Reinforcement Learning for Optimal Portfolio Management in Dynamic Financial Environments : An Exploratory Study

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Ishwar Govind
(111901024)

and

Jerry John Thomas
(111901055)

INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

CERTIFICATE

*This is to certify that the work contained in the project entitled "**Reinforcement Learning for Optimal Portfolio Management in Dynamic Financial Environments : An Exploratory Study**" is a bonafide work of **Ishwar Govind** (**Roll No. 111901024**) and **Jerry John Thomas** (**Roll No. 111901055**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Palakkad under my guidance and that it has not been submitted elsewhere for a degree.*

Dr. Chandrashekar Lakshminarayan

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Madras

Acknowledgements

We would like to express our sincere gratitude to our mentor Dr. Chandrashekar Lakshminarayan for providing his invaluable guidance, comments and suggestions throughout the course of the project. We would also like to thank Dr. Albert Sunny for his advice and support.

Contents

Chapter 1

Introduction

1.1 Background

With the help of artificial intelligence, banking and finance industries have new ways to satisfy client needs for smarter, safer, and more practical ways to access, spend, save, and invest money. Today machine learning is used extensively in the field of finance for algorithmic trading, fraud detection and prevention, credit score calculation and many other use cases. With the breakthroughs in AI, these approaches have a better performance than traditionally employed methods.

We are exploring the problem of Portfolio Optimisation which involves the dynamic allocation of funds into different stocks assets optimally to get the best financial outcome. To maximize the returns on the stock we have to estimate the returns on the stock, often this comes out to be quite a challenge in a complex and dynamic market where over reliance on the predictive models could lead to bad outcomes. Errors in this estimation could lead to traditional naive approaches outperforming ML optimized strategies[2]. Feature extraction and trading strategy design are other important aspects of this problem.

Although many approaches have been proposed in this field, most of them have too many assumptions that prevent these methods from being used in the real market. Some of these unrealistic assumptions are around the financial signals' second-order and higher-

order statistical moments[3]. Models are usually limited to discrete action spaces[4]. Some models also do not consider the transaction and other costs involved in trading. Methods with reduced assumptions tend to have better performance in the real market and we try to tackle this.

Portfolio optimization can be modeled as a Markov Decision Process (MDP) and Reinforcement Learning (RL) can be used on it. Due to the large state space involved in dealing with the stock market, dynamic programming cannot be used because of limited scalability, hence RL is a better suited candidate for tackling the unpredictive market and dynamically changing the strategy.

1.2 Problem Definition

This is an optimisation problem where we try to maximize the portfolio returns (risk-adjusted) for our portfolio. We have an initial capital that we could invest into the market having m assets. After the initial allocation, we wait for the market's response and reallocate the portfolio to have a better prospect of obtaining the maximum returns. We continue this process till the end of a fixed time period. We are also allowed to have uninvested capital. The strategy needs to optimize for both bear and bull market scenarios.

We have some assumptions about the financial market [5][4]

1. Zero slippage (orders executed at the expected price)
2. Possible to trade at the market at any time
3. Our transactions do not affect the market price of the assets

This problem can be modeled as a Markov Decision Process (MDP) and Reinforcement Learning(RL) could be employed to solve this.

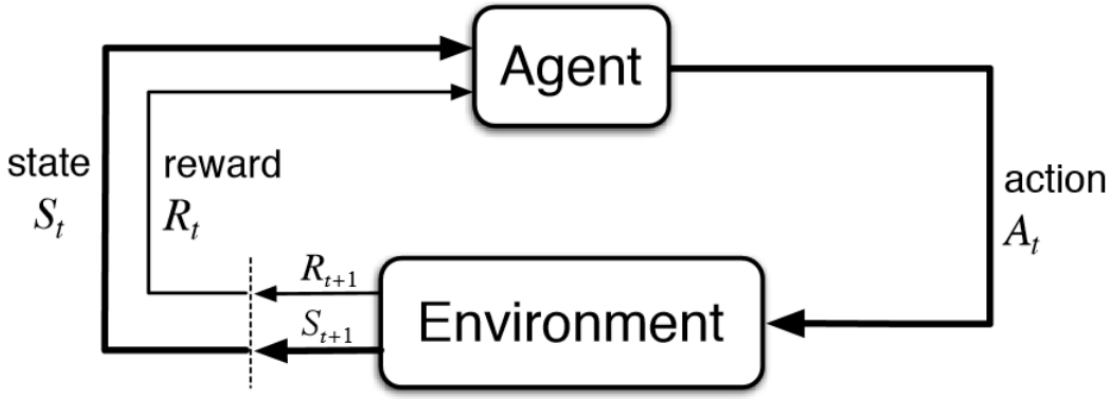

Fig. 1.1 Components of a RL system [6]

State s, consists of $[b_t, p_t, h_t, \text{Market Indicators}_t]$. b_t refers to the balance at time t. p_t refers to the closing prices (adjusted) of each stock at time t. h_t refers to the shares owned of each stock. Market indicators [5] include the Relative Strength Index (RSI), Commodity Channel Index (CCI), Average Directional Index (ADX), etc. All these indicators could be calculated with all or combinations of high, low, and close prices over a period of time.

Action a, consists of a vector of actions over each stock, indicating how much to sell or buy each stock. Not selling nor buying results in holding the stock.

Reward r, aims to quantize the change in portfolio value when we take action a from state s to reach a new state s'. Log returns (weighted sum of log returns for the portfolio) and differential Sharpe (instantaneous risk-adjusted Sharpe ratio) are two methods used for estimating rewards.

1.3 Aim and Objectives

1. Compare the performance of different reinforcement learning and traditional algorithms in a financial market.

2. The goal is to design a trading strategy that maximizes the positive cumulative change of the portfolio value in the dynamic environment.

Chapter 2

Reinforcement Learning in Finance

2.1 Reinforcement Learning

Reinforcement learning is a branch of machine learning that studies how agents should behave in a given environment in order to maximize the concept of cumulative reward. Here agents are virtual beings capable of making decisions based on observations made from the environment. The agent is not pre-programmed and learns based on previous experiences. The experience is from the reward received based on actions made in the environment.

2.2 Why Reinforcement Learning ?

The real-world financial market is a very complex dynamic system. The system is stochastic in nature and keeps changing with each transaction that happens in it. Many supervised deep machine learning algorithms have been tried out on the financial markets. Most of them are used to predict future stock price movements.

The performance of these algorithms is based on the accuracy of the price prediction. The prediction of stock prices is difficult. The actions of buying and selling a certain quantity of stock cannot be done using supervised learning. Although we could program

a model to buy and sell based on the price prediction. This does not make the model "intelligent" and adaptive to market changes.[7]

Reinforcement learning provides algorithms that can train agents capable of doing these actions by using "intelligence". Deep reinforcement learning can be used to create agents capable of beating the best human player in one of the most challenging classical board games, Go.[8]. Model-free deep RL was shown to be successful in its previous attempts at algorithmic trading.[9].

