Chaire LUSIS - CentraleSupélec

Deep Reinforcement Learning for Stock Portfolio Allocation

Explainable approach

Ismail Benaija, Daniel Lopez Herranz and Taha Boukhari
InfoNum

Supervision:

Fabrice Popineau	LISN/CentraleSupélec	Fabrice Daniel	LUSIS
Arpad Rimmel	LISN/CentraleSupélec	François de la Bourdonnaye	LUSIS
Marc Velay	LISN/CentraleSupélec		

Contents

List of Figures

List of Tables

Chapter

1 Introduction

Investment firms and hedge funds are challenged to define cost-effective automated trading strategies for their businesses to optimize capital allocation and maximize returns while limiting risk. Thus, the objective of our study is to propose a DRL program that automatically learns a stock trading strategy while maximizing the return on investment.
Among the methods used for portfolio allocation we can use the algorithms of ML [1], It offers great risk-adjusted returns and utility gains .Also Machine learning methods have been shown to be appropriate and advantageous for the difficult task of identifying patterns in markets (Gu et al., 2020). Gu et al. find an advantage to using machine learning for market timing with return predictions of 26% and 18% increase in Sharpe ratios with Neural Networks and Random Forest, respectively, over that of buy-and-hold.
Our paper will consist in implementing Reinforcement learning methods for portfolio allocation. Other works have been done in this field[2], our objective will be to make a new approach on the subject and to be able to introduce the explicability for more transparency. We test our algorithm on 30 Dow Jones stocks. Evaluate the performance of reinforcement learning agents and compare against the Dow Jones Industrial Average and traditionals ML methods. This strategy was found to outperform all of them.

Chapter

2 Related works

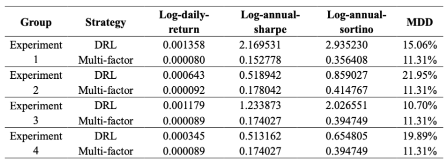
In order to understand the big picture of what it's being currently done with DRL in the field on portfolio allocation, we are gonna present three different papers that summarize some of the recent works on the field.

2.1 continuous reinforcement learning algorithms in portfolio management

In this paper [3] three continuous reinforcement learning algorithms are implemented. Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO) and Policy Gradient (PG) for portfolio management and present their performances under different settings, including different learning rates, objective functions and feature combinations.

2.2 DRL for portfolio management

In this paper [2] a deep reinforcement learning approach is applied to optimize investment decisions in portfolio management by making several innovations, such as adding a short mechanism and designing an arbitrage mechanism. They apply the model to make decision optimization for several randomly selected portfolios and to obtain excess return in stock market. The agent is optimized by maintaining the asset weights at fixed value throughout the trading periods and trades at a very low transaction cost rate. They use Deep Deterministic Policy Gradient (DDPG) for reinforcement learning. More details about DDPG algorithm can be found in the related literatures (Silver et al. 2014; Lillicrap et al. 2015).



Group	Strategy	Log-daily-return	Log-annual-sharpe	Log-annual-sortino	MDD
Experiment	DRL	0.001358	2.169531	2.935230	15.06%
1	Multi-factor	0.000080	0.152778	0.356408	11.31%
Experiment	DRL	0.000643	0.518942	0.859027	21.95%
2	Multi-factor	0.000092	0.178042	0.414767	11.31%
Experiment	DRL	0.001179	1.233873	2.026551	10.70%
3	Multi-factor	0.000089	0.174027	0.394749	11.31%
Experiment	DRL	0.000345	0.513162	0.654805	19.89%
4	Multi-factor	0.000089	0.174027	0.394749	11.31%

Figure 2.1: DRL and multifactor model strategy

we can see that the strategy of DRL outperforms the strategy of multi-multi-factor model .

2.3 Explainability

There are two types of explainability techniques in machine learning, post-hoc methods that build a layer of explainability above the model and transparent models that have built-in explainability techniques. Integrated Gradients is a post-hoc method that is used in classification problems in computer vision to determine a saliency map highlighting the pixels that contributed the most to the classification.

Chapter

3 Background

3.1 Reinforcement learning

Reinforcement learning is a machine learning method that differs from other machine learning methods in that the algorithm is not explicitly told how to perform a task, but instead deals with the problem independently.

Because the agent interacts with its environment, depending on its performance, it receives a rewarded state. The agent makes decisions over time to maximize its reward and minimize its penalty using which is what we call a policy. The benefit of this AI approach is that the AI program can learn without the programmer instructing the agent how it should perform the task.

MDP and overview of RL

In Reinforcement Learning (RL), the problem to solve is described as a Markov Decision Process (MDP). Theoretical results in RL rely on the MDP description being a correct match to the problem. If your problem is well described as a MDP, then RL may be a good framework to use to find a solution to the problem. Conversely, if you cannot convert your problem onto a Markov decision process, then the theory behind RL makes no guarantees of any useful result.

