

Deep Reinforcement Learning for Portfolio Management

RL Project Group 4

Table of contents

Abstract	2
1. Introduction	2
2. System Overview	3
3. Data and Experimental Setup	4
3.1 Data Universe	4
3.2 Action Spaces	5
3.3 Algorithms and Implementation	6
3.4 Evaluation Metrics	6
4. Experiments	7
4.1 Experiment 1 – Single-Asset SPY (PPO / A2C / DQN vs Buy-and-Hold)	7
4.1.1 Setup	7
4.1.2 Results	7
4.2 Experiment 2 – Multi-Asset Uptrend, No Cash	8
4.2.1 Setup	8
4.2.2 Results	8
4.3 Experiment 3 – Three Regimes with Cash (Enhanced DQN + Templates)	9
4.3.1 Setup	9
4.3.2 Results – Uptrend with Cash	10
4.3.3 Results – Range-Bound Basket	11
4.3.4 Results – Downtrend Basket	11
5. Discussion	12
5.1 Why Is Beating SPY So Hard?	12
5.2 Equal-Weight as a “Boss Level” Baseline	13

5.3 RL vs. Financial Reality	13
6. Practical Takeaways for a Quant Team	14
7. Limitations and Future Work	14
8. Conclusion	15

Abstract

Deep reinforcement learning (DRL) has achieved impressive results in games and robotics, and there is a growing literature applying similar methods to trading and portfolio management ([Mnih et al. 2015](#); [Schulman et al. 2017](#); [Jiang, Xu, and Liang 2017](#)). At the same time, practical asset managers know that simple heuristics such as buy-and-hold or equal-weight allocations are surprisingly hard to beat out-of-sample, especially once we penalize risk and transaction costs ([DeMiguel, Garlappi, and Uppal 2009](#)).

This project builds a fully reproducible DRL backtesting pipeline for daily U.S. equities and systematically evaluates four algorithms—PPO, A2C, vanilla DQN and an enhanced DQN—across three scenarios:

1. Single-asset market timing on SPY versus buy-and-hold.
2. Multi-asset allocation across 10 large-cap “uptrend” stocks versus an equal-weight portfolio and the S&P 500.
3. Cash-aware allocation in three different regimes (uptrend, range-bound and downtrend baskets) using a template-based enhanced DQN.

Across all experiments, our main result is negative but informative: after transaction costs and risk penalties, DRL policies rarely deliver a robust edge over buy-and-hold or equal-weight benchmarks in realistic out-of-sample windows (2010–2022 train, 2023–2025 test). In several cases the DRL agent effectively rediscovers buy-and-hold; in others, an equal-weight $1/N$ portfolio remains extremely competitive. We conclude with practical lessons for deploying DRL in production: where it helps, where it is fragile, and how it might be better used as a layer on top of robust portfolio building blocks rather than as a standalone alpha engine.

1. Introduction

Reinforcement learning (RL) provides a general framework for learning decision policies from interaction with an environment ([Sutton and Barto 2018](#)). In the last decade, deep reinforcement learning (DRL) has produced striking successes: human-level control on Atari ([Mnih et al. 2015](#)), strong performance in continuous-control robotics and simulated locomotion ([Schulman et al. 2017](#)), and numerous domain-specific applications.

Finance has naturally become a target for DRL applications. Surveys such as Meng and Khushi (2019) and Fischer (2018) document a rapidly growing literature where agents learn trading rules, execution strategies, or dynamic portfolio allocations (Deng et al. 2017; Jiang, Xu, and Liang 2017; Zhang, Zohren, and Roberts 2020). Many of these papers show promising backtest results, often outperforming classical benchmarks on specific datasets.

However, practitioners remain skeptical. Real markets are noisy, non-stationary and expensive to trade. A key question for a practical quant team is not “can DRL work in *some* backtest?” but rather “does DRL reliably beat simple, robust baselines after costs, across regimes, on realistic data?”