2.3 Reinforcement Learning Methods

2.3.1 Policy Gradient

It is the most basic policy based reinforcement learning algorithm. The agent starts with a random initial policy. The agent makes actions based on the policy and the trajectory made from the policy action is stored. The rewards are computed based on the trajectory. The policy is updated to give a higher probability to an action that will lead to bigger rewards and a lower probability to actions that will lead to lower rewards. The objective function is based on the cumulative future reward. The objective function is maximized by using gradient ascent.

2.3.2 Actor-Critic

Model free reinforcement learning method which has two networks. The Actor network learns based on the policy gradient method. The Critic network is used to evaluate the action produced by the actor using the value function predicted by the network.

Chapter 3

Risk in Finance

3.1 Risk Sensitivity

Risk refers to the potential for financial loss or uncertainty associated with investment decisions. When it comes to investing, risk can be defined as the level of price volatility associated with a particular asset or investment. Sensitivity in finance refers to the extent to which a particular market instrument reacts to fluctuations in underlying factors, observed through changes in its price response to other factors.

The standard optimization criteria in a MDP is to maximize the expected sum of discounted rewards and the average reward. But in finance, we might also want to minimize some measure of risk as well as take care of the optimization problem. The way to do this is to include a penalty for the variability induced by a given policy. This variability can be due to two reasons [10]; either due to uncertainties in the model parameters or the inherent unpredictability arising from the stochastic nature of a system that is studied in risk-sensitive MDP's.

3.2 Types of Risk Sensitivity

Traders/Investors can be broadly classified into one of the following categories.

3.2.1 Risk Averse

Risk aversion refers to the tendency to avoid risks[11]. The trader/investor prefers to safeguard the capital over the possibility of a higher than average reward. Being risk averse has its benefits as it guarantees a stable income while reducing the chances of incurring losses. However, there is a flip side to it such as the depreciation of savings due to inflation, lower expected returns, and missed opportunities. Thus, being risk averse can be both advantageous and disadvantageous.

3.2.2 Risk Seeking

Risk seeking is the opposite of risk averse. Here the trader/investor prefers to have the possibility for a higher than average reward over safeguarding the capital. They might be keen on investing in speculative assets over low risk assets.

3.2.3 Risk Neutral

When evaluating investment options, a risk-neutral person disregards the potential risks involved and concentrates solely on the potential gains. This could be coming from an emotional state and may not always be the advised thing to do. This way of thinking is frequently contextual and may be influenced by factors such as price or other external circumstances.

3.3 A sensitivity formula for risk-sensitive cost

In Borkar's paper[12] we find some fundamental results. The sensitivity formula derived in this paper allows for the efficient computation of the gradient of the expected cost with respect to the parameters of the policy, which can be used for policy optimization. The paper also discusses how the sensitivity formula can be extended to incorporate risk-sensitive cost functions, which take into account not only the expected cost but also the

variance of the cost.

The paper has developed a parametric sensitivity formula for the risk-sensitive cost

$$\nabla^{\theta}\Lambda(\theta) = \sum_{i,} \eta^{\theta}(i)\nabla^{\theta}\pi^{\theta}(i,u)q^{\theta}(i,u)$$

where λ is the risk-sensitive cost, a family of stationary randomized policies given by π^{θ}, θ is a vector parameter and η is a unique invariant probability vector.

Now this gets reduced to the (i, u)th component of $\nabla^{\pi}\Lambda(\pi)$ as $\eta^{\pi}(i)\left(q^{\pi}(i,u) - q^{\pi}\left(i,u_0\right)\right)$.

3.3.1 Risk Sensitivity in Actor-Critic Algorithm

Actor-critic algorithms simultaneously learn policies (actors) and value functions (critics). These algorithms use the actor to select actions and the critic to evaluate those actions and provide feedback to the actor. In the risk-sensitive setting, the following equations are used for updating Q and policy values respectively. 'I' refers to the indicator function and a and b are two scalar sequences that sum to ∞ when summed and less than ∞ when summed as a squared sum, also $\frac{a(n)}{b(n)} \to 0$

$$Q_{n+1}(i,u) = Q_n(i,u) + a(v(i,u,n))I\left\{X_n = i, Z_n = u\right\}\left(\frac{\mathrm{e}^{c(X_n,Z_n)}Q_n\left(X_{n+1},Z_{n+1}\right)}{Q_n\left(i_0,u_0\right)} - Q_n(i,u)\right)$$

$$\pi^{n+1}(i,u) = \Gamma\left(\pi^n(i,u) + b(v(i,u,n))I\left\{X_n = i, Z_n = u\right\}\left(Q_n(i,u) - Q_n\left(i,u_0\right)\right)\right)$$

Here Γ refers to the projection and I refers to the indicator function and the rest follows from before.

An important observation in the Q updating equations is that there is an exponential term in the Q target which acts as the risk-sensitive part and Q gets updated based on other Q values only. The policy gets updated by contributions from both the policy and Q functions. Hence the policy will also reflect the risk-sensitive formulations. The above equations are for discrete action space and we need to use continuous action space hence

we modify the equations accordingly.

Risk Sensitivity in Actor-Critic Algorithm for continuous action space

The following equations have been coded and their results are presented in the experimentation section 5.2.3. The Q loss and policy gradient is defined as follows

$$Q_{loss} = (Q_{target} - Q(s_t, a_t))^2$$

$$Q_{target} = \frac{Q(s_{t+1}, a_{t+1})}{Q(s_{ref}, a_{ref})} * e^{-R} - \alpha * log(\pi(s', a'))$$

$$\nabla_\theta J = \nabla_\theta \frac{1}{|B|} \sum_{s \in B} (Q_\phi(s, a) - \alpha \log \pi_\theta(s', a'))$$

The $\alpha * log(\pi(s, a))$ component is helpful for exploration purposes of the actor as the entropy of the policy is increased.[13] We also used replay memory to store the tuple (state, action, next action, reward) during every experience in the environment. A random subset from the memory is replayed to train the actor and critic during certain intervals. This is helpful for removing correlation between consecutive samples, therefore leading to more efficient learning.[14]

Chapter 4

Transformer-based Proximal Policy Optimization

4.1 Motivation

In recent years, deep reinforcement learning using both deep learning and reinforcement learning approaches has been found to solve complex problems in various domains such as healthcare, finance, robotics etc. In traditional reinforcement learning methods, when the state or action space grows larger it becomes difficult to store the information. Instead of looking up the values we can learn the generalizing function by approximating Q with a deep neural network.

Utilizing neural networks, such as deep convolutional networks, in conjunction with reinforcement learning algorithms, such as Q-learning, Actor-Critic, or Policy Search, produces a robust model (DRL) that can effectively handle previously hard problems. This is possible because DRL typically operates with raw sensory data or image signals as inputs, allowing for greater scalability and flexibility in its applications.

A variant of DQN known as Deep Recurrent Q Networks (DRQN), which includes recurrent layers, has demonstrated promising results in solving reinforcement learning problems.

