FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF
HIGHER EDUCATION
«NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS»

Faculty of Computer Science

Babynin Andrey Igorevich

Graph Neural Networks for Stock Portfolio Optimization

Master thesis

For the degree 01.04.02 Applied Mathematics and Computer Science

Educational program: "Machine Learning and High-Load Systems"

Reviewer

Abroskin Ilya Dmitrievich

Data Scientist, SBER

Academic advisor

Maksimovskaya Anastasia Maksimovna

Senior Product Analyst, Elsevier

Moscow, 2023

TABLE OF CONTENT

Abstract

This research focuses on addressing portfolio optimization problem by leveraging the capabilities of graph neural networks (GNNs) for predicting future returns. The study adopts an end-to-end optimization approach, combining the predictive power of GNNs with the optimization layer to enhance portfolio allocation decisions. In order to improve the robustness of the algorithm, the use of Prioritized Experience Replay (PER) is proposed.

The evaluation of the approach is conducted on a dataset comprising major US Exchange-Traded Funds (ETF), from different classes of assets. The performance of the model is assessed using financial metrics, including the Information ratio, portfolio returns, etc.

The results highlight the usefulness of the GNN-based approach in capturing the complex relationships among assets. The incorporation of PER proved useful in enhancing the model's ability to learn from outlier cases and optimize portfolio allocations under short periods of learning.

Overall, the research suggests that GNNs can be a useful analytical tool in analyzing stock performance interconnectedness. The research opens avenues for further exploration of GNNs in the context of portfolio optimization problem.

Аннотация

Данное исследование посвящено решению проблемы оптимизации инвестиционного портфеля путем использования графовых нейронных сетей (Graph Neural Networks, GNN) для прогнозирования будущих доходностей. В работе применяется комплексный подход к оптимизации, объединяющий предсказательную силу GNN с оптимизационным слоем для улучшения принятия решений о распределении весов портфеля. Для повышения устойчивости алгоритма предлагается использование метода Prioritized Experience Replay.

Оценка предложенного подхода проводится на наборе данных, включающем в себя основные биржевые фонды США (Exchange-Traded Funds, ETF) из различных классов активов. Оценка производится с использованием финансовых метрик, включая Information ratio, накопленную доходность и другие.

Результаты исследования подчеркивают полезность подхода, основанного на GNN, для улавливания сложных взаимосвязей между активами. Использование PER представляется положительным для способности модели обучаться на аномальных случаях и оптимизировать распределение портфеля в короткие периоды обучения.

В целом, исследование предлагает использование GNN в качестве полезного аналитического инструмента для анализа взаимосвязей финансовых активов. Данное исследование открывает возможности для дальнейшего изучения GNN в контексте проблемы оптимизации инвестиционного портфеля.

Key words: *end-to-end learning, machine learning, portfolio optimization, graph neural networks, attention in neural networks, prioritized experience replay*

CHAPTER 1. INTRODUCTION

Section 1.1 Description of the field

Stock market analysis has long been centered around two key elements: stock forecasting and stock portfolio allocation. The ability to accurately predict stock movements simplifies the task of portfolio allocation, while allocation decisions become more challenging under uncertain future market conditions. Traditional approaches have tended to focus solely on one aspect, limiting the breadth of the problem. However, this study takes an alternative approach by adopting the end-to-end optimization framework, which integrates prediction and optimization tasks for more effective stock portfolio optimization.

The usefulness of end-to-end optimization firstly proposed by Donti in the paper "Task-based end-to-end model learning in stochastic optimization" (Donti, et al., 2019) is that end-to-end optimization allows to train models on the same criteria (e.g., Information Ratio) that used to evaluate them, thus overcoming the problem of setting intermediate goals or finding the best approximations to them. The development of machine learning methods has allowed to create robust frameworks utilizing stochastic optimization to achieve this task.

Section 1.2 Purpose of the study

The purpose of this study is to develop a novel approach to portfolio optimization utilizing end-to-end optimization approach with GNN. The developed model is applied to a set of ETF tickers representing different asset classes. GNN model serves as a prediction layer, while cvxpylayer serves as on optimization layer. The model is trained to optimize Information Ratio. For the task of improving the stability of training and prediction a use Prioritized Experience Replay buffer is proposed and examined. At the end, a comparison is made with traditional and wide-known methods of portfolio optimization.

