Final Report

Sam Zhong

CSC 461

December 13, 2024

Introduction

The financial market is inherently dynamic presenting a challenge for investors to make optimal portfolio allocation decisions. Portfolio optimization, which is a cornerstone of financial research, seeks to balance risk and return for an investment portfolio. Traditional approaches, such as Markowitz's Mean-Variance Optimization (MVO) and the Capital Asset Pricing Model (CAPM), have foundations for this field. However, these methods often assume static market conditions and linear relationships, which limit their ability to adapt to rapidly changing environments.

The Efficient Market Hypothesis (EMH), a fundamental theory in finance, posits that financial markets are "informationally efficient." This implies that asset prices fully reflect all available information, leaving no opportunity for excess returns beyond what is achievable through random chance (Malkiel, 2003). While the EMH shaped much of modern finance, it faces criticism for oversimplifying market dynamics and failing to account for anomalies, behavioral biases, and temporary inefficiencies that come up in real-world markets.

This project challenges the implications of the EMH by leveraging Deep Reinforcement Learning (DRL) to identify and exploit patterns in market data. Unlike traditional approaches, DRL models adapt dynamically to changing market conditions and can uncover relationships that may not be immediately apparent (Yashaswi, 2021). By training an agent in a custom environment that integrates historical stock prices, technical indicators, and macroeconomic data, this project assumes that exploitable inefficiencies exist, even if only temporarily. The reinforcement learning framework uses advanced metrics such as Sharpe Ratio, Sortino Ratio, and Cumulative Returns to guide the agent toward optimal portfolio allocation strategies.

The application of DRL in portfolio optimization is unconventional from the static assumptions of traditional methods and the constraints of the EMH. By embracing the complexity of financial markets and adapting to their non-linear, dynamic nature, this project explores the potential for intelligent systems to generate consistent returns while managing risk (Sun et al., 2024).

Problem Definition

While foundational to modern portfolio theory, approaches like MVO and the CAPM have significant limitations in dynamic market environments. MVO assumes that asset returns follow a normal distribution and that correlations between assets remain constant over time. In reality, financial markets are characterized by time-varying correlations, non-linear relationships, and

unexpected shocks, which make these static assumptions unrealistic (ABC Quant Knowledge Base).

Similarly, CAPM relies on the assumption of a single-factor model where market risk (beta) is the primary determinant of an asset's returns. However, empirical evidence has shown that other factors—such as momentum, size, and value—play significant roles in asset pricing. Furthermore, both MVO and CAPM struggle to adapt to rapidly changing market conditions, as they do not incorporate the temporal dependencies or complex interactions present in real-world data (CFA Journal).

In contrast, reinforcement learning offers a dynamic framework that can adapt to evolving market behaviors.

The environment must address several challenges:

1. Data Complexity: Integrating diverse data sources from severe sectors, such as historical stock prices, technical indicators, and macroeconomic variables, to create a rich and realistic simulation (Bianchi, 2022).
2. Dynamic Market Behavior: Accounting for changing correlations between assets, the impact of market shocks, and other time-varying factors (Benhamou et al., 2021).
3. Reward Design: Developing a reward function that balances risk and return, with metrics like Sharpe Ratio, Sortino Ratio, and Cumulative Returns to guide the agent toward stable, risk-adjusted performance (Eschmann, 2021).

The desired outcome is a flexible and robust environment where a reinforcement learning agent can:

1. Learn adaptive strategies for optimizing returns while managing risks effectively.
2. Explore and exploit temporary market inefficiencies.
3. Generalize across varying market conditions.

Data

The dataset used in this project is daily data from 1996 to 2024 retrieved from Yahoo Finance. While this time frame provides a historical perspective on market trends, it is relatively limited in scope due to challenges inherent in time-series predictions and the availability of consistent financial data. Despite these constraints, the dataset is designed to ensure a diverse

representation of the market by including stocks from multiple sectors and an Exchange-Traded Fund (ETF), enabling a well-rounded analysis:

Stock Data Overview:

1. **Technology Sector:** MSFT (Microsoft), AAPL (Apple), INTC (Intel), CSCO (Cisco Systems)
2. **Healthcare Sector:** JNJ (Johnson & Johnson), MRK (Merck & Co.)
3. **Consumer Staples Sector:** PG (Procter & Gamble), KO (Coca-Cola), PEP (PepsiCo), WMT (Walmart)
4. **Other Sectors:**
    - **Consumer Discretionary:** DIS (Walt Disney), MCD (McDonald's)
    - **Financials:** JPM (JPMorgan Chase)
    - **Energy:** XOM (Exxon Mobil), CVX (Chevron)
    - **Industrials:** GE (General Electric)
    - **Communication Services:** T (AT&T)
5. **ETF:** SPY (S&P 500 ETF Trust), representing broader market trends.