A2C

The Advantage Actor Critic (A2C) algorithm combines two types of Reinforcement Learning algorithms (Policy Based and Value Based) together. Policy Based agents directly learn a policy (a probability distribution of actions) mapping input states to output actions. Value Based algorithms learn to select actions based on the predictable values of the input state or action. Overall the actor critic algorithm consists of different networks the actor and the critic as they work together to found a solution and has good results while using large batch size. As the critic network learns which states are better or worse, the actor uses this information to teach the agent to seek out good states and avoid bad states. Now let's see the actor critic architecture : First let's define the variables : G_t : the advantage estimate. $\pi_\theta(a \mid s)$: the policy and $\nabla_\theta J(\theta)$: The policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta \left(a_t \mid s_t \right) G_t \right]$$

After decomposing the equation we got :

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_0, a_0, \ldots, s_t, a_t} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta \left(a_t \mid s_t \right) \right] \mathbb{E}_{r_{t+1}, s_{t+1}, \ldots, r_T, s_T} \left[G_t \right]$$

then :

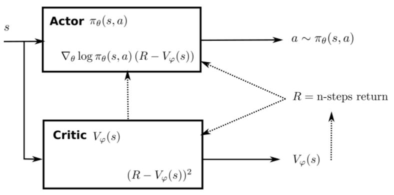
$$\nabla_\theta J(\theta) = \mathbb{E}_{s_0, a_0, \ldots, s_t, a_t} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta \left(a_t \mid s_t \right) \right] \mathbb{E}_{r_{t+1}, s_{t+1}, \ldots, r_T, s_T} \left[G_t \right]$$

combining it we got :

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{s_0, a_0, \ldots, s_t, a_t} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta \left(a_t \mid s_t \right) \right] Q_w \left(s_t, a_t \right) \\ &= \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta \left(a_t \mid s_t \right) Q_w \left(s_t, a_t \right) \right] \end{aligned}$$

all this lead us to The Critic estimates the value function. This could be the action-value (the Q value) or state-value (the V value) and the actor updates the policy and we found the advantage actor critic after using the relationship between the Q and the V from the Bellman optimality equation :

$$\begin{aligned} \nabla_\theta J(\theta) &\sim \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta \left(a_t \mid s_t \right) \left(r_{t+1} + \gamma V_v \left(s_{t+1} \right) - V_v \left(s_t \right) \right) \\ &= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta \left(a_t \mid s_t \right) A \left(s_t, a_t \right) \end{aligned}$$



s
Actor $\pi_\theta(s,a)$
$\nabla_\theta \log \pi_\theta(s,a)\,(R - V_\varphi(s))$
$a \sim \pi_\theta(s,a)$
R = n-steps return
Critic $V_\varphi(s)$
$(R - V_\varphi(s))^2$
$V_\varphi(s)$

Figure 3.1: A2C architecture

PPO

PPO is an actor critic method in which the policy is explicitly updated in order to be stable, rather than imposing hard constraints, it formalizes the constraints as penalties in the objective function. By avoiding constraints at all costs, we can use a first-order optimizer like gradient descent to optimize the objective. It defines the ratio between the new and old policy which we can express as :

$r(\theta)$: $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta old}(a|s)}$ by modifying it :

$$J(\theta)^{TRPO} = E\left[r(\theta)\hat{A}_{\theta old}(s,a)\right]$$

PPO imposes policy ratio, $r(\theta)$ to stay within a small interval around 1. That is the interval between $1-\epsilon$ and $1+\epsilon$. Now we can write the objective function of PPO:

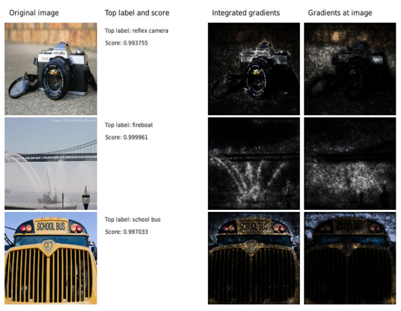
$$J^{\text{CLIP}}(\theta) = \mathbb{E}\left[\min\left(r(\theta)\hat{A}_{\theta_{\text{old}}}(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{\text{old}}}(s,a)\right)\right]$$

Also PPO used to be a great algorithm as he's simple to use and provide very good results overall.

3.2 Integrated gradient

Integrated gradients technique aims to explain the relationship between a model's predictions based on its features. It has many use cases, including understanding the importance of features, identifying data skewness, and debugging model performance. However, it does not provide global feature importances on an entire data set and do not explain the interactions and combinations of features. It has been widely used in Computer Vision for example for object recognition to study pixel importance in predictions made by the network using the Googlenet architecture and trained over ImageNet [4].

Figure 3.2: Importance Object Detection



Chapter

4 Contextualization

Many recent works have applied DRL to quantitative finance. Stock market trading is considered the most challenging task due to its noisy and volatile characteristics, and various DRL-based approaches have been applied. Volatility scaling has been incorporated into DRL algorithms for trading futures contracts, which consider market volatility in a reward function. News headline sentiment and knowledge graphs, as alternative data, can be combined with price-volume data as time series to form a DRL trading agent. High frequency trading using DRL is a hot topic.

Deep Hedging has designed hedging strategies with DRL algorithms that manage the risk of liquid derivatives. It has shown two advantages of DRL in mathematical finance, namely scalability and model-free. The DRL-driven strategy becomes more efficient as the portfolio scale increases. It uses FinRL to manage the risk of liquid derivatives, indicating an extension of our FinRL library to other asset classes and topics in mathematical finance.

With regard to explainability, explainable AI is a hot topic nowadays. If a company wants to adopt an AI solution and use it to make decisions that directly impact the customer, it also needs to have the ability to explain these decisions to customers and regulators. There are various approaches to explain a model, in particular they can be divided into model-agnostic and model-specific methods.