This project tackles that question in a narrowly defined but transparent setting:

- **Daily frequency.** We work with daily OHLCV data, which aligns with many medium-term equity strategies and avoids microstructure issues.
- **Liquid U.S. equities.** Experiments use SPY and a basket of large, well-known stocks (e.g., NVDA, AAPL, TSLA, GS, BAC, XOM, WMT, KO, UNH, MCD).
- **Strong baselines.** We compare against SPY buy-and-hold and $1/N$ equal-weight portfolios, which are known to be very competitive out-of-sample (DeMiguel, Garlappi, and Uppal 2009).
- **Open-source stack.** Algorithms are implemented with Stable-Baselines3 (Raffin et al. 2021) in a custom Gymnasium environment, with all training and evaluation code in this repository.

The goal is deliberately modest but practically important: **stress-test DRL portfolio strategies against strong, simple baselines and document what they really learn.**

2. System Overview

The repository is organized around a clear separation of concerns:

- **Data loading (src/data/).**
 - `single_asset_loader.py` pulls OHLCV data for a single ticker (e.g., SPY).
 - `multi_asset_loader.py` builds price and return matrices for a list of tickers using a common calendar.
- **Environments (src/envs/).**
 - `single_asset_env.py`: continuous position in a single asset, with weights in $[-1, 1]$ (short to long), rebalanced daily.

- `multi_asset_portfolio_env.py`: multi-asset portfolio without explicit cash; weights across N stocks sum to 1.

- `multi_asset_template_env.py` and `multi_asset_template_cash_env.py`: discrete, template-based action spaces where the agent chooses from a small menu of portfolio “styles” (keep, equal-weight, growth tilt, defensive tilt, high-cash).

- **Models (`src/models/`).**

- Training scripts for single-asset PPO, A2C and DQN.

- Training scripts for multi-asset PPO/A2C/DQN and an enhanced DQN (dueling architecture, Double Q-learning, prioritized replay, template actions).

- A dedicated script for enhanced DQN with explicit cash templates.

- **Backtesting & plotting (`src/backtest/`, `src/utils/`).**

- Evaluation scripts generate equity curves, metric tables (annualized return, volatility, Sharpe, max drawdown, etc.) and CSV logs.

- Plotting utilities create the figures used in this report.

All experiments share a **global time split**:

- Training: **2010-01-01 to 2022-12-31**

- Evaluation: **2023-01-01 to 2025-11-12**

Daily rewards are designed to encourage **high risk-adjusted return with low turnover**: the reward combines portfolio return with penalties for volatility and drawdowns, plus explicit transaction costs. This is intended to discourage over-trading and to push policies toward smoother equity curves.

3. Data and Experimental Setup

3.1 Data Universe

We use daily OHLCV data for U.S. listed securities obtained via `yfinance` (as implemented in the loaders). The specific universes per experiment are:

- **Experiment 1 – Single-asset SPY.**

- Asset: SPY (S&P 500 ETF).

- Task: learn to time SPY versus a buy-and-hold baseline.
- **Experiment 2 – Multi-asset uptrend, no cash.**
 - Assets: NVDA, AAPL, TSLA, GS, BAC, XOM, WMT, KO, UNH, MCD.
 - All have strong long-term performance, giving the equal-weight portfolio a structural tailwind.
- **Experiment 3 – Regime-specific baskets with cash.**
 - Uptrend basket (same 10 names as Experiment 2).
 - Range-bound basket: INTC, F, AA.
 - Downtrend basket: VEON, PRGO.

For each universe, the environment builds a rolling window of the last ($W=30$) trading days as the state input, together with the current portfolio weights and any risk features used for the reward. The exact feature engineering is deliberately kept moderate to mimic what a practical team might deploy without over-fitting a huge feature space.

