PLOS ONE

RESEARCH ARTICLE

Explainable post hoc portfolio management financial policy of a Deep Reinforcement Learning agent

Alejandra de-la-Rica-Escudero[1,2☯], **Eduardo C. Garrido-Merchán**[2☯], **María Coronado-Vaca**[2☯*]

1 School of Engineering (ICAI), Universidad Pontificia Comillas, Madrid, Spain, **2** Faculty of Economics and Business (ICADE), Universidad Pontificia Comillas, Madrid, Spain

☯ These authors contributed equally to this work.
* mcoronado@comillas.edu

Citation: de-la-Rica-Escudero A, Garrido-Merchán EC, Coronado-Vaca M (2025) Explainable post hoc portfolio management financial policy of a Deep Reinforcement Learning agent. PLoS ONE 20(1): e0315528. https://doi.org/10.1371/journal.pone.0315528

Editor: Dr. Naeem Saleem, University of Management and Technology, PAKISTAN

Received: August 11, 2024

Accepted: November 26, 2024

Published: January 16, 2025

Data Availability Statement: -All relevant data are within the manuscript and its Supporting Information files. -All Data and Code of the paper are freely and fully available at GitHub https://github.com/aleedelarica/XDRL-for-finance.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Abstract

Financial portfolio management investment policies computed quantitatively by modern portfolio theory techniques like the Markowitz model rely on a set of assumptions that are not supported by data in high volatility markets such as the technological sector or cryptocurrencies. Hence, quantitative researchers are looking for alternative models to tackle this problem. Concretely, portfolio management (PM) is a problem that has been successfully addressed recently by Deep Reinforcement Learning (DRL) approaches. In particular, DRL algorithms train an agent by estimating the distribution of the expected reward of every action performed by an agent given any financial state in a simulator, also called gymnasium. However, these methods rely on Deep Neural Networks model to represent such a distribution, that although they are universal approximator models, capable of representing this distribution over time, they cannot explain its behaviour, given by a set of parameters that are not interpretable. Critically, financial investors policies require predictions to be interpretable, to assess whether they follow a reasonable behaviour, so DRL agents are not suited to follow a particular policy or explain their actions. In this work, driven by the motivation of making DRL explainable, we developed a novel Explainable DRL (XDRL) approach for PM, integrating the Proximal Policy Optimization (PPO) DRL algorithm with the model agnostic explainable machine learning techniques of feature importance, SHAP and LIME to enhance transparency in prediction time. By executing our methodology, we can interpret in prediction time the actions of the agent to assess whether they follow the requisites of an investment policy or to assess the risk of following the agent's suggestions. We empirically illustrate it by successfully identifying key features influencing investment decisions, which demonstrate the ability to explain the agent actions in prediction time. We propose the first explainable post hoc PM financial policy of a DRL agent.

Introduction

The application of Deep Reinforcement Learning (DRL) [1] to portfolio management (PM) has gained popularity in recent years [2] as an alternative method to the less realistic modern portfolio theory techniques such as the Markowitz model [3, 4], whose assumptions often fail to hold true in high-volatility markets such as the technological sector or cryptocurrencies [5]. Concretely, DRL algorithms train agents to maximize expected returns by learning optimal actions through interaction with a simulated environment [6], also known as gymnasium [7]. These agents use Deep Neural Networks models (DNNs) [8] to approximate the distribution of expected rewards for different actions in varying financial states. Despite their capability as universal function approximators [9], DNNs lack interpretability [10]. This means a challenge in financial contexts, where decision-making transparency and interpretability is crucial for investors to trust and adopt automated strategies [11].

The need for explainability in financial decision-making drives the development of Explainable Artificial Intelligence (XAI) techniques [12–14], whose incorporation in the context of DRL enhance the high performance of DRL agents. But, as we will show in section 2 (state of the art), despite the growing literature that analyzes the application of DRL to PM, the literature on the explainability of DRL algorithms applied to PM is very scarce and underdeveloped, with only four recent studies [15–18], to the best of our knowledge. Moreover, these four published DRL explainability methods in PM, only offer explanations of the model in training time, not being able to monitor the predictions done by the agent in the trading time.

Driven by this motivation, and to respond to this research gap in the literature, in our work, we introduce a novel Explainable Deep Reinforcement Learning (XDRL) framework for PM. Concretely, our approach combines the popular Proximal Policy Optimization (PPO) algorithm [19], a state-of-the-art DRL technique, with model-agnostic XAI methods such as feature importance [20], SHAP (SHapley Additive exPlanations) [21], and LIME (Local Interpretable Model-agnostic Explanations) [22]. The three explainability techniques can be implemented independently or jointly, being able to explain the DRL agent predictions in trading time, being able to track throughout the time whether the policy is acting as it is expected or not, what is an advantage with respect to the rest of four published DRL explainability methods just mentioned. Our working hypothesis is that the predictions of DRL agents can be explained in a post-hoc fashion, offering an interpretation that can be tested with respect to financial investment policies.

This work contributes to the underdeveloped literature on the applications of XDRL models in the realm of PM in two ways: We add to this emerging literature of only four previous studies but with promising results. Our research novelty entails, to the best of our knowledge, the first-ever application of an explainable post-hoc portfolio management policy of a DRL agent. Our paper has also important implications in the real financial sector, where there is a need to make models explainable so that investors and other stakeholders can see why certain investment decisions are made. Thus, our proposed methodology also helps practitioners (whether they are individual investors, financial experts, institutional investors, policymakers and other stakeholders).

The rest of the paper is organised as follows. First, we begin with a state-of-the-art section. Afterwards, in a methodology section we explain the fundamental details of DRL applied to financial PM, the PPO algorithm and the explainability techniques that are going to be used to explain the agents' predictions. Then, in an experiments section we provide empirical evidence about the usefulness of our approach that supports our claim that DRL predictions can be explained and hence compared with a particular financial policy. Finally, we illustrate conclusions and further research lines.

State of the art

The application of DRL in financial PM is gaining popularity in the recent years, mainly due to the rise of computing power and architectures that enables a reasonable estimation of the rewards distribution of the actions with respect to the states given by the training process of the agents with respect to financial data [23]. But the literature on the explainability of DRL algorithms applied to PM is very scarce and underdeveloped, with only four recent studies [15–18], to the best of our knowledge. In this section, we show a detailed state-of-the-art of DRL and XDRL applied to financial PM to show the research gap in the literature to which our work responds.