Chapter

5 Our model

5.1 Problem definition

The goal of a portfolio allocation stock problem is to assign a weight to each stock at each time step in order to maximize the cumulative return over a certain period. Let's formalize that: Consider that we have a portfolio of K assets on a timeframe of n timesteps. We note W the vector $w(t) \in \mathbb{R}^N$ as the vector of the weight of each asset at the timestep t, $R(t)$ as the portfolio value at the timestep t and $p_k(t)$ as the asset k price at the timestep t and λ the risk aversion factor and finally the price variation vector : $\mathbf{y}(t) \triangleq \left[\frac{p_1(t)}{p_1(t-1)}, \frac{p_2(t)}{p_2(t-1)}, \dots, \frac{p_K(t)}{p_K(t-1)}\right]^\top$

The problem thus reduces down to the following:

$$\underset{w}{\text{Maximize}}\, \boldsymbol{w}^\top(t)\mathbf{R}(t) - \lambda \boldsymbol{w}^\top(t)\Sigma(t)\boldsymbol{w}(t)$$

$$\text{subject to} \sum_{i=1...K} w_i = 1, 1 \geqslant w \geqslant 0$$

5.2 Reinforcement learning framework

Now that we have defined the problem variables, we can look at the use of reinforcement learning to train an agent to find the best portfolio allocation at each time t. To do this, we need to define the agent's states, the action space and the reward. It can be summarized in the following figure:

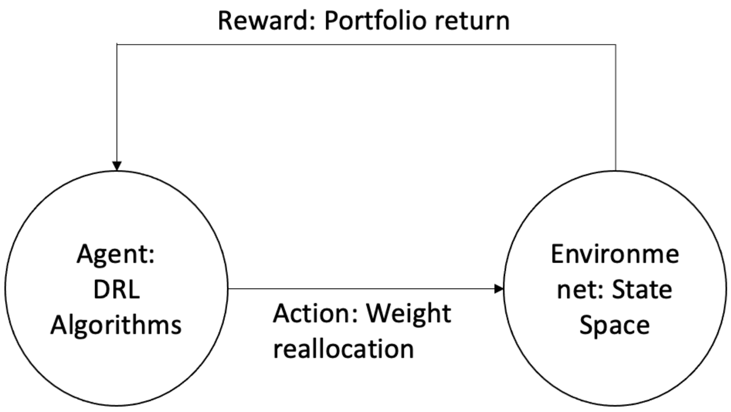


Figure 5.1: Framework

State space: It corresponds to the environmnent perception of the agent, in our problem it will be the asset prices in addition to some technical indicators that we will develop in the next subsection.

Action space: The agent can modify the vector $w(t)$ of the weights at each time step with the constraint that the sum of the weights is equal to 1.

Reward: Several rewards can be used, we can take the portfolio value $R(t)$ or we can take the portfolio value modification between two timesteps $\frac{R(t)-R(t-1)}{R(t-1)}$.

Deep Reinforcement Learning Algorithms: We used two well known DRL algorithms A2C and PPO

5.3 Technical Indicators

Drawing on techniques used in market finance trading, we can use technical indicators that capture trends and volatility to enhance the data environment. Thus we define:

Relative Strength Index (RSI): It is a momentum indicator that measures the speed and change in the movement of a price. It can be calculated with the formula

$$\text{RSI} = 100 - \left[\frac{100}{1 + \frac{\text{Average gain}}{\text{Average loss}}}\right]$$

Two areas are very important, when the RSI is below 30, the asset is said to be oversold and when the RSI is above 70, the asset is said to be overbought.

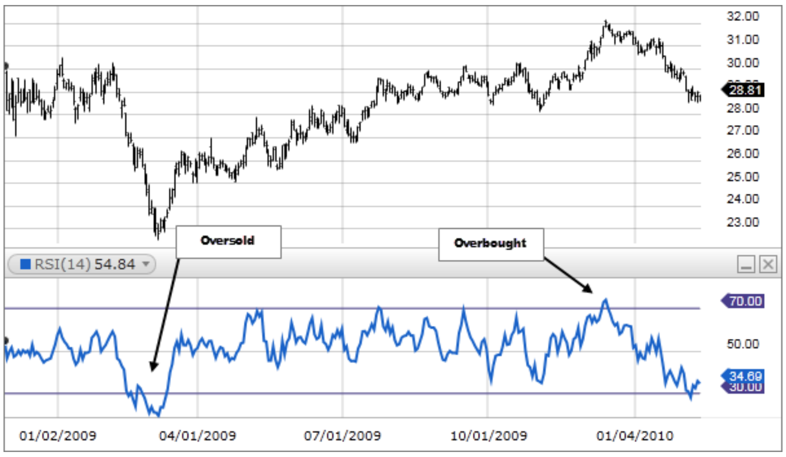


Figure 5.2: Illustration of the two RSI areas. Source: Fidelity broker website

Moving Average Convergence/Divergence (MACD): It is a trend indicator that is equal to the difference between the 12 day exponential moving average and the 26 day one.

$$MACD = EMA\ 12days - EMA\ 26days$$

When MACD crosses above zero, buying the asset is bullish and the opposite is considered bearish.

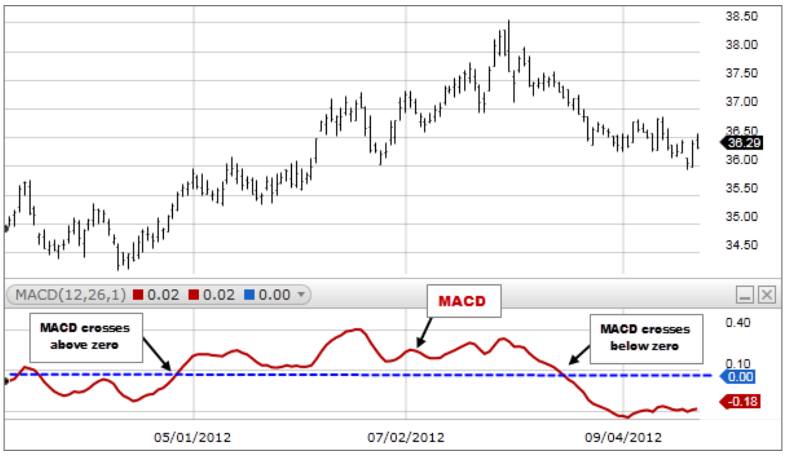


Figure 5.3: Illustration of how MACD works. Source: Fidelity broker website

Average True Range (ATR): It is a volatility indicator, it is the difference between today's high and today's low in an asset price.

