Expert Systems With Applications 238 (2024) 122027

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

GraphSAGE with deep reinforcement learning for financial portfolio optimization

Qiguo Sun [a,*], Xueying Wei [b], Xibei Yang [a]

[a] School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, 212003, Jiangsu Province, China
[b] Fenyang College of Shanxi Medical University, Fenyang, 032200, Shanxi Province, China

ARTICLE INFO

Keywords:
Portfolio management
Deep reinforcement learning
GraphSAGE
Explainable AI
SHAP

ABSTRACT

Portfolio optimization is an active management strategy that aims to maximize returns and control risk within reasonable limits. The Proximal Policy Optimization (PPO), a robust on-policy actor-critic deep reinforcement learning (DRL) model, is gaining popularity in portfolio optimization because it can help reduce emotional biases and take systematic investment actions. However, some research has found that the PPO model cannot achieve such remarkable performance in portfolio optimization as in games or robot control. In this paper, a novel GraphSAGE and DRL coupled model (GRL) is proposed to improve the architecture of the PPO agent by introducing a GraphSAGE-based feature extractor to capture the complex non-Euclidean relationships among market indexes, industry indexes and stocks. In addition, the explainable model SHAP is used to select a few but important features for GRL learning, and a method for generating a static financial graph is defined. This improves the robustness and training efficiency of the GRL model. We provide a holistic performance evaluation for GRL on three datasets using five metrics, i.e., Return On Investment (ROI), Sharpe Ratio, Sortino Ratio, Maximum Drawdown, and Calmar Ratio. The results show that the GRL model outperforms the Equal Weight strategy and the *S&P* 500 index. In addition, the results of the comparative analysis show that the Share-Extractor GRL and the Separate-Extractor GRL significantly outperform the PPO baseline without a feature extractor. This implies that integrating a GraphSAGE-based feature extractor into the PPO agent can improve its performance and robustness in portfolio optimization tasks.

1. Introduction

Portfolio management is the art and science of selecting and managing a group of financial instruments such as stocks, bonds, and futures to obtain long-term profit while minimizing the risk (Markowitz, 1991). Generally, passive portfolio management adopts a buy-and-hold favor to track an index or other benchmark, and Active management aims to beat the performance of an index by actively buying and selling stocks (Grinold & Kahn, 2000). Given a set of stocks, active portfolio management can be formed as a time-dependent optimization problem aiming to find an optimal strategy for stock allocation to achieve high cumulative returns and an appropriate level of risk exposure (Ozbayoglu, Gudelek, & Sezer, 2020; Si, Li, Ding, & Rao, 2017).

In recent years, machine learning (ML) models is gaining popularity in the area of portfolio management as they can help reduce emotional bias and take systematic investment actions (Benhamou, Saltiel, Ungari, & Mukhopadhyay, 2020; Heaton, Polson, & Witte, 2017; Iwasaki, Chen, Du, & Tu, 2018; Krauss, Do, & Huck, 2017; Lee & Yoo, 2020). As a fundamental ML paradigm, Deep Reinforcement Learning (DRL) models are inherently suited for portfolio allocation tasks and could generate significant excess returns with low risk in various markets (Betancourt & Chen, 2021; Jeong & Kim, 2019; Jiang & Liang, 2017; Liang, Chen, Zhu, Jiang, & Li, 2018; Si et al., 2017). Jeong and Kim (2019) proposed a trading system based on the Deep Q-learning Network (DQN). The system could determine the optimal number of trading stocks and increase the total profit by four times when trading the S&P500 index. However, the DQN does not operate on a continuous action space, so the system cannot be directly generalized to the task of assigning asset weights. To solve this problem, Betancourt and Chen (2021) employed the Proximal Policy Optimization (PPO) algorithm to construct the portfolio optimization agent. PPO is known for its stability, as it uses a clip function to avoid destructive updates of large weights (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017). Liang et al. (2018) however, points out that for portfolio management task the PPO algorithm does not perform as remarkably as it does in gaming or robot control.

* Corresponding author.
E-mail address: sunqiguo1992@gmail.com (Q. Sun).

https://doi.org/10.1016/j.eswa.2023.122027
Received 22 July 2022; Received in revised form 12 August 2023; Accepted 3 October 2023
Available online 7 October 2023

The financial market is inherently complex. The actual performance of the portfolio optimization model is affected by the features and data structure chosen (Mosavi et al., 2020; Ozbayoglu et al., 2020). In practice, the selection of less important features is crucial to increase the performance and generalizability of DRL models (Benhamou, Ohana, Saltiel, & Guez, 2021; Mosavi et al., 2020). Guan and Liu (2021) has developed an empirical approach that uses integrated gradients to quantify the impact of input features on DRL results. Their work provides insights for the application of an explainable method for feature selection in DRL models. Moreover, matrix algorithms are well suited for dealing with Euclidean data, while the data used in portfolio optimization usually define a non-Euclidean space where assets are complexly interconnected and therefore cannot be captured by Euclidean geometry. Graph Neural Networks (GNN) have recently been used to solve stock trend forecasting and ranking problems based on supervised learning, but have rarely been used in DRL models (Chen, Jiang, Zhang, & Chen, 2021; Chen, Wei, & Huang, 2018; Matsunaga, Suzumura, & Takahashi, 2019; Soleymani & Paquet, 2021). Soleymani and Paquet (2021) developed the first Graph Convolution Network and a DRL-coupled architecture (GRL) called Pocket, which outperformed market indexes and generated positive returns during the Covid-19 crisis. Research on GRL is still in its infancy. More research is needed to further improve their applicability and effectiveness in portfolio optimization.

In this paper, we proposed a new GRL model for portfolio optimization by combining GraphSAGE and the PPO algorithm. Our contribution is threefold:

- This study is unique in deploying the explainable model SHAP for feature selection to improve the performance and training efficiency of DRL models.
- The PPO agent architecture is enhanced by the implementation of GraphSAGE-based Share/Separate Feature Extractors.
- A holistic performance evaluation for GRL is performed based on different portfolios using five metrics, i.e. Return on Investment (ROI), Sharpe Ratio, Sortino Ratio, Max Drawdown and Calmar Ratio.
- A comparative analysis is performed to investigate the reasons for the better performance of GRL compared to the PPO agent with MLP policy.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature. Section 3 the methodology, Section 4 presents the results and discussion. Finally, Section 5 concludes and provides directions for further research.

2. Literature review

Our work can be related to the ever-growing fields of XAI, GNN, and DRL applications to financial markets. These state-of-the-art methods have recently garnered a lot of interest.

Interpretability of a ML model can be classified into two granular levels: global and local. Financial markets, as a complex multi-agent environment, need a global explainability thereby Shapley value is suitable to explain the model outputs (Babaei, Giudici, & Raffinetti, 2022). Based on the game theory, Lundberg and Lee (2017) proposed the *SHapley Additive exPlanations* (SHAP) model, which can generate Shapley values as a measure of the importance of each input feature. Krauss et al. (2017) implemented and evaluated deep neural networks, gradient-boosted-trees (GBT), random forests (RAF), and several ensembles on $S\&P500$ markets. They extract the variable importances using H2O software and found that returns corresponding to the past 4 to 5 days have the highest relative importance. Ohana, Ohana, Benhamou, Saltiel, and Guez (2021) constructed a gradient boosting decision trees (GBDT) to predict the $S\&P$ 500 futures price drops. They employed SHAP for feature engineering and found that retaining fewer and carefully selected features improves model performance. Ghosh and Sanyal (2021) constructed multiple ML and deep learning algorithms to investigate India VIX during the COVID-19. They employed the SHAP model to analyze the influence of respective features on model prediction. Babaei et al. (2022) proposed an explainable portfolio management model using SHAP model for cryptocurrencies optimization. They aim to improve the trustworthiness of robot advisors by explaining what is behind the selected portfolio weights.

Recently, applying GNN for representation learning of graphs has received a lot of attention. A series of important GNN frameworks have been proposed successively such as graph convolution network (GCN) (Kipf & Welling, 2016), GraphSAGE (Hamilton, Ying, & Leskovec, 2017), Graph Isomorphism Network (GIN) (Xu, Hu, Leskovec, & Jegelka, 2018) and Graph Attention Network (Velickovic et al., 2017). In the last few years, the Graph data structure has been employed to represent sophisticated relational data in the financial markets (Wang, Zhang, Xiao, & Song, 2021). For example, Matsunaga et al. (2019) used GCN to embed the company knowledge graph and evaluated the model on different stocks in Nikkei 225 market. They suggest that the GCN and knowledge graph hold strong promise in designing stock market prediction models. Feng et al. (2019) proposed a GCN-based Relational Stock Ranking model for stock prediction, which captured the rich relations between stocks in the same sector or stocks have a supplier-customer relation. They evaluate their model on NYSE and NASDAQ markets and found that the model outperforms other ML models achieving an average return ratio of 98% and 71% on NYSE and NASDAQ markets. Based on the idea of using related companies' information to increase the prediction accuracy of the target company, Chen et al. (2018) developed a GCN-based stock prediction model. They evaluated the model on China stock market and found it outperforms LSTM models. Furthermore, Chen et al. (2021) proposed a novel GNN-CNN coupled model considering both stock market information and individual stock information. They demonstrated that the model outperforms several state-of-the-art ML models in stock trend prediction.