Technical Indicators and Metrics:

1. **Adjusted Returns:** Incorporates dividends and stock splits, offering a holistic measure of stock performance.
2. **Volume:** Tracks the number of shares traded, providing insights into liquidity and market activity.
3. **Simple Returns:** Measures the percentage change in price from one period to the next.
4. **Relative Strength Index (RSI):** Identifies momentum and potential overbought or oversold conditions.
5. **Moving Averages (20, 50, 100 Days):** Highlights price trends over various time windows.

Macroeconomic Data:

1. **Daily Interest Rates:** Reflects the cost of borrowing, influencing equity valuations and the relative appeal of fixed-income securities (Benhamou et al., 2021).
2. **Unemployment Rates:** Serves as an indicator of economic health, impacting market trends (Bianchi, 2022).

By combining diverse sectoral data, commonly used technical indicators, and macroeconomic variables, the dataset provides a solid foundation for training reinforcement learning models. The inclusion of well-established metrics ensures alignment with traditional portfolio management approaches while exploring their interplay with market dynamics.

Limitations:

Although the dataset is diverse, it is not expansive. Expanding the dataset to include a greater number of stocks, additional global markets, and more advanced financial indicators would do a much better job improving the model's ability to generalize across different conditions. Moreover, incorporating higher-frequency data and alternative data sources (e.g., social sentiment, news) could unlock new dimensions of analysis. However, these advancements require significant computational resources and storage capabilities (Bianchi, 2022).

Methods

Custom Reinforcement Learning Environment:

Overview:

To simulate the complexities of financial markets and enable dynamic portfolio allocation, a custom environment was built using the OpenAI Gym framework. This environment integrates realistic market data and financial metrics, allowing reinforcement learning agents to make and evaluate portfolio decisions.

Environment Design:

The environment initializes with data retrieved from Yahoo Finance, which is preprocessed locally. This preprocessing includes handling missing values, normalizing features, and computing financial indicators such as moving averages, RSI, and returns (Bianchi, 2022). The state representation excludes raw price data and instead uses normalized, non-price features over a rolling window of a set number of days. This parameter, the window size, could be optimized to balance leveraging historical information while avoiding overfitting. For this project, two window sizes were tested: 30 days and 63 days (approximately one-quarter of a year). The resulting state is a flattened array, with dimensions dependent on the number of indicators and the chosen window size.

The action space allows continuous portfolio allocation decisions for each asset and cash, constrained to ensure the weights sum to one. Actions are normalized and clipped to ensure feasibility, with the agent directly controlling asset and cash allocations.

Rewards are calculated using a combination of key financial metrics:

- Cumulative Return: Encourages maximizing portfolio performance.
- Sharpe Ratio and Sortino Ratio: Reward risk-adjusted returns and penalize downside volatility (Eschmann, 2021).
- Maximum Drawdown and CVaR: Penalize extreme losses and large declines from peak portfolio value (Benhamou et al., 2021).

The environment's step function simulates the agent's performance over a horizon of a set number of trading days, updating portfolio returns and evaluating metrics such as Sharpe Ratio and maximum drawdown. The trading horizon is a parameter that can be fine-tuned. In the instances I trained, the horizon was set to one year (252 trading days), as shorter-term horizons could lead to significant tax implications for portfolio reallocations. For episodes, a maximum of 500 steps was used, another parameter that could be fine-tuned. Setting a limit on the steps helps avoid overfitting and encourages better generalization, especially given the relatively small dataset.

Finally, the reset function initializes a new episode, providing a random observation window to the agent. This ensures variability in starting conditions, which is critical for robust training. The environment is designed with flexibility in mind, allowing for future expansion to include additional metrics, constraints, or market features.

Experiments and Analysis

Experimental Setup:

After testing multiple configurations for the simulation environment, significant overfitting was observed in most setups. Through iterative experimentation, the final design was chosen to balance generalization and performance. This setup involved:

- A rolling window size of 63 days to provide sufficient historical context.
- A trading horizon of 252 days (one year), aligning with long-term portfolio allocation practices while avoiding the tax complications associated with short-term reallocations.

- Episodes capped at 500 steps to prevent overfitting and encourage better generalization.

The model was trained using Stable Baselines3's Proximal Policy Optimization (PPO) algorithm with an MLP policy of two hidden layers (64 neurons each). This architecture was selected for its computational efficiency and ability to capture essential patterns in the data.

PPO Hyperparameters:

For this project, the default hyperparameters provided by Stable Baselines3 were used, as computational constraints limited the ability to extensively test different configurations. These parameters include:

- Learning Rate: 0.0003
- Discount Factor: 0.99
- Clip Range: 0.2
- Batch Size: 64
- Number of Epochs: 10

The default values are widely accepted for general reinforcement learning tasks and provide a robust starting point. Fine-tuning these hyperparameters could potentially yield better performance, the focus of this project was on developing the environment and framework rather than optimizing the agent's configuration. Future iterations could explore hyperparameter optimization techniques like grid search or Bayesian optimization to improve the model's learning efficiency and performance (Raffin et al. , 2021).

