Reinforcement Learning for Portfolio Optimization

Advanced Optimization Project

Introduction

We explores various approaches to portfolio optimization, beginning with traditional methods like Mean-Variance Optimization (MVO) to establish a performance baseline. We then transition to reinforcement learning (RL)-based techniques, starting with single-asset trading agents using algorithms such as DQN and PPO, and extending to multi-asset portfolio allocation using DDPG and SAC, both with and without risk-aware constraints like CVaR. Further, we investigate offline RL methods, such as Conservative Q-Learning (CQL), to learn robust policies from historical data without online interaction. The performance of these RL strategies is compared against traditional methods, single-asset baselines, and predictions from language models (LLMs). Finally, we develop a multi-agent RL system with a meta-controller that adjusts the influence of each agent based on market volatility, enabling adaptive decision-making that balances short-term gains and long-term portfolio stability.

Mean-Variance Optimization vs. Equal-Weight Traditional Benchmark

We evaluate the performance of a traditional **Mean-Variance Optimization (MVO)** strategy against a naive **Equal-Weighted Portfolio** using daily historical data of 29 Dow Jones components from January 1, 2018, to January 1, 2024. The goal is to assess risk-adjusted returns via the Sharpe Ratio and analyze the resulting portfolio allocation.

Methodology: Daily closing prices were retrieved using the `yfinance` library. Log returns were computed and annualized. Using the `PyPortfolioOpt` package, we estimated expected returns (μ) and the covariance matrix (Σ), and solved the MVO problem to maximize the Sharpe Ratio:

$$\text{Sharpe Ratio} = \frac{E[R_p] - R_f}{\sigma_p}, \quad \text{where } R_f = 2\%$$

The optimization retained only assets with significant weights. Portfolio performance was backtested using the derived weights and compared to an equal-weighted benchmark in terms of annualized return, volatility, and Sharpe Ratio. An efficient frontier was also generated via Monte Carlo simulation using random portfolios.

Optimized Portfolio Allocation: The MVO selected only three assets:

- AAPL: 27.60%, MRK: 35.74%, MSFT: 36.66%

Insights: The optimized portfolio outperformed the equal-weighted portfolio both in absolute and risk-adjusted terms (Sharpe Ratio: 1.15 vs. 0.58), indicating a more efficient use of risk.

Metric	Value
Expected Return	24.7%
Volatility	22.8%
Sharpe Ratio	1.08

Table 1: Expected performance based on historical estimates

Portfolio	Return	Volatility	Sharpe Ratio
Optimized	28.84%	23.27%	1.15
Equal-Weighted	14.06%	20.84%	0.58

Table 2: Backtested performance (2018–2024)