Algorithmic trading with DRL has shown huge applicability potential while it is still in its' early stages of development (Pricope, 2021). DRL models have three main frameworks, i.e., the critic-only, the actor-only, and the actor-critic frameworks. Moreover, DRL models can be off-policy or on-policy algorithms depending on whether the generated action is related to the improved policy. For example, the deep deterministic policy gradient (DDPG) is a actor-critic off-policy algorithm that learns a deterministic policy in a continuous space. PPO is a actor-critic algorithm that trains a stochastic policy in an on-policy way. Generally, actor-critic is suitable for portfolio optimization because its' action space and reward are continuous, which is naturally applicable for portfolio allocation.

Jiang, Xu, and Liang (2017) developed a DRL-based model for Cryptocurrencies portfolio management, in which Convolutional Neural Network, Recurrent Neural Network, and a Long Short-Term Memory are employed to build the critic and value networks. Liang et al. (2018) investigated the performance of different DRL agents on portfolio management such as the DDPG, PPO, and Policy Gradient (PG) for portfolio management. They pointed out three significant differences in the environment between the Financial market and game or robot control and proposed the so-called Adversarial Training method, which showed significant promotion in terms of average daily return and Sharpe ratio on the China Stock market. To improve the generalization ability of RL-based portfolio agent, Kuo, Chen, Lin, and Huang (2021) proposed a limit order book-generative adversarial model to simulate the financial market. They showed that the framework can improve out-of-sample portfolio performance by 4%. Recently, Liu et al. (2020) introduced a DRL library called FinRL, which delivers basic market environments and several state-of-the-art DRL models. Furthermore, Soleymani and Paquet (2021) developed a graph convolutional reinforcement learning framework for portfolio management, called DeepPocket. DeepPocket

considered the time-dependent interrelations between stocks and represented the relation by a dynamic graph, which achieved extremely high profitability in short-term investment.

In the environment, Jiang et al. (2017) and Soleymani and Paquet (2021) employed similar mathematical formalism to derive the logarithmic rate of return (r_t) of the portfolio, the formula can be expressed as,

$$r_t := ln\frac{p_t}{p_t - 1} = lny_t \cdot \omega_t - 1, \tag{1}$$

where p_t is the portfolio value at period t, ω_t is the portfolio weight vector at t, y_t is the price relative vector defined as,

$$y_t := (1, \frac{v_{1,t}}{v_{1,t-1}}, \frac{v_{2,t}}{v_{2,t-1}}, \ldots, \frac{v_{m,t}}{v_{m,t-1}})^T, \tag{2}$$

where $v_{i,t}$ is the close price of the ith asset at t. Then the final portfolio value becomes,

$$p_f = p_0 exp(\sum_{t=1}^{t_f+1} r_t) = p_0 \prod_{t=1}^{t_f+1} \mu_t y_t \cdot \omega_{t-1}, \tag{3}$$

where u_t is the transaction remainder factor, which cannot be solved analytically. Jiang et al. (2017) proposed a numerical method to solve u_t in which they used fixed commission rate of 0.25% as the cost of selling or purchasing non-cash assets.

3. Methodology

3.1. Features engineering

The raw features are 93 time series in daily frequency containing price, technical, Economic, and other market information. Specifically, price features consist of the open, high, low, close, and volume (OHLCV) and can be fetched using the open-source API *yfinance*. Moreover, the percent changes of close price and its' lags are considered according to Krauss et al. (2017). We employed *stockstats*, an open-source API provided by Cedric Zhuang, to generate technical indicators such as Simple Moving Average (SMA), Exponential Moving Average (EMA), Raw Stochastic Value (RSV), Relative Strength Index (RSI). *stockstats* provides a wrapper for the "pandas.DataFrame" class, which can be directly applied on the OHLCV data generated by *yfinance*.

Economic indicators are financial time series concerning employment, inflation, growth, consumer, productivity, and sentiment indicators. They are provided by the *FRED*, an online database containing economic data time series from different countries. Specifically, the data is fetched using the official API *fredapi* and stored in the "pandas.DataFrame". Some low-frequency time series have been upsampled to daily frequency and the NaN values have been filled using previous data.

Other indexes include the stock market indexes of different countries, sector fund indexes, and treasury yield indexes. For example, we considered four stock market indexes, i.e., the SSE Composite Index (SSE), Nikkei 225 (NI225), FTSE 100 Index (FTSE), and $S\&P$ 500. The sector funds are chosen to cover all the sectors that appeared in the portfolio stocks. In addition, the Treasury Yield 5 Years (FVX) and Treasury Yield 10 Years (TNX) are considered. Overall, these indexes are considered as the nodes in the financial graph such that their information can be shared by stocks through the message passing mechanism of GNN.

However, including many features may result in overfitting and reducing model robustness. Next, we briefly describe the mechanism of the SHAP model and explain how to use SHAP to evaluate feature importance. For more details on the SHAP model, we recommend readers to refer (Lundberg & Lee, 2017). Given the raw features set M, for each feature $i \in M$, the SHAP method trained two models in which one included feature i and the other exclude feature i. Then the difference between the outputs f of the two models was evaluated by $(f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S))$, where x_S indicates the values of features in S. We retrain the model on all feature subsets ($S \subseteq M \quad \{i\}$) and the Shapley values are the weighted averages of all the differences, defined as,

$$\Phi_i = \sum_{S\subseteq M\setminus\{i\}} \frac{|S|!(|M| - |S| - 1)!}{|M|!}(f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)). \tag{4}$$

In this work, all portfolios are a subset of $S\&P$ 500. Therefore, we assume that if the features are crucial for predicting the trends of $S\&P$ 500, they should also be dominant in predicting the trend of portfolios. Following Benhamou et al. (2020)'s work, we employ gradient boosting decision trees (GBDT) to predict whether the mean value of $S\&P$ 500 close price in the next 5 days is higher than the current $S\&P$ 500 close price. The raw features are 93 time series in daily frequency, we only remain 32 important features as node features.

3.2. Proposed method

3.2.1. Mathematical model

For the policy-gradient reinforcement learning method, the agent can be incredibly sensitive to perturbations as small changes in the network parameters may lead to significant changes in policy space resulting in catastrophic drops in performance. PPO is designed to address this problem by limiting the update of the policy network (Schulman et al., 2017). Different from other methods such as DDPG which random samples a subset from a large transition buffer, PPO defines a small fixed-length trajectory and updates the network's parameters by applying minibatch stochastic gradient descent on the batch-size chunks from the trajectory memory. Accordingly, three influential hyperparameters, i.e., the timesteps of the trajectory (T), the batch size (B), and the epochs of updates on each batch (U), need to be tuned when training a PPO agent.

PPO Framework has been refined for portfolio optimization tasks, in which the critic network evaluates market states and the actor network outputs actions, i.e., the weights of stocks. The objective function of PPO can be expressed as,

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}\hat{A}_t] = \hat{\mathbb{E}}_t[r_t(\theta)\hat{A}_t]. \tag{5}$$

where CPI indicates conservative policy iteration, $\hat{\mathbb{E}}_t[...]$ indicates the average over the batch samples, π_θ is the stochastic policy under the current parameters θ, $\pi_{\theta old}$ is the policy under the old parameters, $\hat{A}_t$ is the advantage estimator, $r_t(\theta)$ is a probability ratio. After T steps the agent will perform a learning update for network parameters. The advantage estimator can be defined as,

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T) \tag{6}$$

where t indicates the time index within the fixed-length trajectory $t \in T$, $V(s_t)$ indicates a state-value function.

Since the policy is a probability distribution, i.e. $\pi_\theta(a_t|s_t), \pi_{\theta old}(a_t|s_t) \in [0, 1]$, the values of $r_t(\theta)$ can be very large, which is not good for gradient calculation. Therefore, PPO adopts a clipped-surrogate-objective loss function to update the actor network. Specifically, the algorithm defines a constraint to clip $r_t(\theta)$ to make the update of parameters in the deep network a relatively small amount. Then Eq. (5) can be written as,

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)] \tag{7}$$

where ϵ is a hyperparameter determining the clip range. Finally, the total objective function can be expressed as the combination of loss functions of the actor and critic networks, as well as an entropy term,

$$L^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta(s_t)]], \tag{8}$$

where c_1, c_2 are coefficients, S is the entropy bonus, $L_t^{VF}(\theta)$ is the squared-error loss between target critic value and critic value in memory. We note that the PPO actually adopts gradient ascent so that c_1 should be positive while c_2 should be negative.