In particular situations, DRQNs have shown to be more efficient than conventional DQNs, resulting in better-trained agents with better outcomes. This is due to the limited or no amount of history that DQN has access to. DQN is trained on a single-state representation corresponding to the current time step and has no contextual awareness of what happened before. They frequently disregard the input's temporal components. Even when state sequences are provided to the DQN model it is not able to extract the temporal aspect of the sequences.

By definition of the Markovian process, a state only depends on its previous state and not on the states before it. Hence any problem that requires a history of state sequences to comment on the present state will appear as non-Markovian. Instead of a Markov Decision Process (MDP), the problem becomes a Partially-Observable Markov Decision Process (POMDP). Hence LSTM's and transformers are considered[15].

4.2 Transformer

LSTM's and transformers have shown great results in NLP. The LSTM-based technique has an intrinsic "attention" mechanism that works by analyzing an input sequence and determining which other sequence elements are of significance at each step. The transformer method [16] was created as an alternative to the LSTM's attention mechanism in order to offer a more effective and focused attention mechanism. In contrast to LSTM, GRU, and other comparable models, the transformer model is an encoder-decoder architecture that does not require recurrent neural networks. This provides the combined advantages of decreasing training time while simultaneously increasing accuracy in NLP tasks.

4.3 Transformer Encoder Architecture

The input to the transformer encoder[16][17] is a matrix $H \in \mathbb{R}^{l \times d}$, where l is the sequence length, d is the input dimension. The next step of the process includes projecting H into

distinct spaces using three matrices - W_q, W_k, W_v, that are learnable. Typically, each of these matrices has a size of $\mathbb{R}^{d \times d_k}$, where d_k is a hyper-parameter. Subsequently, the scaled dot product attention can be computed using the equations that follow.

$$Q, K, V = HW_q, HW_k, HW_v$$

$$A_{t,j} = Q_t K_j^T$$

$$\text{Attn}(K, Q, V) = \text{softmax}\left(\frac{A}{\sqrt{d_k}}\right) V$$

where Q_t refers to the query vector for the t th token, while j denotes the token that the t th token attends to. K_j represents the key vector representation of the j th token. The softmax function is applied along the last dimension. Rather than using a single group of W_q, W_k, W_v, employing multiple groups can improve the effectiveness of self-attention. This approach is known as multi-head self-attention, and the equations can be extended accordingly. Position-wise feedforward networks will be used to further process the results of the multi-head attention operation. Other components of the transformer encoder include layer normalization and residual connection. Visualization of the transformer encoder can be found in fig. 4.1.

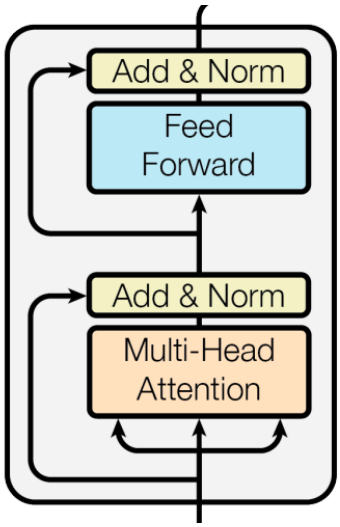

Fig. 4.1 Transformer Encoder [16]

4.4 Proximal Policy Optimization

Proximal Policy Optimization is an on-policy reinforcement learning algorithm that focuses more on sample efficiency and stability.[18] It is inspired by the Trust Region Policy Optimization (TRPO) algorithm and can be optimized by using first-order optimization.[19] The policy is learned by maximizing the clipped objective function given below using some gradient ascent algorithm.

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta (a_t \mid s_t)}{\pi_{\theta_k} (a_t \mid s_t)} A^{\pi_{\theta_k}} (s_t, a_t), \quad g\left(\epsilon, A^{\pi_{\theta_k}} (s_t, a_t)\right) \right)$$

The critic learns the value function by minimizing the loss function using some gradient descent algorithm.

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi (s_t) - \hat{R}_t \right)^2$$

The advantage values $A^{\pi_{\theta k}}$ for the clipped objective function are calculated using the value function V_ϕ.

4.5 Updated Environment

To improve the performance of the agent to learn more temporal patterns, we have increased the dimension of observation space to include a time window of previous stock features. The new state dimension is of the form (time window, number of stocks, close price + indicators + previous weight).

4.6 Model Architecture

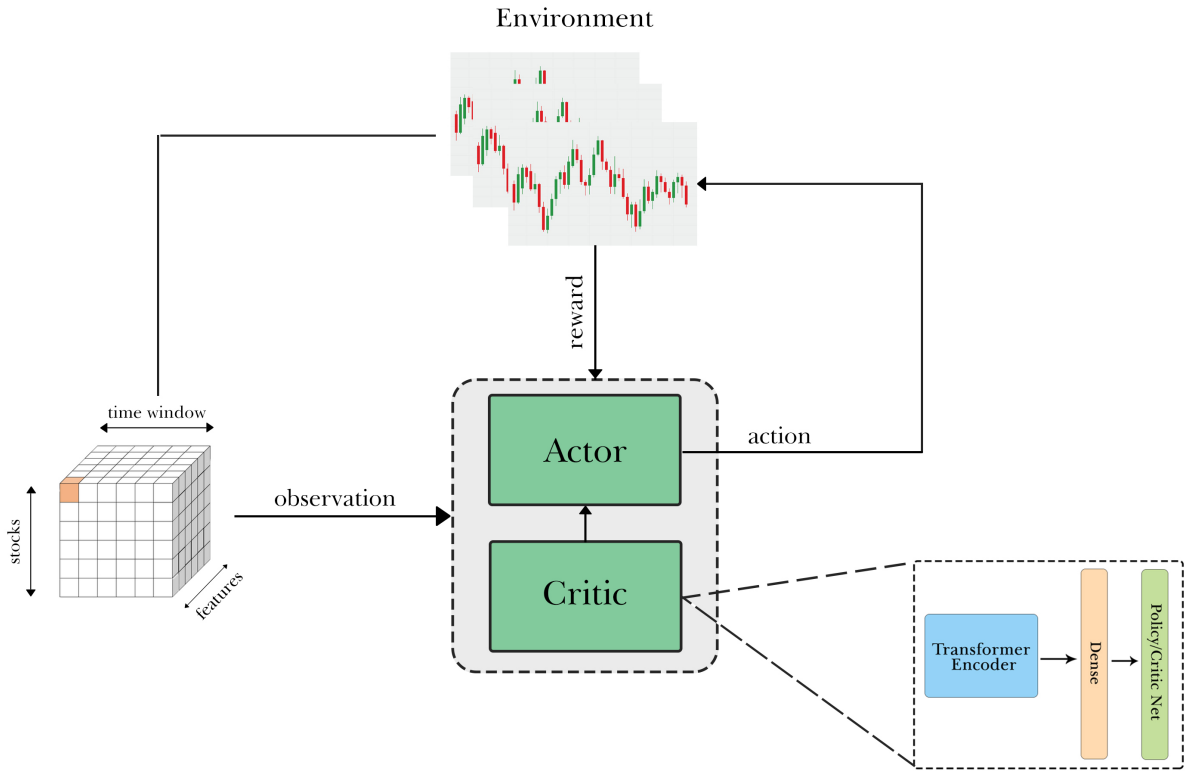

Fig. 4.2 Transformer-based PPO

The reinforcement learning agent uses the actor-critic method to interact and learn from the environment. The environment provides the agent with an observation which is then fed into both actor and critic's transformer encoders. The transformer encoder extracts contextualized features from the environment state which are then fed into the actor and critic's dense networks. Proximal Policy Optimization algorithm is used to update the parameters of the networks.