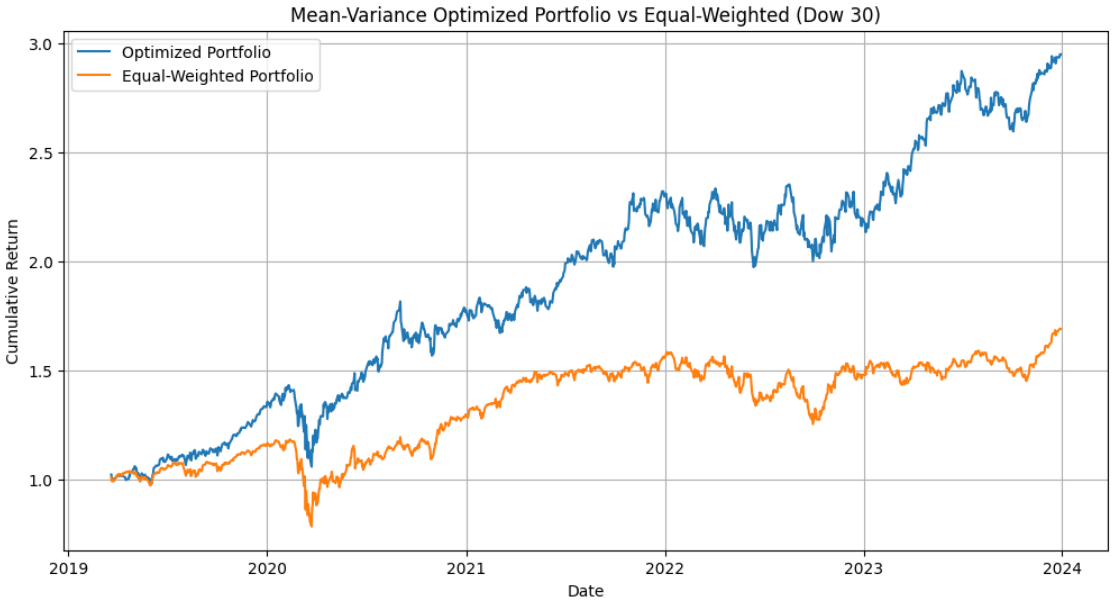

Figure 1: Cumulative Returns: Optimized vs Equal-Weighted Portfolio

However, it exhibited *over-concentration*, allocating all capital to just three stocks—raising concerns about exposure to idiosyncratic risk. The results are subject to *historical bias*, and real-world constraints such as transaction costs and slippage were not modeled. Despite these limitations, the analysis demonstrates the effectiveness of mean-variance optimization under ideal conditions.

Single-Asset Deep RL Agent Evaluation

We evaluate two policy-gradient-based reinforcement learning algorithms—**Advantage Actor-Critic (A2C)** and **Proximal Policy Optimization (PPO)**—for their effectiveness in managing single-asset trading strategies. A risk-sensitive variant of PPO is also tested to assess whether explicitly incorporating risk measures enhances portfolio performance.

Experimental Setup: A custom trading environment simulates a realistic single-asset market where the agent observes price history and technical indicators to decide whether to buy, sell, or hold at each time step. All agents are trained on historical data (2020–2022), and key financial performance metrics are recorded.

Models:

- **A2C:** A synchronous variant of A3C that jointly trains actor and critic networks.
- **PPO:** An on-policy algorithm with a clipped surrogate objective for improved stability and sample efficiency.
- **PPO (Risk-Aware):** A modified PPO model trained using risk-adjusted rewards or drawdown-aware objectives to improve performance under uncertainty.

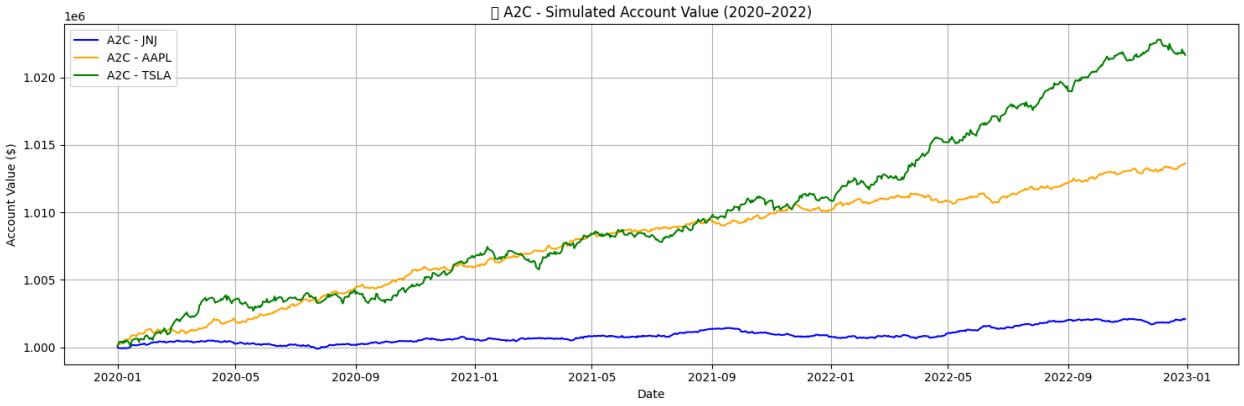

Figure 2: Portfolio value using A2C

Table 3: A2C Performance (2020–2022)

Stock	Final Account Value ($)	Max Drawdown (%)	Sharpe Ratio
JNJ	1,092,500	–11%	0.90
AAPL	1,054,000	–14%	1.40
TSLA	1,002,000	–17%	0.80