5.4 Explainability methods: Integrated gradients

Once we have a trading agent, in order to explain the agent decisions, we will use a classic perturbation methodology based on Integrated Gradients inherited from the computer vision field. This is based on perturbing the state space by introducing a noise on the feature values to estimate the importance of each technical indicator. The formula for the integrated gradient is:

$$\mathrm{IntegratedGradients}_i(x) ::= \left(x_i - x_i'\right) \times \int_{\alpha=0}^{1} \frac{\partial F\left(x' + \alpha \times (x - x')\right)}{\partial x_i} d\alpha$$

where i=feature, x=input, x′=baseline and α an interpolation constant to perturb features by.

In practice, it is hard to compute the integral and we will use the following approximation:

$$\mathrm{IntegratedGradsapprox}(x) ::= \left(x_i - x_i'\right) \times \sum_{k=1}^{m} \frac{\partial F\left(x' + \frac{k}{m} \times (x - x')\right)}{\partial x_i} \times \frac{1}{m}$$

Several types of perturbations can be chosen depending on the value of α and we wil explore that on the next section. The output of this method will give us a saliency map with each technical indicator importance at each timestep for the trading agent decisions.

Chapter 6

Experiments

6.1 Data

We had no constraints on what type of data to use so we decided to decided to use the Dow Jones. The Dow Jones Industrial Average (DJIA), is a price-weighted measurement stock market index of 30 prominent companies listed on stock exchanges in the United States. It is made up, among others, of companies such as Apple, American Express and Disney.

We extracted our Data from Yahoo Finance through the yfinance python package. The extracted data goes from 2008-01-01 to 2021-09-02. Data from 2008 to 2020-07-01 is used for training. Data from 2020-07-01 to 2021-09-02 is used for backtesting.

Backtesting is a key component of effective trading system development. It is accomplished by reconstructing, with historical data, trades that would have occurred in the past using rules defined by a given strategy. The result offers statistics to gauge the effectiveness of the strategy.

No data cleaning is necessary since the data extracted from Yahoo Finance is already clean and ready to use.

6.2 Trading Agent

First, to begin our project, we start with the first part of it: the trading agent.

Environment

Following the logic of an RL project, we start with the environment. After some time looking for a good option, we settle for the framework Gym. Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

Gym supports and variety of ready-made environment but also easily lets us build and create a custom environment that can then be used by the trading agents. We opted for that option and only a small number of methods needed to be overwritten.

For our environment, like with any other RL problem the most important thing was to define our state, reward and action space.

State Since we are working on the Portfolio Management problem, the state needs to be the price action of the considered stocks. In order to have that, we build at each step the covariance matrix of the stock prices which gives us the information we need. That's what's used as input for the algorithm.

Reward For the rewards, we considered two options that were fairly equivalent:

- The first one was to consider the portfolio value variation so we had:

$$r = \frac{new\ portfolio\ vale}{former\ portfolio\ value}$$

- In the second one, we simply had the portfolio value:

$$r = new\ portfolio\ value$$

Actions Our actions are simply a vector corresponding to the portfolio distribution that the agent would allocate to each stock. The sum of that vector is equal to one. The action space is therefore a box of dimension 30 (because there are 30 stocks), ranging from 0 to 1. The vector is then softmax normalized.

Initially, to simplify our problem, we simply considered two stocks and we discretized the action space in order to only consider three options (buy, sell, hold). Once we had a trading agent that worked, we then moved to our actual problem.

Agent

In order to tackle the main problem, after a lot of research and comparison between different options, we decided to use FinRL. FinRL is an open-source project to explore the potential of deep reinforcement learning in finance. The FinRL ecosystem is a unified framework, including various markets, state-of-the-art algorithms, financial tasks (portfolio management, cryptocurrency trading, high-frequency trading), live trading, etc.

FinRL has three layers: applications, drl agents, and market environments. By default, FinRL use Gym for building the environment so it really suited our needs.

With FinRL and the stable baselines 3 library, it was then possible to implement several deep reinforcement learning algorithms. Stable Baselines is a set of improved implementations of Reinforcement Learning (RL) algorithms based on OpenAI Baselines. In our case we decided because our state and action space were continuous, to use and compare two different algorithms that could be used under that conditions:

- Proximal Policy Optimization (PPO)
- Synchronous Advantage Actor Critic (A2C)

Parameters

In order to build the environment and the agent, a lot of parameters needed to be defined. We are not going to review all of them in detail here but we can explain a few of them.

1. Initial amount: initial portfolio value set to 1 000 000
2. Transaction cost pct: The cost of a transaction in percentage. This parameter was set to 0 because it can vary a lot in real applications and what we wanted to evaluate was the real performance of the algorithm.
3. The maximum number of shares to trade. This parameter was set to 100
4. Learning Rate. The learning rate was set to 0.0004 for A2C and to 0.001 for PPO

6.3 Explainability

This was one of the major challenges of our work. At first, we decided that in order to explain our model actions, we needed to explain them in terms a human can understand. For that, we thought that it could be interesting to try to explain the actions in terms of technical indicator changes. For example, we could say that the agent decided to sell a particular stock essentially because of the changes on one or several indicators.