Section 1.3 Structure of the study

The study is structured into several sections. The first chapter discusses current state of research in this area by overviewing existing literature, situating the current study within the existing body of knowledge. The second chapter details the methodology behind the end-to-end optimization approach, including an explanation of Prioritized Experience Replay and its relevance to the study. Furthermore, the utilization of Relational Graph Attention Networks within the prediction layers is outlined.

Moving forward, the third chapter provides an overview of the model architecture, highlighting the structural components of both the prediction and optimization layers. In the fourth chapter, emphasis is placed on data collection, feature generation, the training process, and the conducted experiments. A comparative analysis of the experimental results is also presented in this chapter.

Finally, in the fifth chapter, the study concludes by summarizing the key findings derived from the research and discussing potential avenues for future work in the field of stock portfolio optimization.

CHAPTER 2. LITERATURE REVIEW

Believe me, no. I thank my fortune for it—
My ventures are not in one bottom trusted,
Nor to one place, nor is my whole estate
Upon the fortune of this present year.
Therefore my merchandise makes me not sad.

Merchant of Venice, Act I, Scene 1

The question of wealth allocation has captivated the minds of people since time immemorial, predating the advent of stock exchanges. However, the academic significance of this question can be attributed to the pioneering work of Harry Markowitz (Markowitz, 1952), who laid the foundation of modern portfolio theory. Although the assumptions of this famous approach can be traced to 1940 to a work of Italian mathematician Bruno de Finetti (Pressacco & Serafini, 2007). All in all, mean variance optimization seems to be around almost a century in place without losing its appeal. Mean-variance approach relied on the assumption of investor rationality and employed mean-variance optimization to determine portfolio weights. Subsequently, the development of the Black-Litterman model (Black & Litterman, 1992) relaxed the assumption of homogeneous beliefs among investors, further advancing the field of portfolio optimization.

Concurrently, researchers also directed their attention towards the challenge of stock forecasting, exploring two primary approaches: fundamental analysis and technical analysis. Fundamental analysis, exemplified by the influential three-factor model (Eugene F. Fama, 1993), which seeks to establish a relationship between fundamental factors and stock performance. The further development in the field helped to establish a whole new research area of finding factors that correlate with stocks performance. As of today, more than 400 factors have been covered in academia and believed to add to the explainability of stock movements[1]. In the current investment landscape, there is a notable surge in *thematic* and *style* investing, where stocks are categorized into baskets based on specific social, technological, economic, and geopolitical criteria. This approach aims to replace traditional factor analysis and capture investment opportunities aligned with specific themes, such as *"ESG investing"* or *"AI & Robotics"*. It is important to acknowledge that while these thematic approaches have gained significant popularity, their foundation in academia is relatively limited. The widespread adoption of these themes can be attributed, to a certain extent, to the marketing

[1] https://www.man.com/maninstitute/factors

efforts of prominent ETF providers, exemplified by the activities of Vanguard (Bogle, 2017).

On the other hand, technical analysis investigates patterns and trends in historical price and volume data. Numerous studies have investigated the efficacy of technical indicators in stock forecasting. For instance, (Vargas, et al., 2018) combined news texts and technical indicators as inputs to an LSTM model, while (Huang, et al., 2021) utilized 22 years of S&P companies' quarterly financial data in a Feed-Forward Neural Network. Before neural networks, several attempts were made to predict stock returns using classic machine learning methods such as Support Vector Machines and Random Forests (Mokhtari, et al., 2018).

The emergence of graph neural networks has garnered attention in the realm of stock forecasting. In a notable attempt to predict stock prices (Chen & Wei, 2018) employed a graph representation of shareholders' ownership. Researchers have encoded individual stock features using Recurrent Neural Networks (RNNs) and leveraged the representations of neighboring nodes to forecast stock price performance using Convolutional Graph Networks (CNNs). Another study (Matsunaga, et al., 2019) expanded the universe of relations beyond shareholder types, incorporating "supplier," "customer," and "partner" relations into graph, thus enriching the notion of connectedness.

These diverse approaches and methodologies highlight the multi-faceted nature of stock forecasting and wealth allocation. The fusion of portfolio optimization techniques, fundamental analysis, technical analysis, and the burgeoning field of graph neural networks showcases the ongoing quest for improved accuracy in forecasting stock performance.