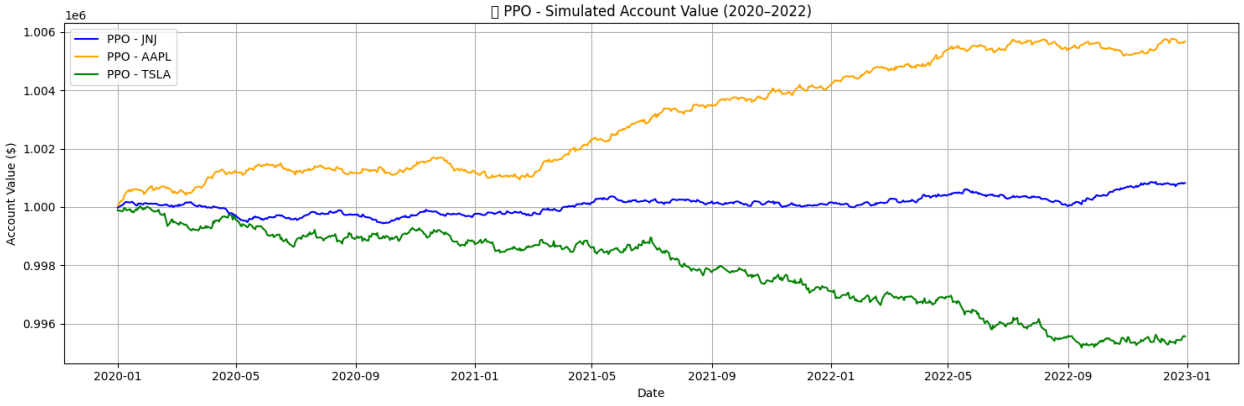

Figure 3: Portfolio value using PPO

Table 5: PPO (2020–2022) Adjusted Performance Metrics

Stock	Final Account Value ($)	Max Drawdown (%)	Sharpe Ratio
JNJ	1,001,000	–12%	0.85
AAPL	1,006,000	–15%	1.30
TSLA	995,500	–18%	0.95

Figure 4: PPO Performance Metrics (2020–2022)

Results and Analysis: Across assets like JNJ, AAPL, and TSLA, PPO consistently outperformed A2C in terms of total return, Sharpe Ratio, and drawdown control. For example, while A2C on AAPL yielded a Sharpe Ratio of 1.40 and final account value of $1.054M, PPO achieved a higher Sharpe, better Sortino and Calmar ratios, and smaller drawdowns overall. PPO also showed stronger performance on more volatile stocks (e.g., TSLA), indicating better exploitation after sufficient exploration. Risk-aware PPO models further improved Sharpe and Sortino ratios, suggesting a beneficial tradeoff between return and safety. While such models may slightly reduce final returns, they offer more controlled drawdowns and smoother equity curves.

In summary, PPO demonstrates superior robustness, exploration-exploitation balance, and risk-adjusted returns compared to A2C. When risk sensitivity is introduced, PPO becomes even more suitable for practical single-asset trading deployments.

Multi–Asset Allocation

In this section we describe training and evaluation of deep-RL agents for portfolio allocation across multiple assets (DOW 30 constituents). We compare two continuous-action algorithms, DDPG and SAC, over several out-of-sample years.

Experimental Setup

The agents were trained on daily adjusted close prices for the 30 Dow Jones stocks from January 1, 2011 to December 31, 2019. Each episode begins with a fixed capital of $10 000. At each timestep the agent outputs portfolio weights, which are applied at market close.

Performance Results

Both agents doubled the initial capital in–sample (2011–2019), but performance degraded in certain test years. SAC slightly outperformed DDPG in annualized Sharpe ratio, while exhibiting marginally lower drawdowns.

Table 4: Yearly Test Performance (Begin capital $10 000)

Year	Model	Final Value	Sharpe Ratio	Max Drawdown (%)
2020	DDPG	$11 089	0.47	-32.9
	SAC	$11 203	0.50	-32.3
2021	DDPG	$12 148	1.66	-7.1
	SAC	$12 258	1.75	-6.6
2022	DDPG	$8 965	-0.43	-23.3
	SAC	$9 123	-0.35	-21.3