3.2 Action Spaces

We study two main classes of action spaces:

1. **Continuous weights (Experiments 1–2).**
 - Single-asset: a scalar position in $[-1, 1]$, interpreted as full short to full long on SPY.
 - Multi-asset: a weight vector w in R^N with components $w_i \geq 0$ and $\sum_i w_i = 1$, rebalanced daily.
2. **Template-based discrete actions with cash (Experiment 3).**
 In `multi_asset_template_cash_env.py` the agent chooses among a small set of portfolio “templates,” for example:
 - Keep current weights.
 - Equal-weight across risky assets (0% cash).
 - Growth tilt (over-weight high-beta stocks).
 - Defensive tilt (over-weight lower-beta or defensive names).
 - High-cash template (e.g., 70% cash, 30% risky assets).

Templates make the action space more interpretable and reduce the dimensionality of the control problem, at the cost of limiting flexibility.

3.3 Algorithms and Implementation

All RL algorithms are implemented using Stable-Baselines3 ([Raffin et al. 2021](#)):

- **PPO (Proximal Policy Optimization).** A clipped policy-gradient method designed for stable updates ([Schulman et al. 2017](#)).
- **A2C (Advantage Actor–Critic).** A synchronous variant of A3C with value-based baselines ([Mnih et al. 2016](#)).
- **DQN (Deep Q-Network).** Value-based method using a neural approximator for Q-values, inspired by early Atari work ([Mnih et al. 2015](#)).
- **Enhanced DQN.** DQN augmented with:
 - Dueling architecture (separate value and advantage streams),
 - Double Q-learning,
 - Prioritized experience replay, and
 - Template-based discrete actions.

Hyperparameters are chosen to be “reasonable but not cherry-picked,” with typical settings such as:

- `total_timesteps` = 300k for PPO/A2C and up to 1M for enhanced DQN.
- Transaction cost of 10 bps per unit turnover.
- Discount factor = 0.99 and moderate entropy regularization for policy methods.

We emphasize robustness over squeezing out the last few basis points.

3.4 Evaluation Metrics

For each policy and baseline we compute:

- Cumulative and annualized return.
- Annualized volatility.
- Sharpe ratio (using risk-free rate = 0).
- Maximum drawdown.

- Turnover (average absolute daily weight change).

These metrics are reported for the out-of-sample period (2023-01-01 to 2025-11-12) only.

4. Experiments

4.1 Experiment 1 – Single-Asset SPY (PPO / A2C / DQN vs Buy-and-Hold)

4.1.1 Setup

- Asset: SPY.
- Train: 2010-01-01 to 2022-12-31.
- Test: 2023-01-01 to 2025-11-12.
- Algorithms: PPO, A2C, DQN.
- Baseline: SPY buy-and-hold.

The agent can be anywhere between fully short and fully long SPY, with daily rebalancing and transaction costs. Intuitively, if SPY shows a strong long-term uptrend and timing is hard, the optimal policy may simply be “stay long most of the time.”

4.1.2 Results

The equity curves for PPO, A2C and DQN are nearly indistinguishable from buy-and-hold in the evaluation window. A typical example is shown in Figure [Figure 1](#).

Key observations:

- All three algorithms converge to **low-turnover, almost always long** policies.
- Out-of-sample Sharpe ratios and drawdowns are extremely close to buy-and-hold.
- Attempts by the agents to time short-term corrections typically hurt performance once costs and risk penalties are applied.

In other words, the RL agents empirically rediscover buy-and-hold as an approximately optimal policy for a single efficient index over a long bull market, consistent with financial theory and prior empirical work on index efficiency.