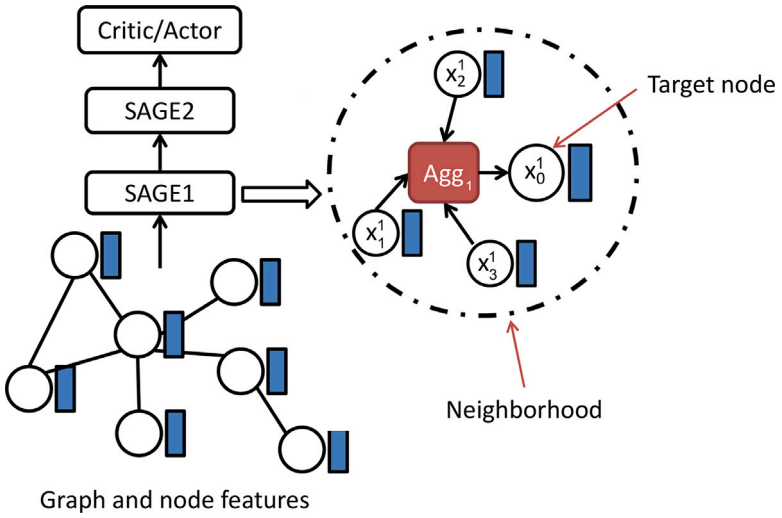

Fig. 1. GraphSAGE-based feature extractor.

3.2.2. GraphSAGE-based feature extractor

Generally, node embedding techniques are designed for distilling high-dimensional features from the neighborhood of the central node into a dense vector embedding using dimensionality reduction techniques. The resulting high-level node embeddings can be fed to downstream neural networks. For example, in Soleymani and Paquet (2021), graph convolution is applied on the dynamic graph to generate high-level node embeddings. The embedded vector is used as the input for the critic-actor DRL model. However, GCN cannot deal with graphs where new nodes (stocks) are continuously being added or removed, which corresponds to, in reality, the adjustment of the stocks in portfolio. GraphSAGE (Hamilton et al., 2017), as a state-of-the-art GNN framework, is capable of learning structural information of a graph and thus can be generalized to unseen nodes. Next, we introduce the mechanism of GraphSAGE and explain how we implemented the feature extractor. In addition, we note that the open-source library *torch-geometric* provides a GraphSAGE framework that can be easily modified and implemented in other models.

The basic idea behind graphSAGE is that instead of learning embeddings for each node, it learns a function to output node embeddings based on the node features and edges joining this node. Fig. 1 shows the message aggregation in the first GraphSAGE layer. The red box is the aggregator which can take functions such as mean and summation to process the aggregated information. The number of GraphSAGE layers indicates the scale of the receptive field. GraphSAGE uses different aggregation functions in different layers, denoted by $AGG_1, AGG_2, \ldots, AGG_k$, to deal with inputs information, which is the concatenation of neighbors' vector and the target vector itself,

$$x_v^k = \sigma[W^{(k)} \cdot \mathrm{CONCAT}(x_v^{k-1}, \mathrm{AGG}_k(x_u^{k-1} \forall u \in N(v)))] \tag{9}$$

where σ is some non-linear function, $W^{(k)}$ is the trainable weight matrix.

The framework of the extractor is shown on the left-hand side of Fig. 1. The inputs are graph-structured data, i.e., a undirected graph $\mathcal{G}$ with N nodes corresponding to stocks and market indexes and F node features $x = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x_N}, \vec{x}_i \in \mathbb{R}^F\}$. Two GraphSAGE layers with mean aggregators are implemented. In the final GraphSAGE layer, the output channel is set to one and the nodes corresponding to portfolio stocks have been extracted as the input for critic and actor networks. Hence the design of the extractor is shown in Algorithm 1,

Algorithm 1: The feature extractor (Hamilton et al., 2017)

input : Financial graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; Node features $x_v, \forall v \in \mathcal{V}$; Learnable weight matrices: $W^k, \forall k \in [1, 2]$; Non-linearity σ; Differential aggregation functions Agg_1, Agg_2, Agg_3

output: Vector representation z_v of nodes in portfolio $v \in \mathcal{V}'$

Initialization: $h_v^0 \leftarrow x_v, \forall v \in \mathcal{V}$;
for k *in* $[1, 2]$ **do**
  for $v \in \mathcal{V}$ **do**
    $h_{N(v)}^k \leftarrow Agg_k(\{h_u^{k-1}, \forall u \in N(v)\})$;
    $h_v^k \leftarrow \sigma(W^k \cdot \mathrm{CONCAT}(h_v^{k-1}, h_{N(v)}^k))$
  $h_v^k \leftarrow h_v^k / \|h_v^k\|_2, \forall v \in \mathcal{V}$,
$z_v \leftarrow h_v^k, \forall v \in \mathcal{V}'$

3.2.3. Model architecture

The portfolio trading system is formulated as a Markov Decision Process comprising five fundamental elements: the state space ($\mathcal{S}$), action space ($\mathcal{A}$), transition probability ($\mathbb{P}$), reward function (r), and discount factor (γ), adhering to the established framework introduced by Liu et al. (2022). In this context, $\mathcal{S} \in \mathbb{R}^{N \times F}$ denotes the state space with N nodes (corresponding to stocks, sector funds, and market indexes) and F features encompassing vital economic, technical, and price indicators chosen based on their SHAP values. Please note that the state space exclusively encompasses node features. These features are then integrated with a predefined edge set in the form of an adjacency matrix, resulting in the creation of the graph structure data. This data is subsequently fed into the agent module for further processing. The action space, denoted as $\mathcal{A} \in \mathbb{R}^m$, characterizes the allocation weights assigned to the m stocks within the portfolio. The transition probability, $\mathbb{P}$, encapsulates the dynamics of an unknown environment. The reward function, $r \in \mathbb{R}$, is designed to evaluate the portfolio's performance, with the discount factor $\gamma \in (0, 1]$ reflecting the temporal preferences. Specifically, the reward function is defined as:

$$r(s, a, s') = v' - v \tag{10}$$

where v' and v represent the portfolio values corresponding to states s' and s, respectively.

Fig. 2 illustrates the structure of the GRL algorithm, providing a visual depiction of its components and interactions. The framework comprises four key components: the Environment module, which characterizes market dynamics and generates the state–action-reward tuple

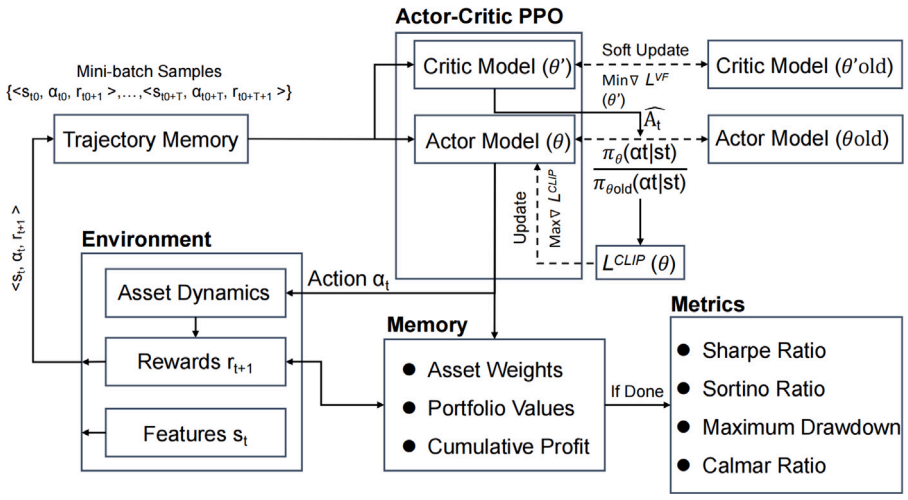

Fig. 2. The GRL algorithm's structure.

$\langle s_t, \alpha_t, r_{t+1} \rangle$ at each time step; the PPO Agent module, responsible for generating action α_t; the Memory module, which retains time-dependent data; and the Metrics module, utilized for assessing portfolio performance.

The design of the Environment module adheres to the standards set by OpenAI gym (Brockman et al., 2016), and the dataset has been seamlessly integrated into this module, following the approach outlined by Liu et al. (2022). Within this module, the computation of the new portfolio value v' involves the Action vector $\alpha_t \in \mathcal{A}$, changes in stock prices (referred to as Asset Dynamics), as well as the transition costs. Building upon this foundation, the preceding portfolio value is retrieved from the Memory module to determine the reward r_{t+1}. Additionally, the new portfolio value and asset weights are appended to the Asset Weights and Portfolio Value vectors within the Memory module, facilitating a comprehensive record of the evolving portfolio dynamics. Subsequently, the outcomes yielded by the Environment module find a temporary residence within the Trajectory Memory, where fixed-length mini-batch samples are aggregated, ultimately serving as the input data for the agent's training.

The PPO agent consists of a Critic Model and an Actor Model. The Actor Model contains the policy function parameterized by θ and takes the current state s_t as input and outputs the action α_t. The Critic Model, with a value function parameterized by θ', is used to support updating the Policy Function. In particular, it takes the states, rewards, and actions as inputs and outputs the advantage $\hat{A}_t$. The Critic Model updates by minimizing $L^{VF}(\theta')$, while the Actor Model updates by maximizing the clipped surrogate objective function $L^{CLIP}(\theta)$.