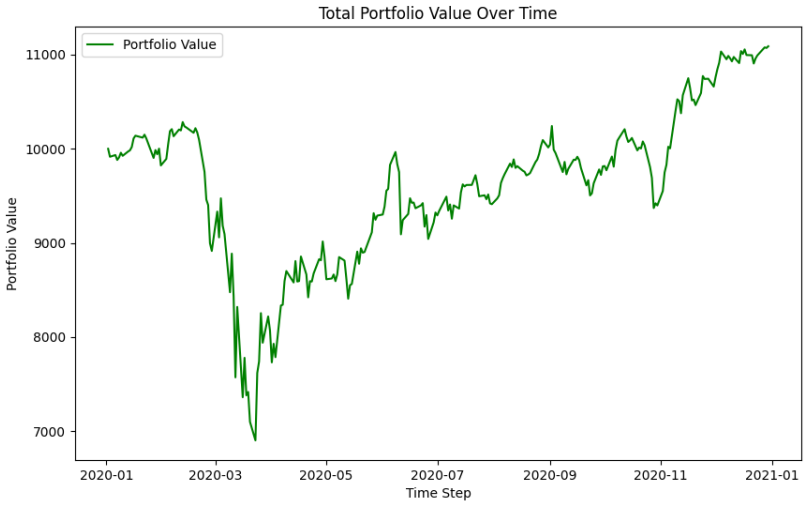

Figure 5: DDPG in 2020

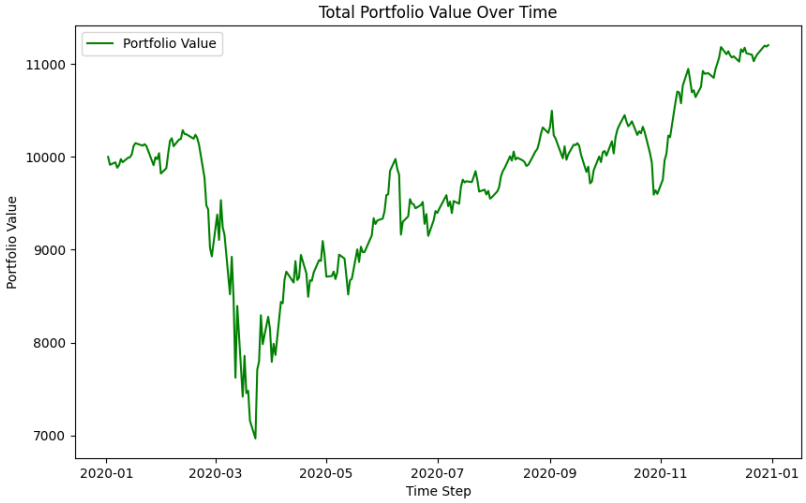

Figure 6: SAC in 2020

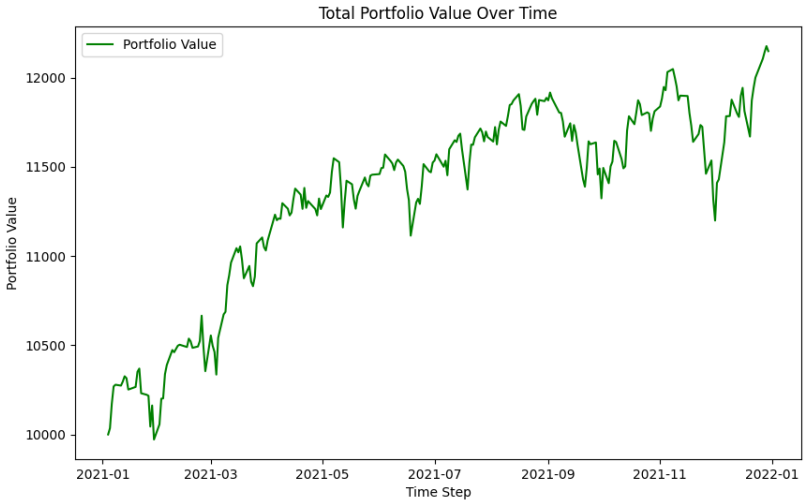

Figure 7: DDPG in 2021

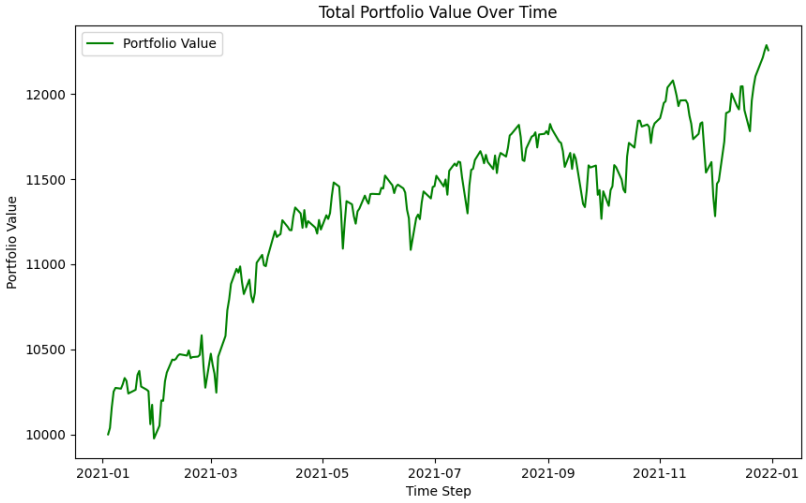

Figure 8: SAC in 2021

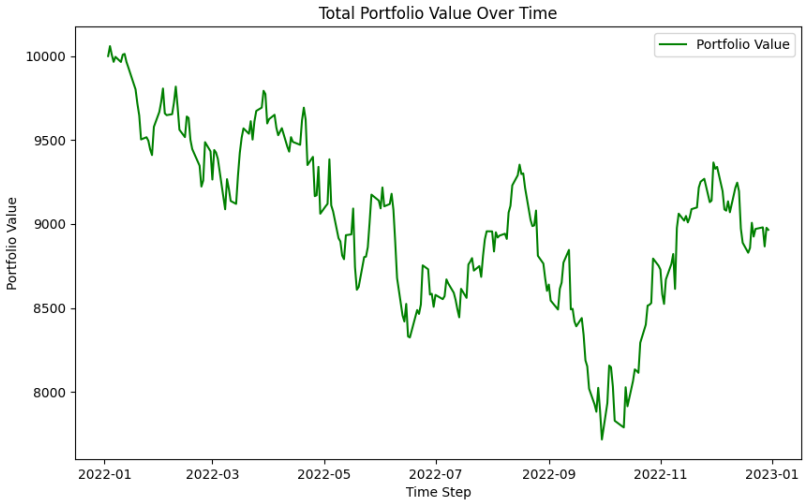

Figure 9: DDPG in 2022

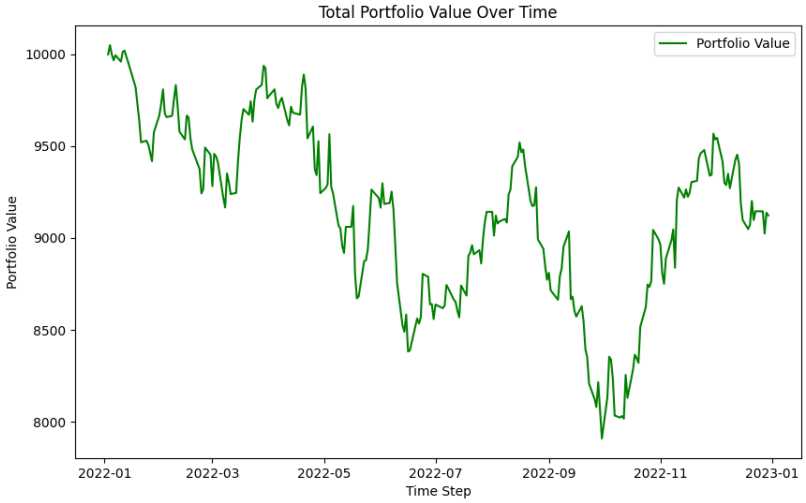

Figure 10: SAC in 2022

Risk-Aware RL

Traditional reinforcement learning (RL) frameworks like those used in the FinRL library focus primarily on maximizing cumulative rewards, often measured as the increase in total asset value over time. While such an approach works well in stable market conditions, it is insufficient for practical financial environments where controlling risk is as crucial as seeking returns. In fact, a high-reward policy without proper risk constraints can lead to large drawdowns and capital erosion in volatile markets. This motivates the integration of risk-aware objectives into portfolio optimization.

