduce a **discrete rebalancing MDP** with actions {-Δ, 0, +Δ} enabling **tabular Q-learning**, whereas the reference paper uses continuous portfolio weight vectors throughout
2. **Algorithm coverage**: We include **tabular methods** (standard Q-learning, Double-Q, UCB-ε) and **DQN** not evaluated in the reference paper, allowing direct tabular-vs-deep comparison
3. **Environment**: We use the **wassname crypto portfolio environment** rather than a custom GBM simulator, which may have different state representations, reward structures, and market dynamics
4. **Analytical baseline**: The reference paper derives **closed-form optimal policies** (Kelly criterion with zero market impact) as performance upper bounds whereas we compare against heuristic baselines (equal-weight, momentum, buy-and-hold)
5. **Focus**: The reference paper investigates **why off-policy methods fail** (noise → bad Q-learning) and we focus on **whether algorithmic improvements** (Double-Q, UCB-ε) and discrete action spaces can make Q-learning viable for portfolio optimization

[4] Li, H. & Hai, M. (2024): Deep Reinforcement Learning Model for Stock Portfolio Management Based on Data Fusion

Summary: Li and Hai propose CMPS (Collaborative Multi-agent reinforcement learning-based stock Portfolio management System), a three-agent DRL model that fuses stock market quotes, financial indices, and stock correlations. They use DQN-based agents with self-attention to combine outputs and introduce CMPS-RF, which includes risk-free asset allocation for risk prevention.

Key Findings:

- Three-agent architecture: market quotes agent, financial index agent, and stock correlation agent
- Each agent uses DQN with convolutional and fully-connected layers
- Self-attention mechanism combines agent outputs for final portfolio weights
- Incorporates Markowitz mean-variance theory for stock correlations
- CMPS-RF variant includes risk-free asset strategy with adaptive weight allocation
- Tested on China Shanghai Stock Exchange (SSE) 50 Index data
- Outperforms FINRL baselines (PPO, DDPG, TD3, SAC) and MAPS on cumulative returns
- CMPS-RF achieves best Sharpe and Calmar ratios, demonstrating superior risk-adjusted performance

Similarity:

Both approaches use multiple information sources to inform portfolio decisions and compare DRL algorithms (including DDPG, TD3, PPO) on portfolio management tasks. Li and Hai also emphasize risk management through their CMPS-RF variant with risk-free assets, which aligns with our risk-aware reward design. Both works use walk-forward validation and standard financial metrics (Sharpe ratio, maximum drawdown, returns).

Difference:

Our work addresses a fundamentally different research question. Li and Hai focus on ***what data sources*** (market quotes + financial indices + correlations) and ***what architecture*** (multi-agent with self-attention fusion) improve deep RL performance. We focus on ***which algorithmic enhancements*** (tabular Q-learning with UCB-ε exploration and Double Q vs. deep baselines) provide better stability and exploration-exploitation balance. Key methodological differences:

- **Agent architecture:** They use three collaborative DQN agents with self-attention fusion and we use single-agent implementations to isolate algorithmic effects
- **State design:** They fuse heterogeneous data (financial reports, correlation matrices) and we deliberately use simpler states (returns/volatility/weights/regime flag) to study algorithmic robustness independent of feature engineering

- **Action space:** We introduce a discrete rebalancing wrapper enabling direct tabular Q-learning comparison whereas they use DQN with continuous-like output projected to simplex
- **Research goal:** We benchmark algorithmic stability (exploration strategies, overestimation bias, seed variance, transaction-cost sensitivity) whereas they advance multi-agent architecture and data fusion for portfolio returns
- **Market focus:** They test on Chinese stocks (SSE 50) and we plan to use crypto assets following the wassname environment
- **Interpretability:** Our tabular methods enable direct Q-value inspection and comparison of exploration strategies (standard ε-greedy vs. UCB-ε)

[5] Huang, G., Zhou, X., Song, Q. (2024): Deep RL for Long–Short Portfolio Optimization

Summary:

This paper develops a DRL portfolio framework with a novel short-selling mechanism for China's A-share market. The key innovation is implementing the constraint $\Sigma|\omega_i| = 1$ (sum of absolute weights equals 1), allowing weights in [-1, 1] for long and short positions. This accurately models continuous trading returns, unlike traditional $\Sigma\omega_i = 1$ constraints that distort calculations with short positions. Using PPO with a modified VGG network and average annualized Sharpe ratio as reward, the model dynamically recalculates short position weights as prices fluctuate while properly accounting for transaction costs.

