International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 11 | November - 2022 Impact Factor: 7.185 ISSN: 2582-3930

Portfolio Optimization using Reinforcement Learning

Authors

Dhruv Joshi , Adwait Patil , Shreyash Mutyalwar

College of Engineering, Pune 411005

Abstract

Reinforcement Learning has contributed to the automation of various tasks, including electronic algorithmic trading. The main benefit of Reinforcement Learning in financial applications is the relaxation of assumptions on market models and hand-picked features, allowing a data-driven, automated process. The performance of Reinforcement Learning agents on the portfolio management problem is measured. A finite universe of financial instruments such as stocks is selected and the trained agent is constructing an internal representation (model) of the market, allowing it to determine how to optimally allocate funds of a finite budget to those assets. The agent is trained on the actual market prices. The performance metrics are then compared with those of standard portfolio management algorithms on a dataset that has not been used before. A successful agent can be used as a consulting software for portfolio managers or it can be used for low-frequency algorithmic trading. Moreover, it can allow us to identify the missing pieces of the existing models and suggest directions to improve them.

1 Introduction

In most developed countries, a high percentage of the population invests in various assets apart from minimal-risk bank deposits. But the same number is quite low for developing countries like India. Financial literacy cannot be created overnight, which is why we need a solution that automates the investments of low-risk groups. We intend to leverage Reinforcement Learning to maximize long-term returns on various assets with minimum risk. Reinforcement learning is the branch of machine learning that deals with optimal sequential decision-making, which has traditionally been used in the gaming industry.

[1] Reinforcement learning addresses a problem that most other learning techniques do not: it takes into account the computational issues that arise when an agent-environment interaction takes place. It does so by using a framework that defines the interaction between the agent and the environment in terms of states, actions, and rewards.[2] Traditionally, statistical methods and processes have been used by equity researchers and traders to find trading opportunities in the financial markets. But a huge chunk of retail investors and small-scale traders are unaware of these methods because of their lack of understanding of these methods. Reinforcement learning presents an opportunity to provide access to the same to the masses oblivious to complex mathematics. In this

paper, we explore the Portfolio Management problem using Reinforcement Learning. The major challenge faced with asset allocation strategies is estimation errors in forecasting expected returns used in the construction of an investment portfolio. This estimation error tends to be magnified to a greater extent in optimization problems, which in most cases leads to under-performance of these strategies compared to naive strategies like ones that allocate capital equally across all assets. An added challenge to the problem is accounting for the transaction costs associated with the frequent rebalancing of a portfolio. Entering and exiting positions frequently to take advantage of short-term or medium-term market movements tends to undermine the long-term objectives of a portfolio by increasing trading costs and thus leading to reduced returns. After the successful implementation of a Deep Q-Network (DQN) by Google DeepMind[3], there has been an explosive adoption of Reinforcement Learning across many fields.

The solution to the portfolio management problem is one that accurately interprets market trends in the short to medium-term time frame and uses that interpretation to optimize the weights of various assets in the portfolio so that the returns are maximized while minimizing risk.

The study aims to investigate the application of Deep Reinforcement Learning (Deep Reinforcement Learning) models to build a risk-reward optimized portfolio. The problem is modeled as a Markov Decision Process(MDP). Markov decision process problems (MDPs) assume a finite number of states and actions. At each time the agent observes a

1 *Introduction*

state and executes an action, which incurs intermediate costs to be minimized (or, in the inverse scenario, rewards to be maximized). The cost and the successor state depend only on the current state and the chosen action. Successor generation may be probabilistic, based on the uncertainty we have on the environment in which the search takes place [4]

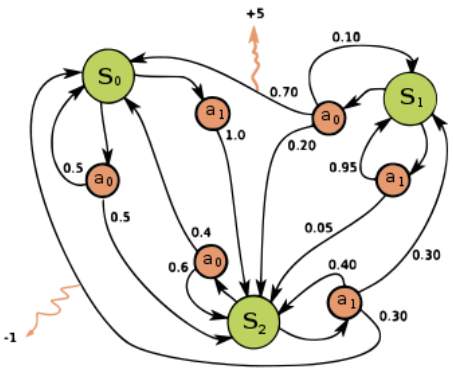

Figure 1.1: Example of a Markov Decision Process

The portfolio allocation problem is modeled as a Markov Decision Process (MDP) and Deep

Reinforcement Learning is used to optimize the allocation strategy.
Three Deep Reinforcement Learning models are explored, namely A2C, DDPG, and PPO RL algorithms. The performance of these models is compared with each other to gain an insight into the individual performance of each Deep Reinforcement Learning model.