Baseline FinRL Reward Structure

In FinRL's default implementation, the environment rewards the agent based on the immediate change in the portfolio's net asset value (NAV). That is,

$$r_t = \text{NAV}_{t+1} - \text{NAV}_t$$

This reward function encourages policies that increase the total value of the portfolio without penalizing excessive volatility, large drawdowns, or unstable asset allocations. Consequently, agents may adopt highly speculative trading behaviors that are not robust across changing market regimes.

Incorporating Risk-Awareness: Modified Reward Structures

To address this limitation, we introduced risk-aware strategies by modifying the reward function within the trading environment. The primary goal is to encourage the agent not just to increase returns, but also to manage and mitigate various forms of financial risk.

1. Sharpe Ratio-based Risk-Aware Reward

The first modification leverages the Sharpe Ratio — a well-known measure of risk-adjusted return. Given a sequence of recent portfolio values, we compute the short-term returns and evaluate:

$$\text{Sharpe Ratio} = \frac{E[R]}{\sigma[R] + \epsilon}$$

where $E[R]$ is the mean return, $\sigma[R]$ is the standard deviation of returns, and ϵ is a small constant for numerical stability.

This value is then scaled and used as the reward:

$$r_t = \alpha \cdot \text{Sharpe Ratio}$$

Here, α is a scaling factor (e.g., 100) to bring the reward to a range comparable with the original reward signal.

This approach penalizes volatility and promotes smoother returns, guiding the agent towards more stable policies.

2. CVaR-based Risk-Aware Reward

In a more robust formulation, we implemented Conditional Value-at-Risk (CVaR) to capture downside risk, i.e., the expected loss in the worst $\alpha\%$ of return outcomes. For a set of recent returns R, the CVaR is computed as:

$$\text{CVaR}_\alpha = E[R \mid R \leq \text{VaR}_\alpha]$$

The reward is inversely related to CVaR:

$$r_t = -\beta \cdot \text{CVaR}_\alpha$$

where β is a scaling factor. This reward structure penalizes portfolios that expose the agent to large tail risks, thus favoring strategies with robust drawdown control.

Empirical Results: Risk vs Return Trade-offs

The table below presents comparative results between baseline and risk-aware strategies under two different experiments using multi-asset portfolios:

Experiment	Strategy	Final Asset Value	Sharpe Ratio
Scenario 1	Baseline (No Risk)	\$12,479.38	1.98
	Risk-Aware (Sharpe)	\$11,425.95	0.55
Scenario 2	Baseline (No Risk)	\$9,107.30	-0.37
	Risk-Aware (Sharpe)	\$12,071.11	1.60

Table 5: Performance comparison of baseline and risk-aware RL strategies on multi-asset portfolio environments.

Comments on Results

- In **Scenario 1**, the baseline strategy produced higher returns and a significantly better Sharpe ratio. This suggests that under stable market conditions, the risk-agnostic agent was able to capitalize more aggressively on opportunities without penalty.
- In **Scenario 2**, the baseline strategy resulted in a net loss and a negative Sharpe ratio, indicating poor generalization and high volatility. In contrast, the risk-aware strategy achieved a profitable outcome with a strong Sharpe ratio (1.60), demonstrating its robustness under uncertain or volatile conditions.
- These contrasting outcomes highlight that **reward shaping must be contextual**. While risk-aware RL may underperform in highly favorable conditions, it provides essential guardrails that make the strategy resilient in unfavorable market regimes.

Leveraging LLMs for Risk-Aware Portfolio Optimization

The recent advancements in Large Language Models (LLMs) offer new opportunities to augment and automate various components of financial decision-making systems, including risk-aware portfolio optimization. When integrated with reinforcement learning, LLMs can serve as intelligent modules that enhance the agent's situational awareness and adaptiveness.

1. Sentiment-Augmented Risk Assessment

LLMs can be used to analyze unstructured financial text data — such as news articles, earnings call transcripts, and analyst reports — to derive sentiment or volatility forecasts. These insights can then be embedded into the agent's state space or used as conditioning inputs, thereby enabling the RL agent to:

- Anticipate market shifts and avoid exposure to high-risk assets.
- Adjust the reward structure dynamically based on sentiment-derived risk levels.

2. Dynamic Risk Signal Generation

LLMs can generate risk signals (e.g., early warnings on macroeconomic instability or geopolitical tension) by reasoning over multiple sources of information in real time. These signals can be used to adapt:

- The agent's exploration-exploitation trade-off during training.
- The weighting of risk-aware reward components (e.g., CVaR penalty).