Multiple DRL algorithms have been proposed recently, which has motivated their application in our area of interest. Liu et al. [24] classify the state-of-art DRL algorithms into three categories: 1) value-based algorithms: those based on Deep Q-Networks (DQN) [25]; 2) policy-based algorithms: directly update the parameters of a policy through policy gradient (PG) [26]; and 3) Actor-Critic based algorithms, such as Advantage Actor Critic (A2C) [25], Proximal Policy Optimization (PPO) [19], Deep Deterministic Policy Gradient (DDPG) [27], Soft Actor-Critic (SAC) [28], or Twin Delayed Deep Deterministic Policy Gradient (TD3) [29]. And all of them have been used to maximize portfolio returns while minimizing risk; for example, [30] apply DDPG for stock trading; [31] apply DQN, DDPG, PPO, A2C, TD3, SAC for automated stock trading in quantitative finance; [24] implement PPO, A2C, DDPG and TD3 for PM. Liang et al. [32] have applied successfully to financial PM three state-of-the-art popular DRL algorithms that can deal with continuous valued actions, namely PPO, DDPG and PG. Specifically, the authors conducted comprehensive experiments on the China stock market, examining various settings, including learning rates, objective functions, and feature combinations, to derive insights for parameter tuning, feature selection, and data preparation. Additionally, driven by the usefulness of DRL in high volatility markets, some authors have applied this methodology in cryptocurrency PM [24, 32–35]. Zhang et al. [36] apply DRL to trade futures contracts and various authors implement hedging strategies with DRL (deep hedging) [37–39]. More concretely, Pham et al. [40] focus on multi-agent DRL to automatically construct hedging strategies. And other authors also apply multi-agent DRL in portfolio management [41–43]. [44] propose a cost-sensitive portfolio selection with DRL, being, thus, the first authors to incorporate transaction costs. While all these DRL approches for PM only consider price changes of assets, but without considering the relation between companies, [45] propose a new DRL framework for PM based on GCN (Graph Convolutional Network) to take into account the relational features of the portfolio (relations between assets and their corresponding companies). [46] incorporate ensemble techniques and fuzzy extension in addition to existing DRL algorithms and use them for PM. [47] propose a novel DRL approach for portfolio optimization that combines the MPT and a DL approach (specifically, they solve the multimodal problem on a dataset of 28 USA stocks through the Tucker decomposition of a model with the input of technical analysis and stock return covariates). Some authors incorporate sentiment analysis in DRL to also perceive market sentiment in portfolio allocation [48–51]. Hambly et al. [23] provide a review of the recent developments and use of RL and DRL in finance, including PM.

But despite the growing literature that analyzes the application of DRL to PM (as we have just reviewed above), the literature on the explainability of DRL algorithms applied to PM is very scarce and underdeveloped, with only four recent studies [15–18], to the best of our knowledge. Guan and Liu (2021) [17] provide an empirical approach of explainable DRL for the PM task in response to the challenge of understanding a DRL-based trading strategy because of the black-box nature of deep neural networks. Specifically, they use a linear layer in hindsight as the reference model and they find the relationship between the reward (the

portfolio return) and the input (the features) by using integrated gradients. Since the linear model (a regression) is interpretable given that the coefficients are interpretable, then the methodology is interpretable. The neural network's capacity as a universal approximator dramatically exceeds that of regression. However, they use a neural network and then make a kind of compensation between the network and the coefficients to see how the linear regression "approximates" the network. If it approximates well, then you can trust of the coefficients. But you lose explainability if the approximation is poor. Additionally, as they are using linear interpretation, correlations may not explain complex patterns. Moreover, their approach differs from ours since it is an explainability approach dependent on the model, while ours is agnostic of the model, it is a post-hoc one. Bougie and Ichise (2020) [16] present a method to combine external knowledge and interpretable reinforcement learning in PM. They derive a rule-based variant version of the Sarsa algorithm ([6], p.140), that is, a neurosymbolic. This way, you can thus add a priori rules and data augmentation to "explain" your policy, since you are "injecting" it a priori. It is not a post-hoc approach like ours but rather an a priori one. While we explain the agent's predictions, they inject the agent with a policy in the form of rules before training. In fact, as we state in section 5 (Conclusions and further research) a line of future work could be to hybridize their approach with ours and see how training modifies the rules injected a priori. Shi et al. (2021) [18] add explainability to their DRL methods in PM through this approach: they use a temporal neural network to extract significant features that explain the patterns as a function of time, that is, a model that manages time, compared to our CNN. Then they apply a regularization technique to simplify them and finally, to explain them, they use class activation mapping (CAM), a way to explain the features of the neural network that they have used in the model, not in prediction time, like our proposed approach. Thus, they explain the model, that is, the policy trained during the training period, but we explain the predictions of the model during trading time. Wang et al. (2019) [15] offer an interpretable DRL investment strategy using interpretable deep attention networks. A ranking of features is obtained through two neural networks that seem to explain the training time data in the best possible way and subsequently a sensitivity analysis is performed to determine the best ones. Their interpretation analysis results reveal that their strategy selects assets by following a principle as "selecting the stocks as winners with high long-term growth, low volatility, high intrinsic value, and being undervalued recently". Once again, this approach differs from ours since it is useful for explanatory purposes, while we make explanations of predictions, for predictive purposes.

Thus, to the best of our knowledge, our study is the first to propose an explainable post hoc PM financial policy of a DRL agent.

Methodology

We now introduce the methodological details of our proposed approach to post-hoc explainable deep reinforcement learning applied to financial portfolio management. First, we will explain the fundamentals of deep reinforcement learning applied to finance, then, we will illustrate the explainable artificial techniques that we have chosen and, finally, we will show how we can integrate those techniques into the deep reinforcement learning method to explain the predictions of the agent.

Fundamentals of Deep Reinforcement Learning applied to financial portfolio management

We will first introduce objections to our methodology for portfolio management and arguments that answer to those objections. Then, we will describe the fundamentals of deep reinforcement learning and how we can apply these algorithms to financial portfolio management.

Although DRL has potential for portfolio management due to its competence in capturing nonlinear features, low prior assumptions, and high similarities with human investing, there are characteristics worth paying attention to as pointed out by Liang et al. [32]: First, a financial market is both highly volatile and non-stationary, totally different to games or robot control [52, 53] which are the main sectors where DRL has been experimented. Second, traditional Reinforcement Learning (RL) aims to maximize rewards over an infinite period, while portfolio management focuses on maximizing returns within a finite time. Third, in finance, it's crucial to test strategies on separate data sets to evaluate their performance, unlike in games or robotics. Lastly, the stock market has an explicit expression for portfolio value; therefore, approximating the value function is useless and can even deteriorate the agent´s performance.

However, deep neural networks are able to approximate any function given enough data and a particular architecture of the network, being universal approximator functions. Regarding maximizing returns within a finite time, we can tune the DRL algorithm via the γ hyperparameter to consider high future rewards. Concretely, $\gamma \in [0, 1]$ controls the focus of the agent in immediate or far rewards as a function of time where a value near to one focus on maximizing returns on a long time period, being γ the same as a discount rate for all DRL algorithms. Next, we can assume that an immediate future behaviour of the stock market is explained technically and by past information, being also DRL suited in this scenario. The agent can be retrained in a constant fashion after its predictions happen in the real-time scenario with the new information. Lastly, although Markowitz model assumes that portfolios are only a function of expected reward and risk, if those assumptions, like normal distributed returns, are not met, then, the function explaining the optimal portfolio is a black-box of an enormous set of features like technical indicators, fundamental ratios or social networks, that DRL algorithms can handle due to neural scaling laws. In this work, we assume that the market can be perfectly explained by technical data, focusing hence only in this kind of data. However, any source of data can also be integrated into the space state of the agent, even multimodal data, so this assumption is not an issue in real-case scenarios. To sum up, we consider that DRL can be successfully applied to the financial portfolio management problem, and now explain the fundamental concepts of this methodology.

DRL is a class of methods that combines reinforcement learning algorithms [6, 23] with deep neural network models [8] to tackle any complex decision-making task. In DRL, an agent interacts with an environment defined by a state space $\mathcal{S}$ and an action space $\mathcal{A}$. Critically, these spaces can be a bounded continuous domain, $\mathcal{S} \in \mathbb{R}^d$ and $\mathcal{A} \in \mathbb{R}^d$ where d is the number of features that the agent perceives in each time step t, and not limited to discrete spaces, as in reinforcement learning. In particular, deep reinforcement learning will encode the policy $\pi(a_t|s_t)$ learnt by trail and error with the environment in the training process in the deep neural network that will map a distribution of states to actions $\mathcal{S} \rightarrow \mathcal{A}$, with the purpose of selecting the action that maximizes an expected reward in a given time period by a $\gamma \in [0, 1]$ hyperparameter. Concretely, at each discrete time step t, the agent observes a state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$ based on the learnt policy $\pi(a_t|s_t)$, that acts as the conditional probability distribution of actions given states being estimated to maximize the expected reward. The environment responds to the action by transitioning to a new state s_{t+1} and providing a reward r_t as an effect of making action a_t given state s_t. Following an iterative process in a simulator, the agent can learn a policy $\pi(a_t|s_t)$ by trail and error that may generalize outside of the simulator if the assumptions made by the deep neural network model are met by the prediction data.