1.1 Understanding the Problem

A Deep Neural Network is proposed as the agent which finds an optimal policy that acts on the state of the environment to produce actions that maximize the reward function. The The deep Reinforcement Learning process is modelled as follows: P_0 is the investment at time zero that we start with. The job of the objective function in the optimization problem is to maximize this cumulative portfolio value at time t.
We define the Reinforcement Learning Configuration of our model as follows:

- The state (S) includes the High, Low, Close, the covariance matrix of close prices and the identified market financial indicators that are used as inputs;
- The action (A) is the desired weights allocation. The action at time t will be represented by the weights vectors at time $t-1$: $A_{t-1} = W_{t-1}$. As a result of the action at A_{t-1} and the state inputs, we will get an action or weights allocation at time t which is $A_t = W_t$. Taking into consideration the transaction costs, our model should make a decision on how the weights are re-balanced from W_{t-1} weights W_t .This is done to maximize the accumulative portfolio value.
- The reward function is given based on the total portfolio value adjusted for thetransaction costs. The reward function adjusted for transaction costs is given by:

https://www.overleaf.com/project/625a592af63258de14692d6d

$$R_t(S_{t-1}, A_{t-1}) = \ln(A_{t-1} \cdot Y_{t-1} - \mu \sum_{i=1}^{n} |A_{i,t-1} - W_{i,t-1}|) \quad (1.1)$$

The reward function is fed back to the agent to reinforce the policy of making actions so that future actions are optimized to reach the objective function.

- Policy, $\pi(s, a)$. The policy determines the action to take that maximizes the reward at each state. In our case the policy is made by a Deep Neural network using a Reinforcement Learning algorithm.

1.2 *Assumptions*

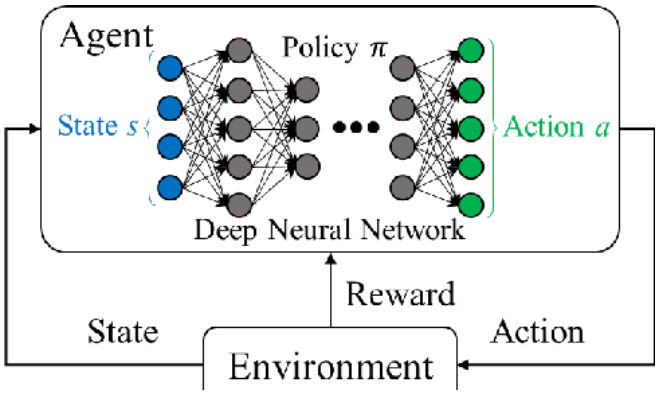

Figure 1.2: Basic Representation of a Deep Reinforcement Learning Framework

1.2 Assumptions

1.2.1 Sufficient Liquidity

Liquidity refers to the ease with which an asset, or security, can be converted into ready cash without affecting its market price. [5] [6] The assumption that we make is that all market assets are sufficiently liquid and every transaction can be executed under the same conditions.

1.2.2 Zero Slippage

Slippage refers to all situations in which a market participant receives a different trade execution price than intended. We assume that the liquidity of all market assets is high enough so that, each trade can be carried out without slippage.
1 Introduction

1.2.3 Free Market

A free market is one where voluntary exchange and the laws of supply and demandprovide the sole basis for the economic system, without government intervention.

1.2.4 Transaction Costs

Transaction costs have been assumed to be 0.1% of the total value of each order. Thisvalue can be changed by the user to best suit their broker's charges.