Chapter 5

Experiments and Observations

5.1 Market Analysis

5.1.1 Bull Market

When the economic conditions are favorable and the stock prices are rising, the market is said to be a bull market. Market conditions are also impacted by the investors' attitudes and how they feel about the market trends. Bull markets have a sustained period where the stock price rises. Investors feel safe when the market is showing booming trends. Economic conditions are strong, and employment is also high. This usually results in a buyer's market. [20]

5.1.2 Bear Market

When the market is on the decline, we say the market is in a bear market. We term it a bear market when it falls by more than 20% from the highs. Stock prices continuously fall, and unemployment rates increase. This also causes anxiety in the investors, contributing to the bear market. Since 1928, there have been twenty-five bear markets; fourteen (56%) have also been followed by recessions, while the other eleven (44%) have not. This results in a seller's market.[1]

5.1.3 Time Lines [1]

1. 2000 - 2002: The dot-com crash caused the S&P 500 to crash by 36.8%
2. 2007 - 2009: The economy goes into recession and enters the second-worst bear market
3. 2014 - 2016: Oil prices and other stocks were in a bear market
4. 2020: Due to the pandemic, the market entered a bear market.
5. 2022 - present: We are currently in a bear market.

5.2 Experiment

We intend to benchmark multiple allocation optimization strategies, including standard deterministic and deep reinforcement learning methods. We conduct a thorough evaluation in which we examine the robustness and performance of such approaches for various market situations and different frequencies of reallocation.

5.2.1 Experiment Setting

When the exchange closes, it does so on the closing price, which may not be an accurate presentation of the stock value; hence we use the adjusted price, which considers the dividends, stock splits, and new stock issues. The dataset is split into training and testing with the 80-20 ratio split for the DRL methods. For traditional deterministic approaches, no split is used. We assume a small transaction cost for each transaction and we limit the total number of transactions on each stock. We try the following methods.

Traditional Methods

1. Max Sharpe
2. Minimum variance portfolio
3. Risk parity
4. Equal weight

Deep Reinforcement Learning Methods

1. A2C (Actor Critic method)
2. PPO (Proximal Policy Optimization)
3. DDPG (Deep Deterministic Policy Gradient)
4. SAC (Soft Actor Critic)
5. TD3 (Temporal Difference)

We do the DRL methods over many runs, and we report three types of results - mean of all the runs, maximum return, and minimum return of all the runs.

5.2.2 Experiment 1: Maximum Sharpe Portfolio

In this initial experiment, we tried to test out our hypothesis on whether choosing the portfolio which maximises Sharpe Ratio showed good results.

- iShares MSCI World ETF (URTH)
- Vanguard Total International Bond Index Fund ETF (BNDX)

The dataset we used for this was daily adjusted stock price data of BNDX and URTH from 06-2013 to 11-2022. Here BNDX is the risk-free asset and URTH the risky asset.

We compared the performance of the Sharpe Ratio maximisation method with 60:40 portfolio. Where 60% of the assets are invested in equities and 40% are invested in bonds. In our scenario, we invested 60 in URTH and 40 in BNDX.

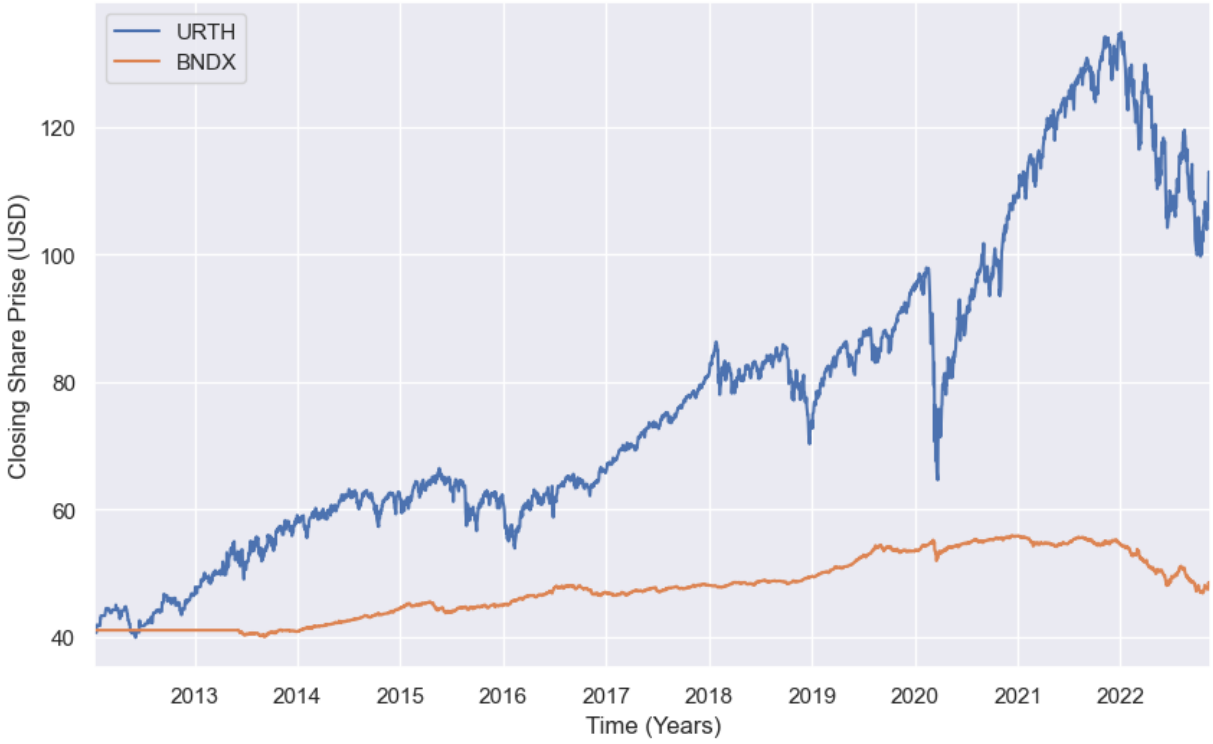

Fig. 5.1 Stock Price over the years

Lagrangian Multiplier

The initial method was maximizing the Lagrangian Equation with constraints. This was done using tensorflow's constrained optimization and the optimizer we used was AdaGrad. [21]

$$L(w_1, w_2, ..., w_n, \lambda) = Sharpe(w_1, w_2, ..., w_n) + \lambda(\sum_{i=1}^{n} w_i - 1)$$