Why PPO was Selected:

PPO is a reinforcement learning algorithm that is particularly well-suited for dynamic and continuous action spaces, such as portfolio allocation. Its design incorporates a clipped objective function, which limits large updates to the policy and ensures a stable learning process (Schulman et al., 2017). This stability is crucial in financial markets, where abrupt changes in strategy can lead to suboptimal outcomes. Additionally, PPO optimizes both the policy and the value function within a single framework, making it more sample-efficient than traditional policy gradient methods(Eschmann, 2021).

PPO is also designed to handle continuous action spaces effectively, enabling smooth adjustments to portfolio allocations across multiple assets and cash. These qualities make PPO

an excellent choice for learning dynamic, risk-aware portfolio strategies in complex and volatile financial environments. Its ability to balance stability and adaptability ensures reliable performance across diverse market conditions (Sun et al., 2024; Yashaswi, 2021).

Comparison Benchmarks:

- **SPY Allocation:** Represents the market benchmark with SPY as a proxy for the S&P 500 index.
- **Equal Allocation:** Divides resources equally across all tickers in the portfolio. This baseline assumes no optimization and acts as a control.

Testing Scenarios:

- Primary Backtesting Data:
    - Data from 2016 to 2024 (unseen during training) was used to evaluate generalization.
- Alternative Ticker Set:
    - An entirely different set of tickers ["F", "COST", "HD", "GOOG", "JNJ", "V", "TSLA", "GE", "INTC", "WBA", "MSFT", "CVS", "PFE", "AMZN", "BA", "CSCO", "IBM", "SPY"] tested the model's robustness. This set includes a mix of high-performing and low-performing stocks to stress-test the agent.

Evaluation Metrics:

To evaluate the agent's performance against SPY and equal allocation baselines, the following metrics were used:

1. Cumulative Return: Measures total portfolio growth over the evaluation period.
2. Sharpe Ratio: Evaluates risk-adjusted returns.

Results:

Figure 1:

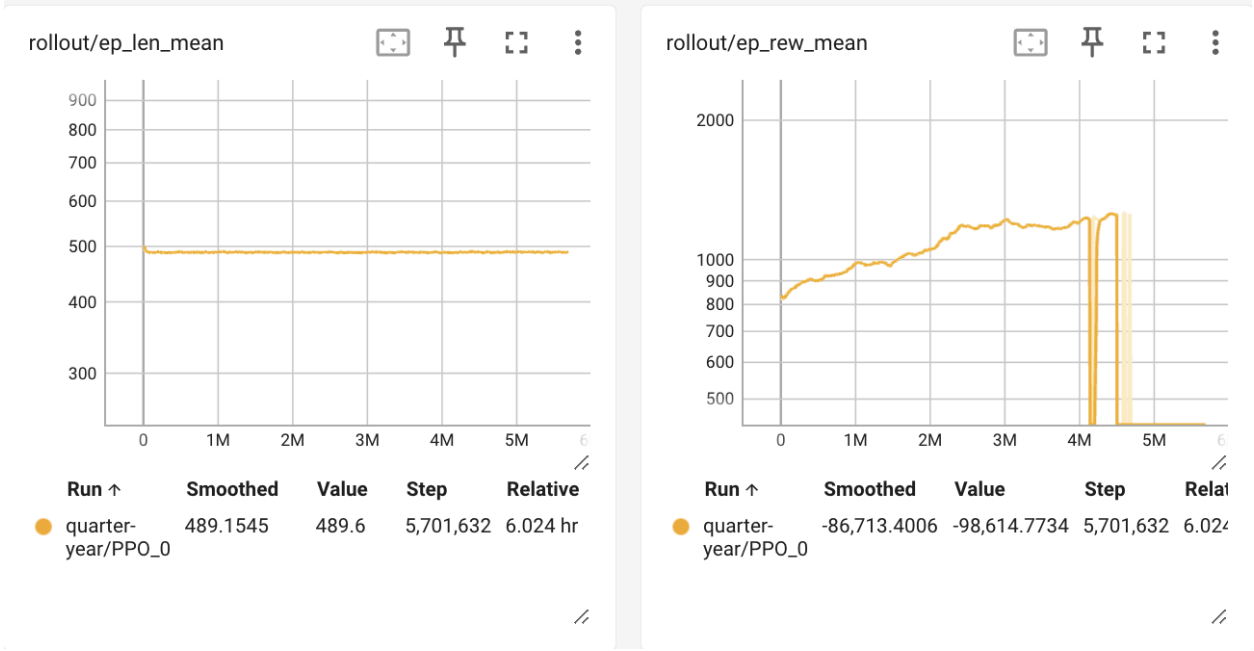
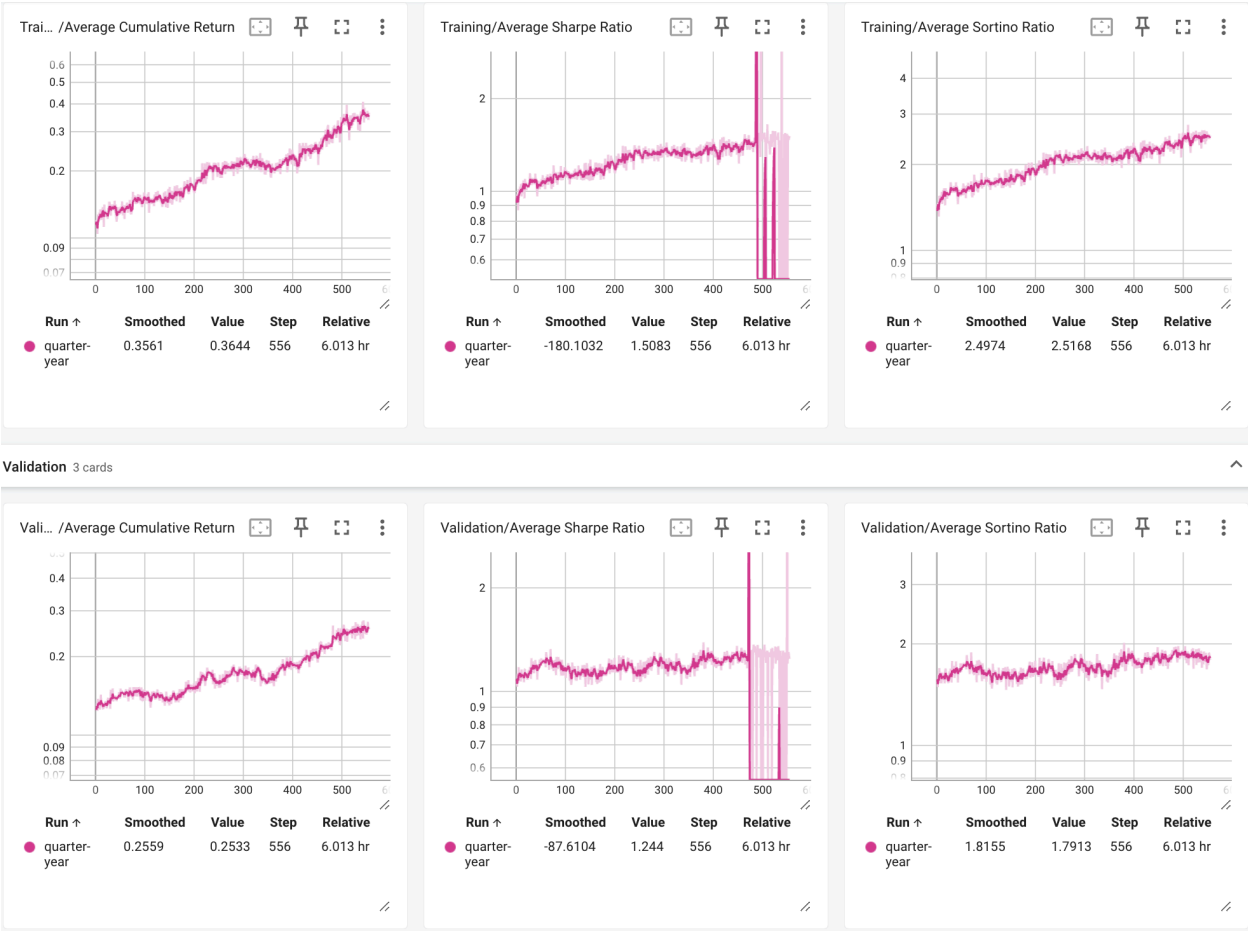


Figure 2:



From the TensorBoard figures above, which log the reward trends during training and validation, several key observations can be made:

Training Rewards:
There is a clear, gradual increase in the reward values over time during the training process indicating the agent is learning and optimizing its policy effectively as the steps progress. However, in the latter steps, we observe the reward starting to oscillate. This oscillation is influenced by the fluctuating Sharpe Ratio, which is a key component of the reward function. The Sharpe Ratio's inherent sensitivity to short-term volatility may lead to these oscillations, particularly as the agent refines its policy and takes more exploratory actions.

Validation Rewards:
Similar to training, the validation dataset also exhibits an overall increase in reward metrics over time, though the trend is less pronounced. This is expected since the validation data represents unseen market conditions and provides a more stringent test of the agent's generalization capabilities. Despite being less significant, the increase in validation metrics suggests that the agent is successfully learning a policy that generalizes to new, unseen data.