More formally, the purpose of the learning process is to learn a policy π that maximizes the expected cumulative reward, which can be defined with a return function in every time step R_t:

$$R_t = \sum_{k=0}^{K} \gamma^k r_{t+k}\,, \tag{1}$$

where $\gamma \in [0, 1]$ is the previously mentioned discount factor that balances immediate and future rewards and K is the end of the episode, or desired prediction period. Consequently, once that we have estimated the policy that maximizes the expected return function R_t, that can be personalized for any problem, we can define, for every time step and for every action and state, a q-value function $Q(s, a)$ that represents the complete distribution of the expected return of taking any action $a \in \mathcal{A}$ in any state $s \in \mathcal{S}$ and following policy $\pi(a_t|s_t)$ as:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[R_t|s_t = s, a_t = a]\,, \tag{2}$$

that is the probability distribution learnt by a DRL algorithm and approximated in the used deep neural network $Q(s, a|\theta)$ by its set of parameters θ. We illustrate this framework in Fig 1.

One of these algorithms, used in our work as its popularity and being a lightweight version of a more expensive algorithm, is Proximal Policy Optimization (PPO) [19], that aims to update policies without making big changes at once to avoid outliers in the training process that can make the estimation of the policy worse. This makes the learning process more stable. This is achieved by creating a "trust region," ensuring that new policies don't deviate too much from old ones. In order to do so, PPO introduces a clipped objective function (clipped probability ratios) that prevents the updating step from moving the new policy too far from the old policy. This clipping mechanism modifies the policy optimization by clipping the probability ratio between the new and old policies, keeping it within a specified range. By maximizing the clipped objective, PPO finds a balance between trying new strategies (exploration) and sticking with known good ones (exploitation).

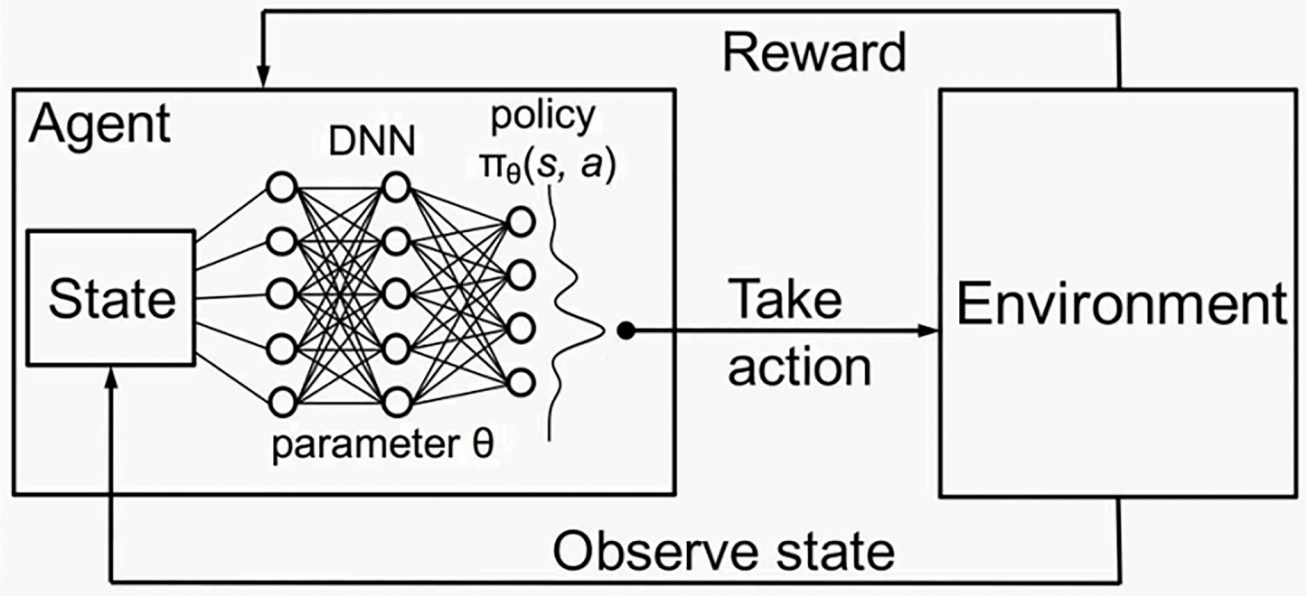

Fig 1. Deep reinforcement learning main components that enable the estimation of the expected reward of any action of the action space of the agent conditioned to any state perceived by the agent. The learnt policy function is encoded by a deep neural network, enabling continuous-valued actions and spaces and any complexity of its mapping.

https://doi.org/10.1371/journal.pone.0315528.g001

Explainable artificial intelligence techniques

As the purpose of our work is to enhance the DRL framework by integrating explainability of financial features to make the decision-making process of the DRL model transparent and understandable, we briefly describe in this section some of the techniques that we have integrated in the DRL framework to make it interpretable for financial experts.

Human decision makers use predictions made by ML models in their process, but the usability of those predictions is limited if the human is unable to justify and understand their trust in said predictions. Explanation is a way of obtaining such understanding, by selecting "what" must be communicated and "how" that information is presented. Moreover,according to [54] XAI is necessary for regulatory issues, and thus, there is also a legal component to be considered. The EU General Data Protection Regulation (GDPR) aims to ensure the 'right to explanation' concerning automated decision-making models.

Interpretability also helps developers understand and improve the model performance. It can aid in troubleshooting and debugging, as well as in detecting potential biases in the AI system and provide insights into how the system would react under different circumstances. This understanding can also provide insights that go beyond predictions, uncovering underlying patterns and relationships within the data.

Vouros [55] provides a comprehensive review of state-of-the-art for explainable DRL methods, categorizing them in classes according to the paradigm they follow, the interpretable models they use, and the surface representation of explanations provided. In fact, it is the unique existing literature review focusing on XDRL. In our work, we have used three explainability techniques: the SHapley Additive exPlanations (SHAP) [56], the Local Interpretable Model-agnostic Explanations (LIME) methodology [57], and feature importance methods.

The SHAP method offers a unified way to explain each instance´s predictions [56]. Imagine you're playing a team sport, and at the end, you want to know who contributed most to the win. It's not enough to say "everyone did their best"; you want specifics, like who scored the most goals. SHAP does this for machine learning. It breaks down a prediction to show the impact of each feature—like how a player's actions affect the game's outcome. Here is how SHAP works in very simple terms: 1. Contribution: It looks at each feature (like a player in a game) of the data the model uses and asks, "What's the contribution of this feature to the final prediction?". 2. Fair Distribution: Then, SHAP uses a fair method for distributing the "credit" of the outcome. It makes sure that the contributions of all features sum up to the total prediction. This is like making sure that all the individual scores from players add up to the final score of the game. 3. Individual Impact: SHAP values can tell you the impact of a single feature on the prediction. For example, just as you might wonder how much scoring that one goal early on helped the team win, SHAP can show how much changing one piece of data can change the prediction. 4. Teamwork: It also considers how features affect the prediction when they work together, kind of like how players might pass the ball to each other. 5. Comparisons: It even lets you compare the importance of different features. Like after a match, you might debate who the most valuable player was, based on their contributions.