We decided to use the integrated gradients method as explained above. For that, we had to add the technical indicators that we wanted to use as explanatory variables to the input of the model. The new state of the agent was then built with the covariance matrix which summarizes the price changes and the technical indicators in order to be able to explain the results.

The technical indicators added needed to add as much variance as possible to be able to explain the actions. For that reason we added the indicators the we described above (macd, rsi, dx, atr). We added for even more information those indicators for two different timesteps (5 and 30 days).

Now, in order to perturbate the input, we needed to perturbate the technical indicators and see how the output reacted to those changes. The problem is that the scale of the indicators is different for each one of them. For that reason, we tested two methods for perturbing the input:

1. Total suppression. This method was used in the paper [5]. In this paper, the authors try to explain an DRL model by perturbing the input and the way they do that is by removing a piece from the board (It's an agent that plays chess). It's interesting because for suppressing the signal, the scale doesn't need to be taken into account.

2. Random perturbation. With this method, we simply perturb the model following a random distribution.

6.4 Sectorization

We also grouped the different stocks by sector in order to analyse and check if we could find any interesting pattern. For that, we labelled each stock to a particular sector (Financial Service, Consumer, Healthcare, Technology...) and we computed for each sector the distribution in percentage of that sector in the portfolio.

After that, we also check for each sector which were the technical indicators that were the most influential. For that we simply computed the average explanation for each indicator for all the stocks belonging to the same sector.

6.5 Computing power

In order to be able to train and run our models, we quickly found the need to move from our personal computers to actual computing centers. In order to do that, we were given access to the Paris-Saclay mesocentre. The mesocentre brings together 3 major technical platforms for scientific computing/data processing from the partner institutions in the Paris-Saclay project.

We were able to connect to the mesocentre VM and move our code there which allowed us to make use of the GPU nodes. The one we used is equipped with an Nvidia Tesla V100. With this GPU, training was about six times faster which made it possible to explore and to fine tune several environment and agent parameters. Inference was also much faster which was necessary in order to backtest our models and plot the results.

Chapter

7 Results

7.1 Trading Agent

After training our agent, we test how it performs on unseen data:

Table 7.1: Backtesting results (from 2020-07-01 to 2021-09-01)

Algorithm	Initial portfolio	Final portfolio	Sharpe
A2C	1 000 000	1 448 550	2.19
PPO	1 000 000	1 446 500	2.21

We plot how our models compare with the DJIA which is our baseline aswell as other ML algorithms (Support Vector Machine, Decision Tree, Random Forest and Linear Regression).

Figure 7.1: Algorithm Comparison

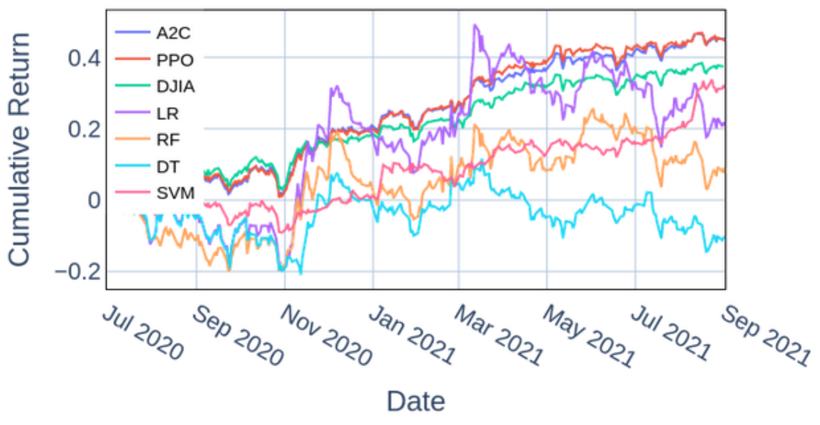
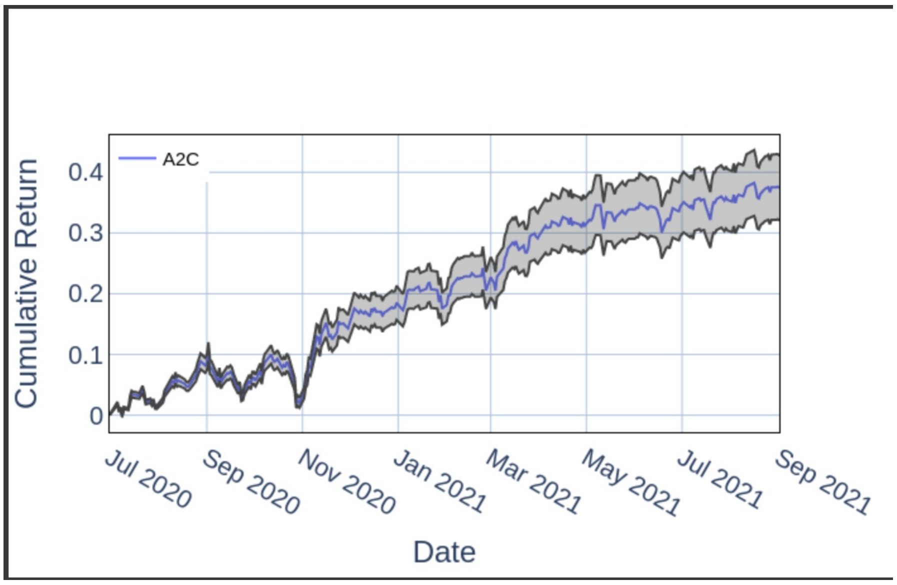
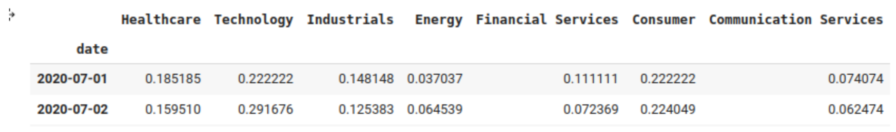