CHAPTER 3. METHODOLOGY

In this chapter I am going to outline main assumptions of an end-to-end optimization process. Following the previous work on this theme (Costa & Iyengar, 2022) I use similar approach to portfolio optimization. The approach proposed in the mentioned paper is based on the combination of two layers: prediction and optimization ones into a single model. Each layer has its own metric to optimize: prediction – nominal loss function, optimization – task loss function. The latter one is used to optimize financial return, while the former one to access model's prediction quality.

The nominal loss function requires both the point prediction and the past prediction errors as inputs to quantify and control model risk. I calculate a nominal loss function, that is composed of Mean Squared errors of past predictions minus possible portfolio future return. At this stage, I got weights as outputs of the optimization layer that are later used in task loss problem. Task loss is the ultimate goal to optimize. In this work I restrain to maximizing Information ratio.

As a mathematical problem, I try to optimize the following:

- $x \in X$ represents features variables.
- $y \in Y$ represents target variables,
- $z \in Z$ represent weights that are chosen during the optimization process.
- r $\in$ R represents realized returns used to indicate.
-

The goal is to find weights θ of a neural network that helps to minimize the task loss function $f(x, r, z^*(x, \theta))$ subject to a nominal cost function $c(x, z)$:

$$\min_{\theta}(f(x, r, z^*(x, \theta)) \\ z^* = argmin(c(x, z)) \quad (1)$$

Nominal loss function:

$$\epsilon_i = r_i - \hat{y}_i$$

$$c(x, z) = \frac{1}{N}(\sum_{i=1}^{N} \epsilon_i)^2 - \gamma * z_t * \hat{y}_t \quad (2)$$

Here, N represents the size of the past prediction sample, and s denotes a prediction window. The prediction window signifies the duration over which return is predicted. The rationale behind employing a longer prediction window is to make predictions less stochastic: over longer time horizon stocks exhibit more stable and

predictable behavior gravitating towards its expected returns. The mechanic of the algorithm is depicted in Figure 1.

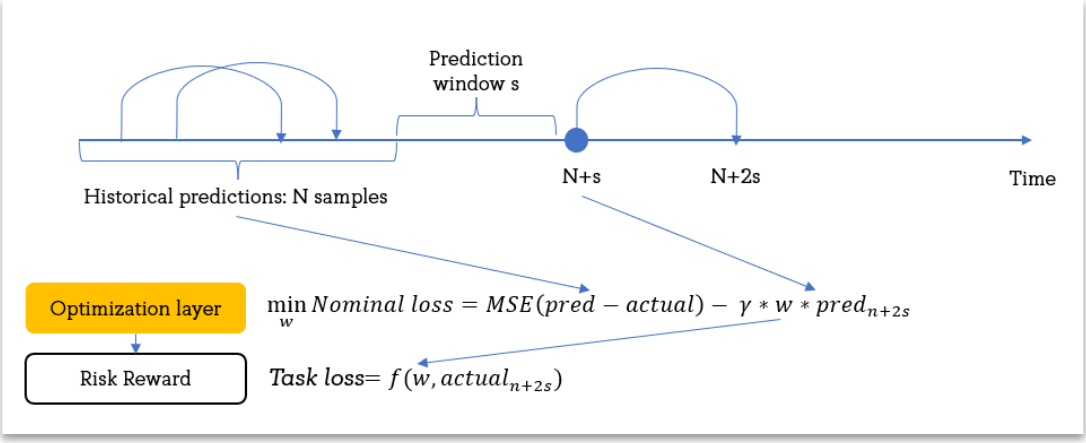

Figure 1. Schema of relationship between *nominal* and *task losses*

Section 3.1 Prioritized experience replay

Similar to the approach adopted in the study by Giorgio Costa (Costa & Iyengar, 2022), I incorporated a set of previous predictions to account for model risk. However, instead of assigning weights to these predictions based on their distribution and deviation from the expected mean, I opted to employ the concept of Prioritized Experience Replay (PER), which has gained popularity in the field of Reinforcement Learning (RL) (Schaul, et al., 2016). The fundamental principle behind PER is to train an RL agent using the specific cases where an agent leans on samples where its results were significantly different from the expected. These samples usually indicate that the model was *"surprised"* and needs further training on these particular set of observations. By storing these *"exceptional"* cases in a buffer and retrieving them during training with some predefined probability is the basic mechanic behind PER. In a broader context, PER addresses three key aspects of agent learning:

1. Efficiency: By prioritizing crucial and informative experiences, the agent can enhance its learning effectiveness and potentially reduce the overall number of experiences required for training.
2. Speed: Focusing on the most valuable experiences enables the agent to learn at an accelerated pace.
3. Performance: Empirical findings have demonstrated that agents employing PER often achieve superior performance across a range of tasks.