Key Findings:

- Novel weight constraint $\Sigma|\omega_i| = 1$ enables accurate return calculation for long-short portfolios in continuous trading
- Uses PPO exclusively (off-policy methods required excessive computation and converged slowly)
- Modified VGG architecture with Actor-Critic structure
- State: normalized price features (open, low, high, close) over sliding window
- Achieved +4.22% to +8.79% annualized returns when traditional models lost -53% to -26%
- Lowest maximum drawdown: 5-8% vs. traditional methods' 18-26%
- Highest Calmar ratio: 1.73 (best risk-adjusted returns)
- Tested on randomly-selected CSI 300 stocks including declining assets

Similarity:

Both apply DRL to portfolio management with actor-critic algorithms (we test multiple including PPO and they use PPO). Both emphasize risk-aware rewards and evaluate using Sharpe, Sortino, drawdown, and Calmar metrics. Both use walk-forward evaluation on real market data with transaction costs.

Difference:

They focus on what mechanism enables long-short modeling, proposing $\Sigma|\omega_i| = 1$ and testing against traditional optimization methods (MV, CVaR, HRP). We focus on which algorithms show better stability, comparing Tabular Q-learning variants (Double Q, UCB-ε) against deep baselines (DDPG, TD3, PPO). Their action space: continuous [-1, 1] with long-short positions. Ours: discrete rebalancing $\{-\Delta, 0, +\Delta\}$ for tabular agents, continuous simplex-constrained long-only for deep RL. Their states incorporate financial indices and stock correlations for data fusion. We use simpler states (returns, volatility, weights, regime flag) to isolate algorithmic effects. They test China SSE 50 stocks and we use crypto assets. Their contribution is a novel short-selling mechanism formulation. Ours is algorithmic benchmarking to understand when improved tabular methods match deep RL stability under transaction costs.

[6] Choudhary, H., Orra, A., Sahoo, K., Thakur, M. (2025): Risk-Adjusted Deep RL for Portfolio Optimization: A Multi-Reward Approach

Summary:

Risk-Adjusted DRL Paper proposes a multi-reward deep reinforcement learning framework for portfolio optimization that trains three separate PPO agents using different reward functions (log returns, Differential Sharpe Ratio, and Maximum Drawdown). These agents' actions are then fused using a CNN to produce a unified risk-adjusted portfolio allocation. The methodology is tested on four global stock markets (Sensex, Dow, TWSE, IBEX) and demonstrates superior performance across multiple risk-return metrics.

Key Findings:

- Combining three reward functions (returns, risk-adjusted returns, downside risk) outperforms single-objective approaches
- CNN-based action fusion successfully integrates diverse agent policies
- Achieved 124.83% cumulative return on Sensex (vs 52.49% market index)
- Sharpe ratios consistently above 1.6 across all markets
- Supervised pre-training using historical data improves convergence
- Statistical significance confirmed via paired t-tests ($p < 0.05$)

Similarity:

Both projects address portfolio optimization using RL with emphasis on risk management. We share common ground in evaluation methodology, using Sharpe ratio, Max Drawdown, and cumulative returns on real market data. Both employ PPO as a core algorithm and recognize the importance of transaction costs. We also both acknowledge that choosing appropriate reward functions is critical for risk-aware trading strategies. Finally, both frameworks use reproducible evaluation with fixed seeds and structured train-test splits.