LIME is a tool designed to explain the predictions of any classifier in a way that is understandable and accurate [57]. It works by learning a simplified model around the prediction, using perturbations of uniformly sampled features. This allows for a better understanding of the relationship between input features and model responses. The goal is to ensure local fidelity, explaining how the model behaves around the specific instance being predicted. While LIME was originally designed to inspect models using binary vectors, for instance representation, it also offers solutions for outcome explanation, as it highlights the importance of features in a local context only. LIME is a universal method that enhances both local and global model

interpretability. In the context of DRL, it can be used to explain any models, such as local decisions for action selection based on interpretable state feature representation, even if it wasn't specifically designed for that task. Functions can provide binary vectors, for instance representation, while any interpretable model can theoretically be used for outcome explanation, like a linear model or a decision tree. Visual means are used to provide surface representations of the instance's interpretable representations for model inspection, as well as explanation logic for local and model explainability. Although Ludenberg et al. [56] showed that LIME is a subset of SHAP and that SHAP outperformed LIME, LIME is still a useful tool, since it's faster compared to SHAP and thus, LIME can be practical in cases where efficiency is important.

Feature importance methods provide insights into which features are most influential in the model's overall decision-making process. By analyzing the importance of each feature, we can understand how the model prioritizes different aspects of the input data when making predictions. This helps identify key drivers of the model's behavior, enhancing transparency and trust in the model's decisions.

Integrating explainability in deep reinforcement learning financial predictions

In this section we describe how we integrate the explainability techniques mentioned in the previous subsection with the DRL methodology illustrated before. Concretely, our work builds on the existing framework from the GitHub repository "Reinforcement learning in portfolio management" https://github.com/deepcrypto/Reinforcement-learning-in-portfolio-management-/tree/master?tab=readme-ov-file. The goal is to enhance this framework by integrating explainability features to make the decision-making process of the DRL model transparent and understandable.

Our starting point is the already developed PPO-based model in the framework, which has demonstrated effectiveness in portfolio management tasks. The PPO algorithm is chosen for its balance between exploration and exploitation, making it well-suited for dynamic and unpredictable environments like financial markets.

We design a training phase consisting on financial data downloaded from Investing.com, Wind and the Shinging-Midas Private Fund with OHCL (Open, High, Close, and Low) price information about several tickers to build a portfolio in a certain period of time (for a detailed description of data, see section of Experiments and Results). Once the agent is ready, after a preprocessing phase to ensure the data is clean and ready for analysis. This involves normalizing values, handling missing data and structuring the data so that the model can work with it. Then, the training phase starts on the financial market loaded, the agent interacts with the environment, learns from the data, and refines its policy to improve decision-making.

To implement the explainability techniques in the already analyzed framework, we will use post-hoc interpretability methods. This requires saving the state-action pairs for all steps during the training process. These state-action pairs will be used later to create explainability models. To achieve our objective, we implement explainability techniques such as SHAP, LIME, and feature important methods. These methods reveal which features and inputs are most influential in the agent decision-making process, providing insights both at a global level and for specific predictions, leading to an interpretable DRL model.

Experiments and results

In this section, we will describe the different experiments that we have performed to show the usefulness of our explainable deep reinforcement learning methodology. For reproducibility and transparency, we make fully accessible the specific dataset used, as well as the code (please

see our Data and Code availability statement). The specific dataset used is included as S1 Data (excel file) and both, the code and data of the paper are available at GitHub https://github.com/aleedelarica/XDRL-for-finance.

Data description

The dataset used for training and evaluating our Deep Reinforcement Learning (DRL) agent comes from Investing.com, Wind, and the Shinging-Midas Private Fund, covering the American stock market. We selected a basket of five American technological stocks with large trading volumes to ensure that our trades do not affect the market: Apple (AAPL), Adobe (ADBE), Alibaba (BABA), Sony (SNE) and Visa (V). The dataset spans a three-year period, with the training phase running from 2015/01/01 to 2016/12/31 and the testing phase from 2017/01/01 to 2018/01/01. For each stock, the dataset includes daily Open, High, Low, and Close (OHLC) price information. To ensure that our DRL agent is robust and able to generalize across different stocks, we normalized the price data. Specifically, the opening, closing, high, and low prices were divided by the closing price on the last day of the dataset. This normalization makes the data comparable across various price ranges and ensures consistency in model training and evaluation. Missing data, which typically occurs on weekends and holidays, was handled by filling the missing values with the closing price from the previous trading day. On such days, the trading volume was set to 0, indicating that the market was closed. This cleaned dataset is included as S1 Data and provides the foundation for training the DRL agent to make portfolio allocation decisions. The agent's ability to allocate between different assets was tested on these low-correlation or negatively correlated stocks to demonstrate its effectiveness in managing diversified portfolios.

Experiments setup

For our portfolio management experiments we have considered the OHCL information of the aforementioned five different technological assets, having a total of 20 features in the state space of the agent. We have considered a PPO DRL algorithm to learn the weights of the deep neural network with default hyperparameters and 100 epochs across all the financial data. Having all that information and performing the training period of the neural network, we interpret the predictions of the deep neural network in the trading period using feature importance, SHAP and LIME methods as we will describe in the following subsections.

Feature importance analysis

We begin with experiments that show how the feature importance can be extracted for the predictions of the agent during the trading period. To do so, we configure a portfolio of several technological assets and measure the importance of their OHCL features in the trading period, information that we illustrate in Fig 2. We will then illustrate how can SHAP and LIME methods can offer interpretability and explainability of the actions performed by the agent in such a scenario.

Apple (AAPL) as a key market indicator. As shown in Fig 2, Apple's closing price consistently emerged as the most critical feature in the agent's decision-making process. This indicates that the DRL agent heavily relied on Apple's performance to guide its portfolio allocation. For example, during periods of strong performance for Apple, the agent significantly increased its allocation to Apple, while reducing allocations to other assets. This suggests that the agent recognized Apple's market leadership and its influence on the broader technology sector.

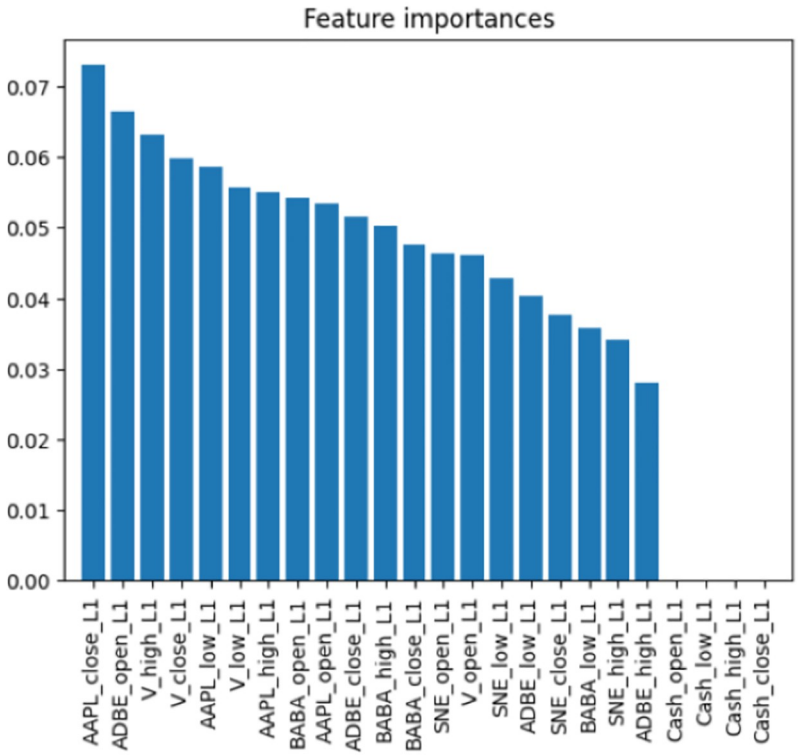

Fig 2. Importance of the features used as the state space of the DRL agent for financial portfolio management experiments. We can see how the APPLE close value is the most important for the estimated policy of the DRL agent.