We adopted the PPO source code in *Stable Baseline3* (Raffin et al., 2021) and modified the critic and actor networks by using the GraphSAGE network as a feature extractor in the agent, as shown in Fig. 3. Under the GRL framework, we consider two variants: (1) a GRL where the Actor model and the Critic model use the same GraphSAGE-based extractor to learn the new embeddings of the SHAP-selected features, which we refer to as the Share-Extractor GRL, and (2) a GRL where the Actor model and the Critic model independently have their own feature extractors, which we refer to as the Separate-Extractor GRL. Specifically, all extractors map the high-dimensional observed state with the shape of $[N, F]$ to a vector z_v with the shape of $[m, 1]$, where N is the number of total nodes including market indexes, industry indexes and stock prices, and m is the number of stocks in the portfolio. Moreover, the parameters in the extractors and MLP are trained simultaneously.

The Memory module is used to store the time series of Asset Weights, Portfolio Returns, and Portfolio Value during training. When the current trajectory is finished, Done is set to True and the metrics in the Metrics module are calculated.

3.3. Evaluation metrics

We record date time, portfolio weights (ω_t), portfolio value (p), and portfolio returns in the environment every step based on Eqs. (1)–(3). At the end of the investment period, the final portfolio value is fetched from the environment and can be used to calculate Return on Investment (ROI),

$$\text{ROI} = \frac{p_f - p_0}{p_0} \tag{11}$$

Four more metrics are employed to evaluate portfolio performance, i.e., the Sharpe ratio, the Sortino ratio, the Max drawdowns, and the Calmar ratio. The annual Sharpe ratio can be defined as,

$$\text{Sharpe ratio} = \frac{\bar{R} - R_f}{\sigma} \tag{12}$$

where $\bar{R}$ is the annual expected portfolio return, R_f is the annual risk-free rate ($R_f = 0$) and σ is the annualized standard deviation of returns.

Instead of using all observations for calculating the standard deviation, Sortino ratio only considers the harmfully negative deviations σ^- and can be defined as,

$$\text{Sortino ratio} = \frac{\bar{R} - R_f}{\sigma^-} \tag{13}$$

Max Drawdown is used to quantify the largest decline in portfolio values, which can be defined as the difference between the maximum portfolio value P and the minimum portfolio value L,

$$\text{Maximum drawdown} = \frac{P - L}{P} \tag{14}$$

Calmar ratio uses max drawdown in the denominator which can be defined as,

$$\text{Calmar ratio} = \frac{\bar{R}}{\text{Maximum drawdown}}. \tag{15}$$

4. Experiments

4.1. Feature engineering

Fig. 4 rank the features by their SHAP values. We represent indicators using their id names on the FRED website so that readers can uniquely determine them. The full names and parameters of selected indicators are shown in Table 1. Since the model performance is related to the train-test splitting, two datasets are generated to experiment. The best performance is found in case (a) with F1 score, Accuracy,

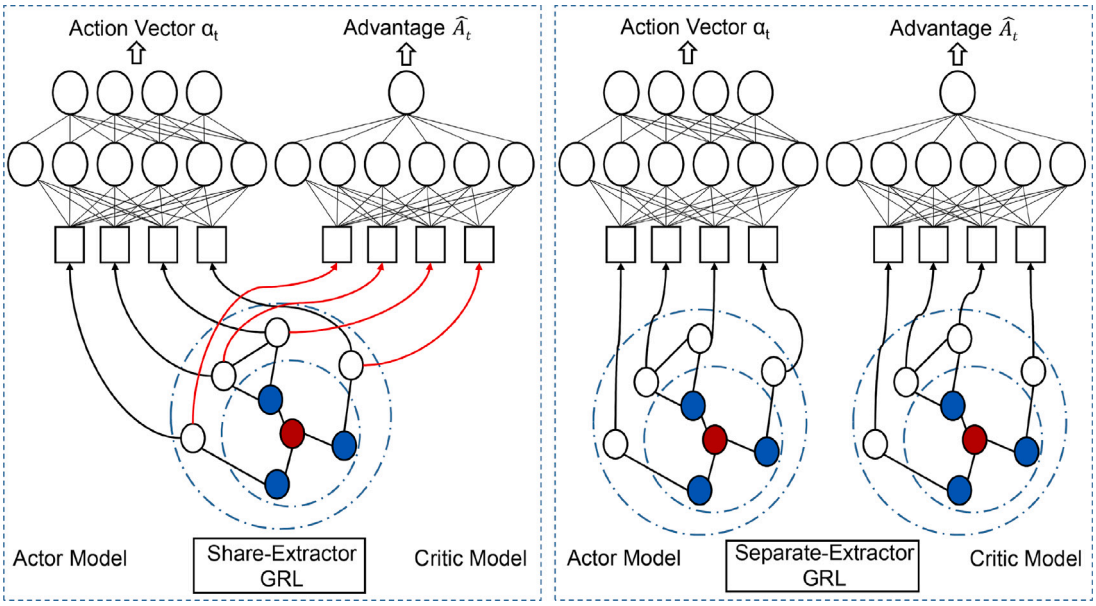

Fig. 3. The architectures of Share-Extractor GRL and Separate-Extractor GRL agents.

Precision, and Recall of 0.6406, 0.6498, 0.6483, 0.7562, respectively, and the second best case (b) is close to case (a).

It can be seen that the selected features are almost coincidental in Fig. 4(a) and (b), which implies the robustness of the feature selection method. For example, the economic indicator "RECPROUSM156N" representing "Smoothed U.S. Recession Probabilities" is determined as the most influencing economic indicator in both cases. The change_5_kama_5_30, which indicates the Kaufman's Adaptive Moving Average generally used to account for market noise or volatility, is identified as the most influencing technical indicator in both cases.

Benhamou et al. (2021) investigated market crises based on the SHAP method and proposed that the features can be divided into two categories, i.e., the pro-cyclical features which support positive market development and counter-cyclical features that predict negative shocks involving a reduced risk of equity downside moves. Specifically, in Fig. 4a pro-cyclical feature has red samples clustered on the right-hand side indicating that large feature values correspond to high SHAP values and vice versa. We find important Pro-cyclical features are technical indicators of adxr, chop-14, 20-d simple moving average (sma20), economic indicators of AISRSA (Auto Inventory/Sales Ratio), PSAVERT (Personal Saving Rate), and price indicators of historical percent change of close price such as change-9-s, change-10-s. The important counter-cyclical are economic indicators of RECPROUSM156N (Smoothed U.S. Recession Probabilities), CORESTICKM159SFRBATL (Sticky Price Consumer Price Index less Food and Energy), PCETRIM12M159SFRBDAL (Trimmed Mean PCE Inflation Rate), technical indicators of cr, short-term moving average sma5 and ema5, mfi-5, change-5-kama-5-30, as well as short-term price indicators such as change-5-s. The importance of short-term price indicators has also been identified in Krauss et al. (2017). Finally, eight economic indicators, fourteen technical indicators, and ten price indicators have been selected as the node features, which are listed in Table 1.

4.2. Financial graph

Portfolios were constructed using stocks in the $S\&P$ 500 market, which consists of the leading 500 companies in the U.S. With highly liquid, intense analyst coverage, and worldwide concentration, the $S\&P$ 500 market is considered an optimal environment for testing any trading strategy (Krauss et al., 2017). We considered two portfolios: one consists of the top 10 constituents by index weight and the other is identical to the portfolio in Soleymani and Paquet (2021).

The topological structure of the top10 dataset is shown in Fig. 5. The Graph is connected with the depth of two. The $S\&P$ 500 index (GSPC) has the largest node degree as it is connected to any other nodes in the graph. The black circle indicates the stocks in the portfolio. It can be seen that the stocks belonging to the same sector are connected because they are usually highly correlated. The method has also been used in Chen et al. (2021, 2018).

Moreover, stocks are linked to their sector funds, which are marked by green cycles. For example, half of the stocks belong to the Technology Sector and are linked to the Technology Select Sector SPDR Fund (XLK). The blue cycles indicate other markets indexes including SSE Composite Index in China (SSE), Nikkei 225 index (N225), the Financial Times Stock Exchange 100 Index (FTSE), the Treasury Yield 10 Years (TNX) and the Treasury Yield 5 Years (FVX).

4.3. Portfolio evaluation

In this section, we analyze the optimization results of Share-Extractor GRL and Equal Weight strategy for different portfolios. Table 2 shows the details about the train-validation-test splitting of the three datasets. GRL model is trained on the training dataset and the optimal model is saved by monitoring the total reward on the validation dataset. Then the model evaluation is performed on the test dataset.