In my study, I have chosen to employ PER to store the indices of time periods when the losses incurred by an algorithm were the biggest, indicating notable challenges in portfolio optimization. By employing this approach, I anticipate mitigating the need to model the distribution of portfolio returns, which may exhibit instability over time. Moreover, this approach does not introduce any additional learnable parameters, thus maintaining a simple model structure and reducing the susceptibility to overfitting.

Section 3.2 Relation Graph Attention Networks

In contrast to previous approaches described in the literature, I opted to utilize Graph Neural Networks (GNN) for the prediction layer. GNNs are a specialized type of neural network explicitly designed to process and analyze data structured as graphs. This approach has garnered significant attention in recent years owing to its capability to effectively model relationships in irregular data. Unlike Convolutional Neural Networks (CNNs), which are predominantly suited for Euclidean data, GNNs excel in handling non-Euclidean data, such as social networks, molecular structures, citation networks, recommendation systems, etc.

History of GNN is nearly as long as the history of other types of neural networks. One of the beginning papers was a paper by Franco Scarselli(Scarselli, et al., 2009) that introduced graph processing, state transition and output functions that determine how the information flows through the graph, and how next level embeddings of nodes are obtained. The next step was the emergence of Convolutional GNN that took place in 2016 and was described in the paper by Thomas N. Kipf and Max Welling (Thomas & Welling, 2016). Convolutional GNN helped to make neighborhood calculations more efficient and scalable thus leading to improved performance and flexibility. Later on, in 2018 graph models with attention mechanisms (Graph Attention Networks, GAT) were introduced (Veličković, et al., 2018). The attention mechanism provided a greater level of interpretability as attention weights help to understand the relative importance of the neighbors' embeddings. As in other sphere the introduction of attention mechanism helped to boost neural networks' performances. In some sense Convolutional GNN can be considered to be a special type of GAT where attention mechanism fully determined by graph structure itself without node features. Later on, Relational Graph Attention Networks (R-GAT) (Busbridge, et al., 2019) emerged as the next stage in the development of GAT. R-GATs introduced the concept of multiple relations within a graph, enabling more comprehensive modeling capabilities. By incorporating attention mechanisms across relations, R-GATs excel in capturing interactions among the nodes at multiple levels.

To better explain the mechanic of complex R-GAT networks, I will overview the basic steps identical in all GNN. Essentially, building GNN consists of 3 steps: Graph Representation, Feature Extraction and Message Passing. Graph Representation is the first step that requires to define what are nodes and edges in a graph. For example, in a social network graph, individuals could be nodes, their relationships could be edges, and attributes could be their interests or activities. Secondly, once the graph is formed, information (features) is extracted from each node. These features can be stored in the form of vectors or embeddings. For example, each node can have an embedding representing it. The next step is Message Passing. This is the core of GNNs. In this step, nodes exchange information with their neighbors. This is done by aggregating features from neighboring nodes, often through a sum, mean, or max operation. This aggregated information is then passed through a neural network to update the node's features. The detailed scheme is depicted in the Figure 2.[2]

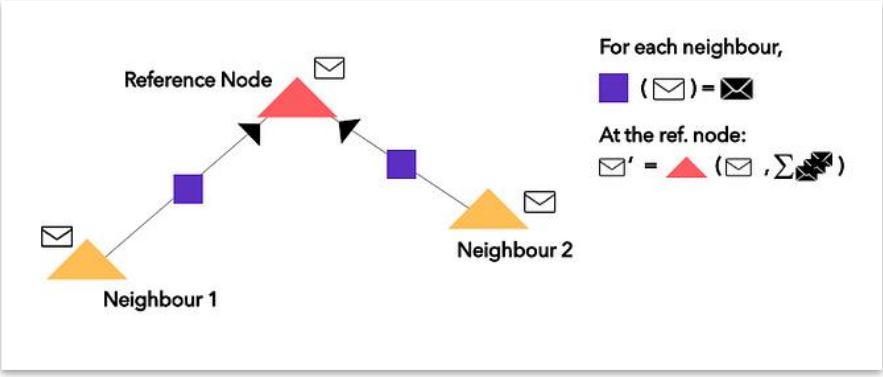

Figure 2. *The violet square is a simple feed-forward NN applied on the embeddings (white envelopes) from the neighboring nodes. The recurrent function (red triangle) applied to the current embedding (white envelope) and summation of edge neural network outputs (black envelopes) to obtain the new embedding (white envelope prime).*