https://doi.org/10.1371/journal.pone.0315528.g002

Asset-Specific Patterns. Fig 3 shows the average importance of each feature across all stocks. Apple (AAPL) remains the most significant, followed by Visa (V), Alibaba (BABA), Adobe (ADBE), and Sony (SNE). This means the DRL agent in the model prioritizes Apple's data in order to make the investing decisions, as it is consistently seen in the previous

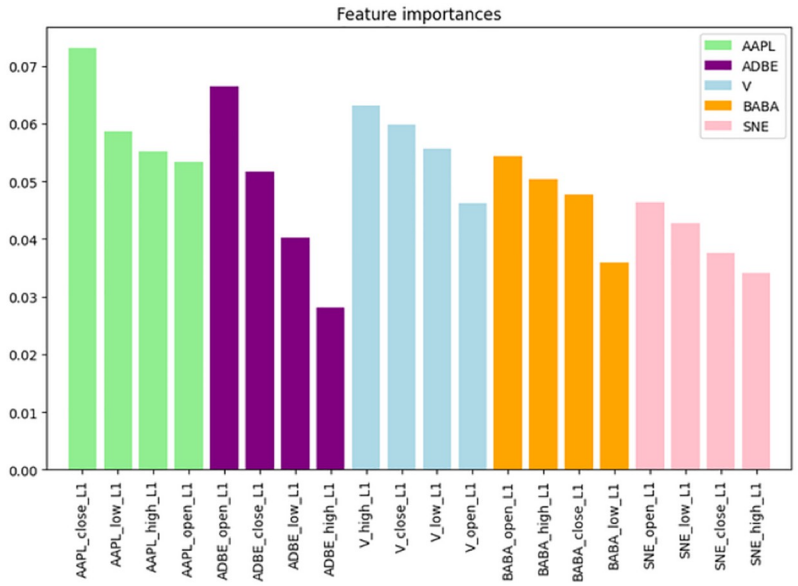

Fig 3. Feature importance of the state space of the DRL agent sorted by assets.

https://doi.org/10.1371/journal.pone.0315528.g003

experiment. Most critically, we can see how, for every asset, it is not clear which of the OHCL is the most important one, as it varies in every asset. Consequently, it is wise to use them all to make the agent more robust to different portfolios. While Apple's closing price dominated, other stocks showed different feature importance profiles: • Visa (V): The opening price of Visa was one of the most important features, reflecting the agent's focus on Visa's initial market activity. For example, when Visa's opening price was higher than expected due to positive market news, the agent increased its allocation to Visa, interpreting this as a bullish signal. • Adobe (ADBE): Interestingly, Adobe's high price was the least significant feature across all stocks, suggesting that the agent did not place much emphasis on extreme daily price highs for Adobe. However, Adobe's opening price ranked as the second most important feature after Apple's closing price. This indicates that the agent paid closer attention to Adobe's initial trading activity to gauge its market sentiment, particularly during volatile periods.

Varying importance of OHLC features. The standout result is that the importance of OHLC features varied significantly across different assets. For some stocks, such as Visa (V), the high price played a more critical role in the agent's decisions. This reflects how the agent leveraged extreme daily price movements to adjust its portfolio. For example, on a volatile trading day, Visa's high price spiked, prompting the agent to reduce its exposure to Visa, as it anticipated a potential price correction after the rapid price increase. This asset-specific variation underscores the agent's ability to adapt its strategy dynamically based on the behavior of individual assets, a critical feature in managing diversified portfolios in volatile markets.

Combined importance of technical indicators. To better understand the broader impact of technical indicators, we plotted the average importance of each OHCL feature across all assets (Fig 4). The results confirmed that close and open prices were the most significant across the portfolio. However, the high and low prices, while less dominant, still played an important role in the agent's predictions.

SHAP analysis

To gain a deeper understanding of the model's predictions, we utilized SHAP, which allows us to break down the contribution of each feature to the DRL agent's portfolio decisions. Since

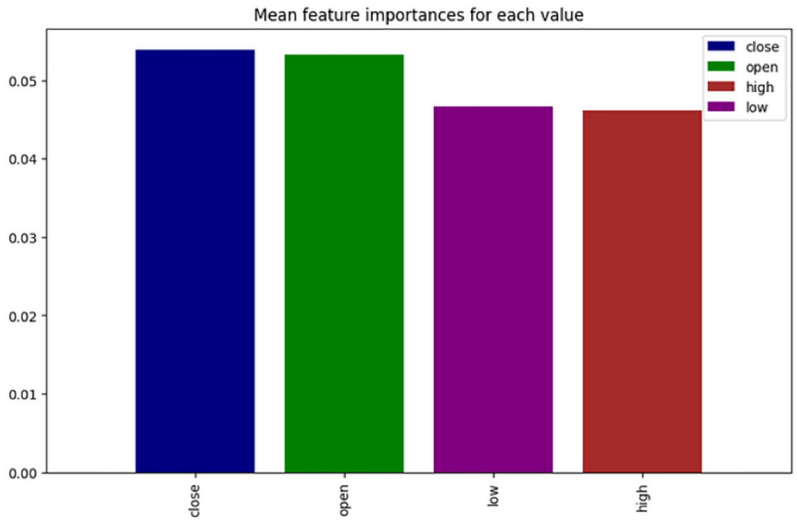

Fig 4. Mean feature importance of the different financial indicators across all the assets of the portfolio.

https://doi.org/10.1371/journal.pone.0315528.g004

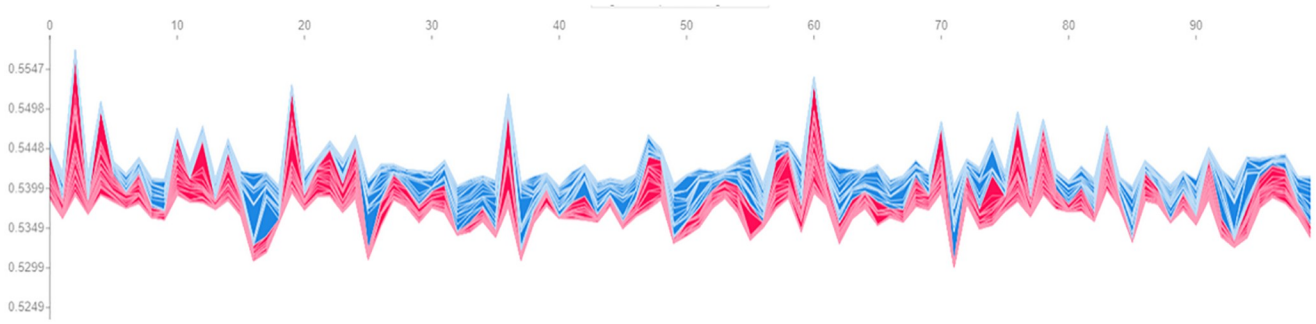

Fig 5. SHAP force plot for AAPL weight allocation with all features contribution.

https://doi.org/10.1371/journal.pone.0315528.g005

our model is multi-output (making weight allocations for six assets, including a cash risk-free asset), SHAP generates individual force plots for each asset´s weight allocation.