Figure 7.2: 95% confidence interval for the cumulative return of the agent using A2C



7.2 Actions

Figure 7.3: Actions by sector per day



date	Healthcare	Technology	Industrials	Energy	Financial Services	Consumer	Communication Services
2020-07-01	0.185185	0.222222	0.148148	0.037037	0.111111	0.222222	0.074074
2020-07-02	0.159510	0.291676	0.125383	0.064539	0.072369	0.224049	0.062474

7.3 Explainability

For each day and for each technical indicator we have computed a value that shows how important that indicator was for the final output of that day. We have summarized for both algorithms and for the two different types of perturbation the results we got:

Table 7.2: PPO, random perturbation

	macd	rsi 30	cci 30	dx 30
min	-0.00013	-0.0028	-0.0024	-0.00085
max	0.00038	92.2	0.0015	0.0072
mean	0.000001	5.19	-0.000027	0.00033
std	0.00001	13.2	0.00024	0.0093

Table 7.3: PPO, null perturbation

	macd	rsi 30	cci 30	dx 30
min	-0.003	-5922	-0.31	-0.78
max	0.0024	-0.018	0.56	0.015
mean	-0.000009	-216	-0.0018	-0.036
std	0.0003	650	0.064	0.1

A2C

Table 7.4: A2C, random perturbation

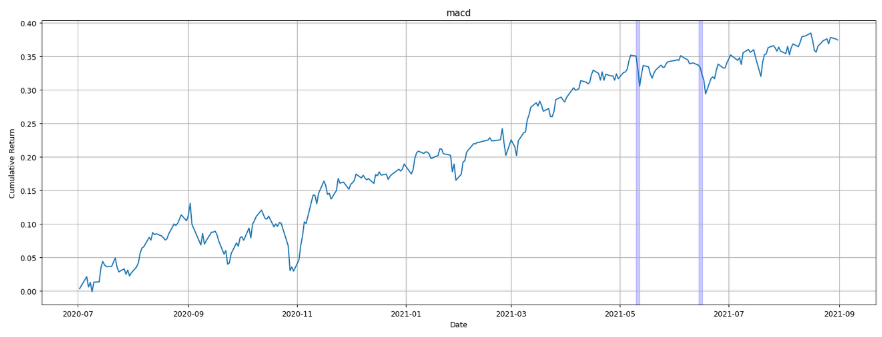
	macd	rsi 30	cci 30	dx 30
min	-1.2e-04	-0.0071	-0.001	-0.0016
max	1.9e-04	103	0	0.011
mean	7.2e-07	4.7	-0.000003	0.0002
std	1.5e-05	10.4	0.000058	0.0011

We plot here the DJIA (independent from our agent) and we then highlight for each indicator, the moments where the explainability value from that indicator is high (top 10%)

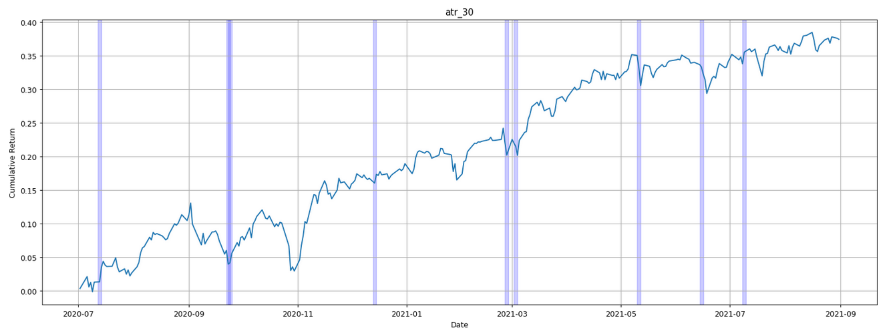
Table 7.5: A2C, null perturbation

	macd	rsi 30	cci 30	dx 30
min	-0.019	-54629	-1.6	-6.5
max	0.051	-0.27	2.2	0.095
mean	-0.00017	-1953	-0.023	-0.21
std	0.0037	5025	0.24	0.56