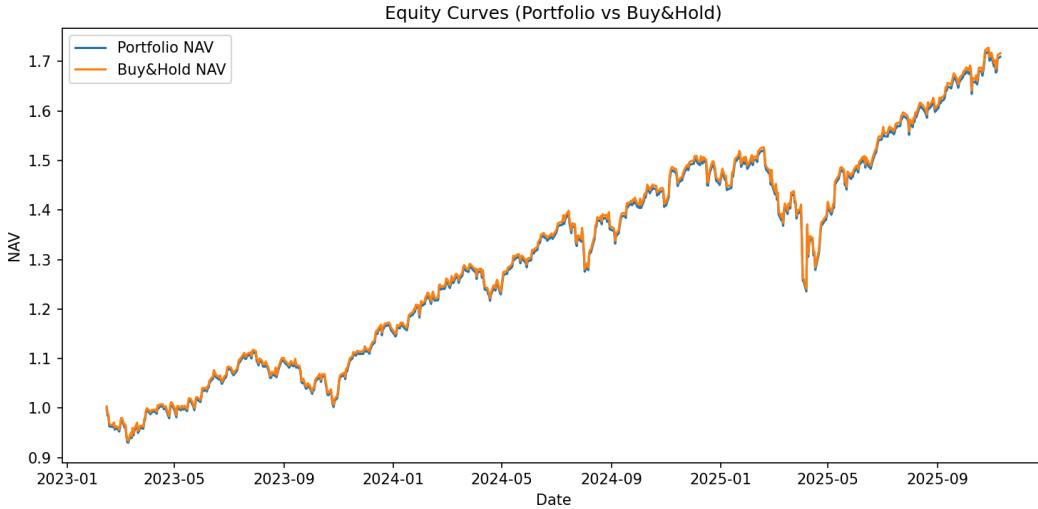


Figure 1: Single-asset SPY: PPO vs SPY buy-and-hold (test period).

4.2 Experiment 2 – Multi-Asset Uptrend, No Cash

4.2.1 Setup

- Assets: NVDA, AAPL, TSLA, GS, BAC, XOM, WMT, KO, UNH, MCD.
- Train/test split: same as Experiment 1.
- Algorithms: PPO, A2C, vanilla DQN, enhanced DQN.
- Baselines:
 - 1/N equal-weight portfolio over the 10 stocks.
 - S&P 500 index.

The agent now allocates across multiple strong performers, always fully invested (no explicit cash). The question becomes: **can DRL exploit cross-sectional structure to beat 1/N and SPY after costs?**

4.2.2 Results

Figure Figure 2 shows the normalized NAV of all algorithms and benchmarks.

Qualitatively:

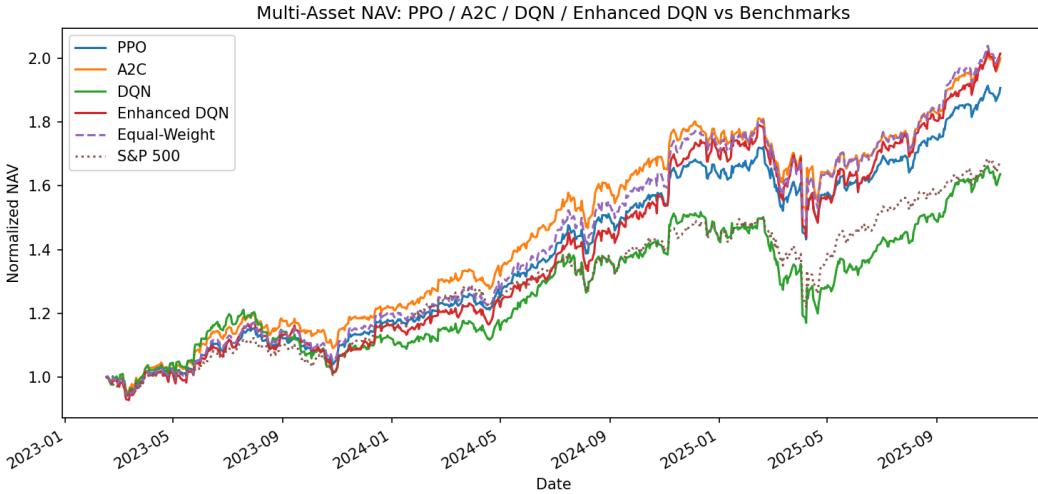


Figure 2: Multi-asset uptrend: PPO / A2C / DQN / enhanced DQN vs equal-weight and S&P 500 (test period).