Apple's weight allocation. In Fig 5, we observe the SHAP force plot for Apple's weight allocation. SHAP values indicate how each feature contributed to the model's decision for a specific sample: Positive SHAP values (red) push the prediction higher, suggesting an increase in the weight allocated to Apple. Negative SHAP values (blue) lowered the prediction, decreasing the weigh allocation to Apple. The baseline, represented by the horizontal line, reflects the average model output (the logit) over the entire dataset. By analyzing individual SHAP values, we can see the direct influence of features like Apple's closing price (AAPL_close_L1) on the agent's decisions.

Isolating apple's closing price. Conversely, we can do the same process with only one feature, to test how and whether it impacts on the portfolio, as we illustrate in Fig 6, where we can see how the red and blue segments still indicate positive and negative contributions, respectively, but they are now isolated into a single feature (Apple's closing price value contribution, in this case).

This plot helps in understanding both the specific impact of this particular feature through the entirety of the dataset and in one specific prediction without the interference of others. For instance, imagine an investor who wants to understand why the model lowered the weight allocation of Apple on a certain instance. The investor could analyze the impact of the different features to that prediction, realizing that the closing price of Apple stock lowered significantly this weight allocation prediction. The combination of the investor's knowledge and the model ´s explanation gives a complete understanding of the prediction. Continuing with the storyline, let's say the investor also examines another instance where the model increased the weight

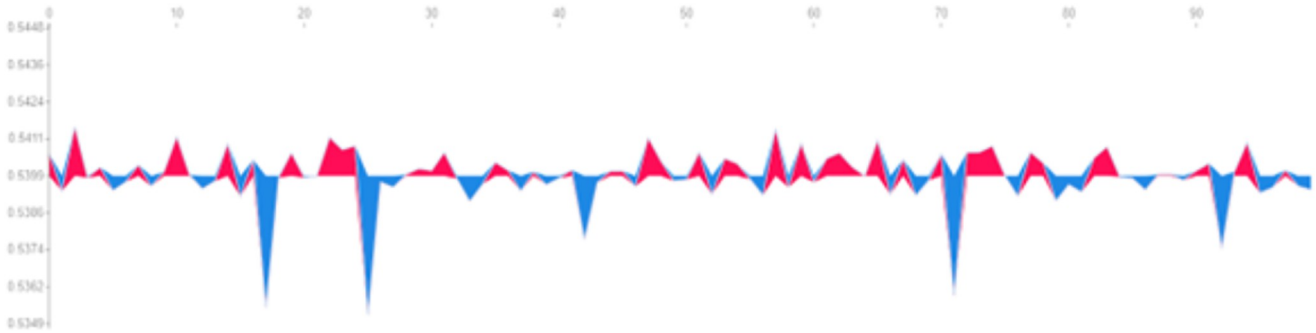

Fig 6. SHAP force plot for AAPL weight allocation with only Apple's closing price value contribution.

https://doi.org/10.1371/journal.pone.0315528.g006

allocation of Apple. By analyzing this specific prediction, the investor notices a red segment indicating that the BABA_open_L1 value had a positive impact. This might initially seem puzzling, as BABA_open_L1 refers to the opening price of Alibaba stock from the previous day. However, the investor recalls that on this particular day, positive news about Alibaba's strong quarterly performance had a ripple effect on the tech sector, boosting overall market sentiment and indirectly benefiting Apple's stock price. The model's sensitivity to such interconnected market dynamics is reassuring to the investor. It demonstrates that the model doesn't just consider isolated stock movements but also understands broader market trends and their impacts on individual assets. This interconnected understanding is crucial for making informed allocation decisions in a diversified portfolio. This comprehensive explainability method enhances the investor's trust in the model, ensuring they can confidently rely on its predictions to guide their investment strategies.

LIME analysis

By using LIME explanations, we can delve deeper into the feature contributions for each asset's weight allocation at a specific point in time. LIME provides instance-specific insights, helping us interpret the decisions made by the DRL agent. We analyze two instances where LIME explains the model's predictions for Apple and Adobe.

Apple's weight allocation. In Fig 7, we present the LIME explanation for Apple's predicted weight allocation at a particular instance, where the allocation reached its maximum value of 0.21. This high allocation is driven by several features. The positive contributors include V_high_L1, BABA_close_L1, AAPL_open_L1, ADBE_close_L1, and BABA_low_L1. These features collectively push the prediction higher. Specifically, the feature V_high_L1 (the high price of Visa on the previous day) has the largest positive contribution, with a value of 1.02, signaling favorable market conditions. BABA_close_L1 (Alibaba's closing price from the previous day) also plays a significant role, contributing 0.96 to Apple's allocation. This could indicate a broader positive sentiment in the technology sector, indirectly boosting Apple. Both Apple's opening price (AAPL_open_L1) and Adobe's closing price (ADBE_close_L1) further push the prediction higher. Notably, in this instance there are no significant negative contributors, indicating a strong overall positive sentiment for Apple. The combined effects of Visa, Alibaba, and Adobe's performance strengthen Apple's allocation, highlighting the model's sensitivity to multiple assets' interdependencies within the technology sector.

Adobe's weight allocation. In Fig 8, we turn to the LIME explanation for Adobe's weight allocation. In this instance, the predicted allocation is 0.16 within a range of 0.15 to 0.25. Positive contributions come from V_low_L1 and BABA_open_L1. Visa's low price from the previous day (V_low_L1) has a strong positive contribution, pushing Adobe's allocation higher. Alibaba's opening price from the previous day (BABA_open_L1) also contributes positively, signaling bullish market conditions. Unlike the Apple example, Adobe's allocation faces negative influences from AAPL_low_L1 and AAPL_high_L1 (Apple's low and high prices), which

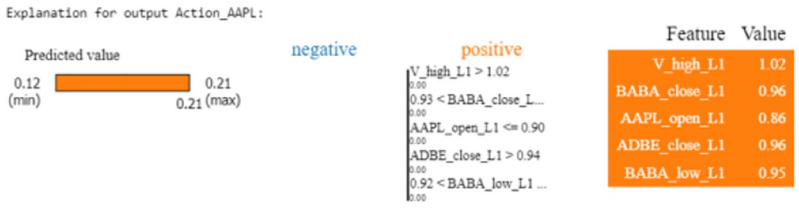

Fig 7. LIME explanation for Apple's weight allocation prediction at a particular instance.

https://doi.org/10.1371/journal.pone.0315528.g007

Fig 8. LIME explanation for Adobe's weight allocation prediction at a particular instance.

https://doi.org/10.1371/journal.pone.0315528.g008

slightly reduce Adobe's allocation. This insight suggests that while Adobe benefits from positive conditions in Visa and Alibaba, certain Apple metrics dampen its allocation.

Implications for investors. These LIME explanations can assist investors by offering explicit explanations for each forecast, allowing them to observe how different factors interact and influence the DRL agent decisions. For example, if an investor consistently observes that specific measures from Visa and Alibaba continuously increase allocations to tech stocks such as Apple and Adobe, they may opt to constantly monitor these metrics. This actionable insight can help investors make more informed and proactive decisions.