Coming back to R-GAT, the input to an R-GAT is a graph, defined as a set of nodes and edges. Each node is associated with a feature vector. These vectors serve as the initial representations for nodes. The key component of R-GATs (and GAT, in general) is the attention mechanism. For each node, the model computes an attention score for each of its neighbors. This score determines how much influence the neighbors' embeddings should have on the node's new representation. In R-

[2] https://dair.ai/posts/An_Illustrated_Guide_to_Graph_Neural_Networks/

GATs, the attention score is also dependent on different levels of relations, allowing it to incorporate relational reasoning. Once the attention scores are computed, the model updates each node's embedding by taking a weighted sum of its neighbors' embeddings. Just like in the original GAT, R-GATs often use multiple attention heads. Each head learns to pay attention to different aspects of the neighbors' embeddings, and the outputs from all heads are concatenated (or summed up) to form the final updated node embedding. The scheme of R-GAT mechanic is depicted in Figure 3.

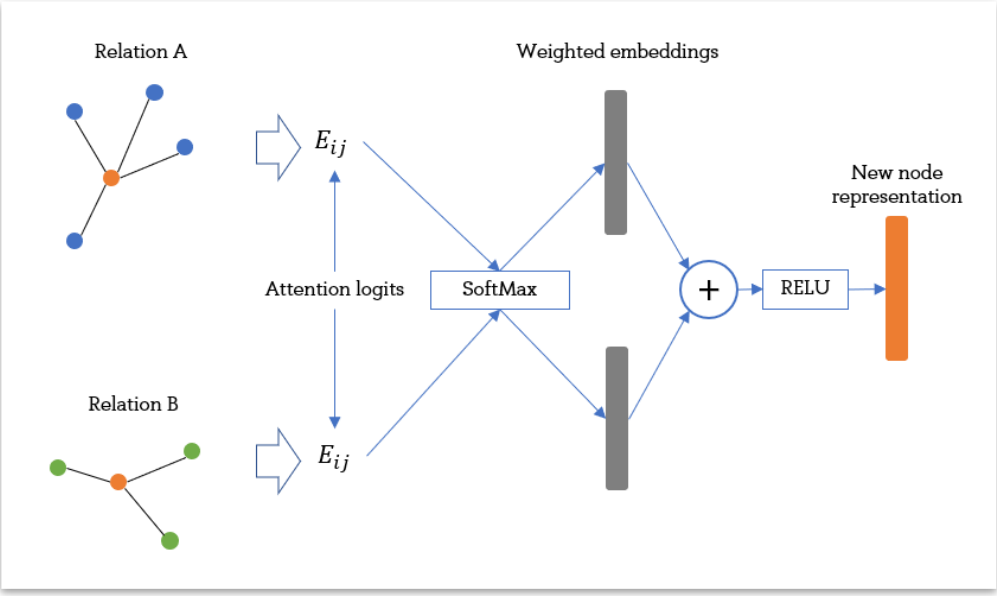

Figure 3. Schema of R-GAT message passing.

There are two ways to normalize attention: Within-Relation Graph Attention (WIRGAT) and Across-Relation Graph Attention (ARGAT). The latter averages attention logits irrespective of the relation type, while the former averages only within each relation type. In this study, I use ARGAT as a default option.