2 Methodology

The methodology proposed for the asset allocation problem follows the following steps:

1. **Stock Selection**: We chose the 50 constituent stocks of the NIFTY50 index as the universe from which stocks are individually chosen depending on their volatility. Neural Network Auto Encoders are proposed to perform this filtration;
2. **Feature Selection and Data Pre-processing**: The use of technical indicators such as EMA, MA, ADX, etc is done to improve the performance of the agent;
3. **Feature Reduction**: To make the data easier for processing, the dimensions are reduced using Neural Network Auto-encoders.
4. **The Model**: The core of the solution lies in the Deep Reinforcement Learning model which is based on the A2C, PPO, and DDPG reinforcement learning algorithms.
5. **Back-testing**: The performance of the models is then tested on the test data set and the results of each model (A2C, PPO, and DDPG) are compared.

2.1 Stock Selection

We use the data for the 50 stocks of the NIFTY50 for the period from 01 January 2014 to 01 January 2022 giving us a total of 1972 data points. The table in the figure 2.1 below shows the head of the downloaded data frame :

```
['AXISBANK.NS', 'TATAMOTORS.NS', 'BAJFINANCE.NS', 'HCLTECH.NS', 'CIPLA.NS', 'DIVISLAB.NS', 'MARUTI.NS', 'COALINDIA.N
S', 'SBIN.NS', 'NTPC.NS', 'HINDALCO.NS', 'APOLLOHOSP.NS', 'INFY.NS', 'ULTRACEMCO.NS', 'BHARTIARTL.NS', 'POWERGRID.NS
', 'BAJAJ-AUTO.NS', 'ICICIBANK.NS', 'INDUSINDBK.NS', 'JSWSTEEL.NS', 'M&M.NS', 'HDFC.NS', 'HINDUNILVR.NS', 'UPL.NS',
'BAJAJFINSV.NS', 'HDFCBANK.NS', 'TATASTEEL.NS', 'ONGC.NS', 'SUNPHARMA.NS', 'LT.NS', 'BPCL.NS', 'HEROMOTOCO.NS', 'REL
IANCE.NS', 'DRREDDY.NS', 'TECHM.NS', 'ASIANPAINT.NS', 'NESTLEIND.NS', 'GRASIM.NS', 'BRITANNIA.NS', 'ITC.NS', 'WIPRO.
NS', 'KOTAKBANK.NS', 'EICHERMOT.NS', 'TITAN.NS', 'TCS.NS', 'ADANIPORTS.NS', 'TATACONSUM.NS', 'SHREECEM.NS']
```

Figure 2.1: List of partially filtered NIFTY50 stocks

3 stocks for which the number of data points was different (less than 1972) are filtered out, leaving us with 47 stocks. From this set of 47 stocks, 20 with lower volatility are selected using Auto-encoders. We construct an Auto-encoder with the total number of stocks in our input layer, we have an encoder layer with dimension 5 and a decoder layer with dimension 47. The stock data is reconstructed by passing it through the Auto-encoder and then the reconstruction error of each stock is calculated. We then pick the 20 stocks

2 Methodology

with the lowest reconstruction errors. Below is the summary of the Auto-encoder model:

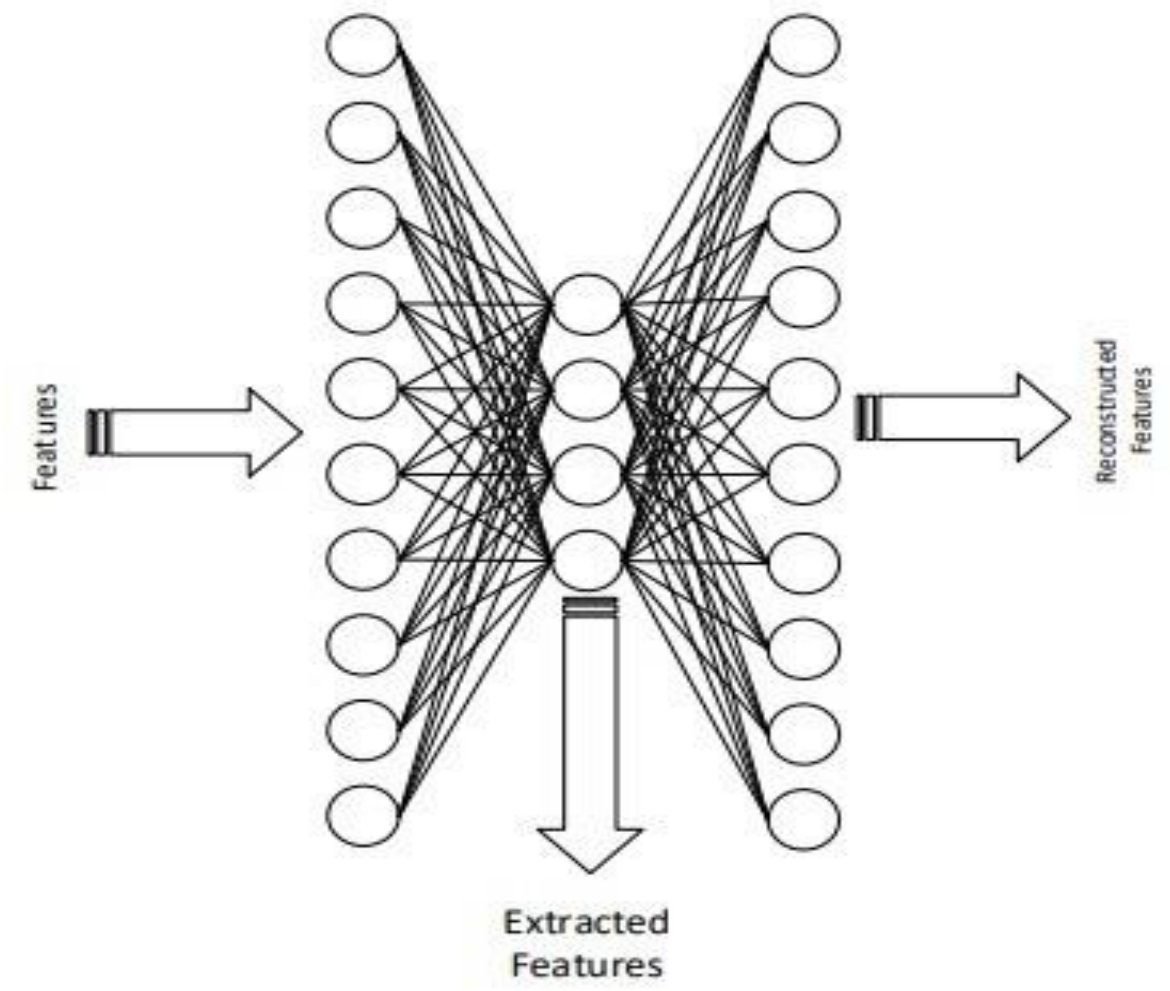

Figure 2.2: Auto-encoder model representation

2.2 Feature Selection and Data Pre-processing

The mechanism of the Auto-encoder model can be summed up using the following figures:

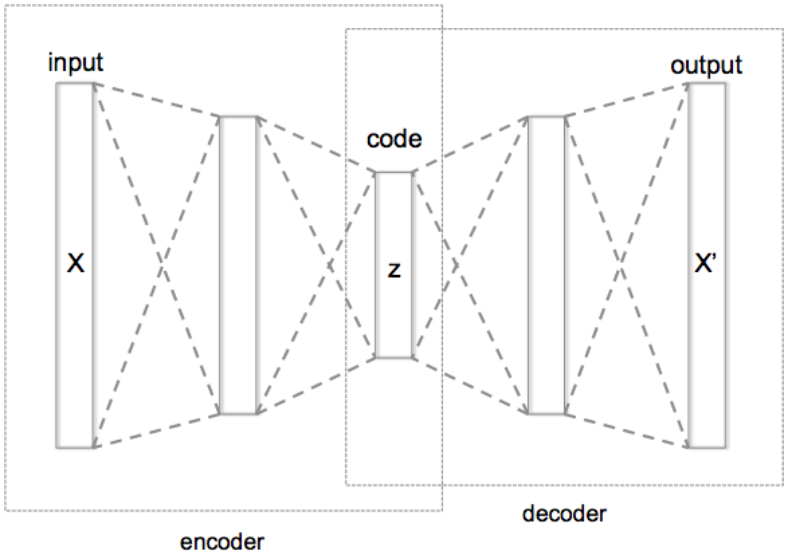

Figure 2.3: Representation of an Auto-Encoder Model

The following is the list of the selected stocks by the Auto-encoder model:

```
Index(['HDFCBANK.NS', 'POWERGRID.NS', 'LT.NS', 'HINDUNILVR.NS', 'RELIANCE.NS',
       'NESTLEIND.NS', 'BAJAJ-AUTO.NS', 'ASIANPAINT.NS', 'HEROMOTOCO.NS',
       'SHREECEM.NS', 'TCS.NS', 'ITC.NS', 'NTPC.NS', 'GRASIM.NS',
       'COALINDIA.NS', 'ULTRACEMCO.NS', 'WIPRO.NS', 'CIPLA.NS', 'M&M.NS',
       'MARUTI.NS'],
      dtype='object', name='stock_name')
```

Figure 2.4: List of Selected Stocks by the Auto-encoder Model

2.2 Feature Selection and Data Pre-processing

The following indicators are considered:

1. Relative Strength Index (RSI)
2. Simple Moving Average (SMA

3. Exponential Moving Average (EMA)
4. Average True Range (ATR)
5. Moving Average Convergence Divergence (MACD)
6. Bollinger Band Width (BBW)
7. On-balance Volume (OBV)
8. Chaikin Money Flow (CMF)
9. Average Directional Index (ADX)
10. Commodity Channel Index (CCI)

Dimensionality reduction using unsupervised stacked restricted auto-encoder model is performed to reduce the number of input features from ten to four. Auto-encoders consist of two parts which are; the encoder, which reduces the dimensions of the input data in the latent hidden layer and the decoder, which reconstructs the data from the hiddenlayer.

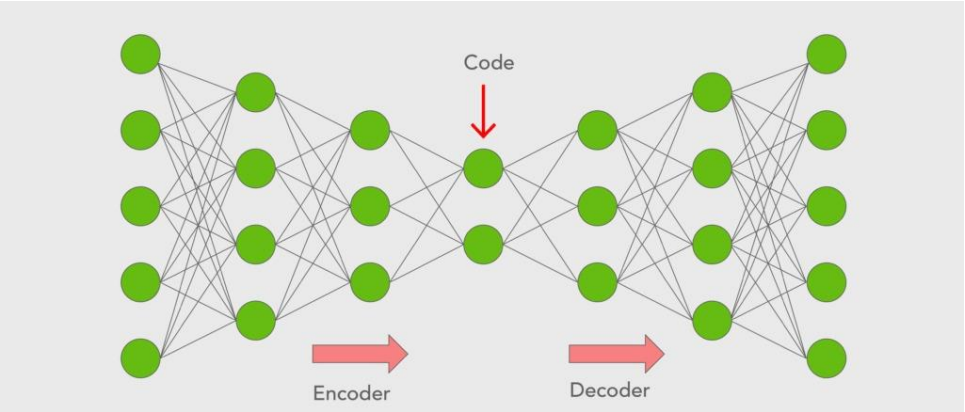

Figure 2.5: Flow Diagram of an Auto-Encoder Model

2.3 Feature Reduction

We perform the dimensional reduction of the added features (technical indicators) to optimize the learning process of the model. Neural Network Auto-encoders are used to perform feature reduction.

The newly added 10 technical indicators are reduced to four new features.