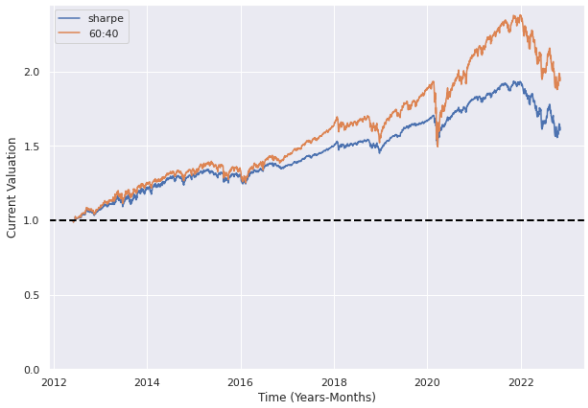

Fig. 5.2 Performance of Max Sharpe v/s 60:40

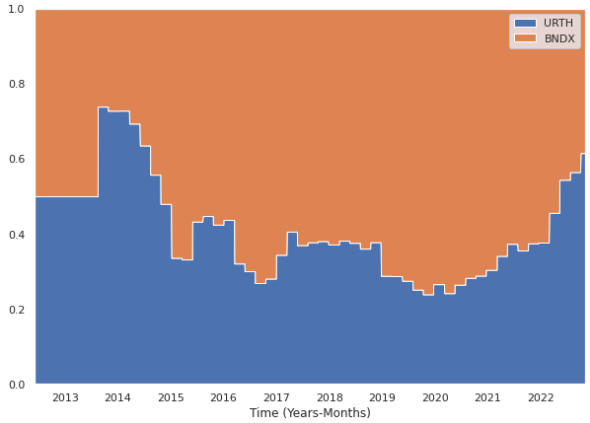

Fig. 5.3 Weights Distribution in Max Sharpe

Convex Optimization

Method 2 involved converting the Sharpe Function into a convex minimization problem as mentioned in *Optimization Methods in Finance.*[22].

Fig. 5.4 Performance of Max Sharpe v/s 60:40

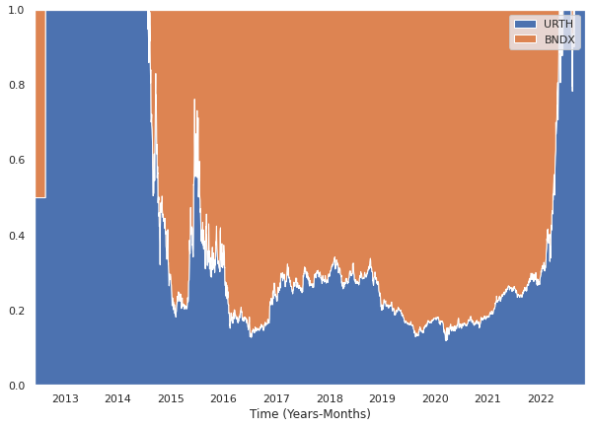

Fig. 5.5 Weights Distribution in Max Sharpe

PPO reward

In the 3rd method, we used the Sharpe ratio of the portfolio as the reward for the PPO Agent.

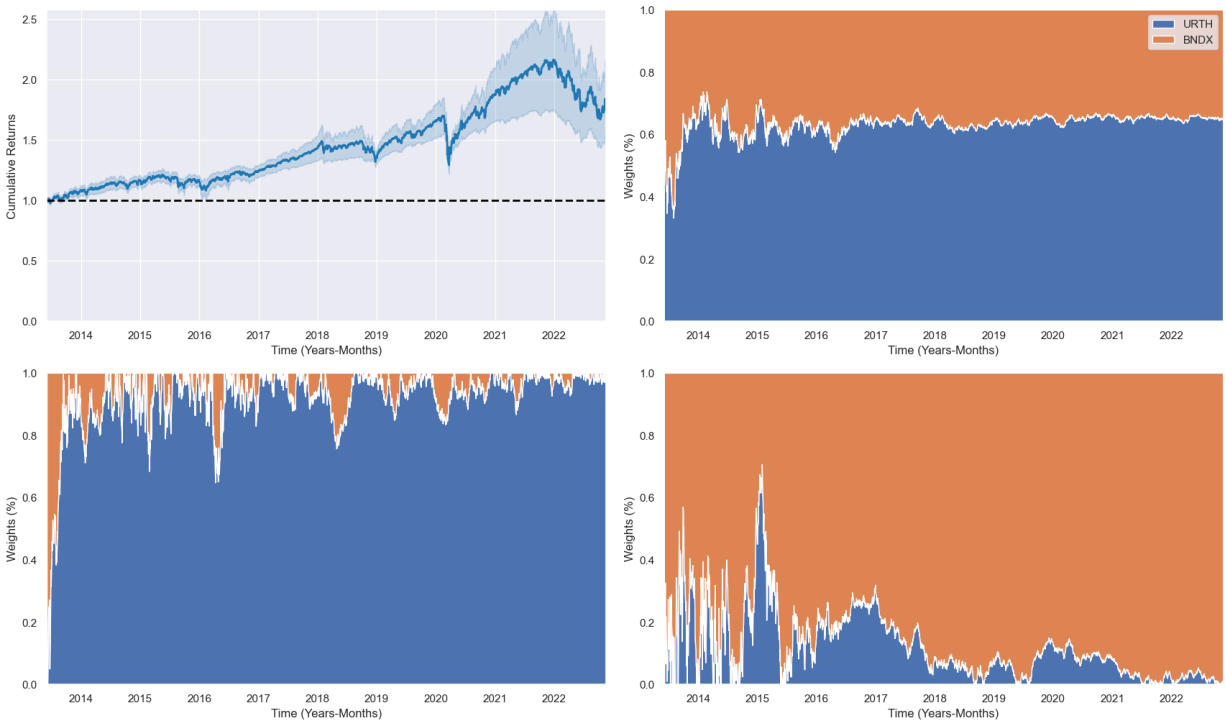

Fig. 5.6 Performance of PPO with reward as Sharpe Ratio. (i) The graph on the top right is the weight distribution for the median reward run of the Agent. (ii) The graph on the bottom left is the weight's distribution for the best reward run of the Agent. (iii) The graph on the bottom right is the weight's distribution for the worst reward run of the Agent.

5.2.3 Experiment 2: Risk Sensitive Actor-Critic

This experiment aims at achieving risk-sensitive results. The stocks considered for this experiment are the DOW 30. The Dow 30 stocks are a collection of 30 large, well-established companies listed on the New York Stock Exchange (NYSE) and NASDAQ. The training time period is from 2010-01-01 to 2021-10-01 and the test period is from 2021-10-01 to 2023-03-01. The stock data are preprocessed to include many technical indicators such as macd, boll_ub, boll_lb, rsi_30, dx_30 etc.