Understanding why the model cuts allocations in response to specific traits can also assist investors in reducing risks. If Apple's poor metrics routinely reduce allocations to other stocks, the investor may investigate this link further and change their portfolio to balance potential drawbacks.

As we have shown throughout the section, the three explainability techniques can be implemented independently or jointly, being able to explain the DRL agent predictions in trading time, being able to track throughout the time whether the policy is acting as it is expected or not, what is an advantage with respect to the rest of published DRL explainability methods, that only offer explanations of the model in training time, not being able to monitor the predictions done by the agent in the trading time.

Statistical hypothesis testing of the robustness of the Shapley values retrieved by our methodology

Our methodology is able to explain the actions performed by an agent with respect to the states that it observes, however, it is necessary to assess the consistency of its explainability, for example of the Shapley values that we have computed. By guaranteeing the explainability, we can trust the explanations of the agent's action. In order to do so, we repeat 50 times the experiment of retrieving the Shapley values of the random forest trained in the states and actions of the DRL agent across different random seeds in our experiments and employ statistical hypothesis testing to discriminate whether the values of the Shapley factors are consistent.

In particular, we used a mixed-effects model, that allows us to account for both fixed effects, that in this case are the different variables of the model and also test random effects, that are the variabilities of the Shapley values introduced by the different seeds of every experiment repetition. Using this methodology, we want to determine whether the variability in Shapley values is significantly influenced by the choice of the seed, that is, by the randomness inherent to every different experiment, or if the values remain consistent regardless of the seed used, that is what we desire that happens. We also want to test with the fixed effects whether the Shapley values are different for the variables, which is desirable as every variable of the states gives to the agent a different information that conditions its action.

Once that we have fitted the mixed-effects model to our data, we obtained in the summary that the estimated variance of the random effects modelling how the different seed changes

was effectively almost close to zero, with mean of 10^{-7} which makes the model impossible to converge. In other words, the Shapley values for all the models were similar for all the variables with variances lower than 10^{-7}, making the mixed-effects model to struggle to converge. This means that the variability in the Shapley values due to the different repetitions of the experiment is negligible, implying directly that the null hypothesis of the statistical test associated with the model regarding the random effects is completely compatible with our sample of models, as the statistical test uses the variance to determine the compatibility of the sample if this hypothesis is true. Recall that the null hypothesis for the random effects of this statistical test is whether variances are consistent and the alternative hypothesis is that they are different. Variances close to zero means having a p-value of almost 1, having all the evidence in favour of this hypothesis. However, the coefficients for the Shapley values between the variables were statistically significant, which means that the variables have a consistent effect on the Shapley values across seeds, as we desired, with p-values lower than 0.01.

Two main conclusions can be extracted of this statistical procedure: (i): the Shapley values do not vary significantly across different seeds and are therefore consistent and (ii): the variables consistently affect the Shapley values, and this effect is stable across different seeds. Both conclusions imply that this methodology for interpretating the actions performed by an agent through the shapley values of a model is consistent and robust.

Conclusions and further work

In this work, we successfully used SHAP, LIME and feature importance to explain the decisions made by our DRL model in portfolio management tasks. These methods provided clear and detailed insights into why the model made specific investment choices, affirming our hypothesis that it is possible to explain DRL predictions in portfolio management with explainability techniques. Also, our prediction explainability analysis with SHAP and LIME revealed that the importance of features varied across different market conditions and specific predictions. Moreover, our feature importance analysis showed that certain features consistently influenced the model's investment decisions more than others.

We believe that several future directions could enhance and expand the impact of this work.

First, it would be interesting to conduct several studies to evaluate how real investors interpret and react to the explanations provided by the model, which could yield valuable reinforcement learning from human feedback. For a better understanding of non-technical investors, it would be nice to develop an intuitive user interface that presents the model's decisions and their explanations in a user-friendly manner would improve its practicality. Also, it would be interesting to deal with a wider set of explainable techniques and different markets to assess the usefulness of the methodology in different scenarios. Finally, a future line of work would be to hybridize one of the a priori XDRL approaches of Bougie and Ichise [16] or Shi et al. [18] with our proposed post hoc approach (the only one in the literature of XDRL for portfolio management); that is, explaining both the model (the policy trained during the training period) as these authors do, and the predictions of the model during trading time (as we do), and see how training modifies the rules injected a priori [16] or the policy trained [18].

Supporting information

S1 Data. All data and code of the paper are freely and fully available at GitHub https://github.com/aleedelarica/XDRL-for-finance.
(XLSX)

Author Contributions

Conceptualization: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Data curation: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Formal analysis: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Investigation: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Methodology: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Project administration: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Resources: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Software: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Supervision: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Validation: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Visualization: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Writing – original draft: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

Writing – review & editing: Alejandra de-la-Rica-Escudero, Eduardo C. Garrido-Merchán, María Coronado-Vaca.

References

1. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA. Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine. 2017; 34(6):26–38. https://doi.org/10.1109/MSP.2017.2743240
2. Rather AM, Sastry V, Agarwal A. Stock market prediction and Portfolio selection models: a survey. Opsearch. 2017; 54:558–579. https://doi.org/10.1007/s12597-016-0289-y
3. Mangram ME. A simplified perspective of the Markowitz portfolio theory. Global Journal of Business Research. 2013; 7(1):59–70.
4. Wilford DS. True Markowitz or assumptions we break and why it matters. Review of Financial Economics. 2012; 21(3):93–101. https://doi.org/10.1016/j.rfe.2012.06.003
5. Hill JM. The different faces of volatility exposure in portfolio management. The Journal of Alternative Investments. 2013; 15(3):9. https://doi.org/10.3905/jai.2012.15.3.009
6. Sutton RS, Barto AG. Reinforcement learning: An introduction, 2nd ed. Cambridge, MA: MIT press; 2020.
7. Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al. OpenAI Gym. arXiv preprint arXiv:160601540. 2016.
8. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015; 521(7553):436–444. https://doi.org/10.1038/nature14539 PMID: 26017442