Difference:

- Problem Formulation: The Risk-Adjusted DRL paper frames portfolio management as a multi-objective optimization problem solved through ensemble learning. We frame it as an algorithm selection problem, testing when simpler methods suffice versus when deep RL becomes necessary.
- Action Space: They use continuous portfolio weights (softmax-constrained) suitable for deep RL. We implement both discrete rebalancing actions (for tabular methods) and continuous weights (for fair comparison).
- Algorithmic Approach: They train three separate PPO agents and fuse actions via CNN. We compare tabular Q-learning variants (Double Q, UCB-ε) directly against multiple deep RL baselines (DDPG, TD3, PPO, DQN).
- Training Methodology: They use supervised pre-training with historical weight calculations (Equation 1) to initialize the CNN. We use pure RL without supervised bootstrapping.
- Reward Function: They use three distinct rewards requiring separate agents. We test individual reward functions head-to-head while also benchmarking a weighted single-objective formulation.

[7] Liu, S. (2024): An Evaluation of DDPG, TD3, SAC, and PPO: Deep Reinforcement Learning Algorithms for Controlling Continuous Systems

Summary:

This paper evaluates four deep reinforcement learning algorithms (DDPG, TD3, SAC, and PPO) for controlling continuous robotic systems using MuJoCo environments in OpenAI Gym. The study tests these algorithms on four robotic control tasks which are: Ant-v4, HalfCheetah-v4, Hopper-v3, and Swimmer-v4, to compare their ability to learn effective control policies. The research aims to identify which algorithms perform best for continuous control and how different environment characteristics affect algorithm performance.

Key Findings:

- **Experimental Setup**: Used identical neural network architectures (256 hidden nodes in two layers), same activation functions, and fixed random seeds across all algorithms to ensure fair comparison
- **TD3 and SAC Performance**: These algorithms consistently achieved the highest rewards across most environments. TD3 used clipped double Q-learning to address overestimation, while SAC employed entropy regularization for better exploration
- **DDPG Limitations**: DDPG consistently underperformed due to Q-value overestimation bias, demonstrating that the improvements in TD3 (its successor) are necessary for robust performance
- **PPO Instability**: PPO showed high variance and potential instability issues. The authors attribute this to variable episode lengths affecting batch size optimization, a critical insight for environments with inconsistent episode durations

- **Environment-Specific Results**: Performance rankings varied by task complexity. For example, in Ant-v4 (most complex, 27-dim observation), PPO eventually achieved highest rewards despite instability, while in simpler tasks like Swimmer-v4, TD3 dominated

Similarity:

- **Algorithm Selection**: Both studies evaluate DDPG, TD3, and PPO as core deep RL methods and emphasize fair comparison with identical hyperparameters and random seeds
- **Stability Analysis**: Both focus on comparing algorithm performance across multiple seeds and analyzing variance in outcomes
- **Continuous Control**: Both address continuous action spaces, though our proposal also includes discrete formulations for tabular methods

Difference:

- **Algorithmic Scope**: Liu compares only deep RL algorithms (DDPG, TD3, SAC, PPO), while we compare tabular Q-learning variants (Double Q, UCB-ε) against deep RL baselines to test when complexity is justified
- **Evaluation Framework**: Liu uses physics-based rewards (forward motion, action penalties) and measures convergence speed and cumulative reward. We use financial risk-adjusted metrics (Sharpe ratio, Max Drawdown, Turnover) with transaction cost modeling (10-50 bps) and include domain-specific non-RL baselines (equal-weight, momentum strategies) to establish practical performance thresholds.

[8] Pawar, A. A., Muskawar, V. P., Tiku, R. (2024): Portfolio Management using Deep Reinforcement Learning

Summary:

This paper proposes a deep reinforcement learning approach for portfolio management where an agent allocates weights to assets rather than simply making buy/hold/sell decisions. The framework uses Deep Q-Network (DQN) architecture to learn optimal portfolio weight allocations across 28 assets (cryptocurrencies and ETFs). The agent operates in a simulated liquid market without transaction costs, aiming to maximize returns while minimizing risk exposure through weight rebalancing over daily time periods from 2010-2018.