2.4 Train and Test Data Split

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 4)                 240

 repeat_vector (RepeatVector  (None, 20, 4)            0
 )

 lstm_1 (LSTM)               (None, 20, 100)           42000

 time_distributed (TimeDistr  (None, 20, 10)           1010
 ibuted)

=================================================================
Total params: 43,250
Trainable params: 43,250
Non-trainable params: 0
```

Figure 2.6: Model for Feature Reduction

	date	tic	close	high	low	open	volume	cov_list	f01	f02	f03	f04
0	2015-01-16	ASIANPAINT.NS	804.700806	859.000000	840.000000	844.900024	1568660.0	[[0.000289698822461938, 7.614774834896145e-05,...	1.859801	0.397385	0.247793	0.0
1	2015-01-16	BAJAJ-AUTO.NS	2013.077515	2442.000000	2393.800049	2420.000000	270014.0	[[0.000289698822461938, 7.614774834896145e-05,...	1.859801	0.397385	0.247793	0.0
2	2015-01-16	BRITANNIA.NS	882.738281	979.924988	958.500000	972.500000	395802.0	[[0.000289698822461938, 7.614774834896145e-05,...	1.859801	0.397385	0.247793	0.0
3	2015-01-16	CIPLA.NS	629.144958	650.000000	639.000000	641.700012	1245788.0	[[0.000289698822461938, 7.614774834896145e-05,...	1.859801	0.397385	0.247793	0.0
4	2015-01-16	HDFCBANK.NS	480.375183	507.450012	499.274994	501.100006	3910654.0	[[0.000289698822461938, 7.614774834896145e-05,...	1.859801	0.397385	0.247793	0.0

Figure 2.7: Reduced Data Frame

2.4 Train and Test Data Split

We split both the close prices and the whole data set into train and test (trade) data.We used 80% of the data for training and then test on the remaining 20%.