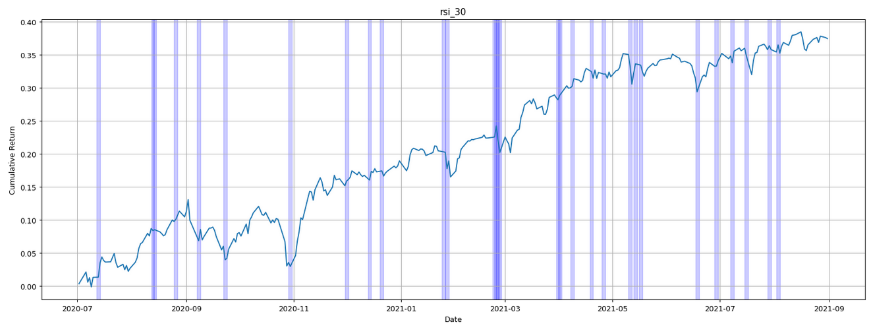
Figure 7.4: macd



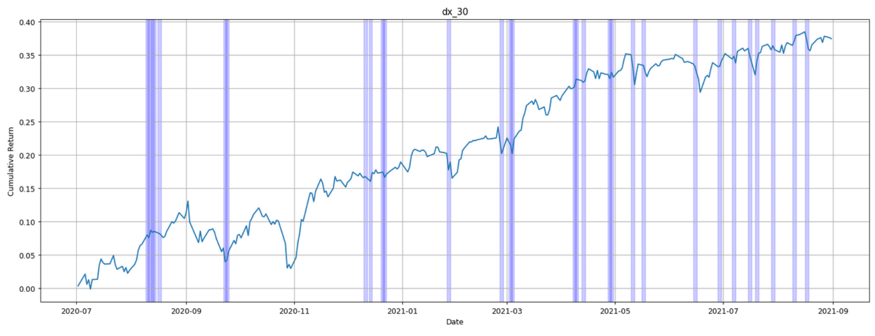
30 days technical indicators



(a) atr

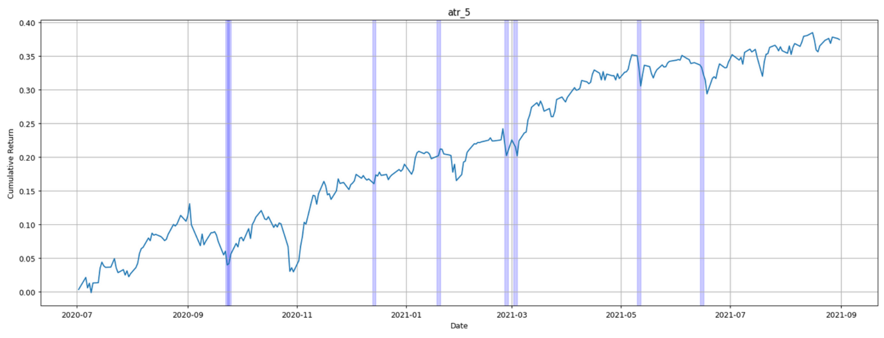


(b) rsi

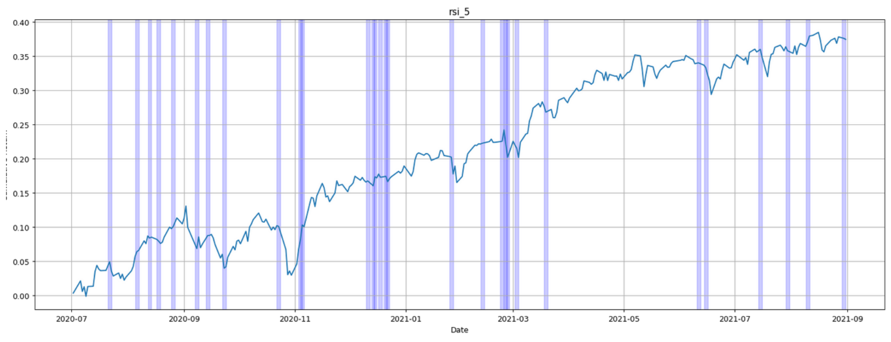


(c) dx

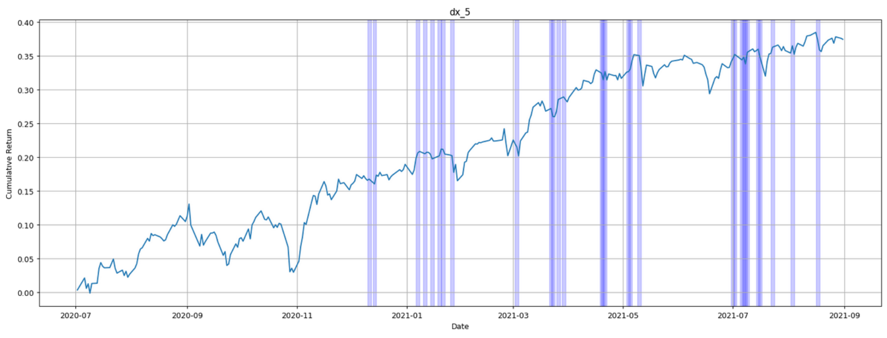
5 days technical indicators



(a) atr



(b) rsi



(c) dx

Figure 7.6: 5 day technical indicators

Chapter

8 Discussion

8.1 Trading

In terms of performance, the metric that has been chosen is the cumulative return over a certain period of trading.

The performance of the trading agents using A2C or PPO are comparable as we can see in Table 7.1, on the backtesting period they both give us a cumulative return of 44%. Compared to other classic Machine Learning techniques or to the DJIA, they seem to be more performant according to the figure 7.1 giving a higher cumulative return.

This can be mitigated by the confidence interval graph 7.2 which shows that the buy and hold strategy with equal weight (DJIA) is sometimes more profitable.

8.2 Explainability

Our work here consists on finding for each day, the importance of each technical indicator. For that as we already explained, we used two types of perturbation (random and null). We summarized our results on tables 7.5 and 7.4 for A2C and 7.3 and 7.2 for PPO.

We can see that the indicator that seems to have the most importance is rsi. The rsi indicator can be seen as a way to see if the stock is oversold. So it gives us an interesting point of view on our particular problem.

Explainability detection zones

As we can see in figures 7.4, 7.6 and 7.5 our explainability module seems to find and highlight important variations of the DJIA. In these figures, we have plotted the DJIA and we highlight the days where the explainibility absolute value of the considered indicator is really high. We do that in order to find interesting patterns and to extract any information that we can.

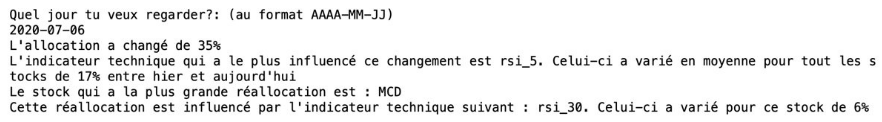
In particular, we can see the difference between the 5 day indicators and the 30 day indicators. For the 5 day indicator, we can in general see that they will detect short time and fast variations whereas the 30 day indicators will in general tend to highlight slow variations.