Key Findings:

- **Novel Action Space**: Instead of traditional discrete actions (buy/hold/sell), the agent directly allocates continuous portfolio weights to each asset, with weights constrained to sum to 1.0

- **State Representation**: Designed a comprehensive state tensor containing: (1) 28 asset prices, (2) 28 ten-day moving averages, (3) 784-element correlation matrix (28×28) computed over previous 10 days
- **DQN Architecture**: Implemented Deep Q-Learning with experience replay buffer (size 32), trained over 1000 episodes with daily rebalancing
- **Superior Performance**: The DRL model achieved the highest Sharpe ratio (1.0424) compared to benchmarks: Benchmark strategy (0.0011), Min-Variance (0), Max Returns (0.0007), and Auto-encoder (0.0019)
- **Long/Short Capability**: Agent can go both long (weights 0 to 1) and short (weights -1 to 1), providing flexibility beyond traditional long-only strategies
- **Replay Buffer Training**: Uses FIFO replay buffer to store experiences (previous state, action, current state, reward) and trains on mini-batches to move toward optimal policy

Similarity:

- **Portfolio Management Focus**: Both apply DRL to portfolio allocation problems with continuous weight assignments
- **Action Space Design**: Both use continuous action spaces for portfolio weights rather than discrete buy/sell/hold actions
- **Risk-Adjusted Metrics**: Both emphasize Sharpe ratio as primary evaluation metric alongside returns
- **Benchmark Comparisons**: Both compare DRL approaches against traditional portfolio strategies (min-variance, equal-weight)

Difference:

- **Transaction Costs**: Pawar et al. assume a frictionless market with no transaction costs, while our proposal explicitly models transaction costs (10-50 bps) as critical constraints
- **Algorithmic Approach**: They use only DQN, while we compare multiple algorithms (tabular Q-learning variants vs. DDPG, TD3, PPO) to test when complexity is justified
- **Market Environment**: They test on stocks and cryptocurrencies over 2010-2018, while we focus specifically on cryptocurrency portfolio management with discrete rebalancing wrapper
- **Evaluation Scope**: Pawar et al. provide single-metric comparison (Sharpe ratio), while we propose comprehensive evaluation including Max Drawdown, Turnover, stability across seeds, and cost sensitivity analysis

[9] Jiang, C., Wang, J. (2023): A Portfolio Model with Risk Control Policy Based on Deep Reinforcement Learning

Summary:

This paper proposes LSTR–TD3, combining the Twin Delayed Deep Deterministic policy gradient (TD3) algorithm with Long- and Short-Term Risk (LSTR) control for portfolio management. The LSTR mechanism dynamically adjusts the proportion of cash assets using

Beta distribution for long-term trends and exponential decay for short-term losses. Tested on Chinese stock market data (CSI500/CSI300, 2012-2022).

Key Findings:

- Risk Control Integration: LSTR adjusts portfolio weights capturing long-term risk (Beta distribution) and short-term risk (exponential function)
- Performance Improvement: LSTR–TD3 outperformed vanilla TD3 across three randomly selected portfolios of Chinese stocks
- Market Effectiveness: Both algorithms beat CSI300 benchmark, validating TD3 applicability to Chinese A-shares

Similarity:

- Both use TD3 for continuous portfolio weight allocation
- Both emphasize risk-adjusted returns (Sharpe ratio, max drawdown)
- Both address portfolio management with continuous action spaces

Difference:

- Risk Mechanism: They adjust cash allocation via LSTR whereas we compare tabular Q-learning improvements vs. deep RL algorithms
- Market & Assets: Chinese stocks vs. cryptocurrency portfolio
- Algorithmic Scope: Single algorithm comparison (TD3 ± risk control) vs. multi-algorithm comparison (tabular methods vs. DDPG/TD3/PPO)
- Transaction Costs: Fixed 0.25% vs. sensitivity analysis across 10-50 bps
- Evaluation: 3 portfolios with basic metrics vs. comprehensive analysis including cost sensitivity and regime robustness

[10] Kayumov, R. A. (2023): Deep Reinforcement Learning for Investment Portfolio Rebalancing

Summary:

This paper implements and tests a Deep Deterministic Policy Gradient (DDPG) model for portfolio rebalancing. The author compares DDPG against A2C and PPO algorithms, then improves DDPG by proposing a compound reward function combining Sharpe ratio and logarithmic return. The model is tested on a portfolio of 23 assets (18 US stocks + 5 commodity futures) over 2009-2023 using weekly data, implemented in PyTorch.