The Stock Environment is based on the FinRL stock trading environment[23]. The environment consists of a stock trading simulation with a defined set of actions, states, and observations. The action space is [-1,1] for each stock; action $< 0 \rightarrow$ sell, action $> 0 \rightarrow$ buy. At initialization, the number of stocks and the action space are specified, with the state represented as a list of the current balance, stock prices, number of stocks owned and technical Indicators (MACD, RSI, CCI, ADX). The observation space is any state, and memory is used to store the history of balances. The sell and buy functions are used to sell or buy a particular stock by the given action amount, and the step function updates the assets and executes the buy or sell action. The reward is calculated as a combination of the Sharpe ratio and the difference between the final assets and the assets before the action. Memory is updated by appending normalized stock units.

The Risk Sensitive Actor Critic agent 3.3.1 is trained and tested to get the following results. The results of the agent are plotted against DDPG, PPO and A2C.

Table 5.1 Performance Comparison of RL Models

	Risk Sensitive AC	DDPG	PPO	A2C	Dow Jones Index
Annual return	-0.644%	2.346%	-10.498%	-5.189%	-3.487%
Cumulative returns	-0.901%	3.302%	-14.39%	-7.192%	-4.864%
Annual volatility	17.335%	17.381%	17.636%	22.838%	18.161%
Sharpe ratio	0.05	0.22	-0.54	-0.12	-10.535%
Max drawdown	-19.726%	-18.096%	-27.131%	-26.917%	-21.940%
Daily value at risk	-2.181%	-2.175%	-2.26%	-2.888%	-2.295%

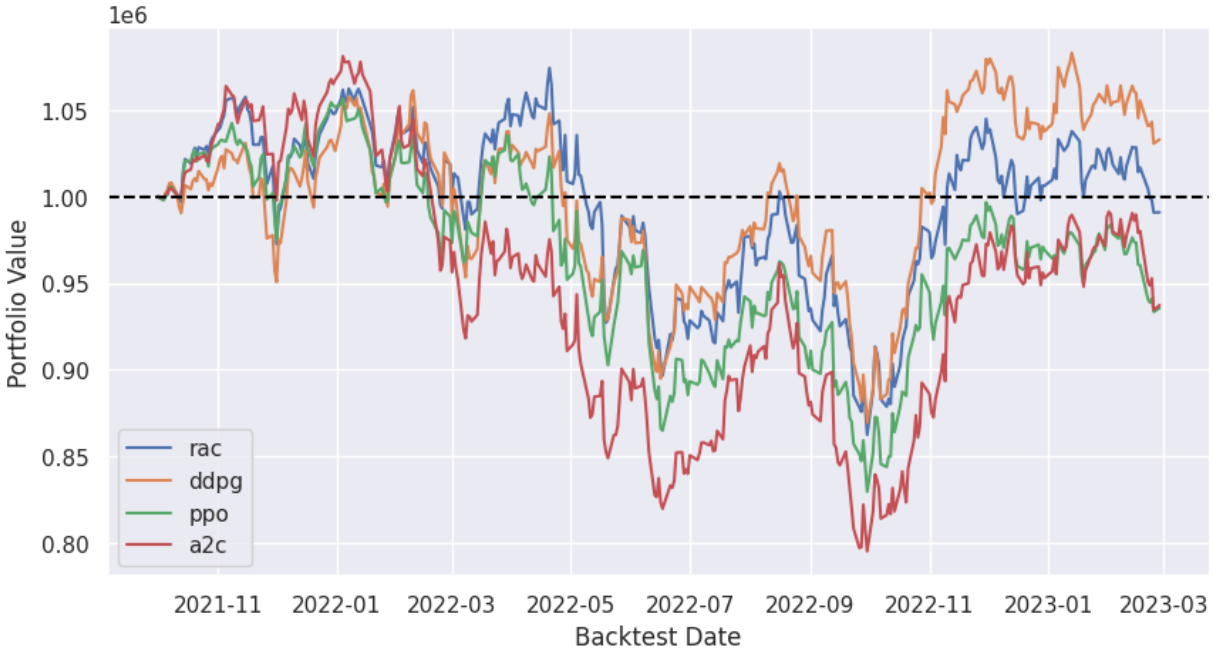

Fig. 5.7 Cumulative Return of different methods vs Time

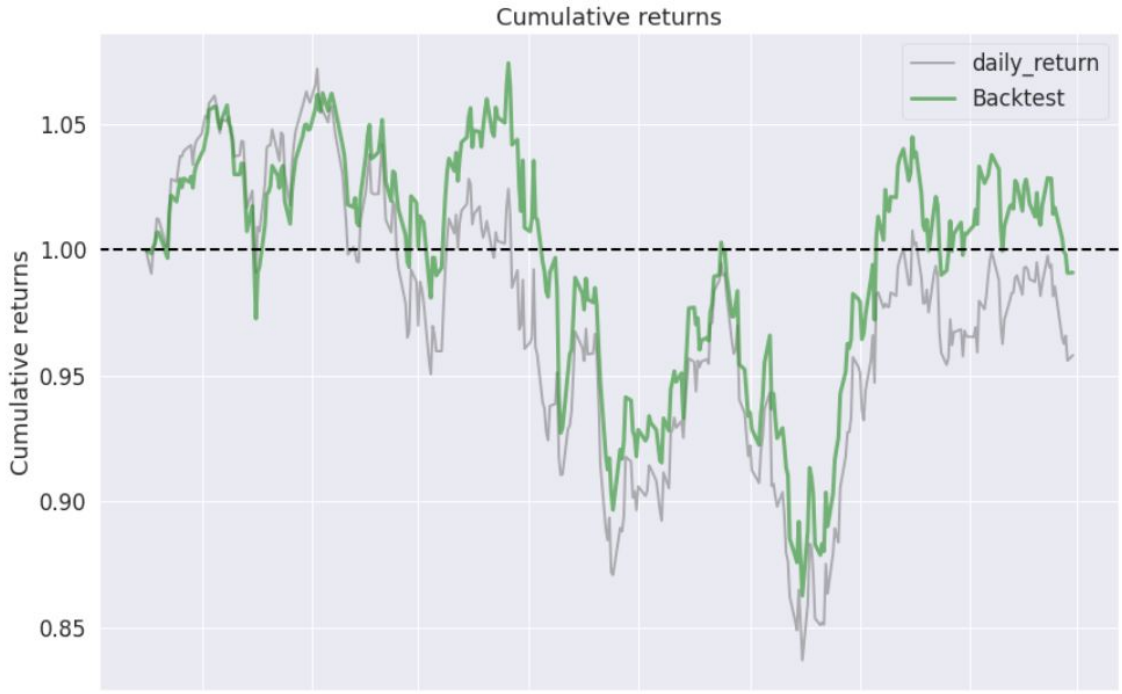

Fig. 5.8 Daily Returns of DJI compared to Risk Sensitive AC's return during Backtest period