Cumulative Returns, Sortino Ratios, and Sharpe Ratios:
During training, these metrics also have positive trends, indicating that the agent is optimizing for both return and risk-adjusted performance. For validation data, while the trends are not as significant, they still reflect a reasonable degree of learning and policy improvement over time.

Oscillations in Later Steps:
The oscillations observed in the reward function during the later training steps may be due to suboptimal exploration, particularly as the agent explores actions that maximize one component of the reward function while inadvertently affecting others.

Conclusion:
Overall, the trends indicate that the agent is learning effectively, as evidenced by increases in rewards, cumulative returns, and risk-adjusted metrics in both training and validation datasets. While the oscillations in the later steps are a point of concern, they do not overshadow the broader positive trends. Future iterations could explore adjustments to the reward function, such

as scaling or smoothing components like the Sharpe Ratio, to reduce oscillations and further stabilize training.

Figure 3:

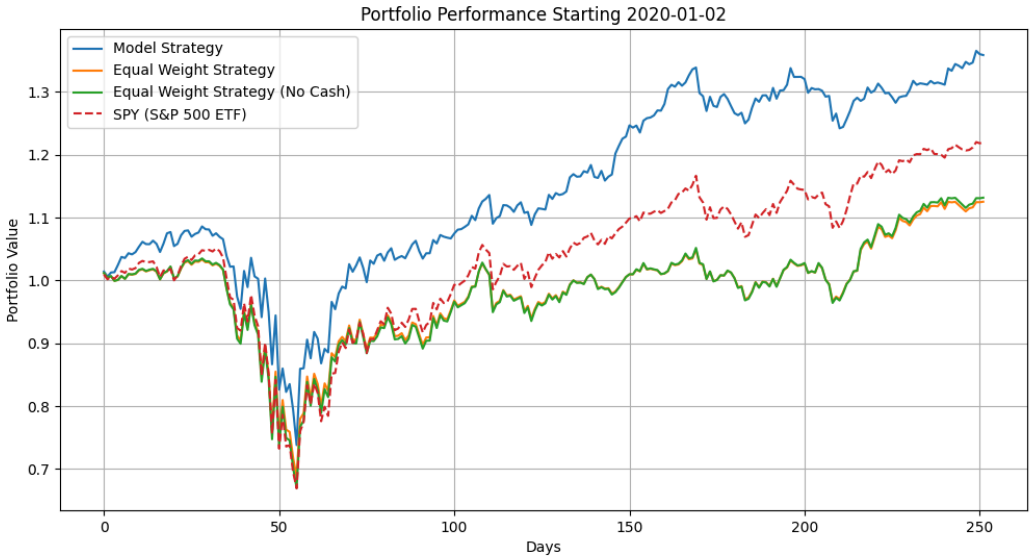


Figure 1 above compares the cumulative returns of the model's allocation strategy and the equal-weight strategy during the 2020 market crash caused by the COVID-19 pandemic. The model allocates MSFT (35%), PG (28%), and WMT (35%). The model's allocation strategy outperformed both the evenly distributed portfolio and the S&P 500.

Performance Metrics:

- **Model Strategy**: Sharpe Ratio: 1.0526, Cumulative Return: 35.02%.
- **SPY**: Sharpe Ratio: 0.6576, Cumulative Return: 17.73%.
- **Equal Weight Strategy**: Sharpe Ratio: 0.3870, Cumulative Return: 7.53%.

The model's allocation strategy outperformed both the evenly distributed portfolio and SPY, achieving superior cumulative returns and the highest Sharpe Ratio, demonstrating effective risk-adjusted performance during extreme market conditions.

Figure 4:

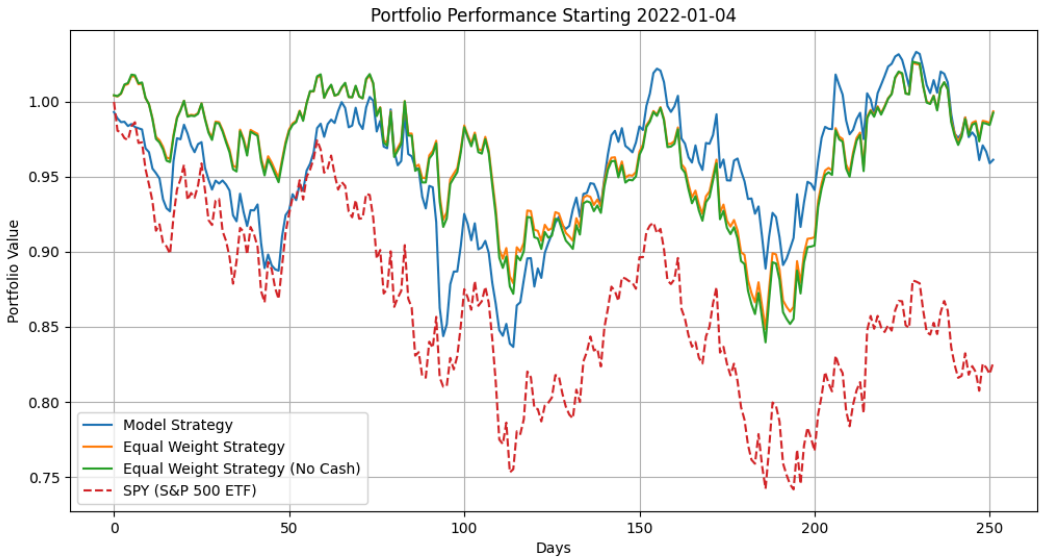


Figure 2 above compares the cumulative returns and allocations of the model's strategy, the equal-weight strategy, and the S&P 500 (SPY) on January 4, 2022, during a challenging economic period following the COVID-19 recovery. The model allocates resources heavily to MSFT (25.81%), WMT(25.81%), JPM (25.81%), and T (22.57%), while avoiding other assets. In contrast, the equal-weight strategy distributes its portfolio evenly across all tickers.
Performance Metrics:

- **Model Strategy:** Sharpe Ratio: -0.1964, Cumulative Return: -5.67%.
- **SPY:** Sharpe Ratio: -0.7188, Cumulative Return: -18.36%.
- **Equal Weight Strategy:** Sharpe Ratio: -0.0373, Cumulative Return: -2.23%.

While the model's allocation demonstrated better performance than the S&P 500 during this period, it underperformed the equal-weight strategy. This highlights that the model does not always outperform the benchmarks, particularly in periods where its learned strategy may not fully align with broader market trends or economic downturns.

Note: The allocations shown in the figures can vary due to the random seed used in the initialization of NumPy and PyTorch. The model's predictions depend on the random seed, which affects stochastic components such as action sampling during inference. This variability underscores the importance of testing the model's average performance over multiple runs with varied seeds to ensure robustness and reliability in real-world applications.

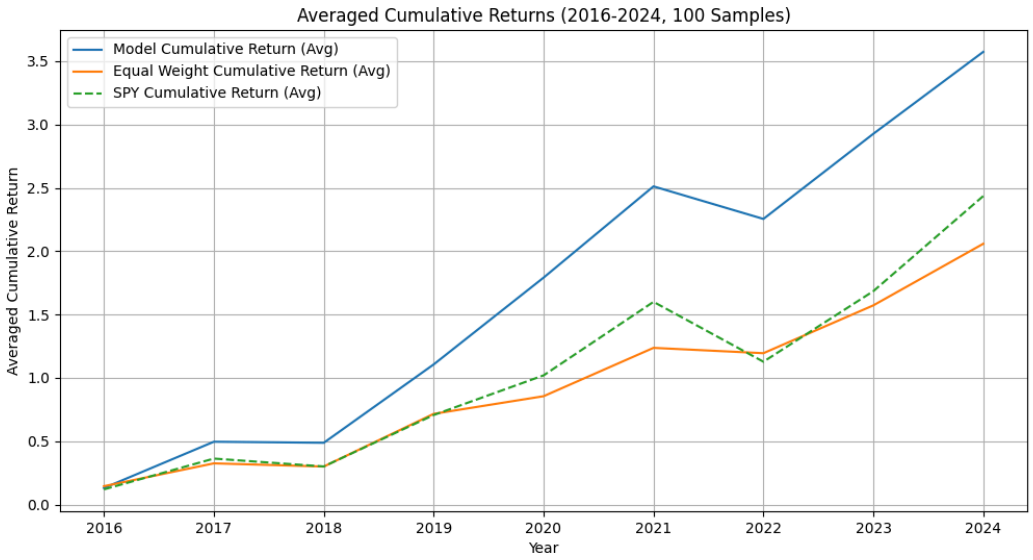
Addressing Variability in Predictions

The variability in the model's predictions, caused by differences in random seeds, is mitigated by aggregating results across multiple simulations with varied seeds. I will systematically evaluate the model's performance over 100 different random seeds. Each seed influences the stochastic components of inference, such as action sampling, ensuring diverse scenarios are tested.

Key Steps in the Solution:

1. **Multiple Seed Evaluation**:
    - By setting unique random seeds for both NumPy and PyTorch, the model is tested under varying prediction scenarios.
    - This approach ensures that the observed performance metrics are not biased by a specific random initialization.
2. **Annual Performance Aggregation**:
    - For each random seed, the model's predictions are evaluated year-by-year from 2016 to 2024. Metrics such as cumulative returns and Sharpe Ratios are recorded for each year and averaged across all simulations.
    - Both the model strategy, equal weight strategy, and SPY are evaluated in parallel.
3. **Sharpe Ratio Averaging**:
    - Calculates Sharpe Ratios for each portfolio across all seeds and computes their averages. This provides a more reliable measure of the model's risk-adjusted returns, mitigating the influence of outliers.