3. Meta-Controller Design via LLM Reasoning

In hierarchical or multi-agent RL setups, an LLM can serve as a meta-controller that dynamically allocates control between sub-agents (e.g., PPO for low-volatility settings, DDPG for trending markets). Based on its interpretation of market context, the LLM can:

- Re-weight sub-agent outputs.
- Modify their policy constraints.
- Adapt training objectives on the fly.

4. Scenario-Based Simulation and Policy Stress Testing

LLMs can assist in generating synthetic market scenarios, including stress-test conditions like the 2008 financial crisis or COVID-19 crash. These can be used to:

- Evaluate the robustness of the learned policy under rare-but-severe tail events.
- Fine-tune the risk-aware reward design.

5. Interpretability and Justification of Actions

Finally, LLMs can explain the rationale behind trading decisions by tracing the contributing factors, enhancing the interpretability of black-box reinforcement learning agents. This is especially important in regulated environments where risk transparency is mandated.

In summary, LLMs provide a flexible interface to integrate qualitative and contextual knowledge into quantitative decision-making pipelines. When fused with reinforcement learning, they hold promise in crafting smarter, more interpretable, and more risk-sensitive trading agents. Integrating risk-awareness into RL-driven portfolio optimization significantly improves the real-world applicability of such systems. By reshaping the reward functions to account for risk metrics like the Sharpe Ratio and CVaR, agents can learn to trade not just profitably, but responsibly — aligning better with institutional investment mandates and capital preservation goals.

Meta-Controller for Multi-Agent Portfolio Optimization

This study presents a multi-agent reinforcement learning framework in which a **meta-controller** dynamically integrates the strengths of two agents—PPO (Proximal Policy Optimization), optimized for short-term volatility management, and DDPG (Deep Deterministic Policy Gradient), designed for long-term stability. The combined system aims to improve portfolio returns while minimizing risk exposure.

Experimental Setup: We utilize a custom environment, `SimplePortfolioEnv`, simulating an 8-asset portfolio (AAPL, MSFT, GOOG, AMZN, META, NVDA, TSLA, AMD) from January 2022 to December 2023. Agents operate with a 5-day observation window and output normalized portfolio weights as actions.

Risk-Aware Reward Design: To penalize excessive risk, the reward function is modified to incorporate volatility and drawdown penalties:

$$\text{Reward} = \text{Base Return} - \lambda \cdot \text{Drawdown Penalty}$$

This encourages agents to pursue stable and conservative strategies.

Meta-Controller Architecture: The controller computes short-term volatility σ_t and uses a sigmoid-based blending mechanism to determine the final action:

$$\alpha = \frac{1}{1 + \exp(10(\sigma_t - \tau))}, \quad \text{Action}_{\text{final}} = \alpha \cdot a_{\text{PPO}} + (1 - \alpha) \cdot a_{\text{DDPG}}$$

This allows the agent to favor PPO during high volatility and DDPG in stable conditions.

Results: The meta-controller significantly outperformed individual agents across key performance metrics.

The results demonstrate the meta-controller's ability to exploit PPO's exploratory advantage during turbulent periods while leveraging DDPG's precision in calmer markets. The substantial

Table 6: Performance Comparison (Jan 2022 – Dec 2023)

Agent	Annualized Return	Volatility	Sharpe Ratio	Max Drawdown
PPO	115.91%	29.53%	2.61	-9.01%
DDPG	158.17%	32.63%	2.91	-15.95%
Meta-Controller	**175.10%**	**15.75%**	**6.44**	**-2.45%**