- The equal-weight portfolio across these 10 strong names is already a **very strong baseline**, delivering high returns and diversification.
- PPO and A2C produce sensible allocations and track equal-weight reasonably well but **do not convincingly dominate it on a risk-adjusted basis**.
- Vanilla DQN tends to over-react; its NAV curve exhibits higher noise and occasional underperformance.
- The enhanced DQN manages to **slightly outperform** $1/N$ over the evaluation window, but the edge is modest and likely fragile.

This result mirrors findings in the portfolio-optimization literature: once you account for estimation error and transaction costs, sophisticated optimization often fails to deliver a large and reliable advantage over $1/N$ ([DeMiguel, Garlappi, and Uppal 2009](#)).

4.3 Experiment 3 – Three Regimes with Cash (Enhanced DQN + Templates)

4.3.1 Setup

To stress-test the approach, we introduce explicit cash and regime-specific universes:

- **Uptrend basket:** same 10-stock universe as Experiment 2.

- **Range-bound basket:** INTC, F, AA.
- **Downtrend basket:** VEON, PRGO.

In each case, enhanced DQN acts in the template-and-cash environment:

- Weights across stocks plus cash sum to 1.
- The action is one of five templates (keep, equal-weight, growth, defensive, high-cash).
- The reward remains risk-adjusted return minus costs.

The ideal behavior in the downtrend basket, for example, would be to **discover the high-cash template and stay there most of the time**.

4.3.2 Results – Uptrend with Cash

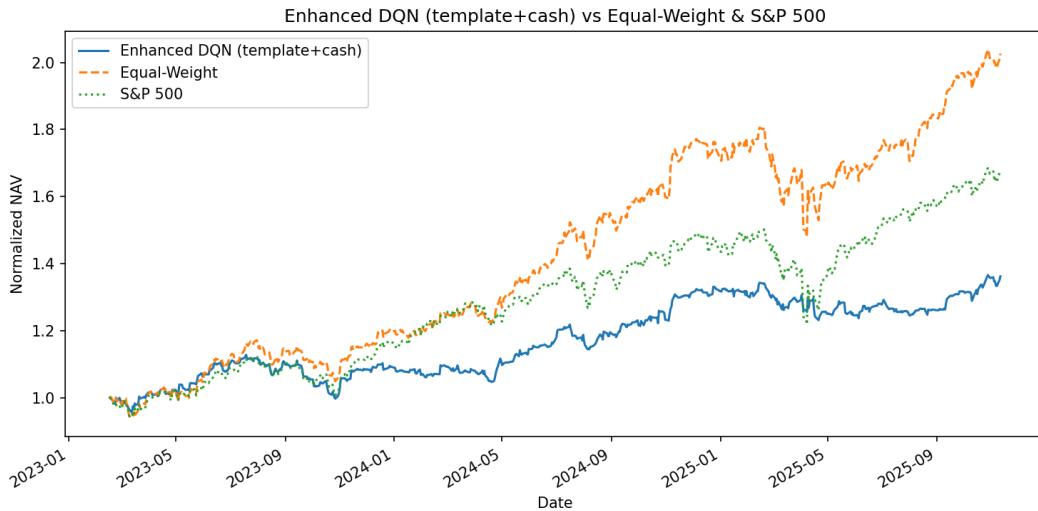


Figure 3: Uptrend basket with cash: enhanced DQN (template+cash) vs equal-weight and S&P 500.

- Equal-weight remains extremely competitive.
- The DRL agent frequently holds **too much cash** in strong uptrends or rotates between styles, generating avoidable transaction costs.
- As a result, enhanced DQN typically **lags** equal-weight in total return, even if drawdowns are sometimes slightly smaller.