Figure 5:



The line graph illustrates the averaged cumulative returns of the Model Strategy, Equal Weight Strategy, and the S&P 500 (SPY) from 2016 to 2024, calculated across 100 random seeds. The cumulative returns reflect the long-term growth of an initial portfolio value under each strategy.

Key Observations:

1. **Cumulative Returns**:
    - The Model Strategy outperforms both the Equal Weight Strategy and SPY in terms of cumulative returns across the years.
    - By 2024, the Model Strategy achieves a cumulative return of 357.34%, compared to 205.85% for Equal Weight and 243.52% for SPY.
2. **Sharpe Ratios**:
    - Despite achieving the highest cumulative returns, the Model Strategy exhibits a Sharpe Ratio of 1.22, slightly lower than the Equal Weight Strategy's 1.34.
    - This suggests that while the model achieves higher absolute returns, it may take on more risk, leading to higher volatility compared to the Equal Weight Strategy.
    - The SPY shows a Sharpe Ratio of 1.00, indicating the lowest risk-adjusted returns among the three.

The current reward function for the Model Strategy was designed to balance multiple objectives, including maximizing cumulative returns and improving risk-adjusted performance through metrics like the Sharpe Ratio and downside risk measures. This approach has led to superior

cumulative returns, but it has not consistently optimized for risk-adjusted performance, as evidenced by the Equal Weight Strategy's higher Sharpe Ratio (1.34 vs. 1.22).

This result highlights an opportunity for improvement. Fine-tuning the reward function to place greater emphasis on the Sharpe Ratio could guide the model toward prioritizing risk-adjusted returns over absolute growth. By incorporating a higher weight on the Sharpe Ratio, the model may achieve more stable performance and reduce volatility, better aligning with long-term portfolio management objectives.

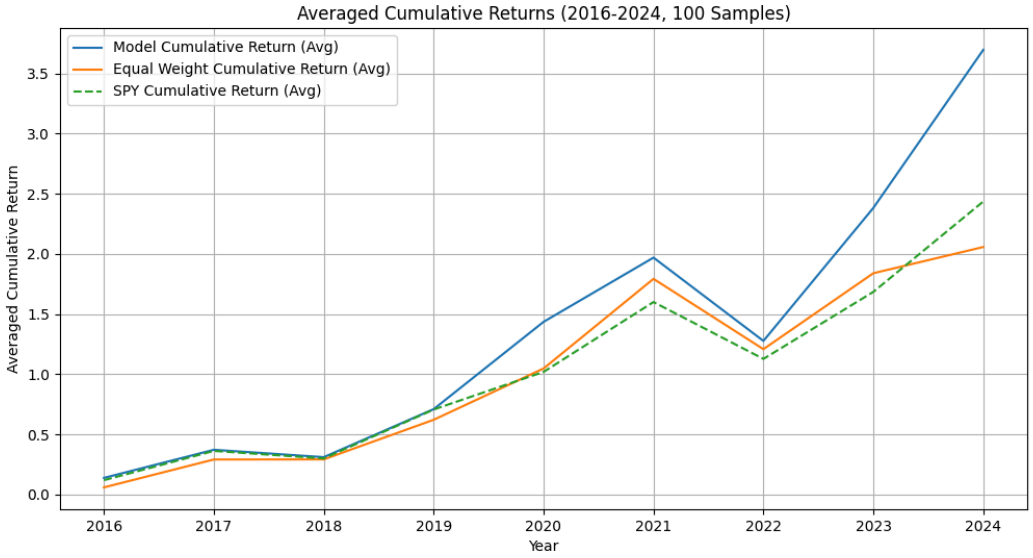
Testing on a Completely Different Set of Tickers:

To evaluate the model's generalization capabilities, a completely new set of tickers was selected, distinct from those used in training. The selected tickers represent a mix of high-performing, average, and underperforming assets:

AMZN, BA, COST, CSCO, CVS, F, GE, GOOG, HD, IBM, INTC, JNJ, MSFT, PFE, SPY, TSLA, V, WBA, SPY.

This set mirrors the shape and structure of the original data while introducing new market dynamics and asset behaviors. The model's cumulative returns and Sharpe Ratios were compared against the Equal Weight Strategy and SPY.