Overall, the mathematics behind R-GAT can be summarized as following:

1. Attention logits $a_{i,j}^r$ are computed for each relation type r with the help of both query Q and key kernels K:

$$q_{i,j}^r = W^r * x_i * Q^r$$
$$k_{i,j}^r = W^r * x_i * K^r \quad (3)$$

2. Then the additive attention is applied taking into account edge features e

$$a_{i,j}^r = \text{LeakyReLU}\left(q_i^r + k_i^r + W^r * e_{i,j}^r\right) \quad (4)$$

3. After that, attention weights are renormalized using *across-relation attention mechanism*:

$$a_{i,j}^{r} = \frac{\exp\left(a_{i,j}^{r}\right)}{\sum_{r}\sum_{k}\exp\left(a_{i,k}^{r}\right)} \quad (5)$$

4. Next, the new node embedding is obtained through propagation:

$$x_{i}^{(2)} = \sigma\left(\sum_{r}\sum_{j} a_{i,j}^{r} * x_{j}\right) \quad (6)$$

5. In case of multiple-head attention, individual embeddings are aggregated by concatenation or summed up.

CHAPTER 4. MODEL ARCHITECTURE

Section 4.1 End to end portfolio construction

The algorithm introduced in this study aims to leverage past prediction errors to gain insights into the local landscape of time series data. To achieve this objective, the algorithm follows a following process. First, N samples are obtained, predictions of future returns are made, and the corresponding errors are calculated. Subsequently, I take a gap of prediction window s and the algorithm predicts the desired day K based on the accumulated information. The specific steps of this algorithm are described in Algorithm 1.

Algorithm 1. Calculation of portfolio performance (without PER)

---

for *each day* $t \in \{K, K+1, \dots, T\}$ **do**
  for each sample in training sample $h \in \{t-N-s, \dots t-N\}$ **do**
    predict future return $\hat{y}_i$
    calculate the error $\epsilon_i = r_i - \hat{y}_i$
    aggregate with previous errors to the sum $\sum_i \epsilon_i$
  Predict future return for $t = K + S$
  Find optimal weights $z^* = argmin(c(x, z))$
  Use optimal weights to calculate Risk-Reward $\mathcal{R}_\theta(z)$
  Back-propagate $\theta \leftarrow (\theta - learning\ rate * \nabla_\theta \mathcal{R}$

It is important to highlight that there are two versions of this algorithm: one with the inclusion of Prioritized Experience Replay (PER) and one without. In case of PER, some samples are drawn from the buffer with the predefined probability (in my case p=0.25) In my view, this inclusion should enhance the robustness of the algorithm, as consecutive samples drawn without the intervention of PER may exhibit high similarity, potentially leading to a form of local overfitting. The model-based approach is depicted in Figure 4.

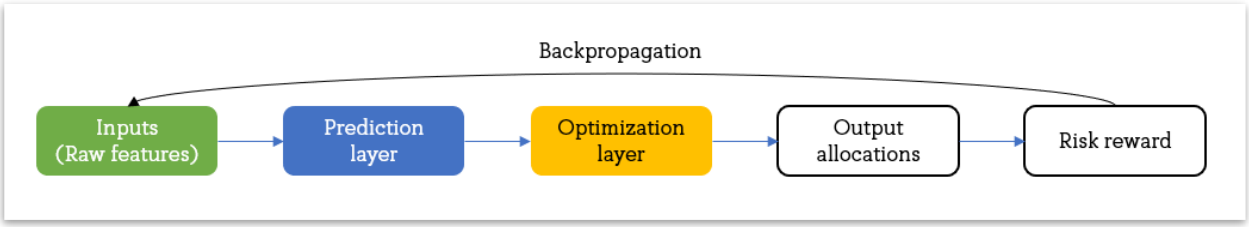

Figure 4. Computational schema of model-based approach

Section. 4.1.1 Prediction layer

Prediction layer consists of 3 layers: 2 relational graph attention layers (R-GAT) and one linear layer. RGAT layers have the same number of attention heads, defined in the hyperparameters. In the base scenario 1 head. Number of heads is stated per relation. They have across-relation attention mechanism meaning that attentions from all relations are weighted to achieve final numbers. Between layers I have a *BatchNorm* and *Dropout* layers serving to stabilize the weights and prevent overfitting. As a nonlinear activation function, I use *ELU*. The detailed scheme of a prediction layer is described in Figure 6.