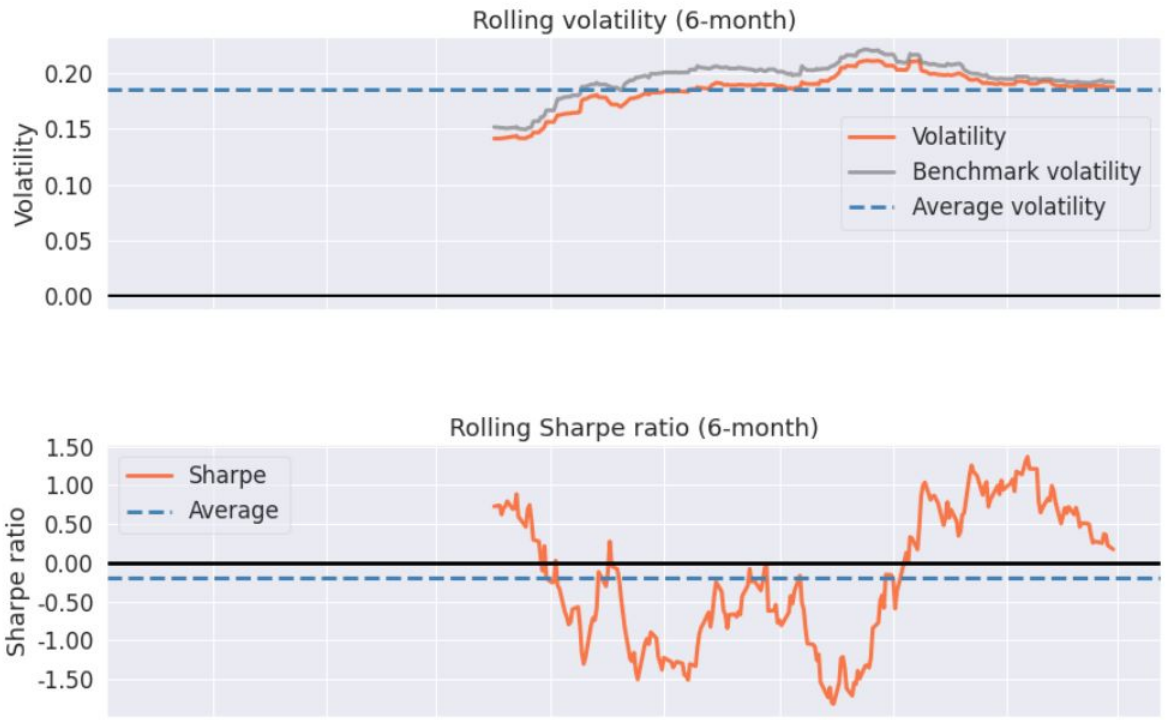

Fig. 5.9 Rolling Volatility and Sharpe Ratio of Risk Sensitive AC

5.2.4 Experiment 3: Transformer-based Proximal Policy Optimization

The primary objective of this experiment is to evaluate the performance of a reinforcement learning-based portfolio optimization strategy utilizing a transformer-encoded proximal policy optimization (PPO) algorithm. The dataset for this experiment (Dow Jones) as well as the timeline and pre-processing is the same as in the previous experiment section 5.2.3. The environment is described in section 4.5.

The transformer encoder based PPO architecture as described in section 4.6 is tested here. The portfolio optimization approach suggested will be assessed by comparing its performance with that of the Dow Jones index. The evaluation criteria will encompass several metrics, such as the Sharpe ratio, annualized return, maximum drawdown etc.

Table 5.2 Backtest results

	Risk Sensitive AC	DDPG	Transformer PPO
Annual return	-0.644%	2.346%	7.758%
Cumulative returns	-0.901%	3.302%	10.771%
Annual volatility	17.335%	17.381%	18.953%
Sharpe ratio	0.05	0.22	0.49
Max drawdown	-19.726%	-18.096%	-16.851%
Daily value at risk	-2.181%	-2.175%	-2.351%