Figure 6:



Average Sharpe Ratios:

- **Model Strategy**: 0.86
- **Equal Weight Strategy**: 0.86
- **SPY**: 1.00

Key Observations:

1. **Cumulative Returns:**
    - The Model Strategy outperformed the Equal Weight Strategy across most years, achieving a cumulative return of 369.55% by 2024 compared to 205.72% for Equal Weight and 243.52% for SPY.
2. **Sharpe Ratios:**
    - The Model Strategy and Equal Weight Strategy both achieved an average Sharpe Ratio of 0.86, indicating similar risk-adjusted returns over the test period.
    - SPY achieved a slightly higher Sharpe Ratio of 1.00, reflecting its lower volatility compared to the dynamic allocation strategies.

The Model Strategy achieved higher cumulative returns than both the Equal Weight Strategy and SPY when tested on a completely different set of tickers. While the Sharpe Ratio of the Model Strategy was on par with the Equal Weight Strategy, this is not necessarily a drawback, as the model managed to achieve higher returns without introducing additional risk. However, the Sharpe Ratio underperformed SPY for this dataset, which may be influenced by the selection of tickers, as many of them are inherently more volatile. Future iterations of the reward function could explore placing greater emphasis on risk-adjusted metrics like the Sharpe Ratio to better balance returns and volatility, especially when managing a portfolio with high-risk assets.

Conclusion and Future Directions

This project demonstrates promising results, showcasing the potential of reinforcement learning for dynamic portfolio optimization. However, it is clear that the approach is not without its limitations. The model performed well in scenarios involving diverse datasets but struggled when tested on a dataset comprising the 17 largest market-cap stocks, most of which are concentrated in the technology sector and have seen tremendous growth in recent years. In contrast, the training dataset was smaller, more diversified across sectors, and included stocks with varying levels of performance. This allowed the model to identify opportunities more

effectively. The discrepancy highlights the importance of dataset diversity and its influence on the model's ability to generalize across different market conditions.

Moving forward, several enhancements could significantly improve the model's performance and robustness. A key improvement would be using a larger and more comprehensive dataset, potentially containing all stocks in the S&P 500 as well as international markets. This would allow the model to train on a broader range of scenarios and asset behaviors, improving its ability to adapt to concentrated or sector-specific datasets. However, acquiring and processing such extensive data would present logistical and computational challenges.

Additionally, the environment and algorithm could be refined through careful tuning of key parameters, such as the rolling window size and the trading horizon, to better align with the complexities of market behavior. The MLP architecture could also be expanded with more layers or neurons to capture more intricate patterns, although this would require greater computational resources and could increase the risk of overfitting. Another potential avenue for exploration is the use of a multi-agent system, where each agent specializes in a specific task, such as focusing on individual sectors, identifying macroeconomic trends, or managing risk at different levels. A multi-agent approach could enhance the model's ability to decompose and address complex market dynamics, potentially leading to more effective portfolio strategies.

By addressing these limitations and pursuing these future enhancements, this framework could evolve into a more robust and adaptable solution for portfolio optimization in diverse and complex financial markets.

Github Repository:
https://github.com/SamZhong2/Portfolio_Management

References

1. ABC Quant Knowledge Base. (n.d.). Markowitz portfolio optimization and its limitations. ABC Quant Knowledge Base. Retrieved from https://knowledgebase.abcquant.com/index.php?article=59&option=com_kb&task=article
2. Benhamou, E., Saltiel, D., Ohana, J.-J., Atif, J., & Laraki, R. (2020). Deep reinforcement learning (DRL) for portfolio allocation. In ECML PKDD 2020 Workshops (pp. 531–536). Springer. https://doi.org/10.1007/978-3-030-67670-4_32
3. Bianchi, R. (2022). Reinforcement learning for portfolio optimization. (Master's thesis). Universitat Politècnica de Catalunya.
4. CFA Journal. (n.d.). The limitations of the capital asset pricing model (CAPM). CFA Journal. Retrieved from https://www.cfajournal.org/limitations-capm
5. Eschmann, J. (2021). Reward function design in reinforcement learning. In Reinforcement learning algorithms: Analysis and applications (pp. 25–33). Springer.
6. Malkiel, B. G. (2003). The efficient market hypothesis and its critics. Journal of Economic Perspectives, 17(1), 59–82.
7. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym: A toolkit for developing and comparing reinforcement learning algorithms. arXiv preprint arXiv:1606.01540. https://arxiv.org/abs/1606.01540
8. Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., & Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research, 22(1), 1–8.
9. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. https://arxiv.org/abs/1707.06347
10. Sun, R., Stefanidis, A., Jiang, Z., & Su, J. (2024). Combining transformer-based deep reinforcement learning with Black-Litterman model for portfolio optimization. Neural Computing and Applications, 36(32), 20111–20146. https://doi.org/10.1007/s00521-024-09805-9
11. Yashaswi, K. (2021). Deep reinforcement learning for portfolio optimization using latent feature state space (LFSS) module. arXiv preprint arXiv:2101.11921. https://arxiv.org/abs/2101.11921