```
Training Data:  from  2015-01-16  to  2020-08-17
Testing Data:  from  2020-08-18  to  2021-12-31
```

Figure 2.8: Training and Testing data Time frames

2.5 Deep Reinforcement Learning Models

Three Deep Reinforcement Learning models are implemented using the following threeRL algorithms:

2 Methodology

1. Deep Deterministic Policy Gradient (DDPG). This algorithm constantly learns the State-action function (Q function) and uses it to determine the best policy;

2. Advantage Actor-Critic (A2C) Algorithm. The algorithm consists of the policy which is the actor and acts on the environment. The critic calculates the value function based on a set reward function and this helps the actor to determine the optimal policy.

3. Proximal Policy Optimization (PPO). The PPO algorithm is an on-policy algorithm that can be used for discrete or continuous action space environments.

The environment setup within which we implement the algorithms is made up of the stock constituents, the prices, the technical indicators (features), and the covariance matrix (titled 'covs list') derived from the- one-year data at each time step. The Deep Re- inforcement Learning agent interacts with the environment in an exploration-exploitation manner.[7] This involves making decisions that will maximize rewards by considering whether to make previous good decisions (exploitation) or to try new decisions with the potential for greater rewards (exploitation). We make the assumptions that there are no short sales and all of our capital is used to invest in the portfolio stocks. The following algorithm gives a summary of the interaction within our environment.

2.5.1 The Algorithm

Input: *s*, **state space which includes the covariance matrix for stocks and technicalindicators**

Output: The final value of the portfolioStep By Step Process

1. Initialize P_0, $w_0 = (\frac{1}{m}, ..., \frac{1}{m})$ where P_0 is the initial portfolio value and w_0 is the initial portfolio weights and m is the number of assets in the portfolio.

2. Observe the state *s*d output portfolio weights vector w_t at time t;

3. Normalize the weights w_t to sum to 1;

4. Calculate stock returns vector

5. $r_t = \frac{v1,t-v1,t-1}{v1,t-1}, ... \frac{vm,t-vm,t-1}{vm,t-1}$, v is the closing price;

6. Portfolio returns for the period:$(w_t)^T r_t$;

7. Update portfolio value $P_t = P_{t-1}(1 + (w_t)^T r_t)$;

2.6 Back-testing

2.6 Back-testing

2.6.1 Portfolio Weight Allocations

The Deep Reinforcement Learning agents rebalance the weights of all assets at each time step using a policy that would maximize the cumulative returns of the entire portfolio.

2.6.2 Back-testing of the Portfolios

The figure below shows the cumulative returns of the Deep Reinforcement Learning models. The A2C model performs slightly better than the other two Deep Reinforcement Learning models.

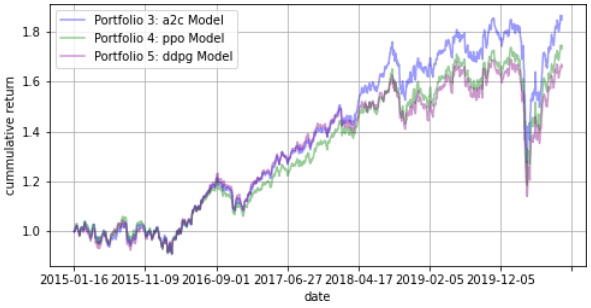

Figure 2.9: Back-testing on the Training Data set

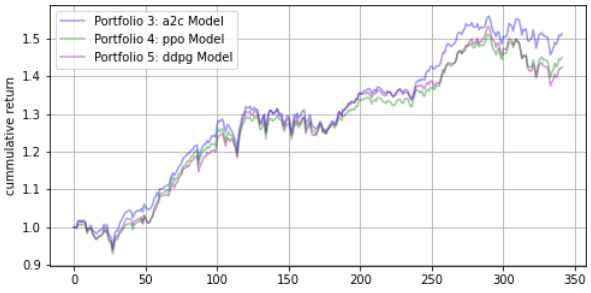

Figure 2.10: Back-testing on the Test Data set

2 Methodology

The following table presents the performance matrices of the portfolios with the A2C model presenting the highest Sharpe ratio followed by the PPO and DDPG models.

	a2c Model	ppo Model	ddpg Model
Annual return	0 356105	0.314278	0.297421
Cumulative returns	0.511962	0.449024	0 423859
Annual volatility	0.138870	0.140052	0.143447
Sharpe ratio	2.264231	2.022385	1 887772
Calmar ratio	5.050155	3.933461	2 911692
Stability	0.907157	0.881500	0 876337
Max drawdown	0.070514	0.079899	0.102147
Omega ratio	1.483862	1.412927	1 388536
Sortino ratio	3.319397	2.921451	2 721123
Skew	0.511494	0.585824	0 530169
Kurtosis	2.267039	2.174643	2 441126
Tail ratio	0 893246	0.911533	0 880278
Daily value at risk	-0 016248	0.016521	0 016998
Alpha	0 000000	0.000000	0 000000
Beta	1 000000	1.000000	1.000000

Figure 2.11: Final Result Statistics

Conclusion

Three Deep Reinforcement Learning models were used to construct portfolios and trade stocks using a fixed initial capital. The A2C model outperformed the other strategies while maximising the Sharpe ratio. But the same might not be true for a different dataset. Further scope lies in identifying the model which is best suited for a particular type of dataset and understanding the metrics to be used for labelling the datasets.

Data Availability Statement

The data used is publicly accessible through Yahoo Finance [8].

Bibliography

[1] V. Franç ois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018. DOI: 10.1561/2200000071. [Online]. Available: https://doi.org/10.1561%2F2200000071.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529– 533, 2015.

[4] S. Edelkamp and S. Schrödl, *Heuristic Search - Theory and Applications*. Academic Press, 2012, ISBN: 978-0-12-372512-7. [Online]. Available: http : / / www . elsevierdirect.com/product.jsp?isbn=9780123725127.

[5] J. M. Keynes, *The Collected Writings of John Maynard Keynes*, E. Johnson and D. Moggridge, Eds., ser. The Collected Writings of John Maynard Keynes. Royal Economic Society, 1978, vol. 6. DOI: 10.1017/UPO9781139520652.

[6] A. Hayes, *What is liquidity?* Mar. 2022. [Online]. Available: https://www.investopedia.com/terms/l/liquidity.asp.

[7] FinRL, *Three layer architecture*, Mar. 2021. [Online]. Available: https://bit.ly/3Ns1oI5.

[8] *Nifty 50 components*, Mar. 2022. [Online]. Available: https://finance.yahoo.com/quote/%5C%5ENSEI/components/.