PPO consists of multiple influencing hyperparameters such as the learning rate, trajectory length, batch size, update intervals, clip range, and coefficients for the state advantage. Moreover, the Graph-based feature extractor has a hyperparameter, i.e., the number of hidden channels. Therefore, using the exhaustive method for hyperparameter tuning is computationally intensive. In this work, we adopted the trial-and-error method to tune hyperparameters and obtained a suboptimal model. Specifically, we set the learning rate to 0.0001, total timesteps to 500 000, trajectory length to 32, batch size to 8, update intervals to 20, clip range to 0.5, coefficient of state advantage to 0.95, and the hidden channels of feature extractor to 16, respectively.

Fig. 6 shows the portfolio values on the Top10 dataset during the test period. We note that the policy network of PPO outputs probability distribution and the action is obtained by sampling the distribution. To evaluate the uncertainty of GRL models, we calculate the mean and variance of ten stochastic trajectories generated by the DRL model and present the mean-error plot. It can be seen that the average portfolio value of GRL model is higher than that in the Equal Weight strategy and

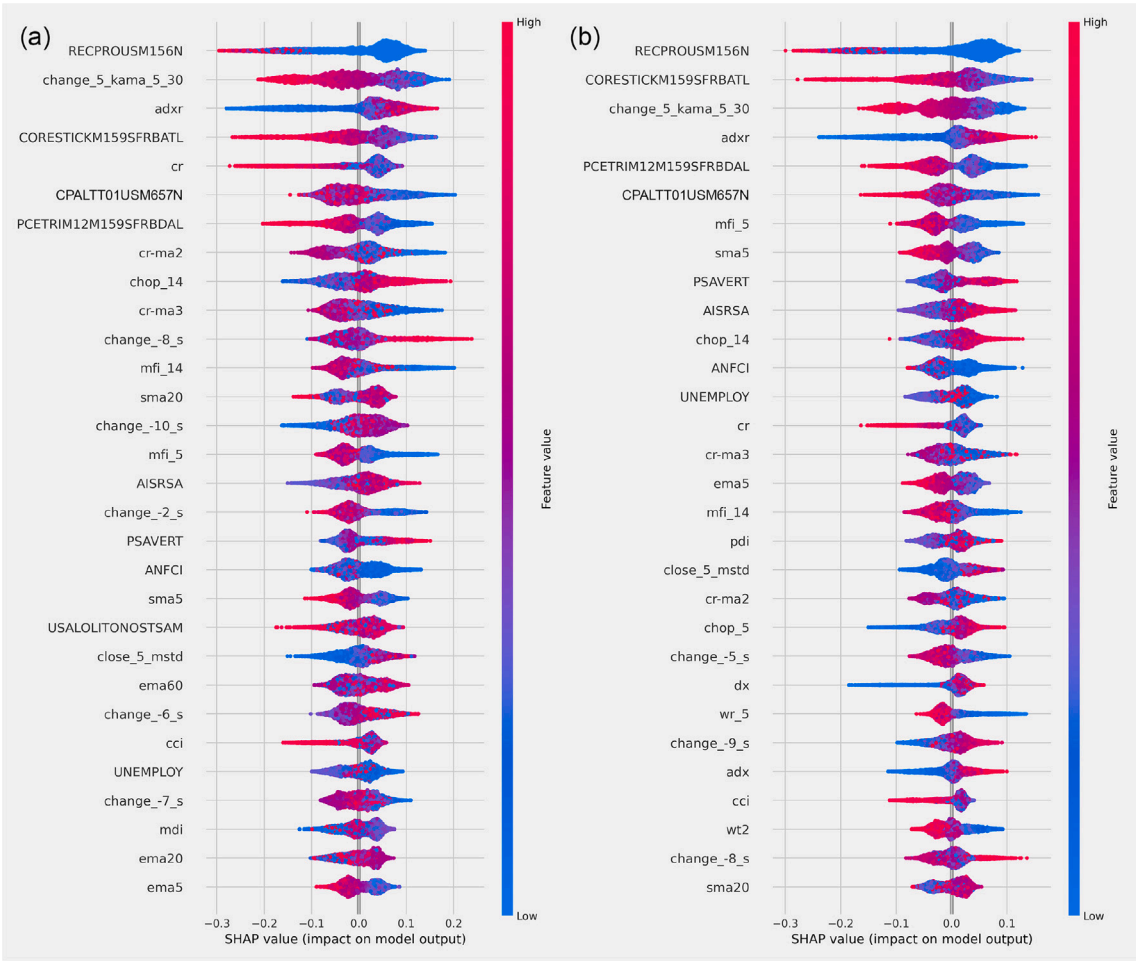

Fig. 4. Marginal contribution of top 30 features for predicting the trend of *S&P* 500 using the GBDT model. (a) and (b) indicate results from two datasets with different train-test splitting.

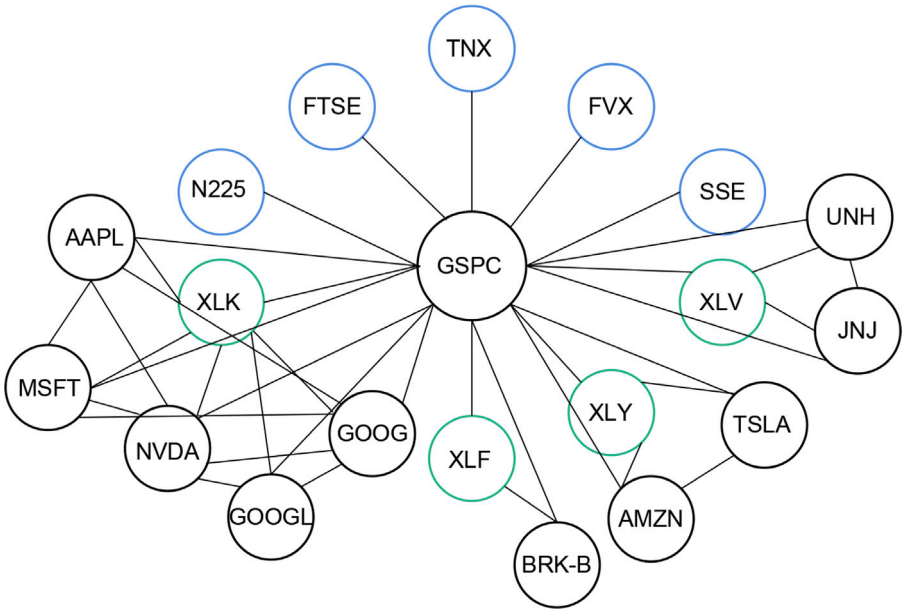

Fig. 5. Financial graph of the Top10 dataset.

Table 1
Selected features by SHAP.

ID	Explaination	Parameters
RECPROUSM156N	Smoothed U.S. Recession Probabilities	–
CORESTICKM159SFRBATL	Sticky Price Consumer Price Index less Food and Energy	–
PCETRIM12M159SFRBDAL	Trimmed Mean PCE Inflation Rate	–
CPALTT01USM657N	Consumer Price Index	–
PSAVERT	Personal saving rate	–
AISRSA	Auto Inventory/Sales Ratio	–
ANFCI	Chicago Fed Adjusted National Financial Conditions Index	–
UNEMPLOY	Unemployment Rate	–
change_5_kama_5_30	Kaufman's Adaptive Moving Average	Window = 5, fast = 5 slow = 30
adxr	Line in Directional Movement Index	–
mdi	Line in Directional Movement Index	–
cr	Energy Index	–
cr_ma2	Line in Energy Index	–
cr_ma3	Line in Energy Index	–
chop_14	Choppiness Index	Window = 14
mfi_5	Money Flow Index	Window = 5
mfi_14	Money Flow Index	Window = 14
cci	Commodity Channel Index	Window = 14
sma_5	Simple Moving Average	Window = 5
sma_20	Simple Moving Average	Window = 20
ema_5	Exponential Moving Average	Window = 5
ema_20	Exponential Moving Average	Window = 20
change_-t_s	Historical percentage change of close price shift by t	$t \in [-1, -10]$

Table 2
Training, validation, and test periods of datasets.

Stocks	Training period	Validation period	Test period
Top10	2002-01-02 to 2015-01-01	2015-01-02 to 2017-01-01	2017-01-02 to 2022-04-01
28-stocks	2002-01-02 to 2012-12-05	2012-12-06 to 2013-11-03	2013-11-04 to 2014-03-14
28-stocks	2002-01-02 to 2011-01-02	2011-01-02 to 2012-01-02	2012-01-02 to 2022-04-01

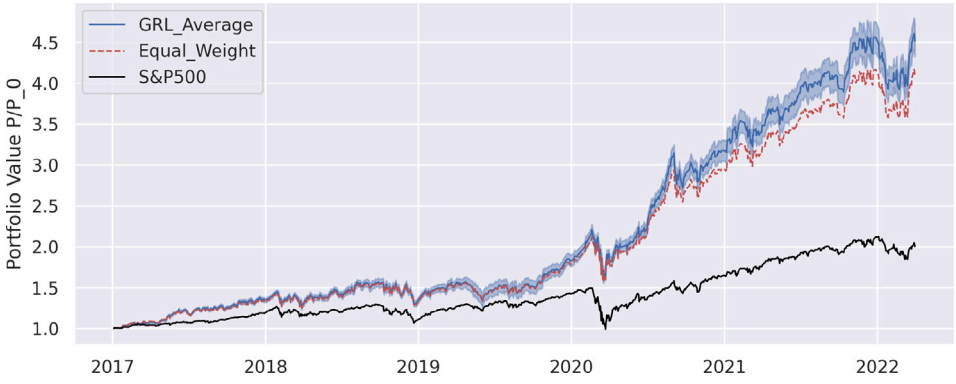

Fig. 6. Portfolio values on Top10 test dataset by GRL model (blue) and Equal Weight strategy (red), the black line indicates $S\&P$.

significantly outperform the $S\&P500$ index. Moreover, the difference in the portfolio values between the GRL Average and Equal Weight strategy gradually increased over time, which implies GRL can achieve better performance for long-term portfolio optimization.