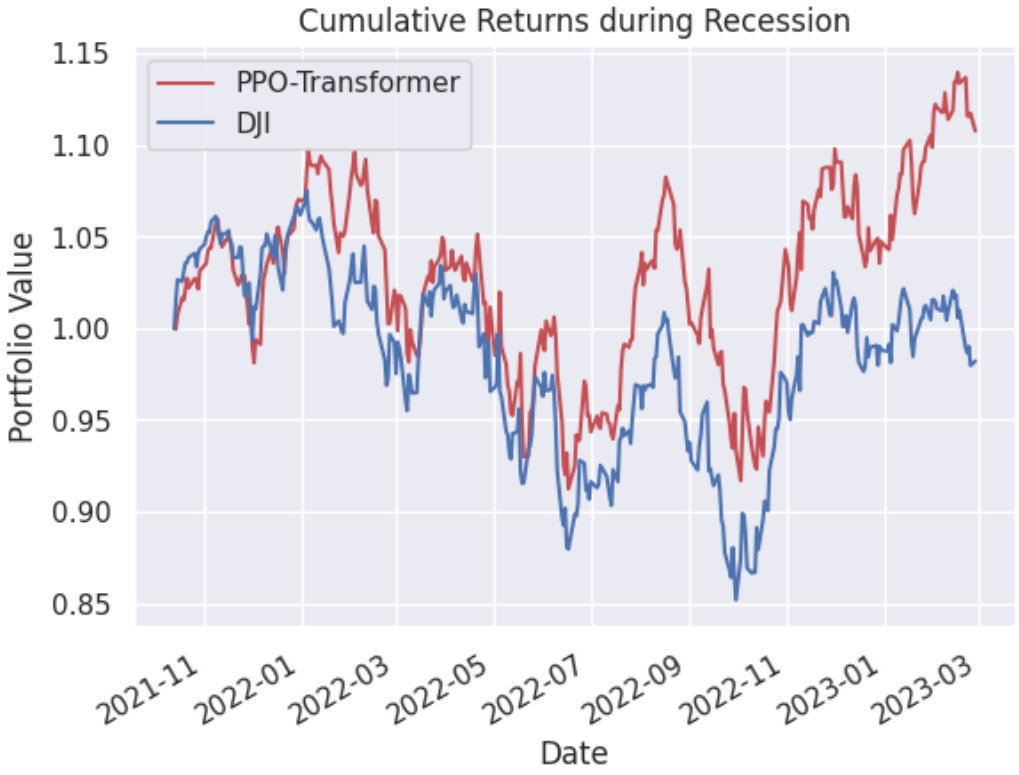

Fig. 5.10 Comparing Transformer-PPO with Dow Jones Index

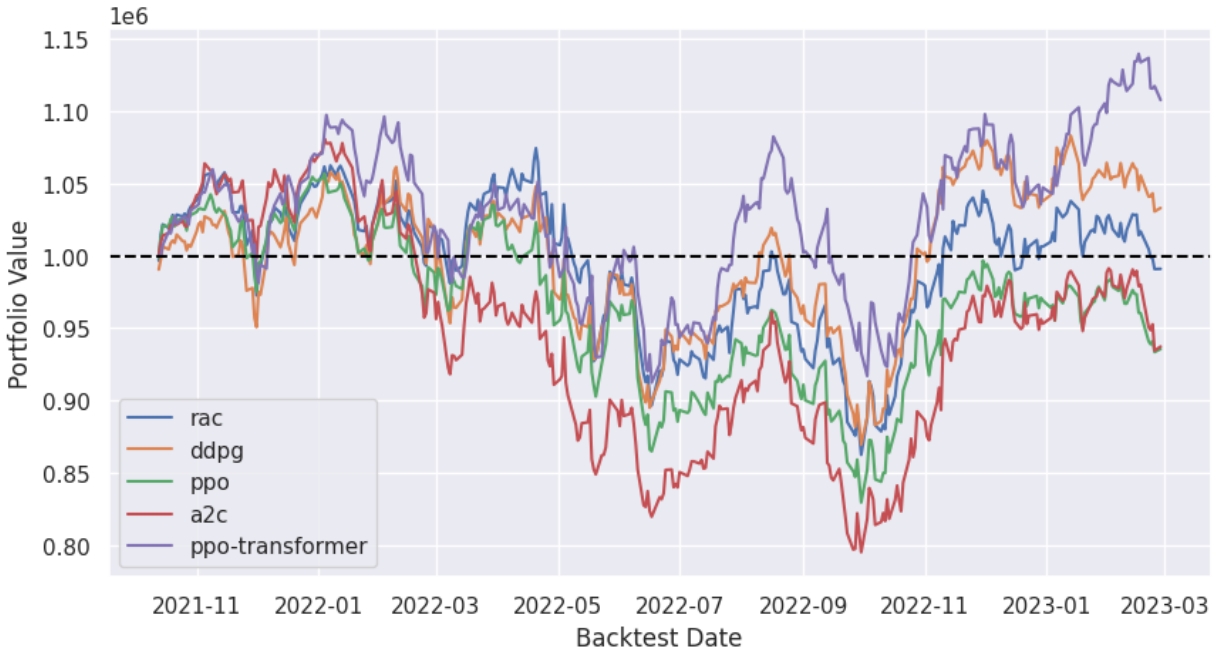

Fig. 5.11 Comparing Transformer-PPO with other models

5.2.5 Observations

The first method used for maximizing Sharpe was very expensive as it required optimizing an equation using Lagrangian method during every time step. The method did not give a global optimum solution for weights (quasi-convex) and did not display good performance. The second method for maximizing Sharpe was more accurate at predicting the most optimal weights and also displayed better results in the early timesteps. The Sharpe Ratio as Proximal Policy Optimization (PPO) reward did perform better when compared to the other two methods during the recession period. We observe that no method can outperform all others; different methods work differently. Deep Reinforcement Learning (DRL) methods are also unstable regarding results, whereas traditional approaches are more stable. Hybrid combinations between them could produce better results.

The Risk-Sensitive Actor-Critic (RAC) agent provides us with some insight on how effective risk-sensitivity is in the financial market during the recession period(2021-2023). Although our algorithm is not as good as Deep Deterministic Policy Gradient (DDPG) model in terms of cumulative returns, it outperforms the Proximal Policy Optimization (PPO) and Advantage Actor-Critic (A2C) agents. RAC achieved a greater peak and performed better than the others during the early training period due to market stability. RAC had the lowest annual volatility among all the other methods (17.33%).

From the transformer PPO experiment results, we observe that the annual return for the Risk-Sensitive AC (RAC) is negative (-0.64%), the DDPG model had a lower value of 2.34% compared to the transformer PPO model's annual return of 7.75%. This shows that the transformer PPO model is better at making a profit even during a recession period. Sharpe ratio was also higher for the Transformer PPO, suggesting a better portfolio allocation strategy. The Max drawdown of the RAC and DDPG models was higher than that of the transformer PPO model, indicating that the former models experienced larger losses in the value of the portfolio during the evaluated period. We can conclude that the transformer PPO model outperformed the RAC and DDPG models in terms of overall performance.

Chapter 6

Future Work and Scope

1. Developing better Reward functions in order to bias risk sensitivity.
2. Develop a better measure for understanding the Agent's performance regardless of the market scenario (Bullish/Bearish/Recessions).
3. Developing an end to end trading software.
4. Crawl and collect external data sources (news articles and social media sentiment analysis) and include them in the environment to improve it's predictive power.
5. Explore imitation Learning and transfer learning in the financial market
6. Creating an interpretable artificial intelligence framework that can assist investors in comprehending the model's decision-making process.
7. Exploring the use of graph neural network encoding to better understand market features and their interconnections, leading to more robust portfolio optimization strategies

References

[1] "History of bear market," https://www.investopedia.com/a-history-of-bear-markets-4582652.

[2] B. Clark, Z. Feinstein, and M. Simaan, "A machine learning efficient frontier," *Operations Research Letters*, vol. 48, no. 5, pp. 630–634, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167637720301139

[3] Y. Gao, Z. Gao, Y. Hu, S. Song, Z. Jiang, and J. Su, "A framework of hierarchical deep q-network for portfolio management," in *ICAART*, 2021.

[4] A. Oshingbesan, E. Ajiboye, P. Kamashazi, and T. Mbaka, "Model-free reinforcement learning for asset allocation," Ph.D. dissertation, 09 2022.

[5] R. Durall López, "Asset allocation: From markowitz to deep reinforcement learning," 07 2022.

[6] 2016. [Online]. Available: https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver

[7] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," 2017. [Online]. Available: https://arxiv.org/abs/1706.10059

[8] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham,

N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 01 2016.

[9] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, 2017.

[10] P. L.A. and M. Ghavamzadeh, "Actor-critic algorithms for risk-sensitive mdps," *Advances in Neural Information Processing Systems*, 02 2013.

[11] J. Chen, "Risk averse: What it means, investment choices and strategies james chen," Sep 2022. [Online]. Available: https://www.investopedia.com/terms/r/riskaverse.asp

[12] V. Borkar, "A sensitivity formula for risk-sensitive cost and the actor–critic algorithm," *Systems Control Letters*, vol. 44, pp. 339–346, 12 2001.

[13] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *CoRR*, vol. abs/1812.05905, 2018. [Online]. Available: http://arxiv.org/abs/1812.05905

[14] R. Liu and J. Zou, "The effects of memory replay in reinforcement learning," *CoRR*, vol. abs/1710.06574, 2017. [Online]. Available: http://arxiv.org/abs/1710.06574

[15] U. Upadhyay, N. Shah, S. Ravikanti, and M. Medhe, "Transformer based reinforcement learning for games," 12 2019.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc.,

2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[17] H. Yan, B. Deng, X. Li, and X. Qiu, "Tener: Adapting transformer encoder for name entity recognition," 11 2019.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[19] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015. [Online]. Available: http://arxiv.org/abs/1502.05477

[20] "Bull and bear markets," https://www.investopedia.com/insights/digging-deeper-bull-and-bear-markets.

[21] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[22] G. Cornuéjols, J. Peña, and R. Tütüncü, *Optimization Methods in Finance*. Cambridge University Press, 2018. [Online]. Available: https://books.google.co.in/books?id=Dq1jDwAAQBAJ

[23] "Finrl stock env." [Online]. Available: https://github.com/AI4Finance-Foundation/FinRL/blob/master/finrl/meta/env_stock_trading/env_stocktrading.py