4.3.3 Results – Range-Bound Basket

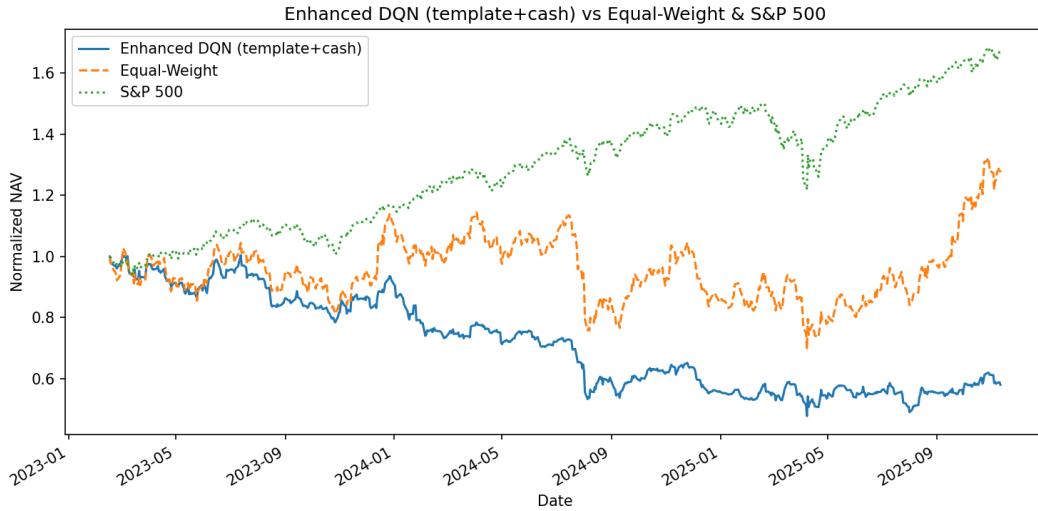


Figure 4: Range-bound basket: enhanced DQN (template+cash) vs equal-weight and S&P 500.

In a noisy, sideways market:

- Prices exhibit no clear long-term trend; short-term moves are dominated by noise.
- The agent struggles to extract a stable signal and **tends to churn** between templates, incurring cost without gaining much edge.
- Equal-weight with low turnover again looks surprisingly robust.

4.3.4 Results – Downtrend Basket

In a persistent downtrend:

- Ideally, the agent should learn “stay in high-cash most of the time.”
- In practice, the DQN often:
 - Over-reacts to short bear-market rallies,
 - Oscillates between risk-on and risk-off templates, and
 - Fails to commit to a mostly-cash stance for extended periods.

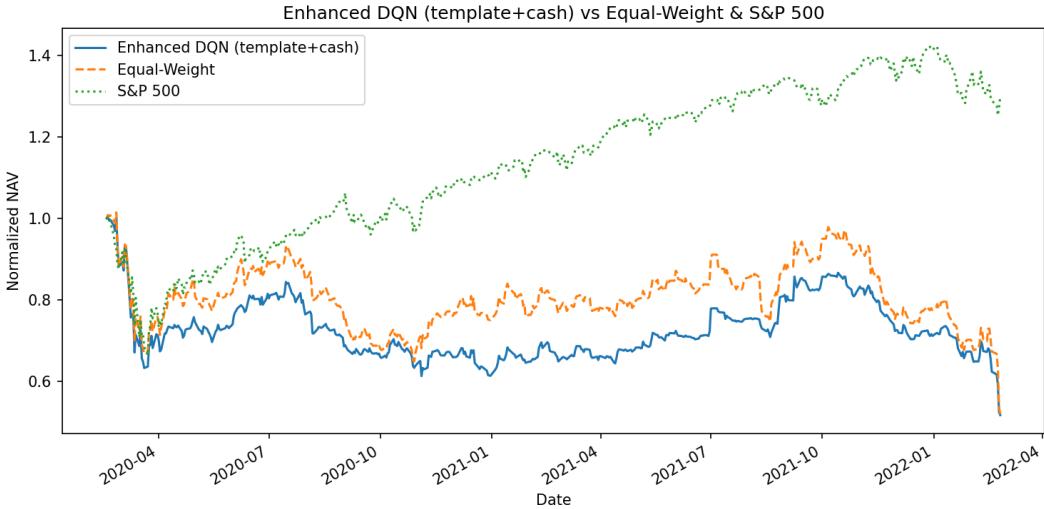


Figure 5: Downtrend basket: enhanced DQN (template+cash) vs equal-weight and S&P 500.