Fig. 7 shows the portfolio values of the GRL model and Equal Weight strategy on the 28-stocks dataset proposed by Soleymani and Paquet (2021). Their test period is relatively short as they used dynamic graphs, while their portfolio is extremely profitable. In contrast, we used a static graph and focused on the long-term performance and robustness of the portfolio. During the 10-year test period, the GRL model outperforms the Equal Weight strategy and $S\&P500$ index. Moreover, like on the Top10 dataset, the difference in portfolio values between the GRL model and the Equal Weight strategy increased over time. Specifically, the advantages of GRL over other strategies became more prominent after 2018, and the advantages remained after the outbreak of COVID-19. This demonstrates that the GRL model is robust in portfolio optimization and can be used for long-term portfolio management.

Fig. 8 illustrates the distribution of the weights before and after the outbreak of the COVID-19 period. The dashed line indicates the initial portfolio weight. Generally, single stock weight is within a range of [0.015, 0.065]. Because the message passing mechanism in GraphSAGE can smooth signal and thus avoids assigning extreme weights for some stocks. Moreover, the weight changes are generally consistent which means that the GRL policy is relatively stable. In addition, during the early stage of the COVID-19 period (Yellow line), the weights of some stocks in the healthcare sector such as MRK, JNJ, and GILD, have increased. It is interesting to see that healthcare-sector stocks generally rose during that period. This implies that the GRL model could capture market trends and make profitable trading decisions.

Table 3 summarized the evaluation of portfolios by DRL model and Equal Weight strategy, and $S\&P500$. On the Top10 dataset GRL model achieved the average ROI of 351.46% over the three investment years from 2017-01-02 to 2022-04-01. Meanwhile, Equal Weight strategy and the $S\&P$ 500 only achieved the ROIs of 308.07% and 100.65%, respectively. Therefore, the GRL model seems more profitable than the

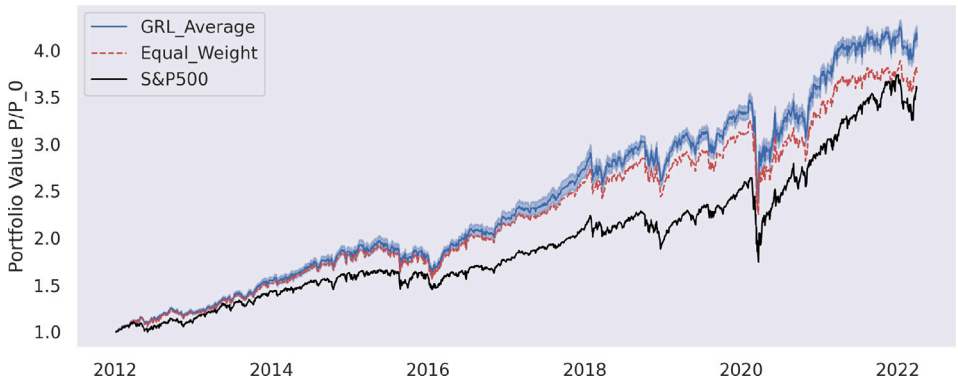

Fig. 7. Portfolio values on 28-stocks test dataset from 2012-01-02 to 2022-04-01 by GRL model (blue) and Equal Weight strategy (red), the black line indicates $S\&P$.

Fig. 8. Portfolio weights of the 28-stocks dataset.

Equal Weight strategy and the $S\&P$ 500. Generally, a Sharpe ratio below one could be suboptimal; a ratio between one and two is fair and a ratio more than two is considered excellent. Although the GRL model achieved the highest Sharpe ratio of 1.5252, it is still below the threshold of two. The main reason is that the market crisis resulting from the COVID-19 leads to significant volatility on financial markets in the United States (Albulescu, 2021).

We employ Calmar ratio and Sortino ratio to evaluate the downside of the portfolios. The Calmar ratio and Sortino ratio in the optimal trajectory of GRL reached 1.3729 and 1.8038, respectively, which is significantly higher than that in the Equal Weight strategy with 1.3375 and 1.6829, respectively, which implies the drawdown risk and negative risk in the GRL portfolio are the lowest. In addition, the average Max Drawdown in the GRL-based portfolio is slightly larger than that in the Equal Weight portfolio but significantly lower than that in $S\&P$ 500. These results suggest that the GRL model can earn higher returns more efficiently compared to the Equal Weight strategy and $S\&P$ 500.

The second dataset consists of 28 stocks from 2013-11-04 to 2014-03-14, which is identical to the second dataset in Soleymani and Paquet (2021). The DeepPocket model, proposed by Soleymani and Paquet (2021), achieved the highest ROI of 79.57% and the Sharp ratio of 3.84 with a Maxdrawn of 0.4374. Compared to DeepPocket, our model achieved a significantly lower Max Drawdown of 0.0375 on average while obtaining a relatively high Sharp ratio of 2.0802 and a reasonable Max ROI of 6.39%. In addition, during the short period, our model showed higher profitability and less risk compared with the Equal Weight strategy and $S\&P$ 500.

Moreover, we evaluated the long-term performance of the GRL portfolio over a 10-years investment period. The GRL model achieved the highest performance in all metrics compared to the Equal Weight strategy and the $S\&P$ 500. Moreover, only the GRL model has a Sharp Ratio larger than 1. However, compared to the TOP10 dataset, the performance of the GRL model on the 28-stocks portfolio reduced significantly, which highlights the impact of stock selection on the GRL model results. In practice, combining the stocks selection algorithm and the GRL model may further improve portfolio performance.

4.4. Comparative analysis

In this section, we analyze the performance of the two types of GRL architectures and a less powerful PPO model with MLP agent (MLP-DRL) proposed by *Stable Baseline3* (Raffin et al., 2021). The training, validation and test data are from the Top10 datasets in the periods: 2002-1-1 to 2015-01-01, 2015-01-02 to 2015-06-01, and 2015-06-02 to 2016-01-01.

Fig. 9 illustrates the performance of the three models on training and validation sets using TensorBoard powered by Tensorflow (Abadi et al., 2016), with the smoothing factor set to 0.7. The first row shows the changes in mean episodic training reward and validation reward. In the training process, the Share-Extractor GRL achieves the highest reward of more than 4 times the initial value. The mean reward of MLP-DRL is much lower than that of the two GRL models. In the evaluation phase, the optimal rewards of the two GRL models are over 1.32, much higher than the mean reward of MLP-DRL baseline of 1.258.

As can be seen in the second row of Fig. 9, the GRL models would result in a large Approximate KL compared to the MLP-DRL model, indicating that the divergence between the old and current policies is substantial. Since the PPO algorithm uses a clipping method to