There is a big difference in particular between dx 5 and dx 30, where dx 5 can't detect much at the first half of the backtest period but dx 30 has no problem.

We can also point out that there are a lot of zones that are detected by both 5 day and 30 day indicators aswell as zones that are detected by several indicators.

The goal of this explainability module is to give the ability to the one using the trading agent to understand its decisions by deep diving into the key technical indicators that led to a reallocation of the portfolio. This is why the final output will be sentences in natural language with the technical indicator that has the highest explainability value alongside with the stock that has been reallocated the most. An example would be:

Figure 8.1: Example of an output of the explainability module



```
Quel jour tu veux regarder?: (au format AAAA-MM-JJ)
2020-07-06
L'allocation a changé de 35%
L'indicateur technique qui a le plus influencé ce changement est rsi_5. Celui-ci a varié en moyenne pour tout les s
tocks de 17% entre hier et aujourd'hui
Le stock qui a la plus grande réallocation est : MCD
Cette réallocation est influencé par l'indicateur technique suivant : rsi_30. Celui-ci a varié pour ce stock de 6%
```

8.3 Limits

It's important to note that results might not exactly be what our agent is really doing. The reason for that is essentially because, we are trying to explain what our agent is doing using only a few technical indicators. However, the input state of the agent is different.

The input state, is a combination of technical indicators and price variations. It could be the case that technical indicators play no role on the agent decision and that our agent only use the price action of the stocks. We could not however add this information in order to explain our outputs in function of that because we consider that only the technical indicators are useful for explainability.

In addition, we have only considered technical indicators individually but it would probably be more interesting to look for a combination of indicators and see how they react together.

Other indicators could also have been added, specially lagged indicators.

Chapter

9 Conclusion

In our study, we explored a DRL method to trade on a financial market for stock trading. Through this modeling, we were able to train a reinforcement learning agent based on PPO and A2C using historical data from the DOWJONES30. The results obtained on the model prove that it is able to give better results than classical methods as buy and hold and other ML algorithms.

Also the use of integrated gradient has been very useful in order to compute the importance of each technical indicator in relation to our portfolio for each day by using two different types of perturbations and gave us a main idea of how our agent choose to position itself on the financial market to make money.

For future work, it will be interesting to explore more complex models challenges and deal with different stocks, solve empirical approaches, and analyze through natural language the financial market news. This new information could then be fed to the model which would have more information on how the real world economic and politic situation is evolving and make predictions on that.

Bibliography

[1] Michael Pinelis* and David Ruppert. "Machine Learning Portfolio Allocation." In: (). URL: https://arxiv.org/pdf/2003.00656.pdf.

[2] Xiaohua Zhou Gang Huang and Qingyang Song. "Deep Reinforcement Learning for Portfolio Management." In: (2021). URL: https://arxiv.org/pdf/2012.13773.pdf.

[3] Zhipeng Liang; Hao Chen; Junhao Zhu. "Adversarial Deep Reinforcement Learning in Portfolio Management." In: (2018). URL: https://arxiv.org/abs/1808.09940.

[4] Qiqi Yan Mukund Sundararajan Ankur Taly. "Axiomatic Attribution for Deep Networks." In: (2017). URL: https://arxiv.org/abs/1703.01365.

[5] Piyush Gupta et al. Nikaash Puri. "Explain Your Move: Understanding Agent Actions Using Specific and Relevant Feature Attribution." In: (2019). URL: https://arxiv.org/abs/1912.12191.

[6] Prafulla Dhariwal John Schulman Filip Wolski. "Proximal Policy Optimization Algorithms." In: (2017). URL: https://arxiv.org/abs/1707.06347v2.

[7] Adrià Puigdomènech Badia Volodymyr Mnih. "Asynchronous Methods for Deep Reinforcement Learning." In: (2016). URL: https://arxiv.org/abs/1602.01783.

[8] Natalia Díaz-Rodríguez Alexandre Heuillet Fabien Couthouis. "Explainability in Deep Reinforcement Learning." In: (2020). URL: https://arxiv.org/abs/1703.01365.

[9] Jianjun Li; Guohui Li; Peng Pan Si Shi. "XPM: An Explainable Deep Reinforcement Learning Framework for Portfolio Management." In: (2021). URL: https://dl.acm.org/doi/abs/10.1145/3459637.3482494.

[10] Xinyi Li Yuancheng Zhan; Xiao-Yang; Liu. "Optimistic Bull or Pessimistic Bear: Adaptive Deep Reinforcement Learning for Stock Portfolio Allocation." In: (2021). URL: https://arxiv.org/abs/1907.01503.

[11] OLEG SZEHR. "HEDGING OF FINANCIAL DERIVATIVE CONTRACTS VIA MONTE CARLO TREE SEARCH." In: (2021). URL: https://arxiv.org/abs/2102.06274.

[12] Jianfeng Gao Xiujun Li Lihong Li. "Recurrent Reinforcement Learning: A Hybrid Approach." In: (2015). URL: https://arxiv.org/abs/1509.03044.

[13] Jamal Atif Eric Benhamou David Saltiel. "Deep Reinforcement Learning (DRL) for Portfolio Allocation." In: (2020). URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3871071.

[14] Jinjun Liang Zhengyao Jiang Dixing Xu. "A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem." In: (2017). URL: https://arxiv.org/abs/1706.10059.