9. Lu Y, Lu J. A universal approximation theorem of deep neural networks for expressing probability distributions. Advances in Neural Information Processing Systems. 2020; 33:3094–3105.
10. Castelvecchi D. Can we open the black box of AI? Nature News. 2016; 538(7623):20. https://doi.org/10.1038/538020a PMID: 27708329
11. Lowenstein L. Financial transparency and corporate governance: you manage what you measure. Columbia Law Review. 1996; 96(5):1335–1362. https://doi.org/10.2307/1123407
12. Caruana R, Lundberg S, Ribeiro MT, Nori H, Jenkins S. Intelligible and explainable machine learning: Best practices and practical challenges. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining; 2020. p. 3511–3512.
13. Angelov PP, Soares EA, Jiang R, Arnold NI, Atkinson PM. Explainable artificial intelligence: an analytical review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2021; 11(5):e1424.
14. Mehta M, Palade V, Chatterjee I. Explainable AI: Foundations, methodologies and applications. Springer; 2023.
15. Wang J, Zhang Y, Tang K, Wu J, Xiong Z. Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining; 2019. p. 1900–1908.
16. Bougie N, Ichise R. Towards interpretable reinforcement learning with state abstraction driven by external knowledge. IEICE TRANSACTIONS on Information and Systems. 2020; 103(10):2143–2153. https://doi.org/10.1587/transinf.2019EDP7170
17. Guan M, Liu XY. Explainable deep reinforcement learning for portfolio management: an empirical approach. In: Proceedings of the second ACM international conference on AI in finance; 2021. p. 1–9.
18. Shi S, Li J, Li G, Pan P, Liu K. XPM: An explainable deep reinforcement learning framework for portfolio management. In: Proceedings of the 30th ACM international conference on information & knowledge management; 2021. p.1661–1670.
19. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:170706347. 2017.
20. Belle V, Papantonis I. Principles and practice of explainable machine learning. Frontiers in Big Data. 2021; 4:688969. https://doi.org/10.3389/fdata.2021.688969 PMID: 34278297
21. Rozemberczki B, Watson L, Bayer P, Yang HT, Kiss O, Nilsson S, et al. The shapley value in machine learning. arXiv preprint arXiv:220205594. 2022.
22. Visani G, Bagli E, Chesani F, Poluzzi A, Capuzzo D. Statistical stability indices for LIME: Obtaining reliable explanations for machine learning models. Journal of the Operational Research Society. 2022; 73 (1):91–101. https://doi.org/10.1080/01605682.2020.1865846
23. Hambly B, Xu R, Yang H. Recent advances in reinforcement learning in finance. Mathematical Finance. 2023; 33(3):437–503. https://doi.org/10.1111/mafi.12382
24. Liu XY, Yang H, Gao J, Wang CD. FinRL: Deep reinforcement learning framework to automate trading in quantitative finance. In: Proceedings of the second ACM international conference on AI in finance; 2021. p. 1–9.
25. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature. 2015; 518(7540):529–533. https://doi.org/10.1038/nature14236 PMID: 25719670
26. Sutton RS, McAllester D, Singh S, Mansour Y. Policy gradient methods for reinforcement learning with function approximation. Advances in Neural Information Processing Systems. 1999; 12.
27. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, et al. Continuous control with deep reinforcement learning. arXiv preprint arXiv:150902971. 2015.
28. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. PMLR; 2018. p. 1861–1870.
29. Dankwa S, Zheng W. Twin-delayed DDPG: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. In: Proceedings of the 3rd international conference on vision, image and signal processing; 2019. p. 1–5.
30. Liu XY, Xiong Z, Zhong S, Yang H, Walid A. Practical deep reinforcement learning approach for stock trading. arXiv preprint arXiv:181107522. 2018.
31. Liu XY, Yang H, Chen Q, Zhang R, Yang L, Xiao B, et al. FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance. arXiv preprint arXiv:201109607. 2020.
32. Liang Z, Chen H, Zhu J, Jiang K, Li Y. Adversarial deep reinforcement learning in portfolio management. arXiv preprint arXiv:180809940. 2018.

33. Jiang Z, Liang J. Cryptocurrency portfolio management with deep reinforcement learning. In: 2017 Intelligent systems conference (IntelliSys). IEEE; 2017. p. 905–913.

34. Sadighian J. Deep reinforcement learning in cryptocurrency market making. arXiv preprint arXiv:191108647. 2019.

35. Sattarov O, Muminov A, Lee CW, Kang HK, Oh R, Ahn J, et al. Recommending cryptocurrency trading points with deep reinforcement learning approach. Applied Sciences. 2020; 10(4):1506. https://doi.org/10.3390/app10041506

36. Zhang Z, Zohren S, Roberts S. Deep reinforcement learning for trading. arXiv preprint arXiv:191110107. 2019.

37. Buehler H, Gonon L, Teichmann J, Wood B, Mohan B, Kochems J. Deep hedging: hedging derivatives under generic market frictions using reinforcement learning. Swiss Finance Institute Research Paper. 2019;(19-80).

38. Cao J, Chen J, Hull J, Poulos Z. Deep hedging of derivatives using reinforcement learning. arXiv preprint arXiv:210316409. 2021.

39. Carbonneau A. Deep hedging of long-term financial derivatives. Insurance: Mathematics and Economics. 2021; 99:327–340.

40. Pham U, Luu Q, Tran H. Multi-agent reinforcement learning approach for hedging portfolio problem. Soft Computing. 2021; 25(12):7877–7885. https://doi.org/10.1007/s00500-021-05801-6 PMID: 33897298

41. Lussange J, Lazarevich I, Bourgeois-Gironde S, Palminteri S, Gutkin B. Modelling stock markets by multi-agent reinforcement learning. Computational Economics. 2021; 57(1):113–147. https://doi.org/10.1007/s10614-020-10038-w

42. Huang Z, Tanaka F. MSPM: A modularized and scalable multi-agent reinforcement learning-based system for financial portfolio management. PLoS ONE. 2022; 17(2):e0263689. https://doi.org/10.1371/journal.pone.0263689 PMID: 35180235

43. Lee J, Kim R, Yi SW, Kang J. MAPS: Multi-agent reinforcement learning-based portfolio management system. arXiv preprint arXiv:200705402. 2020.

44. Zhang Y, Zhao P, Wu Q, Li B, Huang J, Tan M. Cost-sensitive portfolio selection via deep reinforcement learning. IEEE Transactions on Knowledge and Data Engineering. 2020; 34(1):236–248.

45. Shi S, Li J, Li G, Pan P, Chen Q, Sun Q. GPM: A graph convolutional network based reinforcement learning framework for portfolio management. Neurocomputing. 2022; 498:14–27. https://doi.org/10.1016/j.neucom.2022.04.105

46. Hao Z, Zhang H, Zhang Y. Stock portfolio management by using fuzzy ensemble deep reinforcement learning algorithm. Journal of Risk and Financial Management. 2023; 16(3):201. https://doi.org/10.3390/jrfm16030201

47. Jang J, Seong N. Deep reinforcement learning for stock portfolio optimization by connecting with modern portfolio theory. Expert Systems with Applications. 2023; 218:119556. https://doi.org/10.1016/j.eswa.2023.119556

48. Koratamaddi P, Wadhwani K, Gupta M, Sanjeevi SG. Market sentiment-aware deep reinforcement learning approach for stock portfolio allocation. Engineering Science and Technology, an International Journal. 2021; 24(4):848–859. https://doi.org/10.1016/j.jestch.2021.01.007

49. Li X, Li Y, Zhan Y, Liu XY. Optimistic bull or pessimistic bear: Adaptive deep reinforcement learning for stock portfolio allocation. arXiv preprint arXiv:190701503. 2019.

50. Nousi P, Avramelou L, Rodinos G, Tzelepi M, Manousis T, Tsampazis K, et al. Leveraging Deep Learning and Online Source Sentiment for Financial Portfolio Management. arXiv preprint arXiv:230916679. 2023.

51. Azhikodan AR, Bhat AG, Jadhav MV. Stock trading bot using deep reinforcement learning. In: Innovations in Computer Science and Engineering: Proceedings of the Fifth ICICSE 2017. Springer; 2019. p. 41–49.

52. Liu R, Nageotte F, Zanne P, de Mathelin M, Dresp-Langley B. Deep reinforcement learning for the control of robotic manipulation: a focussed mini-review. Robotics. 2021; 10(1):22. https://doi.org/10.3390/robotics10010022

53. Shao K, Tang Z, Zhu Y, Li N, Zhao D. A survey of deep reinforcement learning in video games. arXiv preprint arXiv:191210944. 2019.

54. Adadi A, Berrada M. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE access. 2018; 6:52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052

55. Vouros GA. Explainable deep reinforcement learning: state of the art and challenges. ACM Computing Surveys. 2022; 55(5):1–39. https://doi.org/10.1145/3527448

56. Lundberg SM, Lee SI. A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems. 2017; 30.

57. Ribeiro MT, Singh S, Guestrin C."Why should I trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining; 2016. p. 1135–1144.