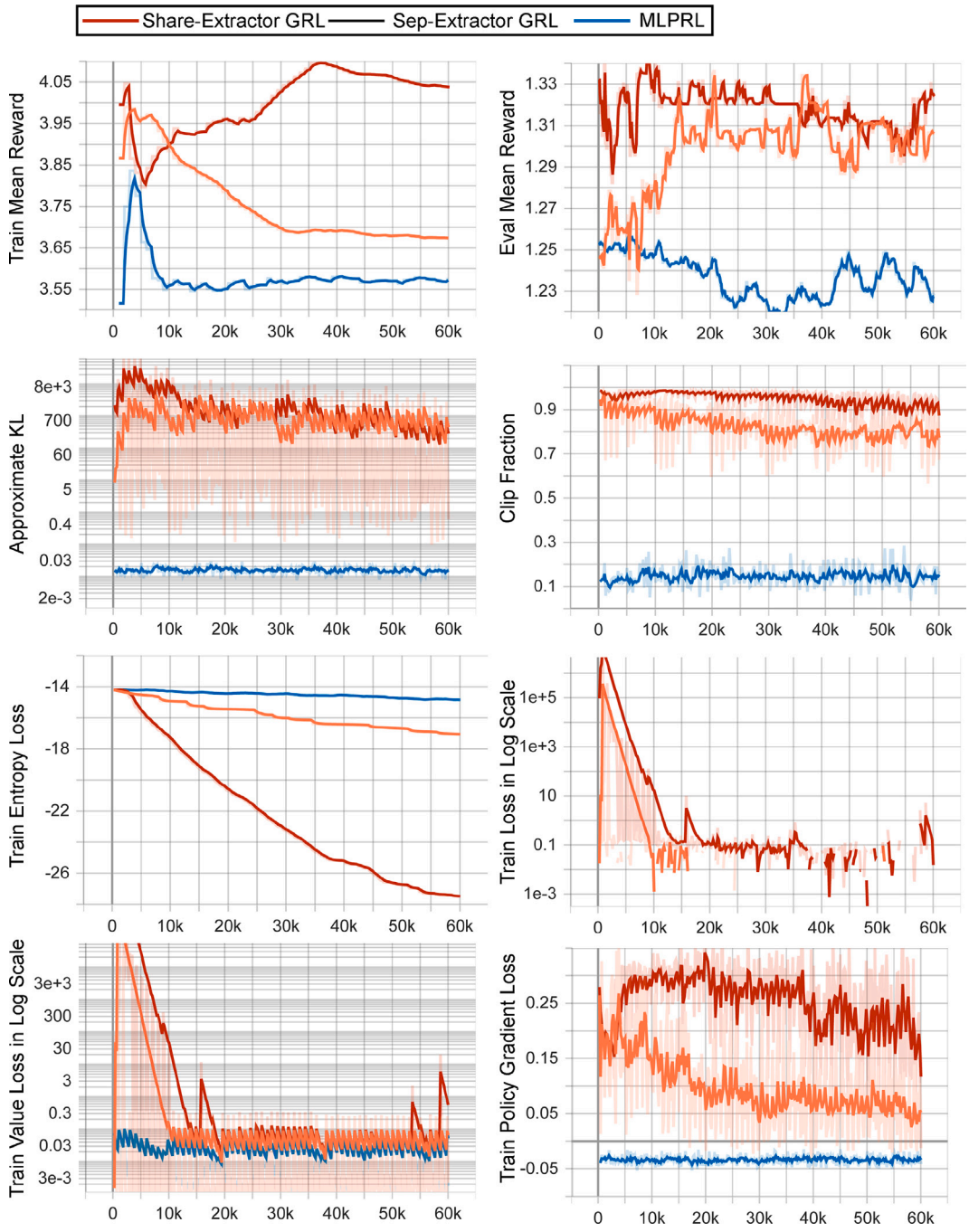

Fig. 9. Share-Extractor GRL, Separate-Extractor GRL and MLP-DRL baseline performance on the training and validation sets.

restrict the policy change to a small range, a large policy update is effectively controlled. In addition, the Approximate KL in the GRL models decreases sharply during training and the Clip Fraction values slowly decrease and fall below the critical value of 0.8. Moreover, the Approximate KL and Clip Fraction in the MLP-DRL model are tiny, but the model has not been optimized.

The last two rows in Fig. 9 show the variation of the different loss functions during model training. The Train Entropy Loss in the Share-Extractor GRL decreases sharply compared to the other models. The model MLP-DRL shows only a small Train Loss and Train Value Loss at the beginning. In contrast, for the two GRL models, the Train Loss and Train Value Loss are large initially and decrease significantly after training for 15_000 time steps. In addition, the Train Policy Gradient Loss in the GRL models are slightly larger than in the MLP-DRL baseline.

Fig. 10 shows the portfolio values during the test period generated by the optimal models Share-Extractor GRL, Separate-Extractor GRL, and PPO-MLP. The two GRL models achieve similar performance and outperform the MLP-DRL baseline. Compared to the baseline model, the excess profit generated by the GRL model increases over time. However, during the short-term decline between August and September 2015, the portfolio value of both GRL models showed significant regressions. This is because we did not include a risk control objective in the reward function. This issue should be investigated in future research.

Although the MLP-DRL model uses the same SHAP-selected features as the GRL models, its profitability dropped significantly (about 8%) after the GraphSAGE-based feature extractor was removed. This is consistent with the conclusion by Liang et al. (2018) that PPO can achieve peak performance in games or robot control, but is less effective in

Table 3
Portfolio evaluation on test datasets.

ID	Duration	ROI (%)	Sharp ratio	Calmar ratio	Max drawdown	Sortino ratio
Top10-opt	1149 d	384.07	1.5252	1.3729	0.2678	1.8038
Top10-avg	1149 d	351.46	1.4749	1.3528	0.2671	1.7573
Top10-Equal Weight	1149 d	308.07	1.4256	1.3375	0.2516	1.6829
S&P500	1149 d	100.65	0.8468	0.5082	0.3392	0.9275
28-stocks-opt	90 d	6.39	2.0802	5.4410	0.0387	3.0135
28-stocks-avg	90 d	5.90	1.9723	5.1874	0.0375	2.8525
DeepPocket	90 d	79.57	3.8400	–	0.4374	–
28-stocks-Equal-W	90 d	4.93	1.6925	4.2402	0.0387	2.4416
S&P500	90 d	4.43	1.5526	3.9467	0.0401	2.1769
28-stocks-opt	2234 d	425.05	1.0489	0.6086	0.2875	1.2589
28-stocks-avg	2234 d	411.83	1.0294	0.5832	0.2990	1.2251
28-stocks-Equal-W	2234 d	375.98	0.9648	0.5374	0.3055	1.1426
S&P500	2234 d	359.26	0.9182	0.4721	0.3392	1.0619

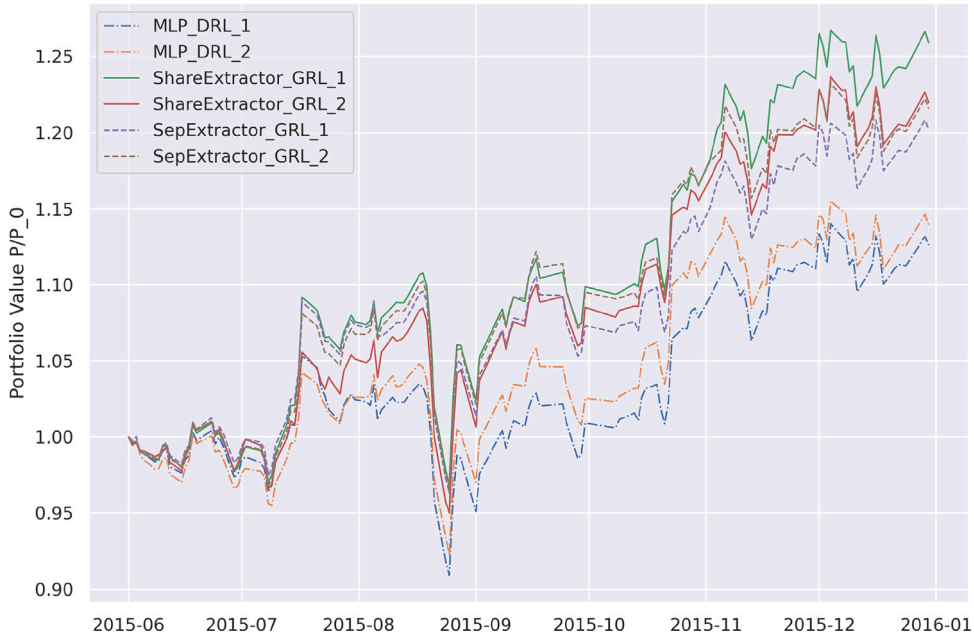

Fig. 10. Portfolio performance on the Top-10 test dataset by Share-Extractor GRL, Separate-Extractor GRL and MLP-DRL baseline models.

portfolio optimization. GRL models use the GraphSAGE-based feature extractor to capture spatial dependencies of different assets, and then use the MLP to learn an optimal strategy that increases the agent's profitability.

Moreover, compared to the MLP-DRL model, GRL models usually require more training time steps to reduce the training loss due to the higher model complexity. When we compare the Share-Extractor GRL with the Separate-Extractor GRL, we find that the former is easier to train and can produce strategies with high returns.

4.5. Discussion

The above experiments show the feasibility of the proposed portfolio optimization method. By comparing with different baseline models, we demonstrate that the proposed model improves the performance of traditional portfolio optimization models in the following ways:

- **Feature Engineering** Understanding why a model provides certain planning is at least as important as its accuracy (Benhamou et al., 2020). We use SHAP model to select a few but important features as model inputs to improve the training efficiency and performance of GRL.
- **Static Financial Graph** Recently, a dynamic graph encoding the temporal relationships of stocks has been proposed and successfully applied to stock ranking tasks (Feng et al., 2019; Saha, Gao, & Gerlach, 2021). However, the performance of GNN can be greatly reduced by introducing noise edges (Geisler et al., 2021). Therefore, we believe that it is important and practical to construct a static graph based on market indexes, industry indexes, and stock classifications and their relationships.
- **GraphSAGE Feature Extractor** The Equal-weight strategy proves to be more robust (Malladi & Fabozzi, 2017) compared to other price or value weighted strategies. Nevertheless, the experimental results of different portfolios suggest that the GRL models outperform the Equal-weight strategy. Moreover, the comparative analysis suggests that a PPO agent with a GraphSAGE-based feature extractor significantly outperforms the PPO baseline without an extractor. Therefore, it is hypothesized that using GraphSAGE-based feature extractor is the main reason for improving the performance and robustness of the PPO agent on portfolio optimization. The GraphSAGE layers use a mean aggregator to combine the feature vectors of neighboring nodes, i.e., the market index, the related industry index, or similar stocks. In this way, the stability of the policy can be ensured when the state of a particular stock suddenly changes. Therefore, the GRL models provide long-term stable profitability.
- **Model Choice** We offer two types of GRL variants: the Share-Extractor GRL and the Separate-Extractor GRL. In general, the two models achieve similar performance on the datasets under

consideration. However, the Share-Extractor GRL is easier to train since it has fewer training parameters. In practice, we suggest performing hyperparameter optimization for both models and selecting the most appropriate model based on its performance on the validation set.