The result is that enhanced DQN typically performs **better than a fully invested portfolio**, but still not clearly better than equal-weight. The agent appears to “know” that cash is useful, but has trouble learning a stable, regime-appropriate policy.

5. Discussion

5.1 Why Is Beating SPY So Hard?

SPY tracks the S&P 500, a broad, liquid, institutionally-dominated index. The Efficient Market Hypothesis suggests that consistently outperforming such indices, net of risk and transaction costs, is extremely difficult.

Our single-asset results align with this view:

- The best DRL policies are **indistinguishable from buy-and-hold** in the evaluation window.
- Trying to time corrections yields minimal benefit once we account for costs and the noise in daily returns.
- From a portfolio-construction perspective, this is a reassuring result: DRL does not magically extract alpha where long-only exposure is already close to optimal.

5.2 Equal-Weight as a “Boss Level” Baseline

The $1/N$ portfolio has a long history as a surprisingly strong baseline in portfolio theory ([DeMiguel, Garlappi, and Uppal 2009](#)). The intuition is simple: when estimates of expected returns and covariances are noisy, sophisticated optimizers tend to overfit; equal-weight avoids this by refusing to “optimize” at all.

Our multi-asset experiments reinforce this lesson:

- Even in a curated uptrend basket, **$1/N$ is hard to beat**.
- Enhanced DQN can nudge ahead in one favorable window, but the margin is small and sensitive to regime and hyperparameters.
- When we add cash and more complex regimes, equal-weight remains competitive or superior in many cases.

For any production use, this strongly suggests that DRL should be judged not just against naive buy-and-hold of a single name, but against realistic, diversified baselines.

5.3 RL vs. Financial Reality

The difficulties we observe are consistent with broader critiques of DRL in financial markets ([Meng and Khushi 2019](#); [Fischer 2018](#); [Zhang, Zohren, and Roberts 2020](#)):

- **Sample inefficiency & non-stationarity.** Financial history provides limited, regime-shifted samples. Agents trained on one regime often fail in another.
- **Low signal-to-noise environments.** Daily returns embed tiny predictive signals within substantial noise. DRL can easily overfit spurious patterns.
- **Objective mismatch.** We compress rich investor objectives (drawdowns, tail risk, liquidity) into a single scalar reward and hope the agent internalizes everything.
- **Transaction costs & market impact.** In backtests we assume fixed costs and zero market impact; real-world frictions are worse.

Our negative results are therefore not a condemnation of DRL itself, but a reminder that **the environment, state, reward and baselines matter as much as the algorithm**.

6. Practical Takeaways for a Quant Team

If you are thinking about bringing DRL into a production investment process, this project suggests several practical lessons:

1. **Start with strong baselines.**

Always benchmark DRL against SPY, equal-weight, and other robust heuristics—not just against random or weak baselines.

2. **Use DRL as a layer, not a monolith.**

Instead of letting DRL directly choose raw asset weights, let it choose among a small set of pre-built, diversified model portfolios (e.g., low-vol, risk-parity, factor tilts). This reduces dimensionality and leverages decades of portfolio-theory insights.

3. **Focus on risk and turnover.**

In production, Sharpe improvement of +0.1 with double the turnover is rarely attractive. Rewards that strongly penalize turnover and drawdown are essential.

4. **Be realistic about data.**

Daily equity data from 2010–2025 is not “big data” in RL terms. Cross-validation, walk-forward evaluation, and robustness checks across universes are crucial.

5. **Invest in monitoring and diagnostics.**

Use the logging and plotting tools (equity curves, action traces, turnover) to understand *why* an agent behaves as it does, not just whether it beats a benchmark in one sample.