Key Findings:

- **DDPG Superiority**: DDPG outperformed A2C and PPO in initial experiments, achieving 79.4% total return vs. 61.3% (PPO), 51.0% (A2C), 57.5% (Markowitz baseline), and 23.3% (DJIA benchmark) over 2020-2022
- **Improved Reward Function**: Proposed compound reward combining Sharpe ratio and log return: Reward = (Sharpe + logReturn × 10^3) / TotalSteps, which improved performance over simple return-based rewards
- **Final Performance**: On extended backtest (2019-2023), DDPG achieved 118.4% total return, 32.9% average annual return, 1.42 Sharpe ratio, and 15.1% maximum drawdown, beating all baselines
- **Training Details**: Used Actor-Critic architecture with Conv2d layers, trained for 100 episodes (~56,000 steps) with 0.1% transaction costs and weekly rebalancing
- **Baseline Comparisons**: Outperformed Markowitz MPT (74.7% return, 0.94 Sharpe), Equal Weighting (66.6% return, 0.92 Sharpe), and DJIA benchmark (45.7% return, 0.63 Sharpe)
- **Technical Features**: Used MACD, Bollinger Bands, RSI, CCI, DX as additional state features, implemented replay buffer and noise injection following standard DDPG architecture

Similarity:

- Both use DDPG for continuous portfolio weight allocation
- Both emphasize Sharpe ratio in reward/evaluation
- Both use weekly rebalancing with transaction cost modeling (0.1% vs. our 10-50 bps)
- Both test on multi-asset portfolios with historical backtesting

Difference:

- **Reward Function**: Kayumov proposes compound reward (Sharpe + logReturn), while we compare algorithmic improvements (Double Q, UCB-ε) and standard reward formulations
- **Asset Universe**: Mixed US stocks + commodities vs. cryptocurrency-only portfolio
- **Algorithmic Focus**: Single algorithm optimization (DDPG variants) vs. comparative study across algorithm classes (tabular vs. deep RL)
- **Evaluation Depth**: Basic metrics (return, Sharpe, MDD) vs. comprehensive analysis including cost sensitivity, regime robustness, and stability across seeds
- **Action Space**: Direct continuous weights vs. discrete rebalancing wrapper for tabular methods

References

[1] M. Alidousti, M. K. Bafruei, and A. H. A. Sedigh, “A Novel Data-Efficient Double DQN Framework for Intelligent Financial Portfolio Management,” *Engineering Applications of Artificial Intelligence*, Volume 162, Part B, 20 December 2025, 112436.

[2] Z. Jiang, D. Xu, and J. Liang, “A Deep Reinforcement Learning Framework for Financial Portfolio Management,” *arXiv preprint arXiv:1706.10059v2*, 2017.

[3] Lu Chung, “Evaluation of Deep Reinforcement Learning Algorithms for Portfolio Optimisation,” *arXiv preprint arXiv:2307.07694v3*, 2025.

[4] Li, H. & Hai, M., “Deep Reinforcement Learning Model for Stock Portfolio Management Based on Data Fusion”, Neural Process Lett 56, 108 (2024)

[5] Huang, G., Zhou, X., Song, Q., “Deep RL for Long–Short Portfolio Optimization”, arXiv:2012.13773

[6] Choudhary, H., Orra, A., Sahoo, K., Thakur, M., “Risk-Adjusted Deep RL for Portfolio Optimization: A Multi-Reward Approach”, Int J Comput Intell Syst 18, 126 (2025)

[7] Liu, S. (2024), “An Evaluation of DDPG, TD3, SAC, and PPO: Deep Reinforcement Learning Algorithms for Controlling Continuous Systems”, Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023)

[8] Pawar, A. A., Muskawar, V. P., Tiku, R., “Portfolio Management using Deep Reinforcement Learning”, arXiv:2405.01604v1, 1 May 2024

[9] Jiang, C., Wang, J., “A Portfolio Model with Risk Control Policy Based on Deep Reinforcement Learning”, Mathematics 2023, 11, 19. https://doi.org/10.3390/math11010019

[10] Kayumov, R. A., “Deep Reinforcement Learning for Investment Portfolio Rebalancing”, 10.13140/RG.2.2.13275.75042 (2023)