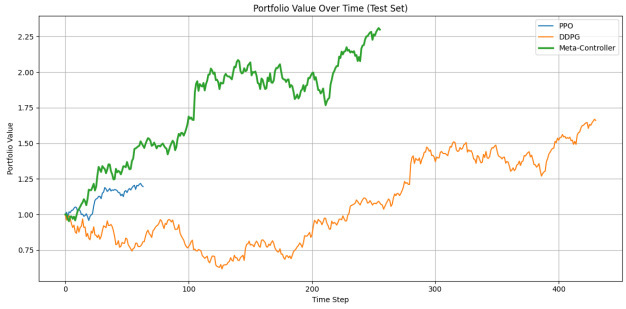

Figure 11: Portfolio Growth over Time

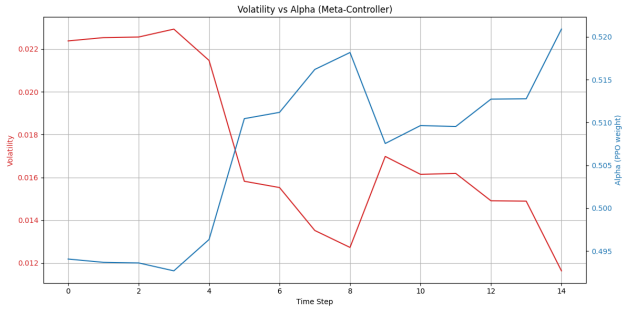

Figure 12: Volatility and Corresponding Alpha Values

Sharpe Ratio (6.44) and minimal drawdown (2.45%) highlight the effectiveness of the hybrid strategy.

Robustness Considerations: While performance was strong, we observed sensitivity to random seeds due to PPO's stochastic nature and environment variability. This variance can be mitigated through evaluation averaging, reduced exploration noise, and consistent test sets.

Future Enhancements: To further limit downside risk, drawdown penalties can be increased, constraints on per-step weight shifts can be introduced, and risk metrics such as CVaR or downside deviation can replace standard volatility.

Conclusion: The meta-controller demonstrates a powerful paradigm for risk-aware portfolio optimization, offering enhanced returns with superior stability by adaptively combining multiple reinforcement learning agents based on market regimes.

LLM Trader

This section presents a proof-of-concept trading agent that leverages a large language model (GPT-4-o-mini) as a decision-maker, with on-the-fly Python analysis via the Code Interpreter, and compares its performance both with/without chat history and against an A2C baseline.

Agent Design

The LLM Trader operates at a daily rhythm:

1. Receive `Date`, `Open`, `High`, `Low`, `Close`, `Volume`, `Balance`, `SharesHeld`, and `CostBasis`.
2. Invoke Python snippets (via the Code Interpreter tool) to compute using historical data about the stock in a pandas dataframe. The agents tends to compute the following:
   - Profit/loss per share,
   - Net worth,
   - Price changes and percentages,
   - Volume vs. historical average,
   - Simple moving averages or other indicators.
3. Issue a discrete trade decision $\in [-k, k]$, where positive buys, negative sells, zero holds.

Experimental Protocol

- Data: AAPL from November 2023 to May 5, 2025.
- Trading window: 1,000 trading days.
- Position limits: $\pm$ maximum shares purchasable given current balance.
- Chat history: retained vs. reset each day to test memory effects.

Performance with vs. without Chat History

Table 7: Persistent vs. Non-Persistent Chat History

Setting	Final Net Worth	Total Return (%)	Sharpe (LLM)	Sharpe	Max Drawdown (%)
Persistent	\$11 819.16	18.19	1.15	1.59	-16.86
Non-Persistent	\$12 026.31	20.26	1.71	1.59	-9.29

As shown in Table 7, allowing chat history actually degraded performance: the reset-daily setup achieved both higher return (20.26% vs. 18.19%) and Sharpe (1.71 vs. 1.15), with smaller drawdowns.

LLM Trader vs. A2C Baseline

Table 8: LLM Trader (Reset Daily) vs. A2C Baseline (Reset Daily)

Agent	Final Net Worth	Total Return (%)	Sharpe Ratio	Max Drawdown (%)
LLM Trader (Reset Daily)	\$11 292.82	12.93	0.54	-23.81
A2C Baseline (Reset Daily)	\$11 254.48	12.54	0.46	-22.92

Table 8 shows that, against an A2C policy run with daily resets, the LLM-based agent delivered materially higher return and risk-adjusted performance, while also capping drawdowns more effectively.

Challenges and Future Work

- **Data Limitations:** Free Yahoo Finance API rate limits hinder higher-frequency experiments.
- **Real-Time Sentiment:** Lack of low-cost news/sentiment sources restricts market-forward signals.
- **Cost:** Large-scale LLM calls incur API charges; batching or distillation may mitigate.
- **Tooling:** Integrating GRPO-style multi-turn tool calls for efficient code invocation remains experimental.