7. Limitations and Future Work

This project is intentionally focused and has several limitations:

- **Universe size.** We work with at most 10 stocks plus cash. Scaling to hundreds of names introduces new challenges in state and action representation.
- **Single frequency.** Only daily data are considered; intraday strategies have very different microstructure characteristics and cost profiles.
- **Simplified transaction costs.** Slippage and market impact are not modeled explicitly.
- **Limited model families.** We test PPO, A2C, DQN and an enhanced DQN, but not more recent algorithms such as SAC or distributional RL, nor model-based methods.

Promising directions for future work include:

- Adding richer state features (macro indicators, volatility regimes, sentiment signals) while controlling for over-fitting.
- Combining DRL with robust portfolio rules (e.g., DRL chooses when to tilt away from a baseline risk-parity allocation).
- Using ensembles of agents to reduce variance and improve robustness.
- Designing evaluation frameworks that emphasize walk-forward performance, hyperparameter robustness, and stress tests across multiple crises.

8. Conclusion

We built and open-sourced a complete DRL trading and portfolio backtesting stack, then deliberately asked a hard question: **when we stress-test PPO, A2C, DQN and enhanced DQN against strong baselines on real daily equity data, do they deliver a robust edge?**

The answer, in this setup, is “**not really.**” RL agents either reproduce buy-and-hold or struggle to beat equal-weight portfolios once we account for risk and transaction costs. From a research perspective, this is a valuable negative result: it highlights the importance of baselines, evaluation design, and realistic assumptions.

From a practitioner’s perspective, the message is nuanced but hopeful. DRL is unlikely to be a magic alpha engine dropped directly onto raw prices. It is more promising as a *component* of a broader portfolio system—used to switch among robust building blocks, adapt to regimes, or augment existing signals—rather than as a standalone black box.

For now, the humble equal-weight portfolio and SPY buy-and-hold remain tough opponents. Any DRL strategy claiming to beat them consistently should be treated with both interest and healthy skepticism.

DeMiguel, Victor, Lorenzo Garlappi, and Raman Uppal. 2009. “Optimal Versus Naive Diversification: How Inefficient Is the $1/n$ Portfolio Strategy?” *Review of Financial Studies* 22 (5): 1915–53.

Deng, Yue, Feng Bao, Youyong Kong, Zhiqian Ren, and Qionghai Dai. 2017. “Deep Direct Reinforcement Learning for Financial Signal Representation and Trading.” *IEEE Transactions on Neural Networks and Learning Systems* 28 (3): 653–64.

Fischer, Thomas G. 2018. “Reinforcement Learning in Financial Markets – a Survey.” FAU Discussion Papers in Economics 12/2018.

Jiang, Zhengyao, Dixin Xu, and Jinjun Liang. 2017. “A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem.” *arXiv Preprint arXiv:1706.10059*.

- Meng, Terry Lingze, and Matloob Khushi. 2019. “Reinforcement Learning in Financial Markets.” *Data* 4 (3): 110.
- Mnih, Volodymyr, Adrià Puigdomènec Badia, Mehdi Mirza, et al. 2016. “Asynchronous Methods for Deep Reinforcement Learning.” In *Proceedings of the 33rd International Conference on Machine Learning*.
- Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, et al. 2015. “Human-Level Control Through Deep Reinforcement Learning.” *Nature* 518 (7540): 529–33. <https://doi.org/10.1038/nature14236>.
- Raffin, Antonin, Ashley Hill, Adam Gleave, et al. 2021. “Stable-Baselines3: Reliable Reinforcement Learning Implementations.” *Journal of Machine Learning Research* 22 (268): 1–8.
- Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. “Proximal Policy Optimization Algorithms.” *arXiv Preprint arXiv:1707.06347*.
- Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. 2nd ed. MIT Press.
- Zhang, Zihao, Stefan Zohren, and Stephen Roberts. 2020. “Deep Reinforcement Learning for Trading.” *Journal of Financial Data Science* 2 (2): 25–40.