5. Conclusion

In this work, we developed the GRL model by combining GraphSAGE and PPO for long-term portfolio optimization. A static financial graph is defined where the nodes represent stocks, bonds, or market indexes and the edges indicate their connection. To select fewer but dominant features from a set of indicators containing the price, technical and economic information, we employed the SHAP model to rank the features based on their importance and keep the top 32 features. A GraphSAGE-based features extractor is designed to obtain high-level node embeddings based on the financial graph. The embedded vector is sent to the refined PPO agent to learn an optimal policy for portfolio allocation.

We employ five metrics, i.e., the Return on Investment (ROI), the Sharpe ratio, the Sortino ratio, the Max Drawdown, and the Calmar ratio. The results demonstrate GRL model can earn higher returns more efficiently compared to the Equal Weight strategy and $S\&P$ 500. Moreover, the GRL model showed robustness in long-term investment. In addition, by analyzing the weight allocation by GRL during the early stage of COVID-19, we argue that GRL could identify market patterns and seek investment opportunities to gain excessive returns. A comparative analysis is performed and the results indicate that the Share-Extractor GRL and Separate-Extractor GRL significantly outperform the PPO baseline without a feature extractor. This implies that integrating a GraphSAGE-based feature extractor into the PPO agent can improve its performance and robustness in portfolio optimization tasks.

The limitations of this work are threefold. First, We adopt a full-position strategy and do not consider stop-profit or stop-loss strategies, which may increase portfolio volatility. Second, GRL consists of multiple influencing hyperparameters making hyperparameter tuning a challenging task. A more advanced hyperparameter optimization method should be investigated in future studies. Third, using ten random trajectories to analyze the uncertainty of the GRL model may not be sufficient. Investing the influence of the number of trajectories on model performance should be interesting.

CRediT authorship contribution statement

Qiguo Sun: Conceptualization, Methodology, Software, Investigation, Writing – original draft. **Xueying Wei:** Data, Review & editing. **Xibei Yang:** Review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). Tensorflow: a system for large-scale machine learning.. *vol. 16*, In *Osdi* (pp. 265–283). Savannah, GA, USA, (2016).

Albulescu, C. T. (2021). COVID-19 and the United States financial markets' volatility. *Finance Research Letters*, *38*, Article 101699.

Babaei, G., Giudici, P., & Raffinetti, E. (2022). Explainable artificial intelligence for crypto asset allocation. *Finance Research Letters*, Article 102941.

Benhamou, E., Ohana, J.-J., Saltiel, D., & Guez, B. (2021). Explainable AI (XAI) models applied to planning in financial markets.

Benhamou, E., Saltiel, D., Ungari, S., & Mukhopadhyay, A. (2020). Time your hedge with deep reinforcement learning. arXiv preprint arXiv:2009.14136.

Betancourt, C., & Chen, W.-H. (2021). Deep reinforcement learning for portfolio management of markets with a dynamic number of assets. *Expert Systems with Applications*, *164*, Article 114002.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). Openai gym. arXiv:arXiv:1606.01540.

Chen, W., Jiang, M., Zhang, W.-G., & Chen, Z. (2021). A novel graph convolutional feature based convolutional neural network for stock trend prediction. *Information Sciences*, *556*, 67–94.

Chen, Y., Wei, Z., & Huang, X. (2018). Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM international conference on information and knowledge management* (pp. 1655–1658).

Feng, F., He, X., Wang, X., Luo, C., Liu, Y., & Chua, T.-S. (2019). Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)*, *37*(2), 1–30.

Geisler, S., Schmidt, T., Şirin, H., Zügner, D., Bojchevski, A., & Günnemann, S. (2021). Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems*, *34*, 7637–7649.

Ghosh, I., & Sanyal, M. K. (2021). Introspecting predictability of market fear in Indian context during COVID-19 pandemic: An integrated approach of applied predictive modelling and explainable AI. *International Journal of Information Management Data Insights*, *1*(2), Article 100039.

Grinold, R. C., & Kahn, R. N. (2000). Active portfolio management.

Guan, M., & Liu, X.-Y. (2021). Explainable deep reinforcement learning for portfolio management: an empirical approach. In *Proceedings of the second ACM international conference on AI in finance* (pp. 1–9).

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, *30*.

Heaton, J. B., Polson, N. G., & Witte, J. H. (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, *33*(1), 3–12.

Iwasaki, H., Chen, Y., Du, Q., & Tu, J. (2018). Topic tones of analyst reports and stock returns: A deep learning approach. Available at SSRN 3228485.

Jeong, G., & Kim, H. Y. (2019). Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, *117*, 125–138.

Jiang, Z., & Liang, J. (2017). Cryptocurrency portfolio management with deep reinforcement learning. In *2017 intelligent systems conference (IntelliSys)* (pp. 905–913). IEEE.

Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. arXiv preprint arXiv:1706.10059.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, *259*(2), 689–702.

Kuo, C.-H., Chen, C.-T., Lin, S.-J., & Huang, S.-H. (2021). Improving generalization in reinforcement learning–based trading by using a generative adversarial market model. *IEEE Access*, *9*, 50738–50754.

Lee, S. I., & Yoo, S. J. (2020). Threshold-based portfolio: the role of the threshold and its applications. *The Journal of Supercomputing*, *76*(10), 8040–8057.

Liang, Z., Chen, H., Zhu, J., Jiang, K., & Li, Y. (2018). Adversarial deep reinforcement learning in portfolio management. arXiv preprint arXiv:1808.09940.

Liu, X.-Y., Xia, Z., Rui, J., Gao, J., Yang, H., Zhu, M., et al. (2022). Finrl-meta: Market environments and benchmarks for data-driven financial reinforcement learning. *Advances in Neural Information Processing Systems*, *35*, 1835–1849.

Liu, X.-Y., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., et al. (2020). Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance. arXiv preprint arXiv:2011.09607.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*.

Malladi, R., & Fabozzi, F. J. (2017). Equal-weighted strategy: Why it outperforms value-weighted strategies? theory and evidence. *Journal of Asset Management*, *18*, 188–208.

Markowitz, H. M. (1991). Foundations of portfolio theory. *The Journal of Finance*, *46*(2), 469–477.

Matsunaga, D., Suzumura, T., & Takahashi, T. (2019). Exploring graph neural networks for stock market predictions with rolling window analysis. arXiv preprint arXiv:1909.10660.

Mosavi, A., Faghan, Y., Ghamisi, P., Duan, P., Ardabili, S. F., Salwana, E., et al. (2020). Comprehensive review of deep reinforcement learning methods and applications in economics. *Mathematics*, *8*(10), 1640.

Ohana, J.-J., Ohana, S., Benhamou, E., Saltiel, D., & Guez, B. (2021). Explainable ai models of stock crashes: A machine-learning explanation of the covid march 2020 equity meltdown. *Université Paris-Dauphine Research Paper*, (3809308).

Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). Deep learning for financial applications: A survey. *Applied Soft Computing*, *93*, Article 106384.

Pricope, T.-V. (2021). Deep reinforcement learning in quantitative algorithmic trading: A review. arXiv preprint arXiv:2106.00123.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1–8, URL http://jmlr.org/papers/v22/20-1364.html.

Saha, S., Gao, J., & Gerlach, R. (2021). Stock ranking prediction using list-wise approach and node embedding technique. *IEEE Access*, *9*, 88981–88996.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Si, W., Li, J., Ding, P., & Rao, R. (2017). A multi-objective deep reinforcement learning approach for stock index future's intraday trading. *vol. 2*, In *2017 10th international symposium on computational intelligence and design (ISCID)* (pp. 431–436). IEEE.

Soleymani, F., & Paquet, E. (2021). Deep graph convolutional reinforcement learning for financial portfolio management–DeepPocket. *Expert Systems with Applications*, *182*, Article 115127.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *Stat*, *1050*, 20.

Wang, J., Zhang, S., Xiao, Y., & Song, R. (2021). A review on graph neural network methods in financial applications. arXiv preprint arXiv:2111.15367.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks? arXiv preprint arXiv:1810.00826.