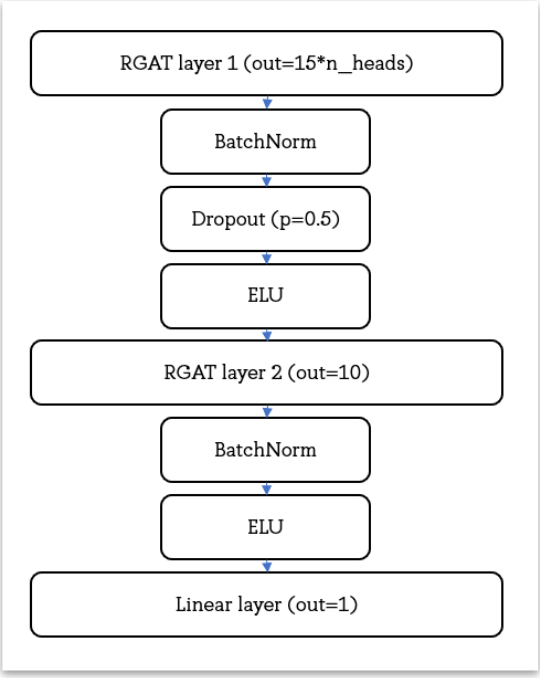

Figure 6. Scheme of prediction layer

Section 4.1.2 Optimization layer

The Python package *CvxpyLayer* (Agrawal, et al., 2019) is a seamlessly integrated tool within the widely used deep learning package *Pytorch*. By incorporating this package, users can construct a computational graph with efficient back-propagation capabilities. *CvxpyLayer* builds upon the foundation of the convex optimization package *cvxpy*, enabling the solution of optimization programs where the output of a layer depends on the solution of the previous layer.

Utilizing a convex optimization layer in a neural network can enhance interpretability, as it explicates the relationship between consecutive layers through the optimization program. When the convex optimization problem has a unique solution, the convex optimization layer functions similarly to a layer with a

deterministic functional relationship. However, in cases where an analytical solution is either nonexistent or impractical, the convex optimization layer offers an elegant approach to encode such relationships.

In the context of the end-to-end model-based portfolio, I employ the convex optimization layer within the network to optimize nominal loss problem by optimizing portfolio weights.

CHAPTER 5. EXPERIMENTS ON MARKET DATA

Section 5.1 Data collection

The decision to utilize ETFs as the target universe instead of individual stocks is based on two primary reasons. Firstly, it addresses the concern of survivorship bias prevalent in stock selection. The survivorship bias is omnipresent across financial markets, and can be found in stocks and mutual funds' performance (Elton, et al., 1995). Survivorship bias stand for the fact that the only financial entities present as of today are the *"winners"* of the past competition, meaning they have not been acquired, liquidated or merged with others. Therefore, selecting stocks solely based on their performance introduces a risk of overfitting, as only the successful companies are observable. Consequently, mitigating this bias requires additional measures, such as obtaining information on ceased tickers, which can be challenging. So, the use of ETFs in the analysis provides a solution to this issue by tracking the performance of indices and automatically adjusting their holdings to account for such cases.

The second reason pertains to the opportunity for diversification across multiple dimensions, including geography, asset classes, and investment styles, among others. Achieving a similar level of diversification with individual stocks would necessitate considering a wide range of tickers. However, due to limitations in data availability and computational costs associated with processing large graphs, this approach is not feasible. Consequently, ETFs emerge as ideal targets for experimentation due to their inherent ability to provide diversification benefits across various dimensions.

The data utilized in this study was obtained by gathering historical quotes from Yahoo Finance. A total of 20 exchange-traded funds (ETFs) were included in the analysis, with a list of these ETFs provided in Appendix A. The selection of ETFs was based on several criteria, including capitalization, asset class, geographical focus, and thematic attributes (such as sectors, dividends, or growth/value characteristics). In order to encompass a broad representation, ETFs from four major asset classes were chosen, namely Equity, Bonds, Commodities, and Real Estate. Furthermore, ETFs targeting various regions around the world, including the United States, Developed Markets, and Developing Markets, were also included, ensuring diversity in the sample.

For each ticker, detailed price information (open, close, high, low) and volume data were collected. As different ETFs were launched in different years, a subset of quotes encompassing their common intersection was utilized for the analysis. The

dataset covers a time span ranging from 25th October 2012 to 27th February 2023, providing a data of the market dynamics over the 10 years.

At Figure 7, there is a graph with cumulative performance of ETFs adjusted for dividends. It is easy to see, that two particular stocks can be considered as outliers, namely: SOXX, which tracks Semiconductor sector and QQQ which track Nasdaq index. Therefore, these two represents the technology sector with the highest valuations and prospects for growth, yet with the elevated risks as well. On the other hand, the bond tickers, namely AGG and BSV generated less impressive results in terms of return, but demonstrated a far lower volatility.

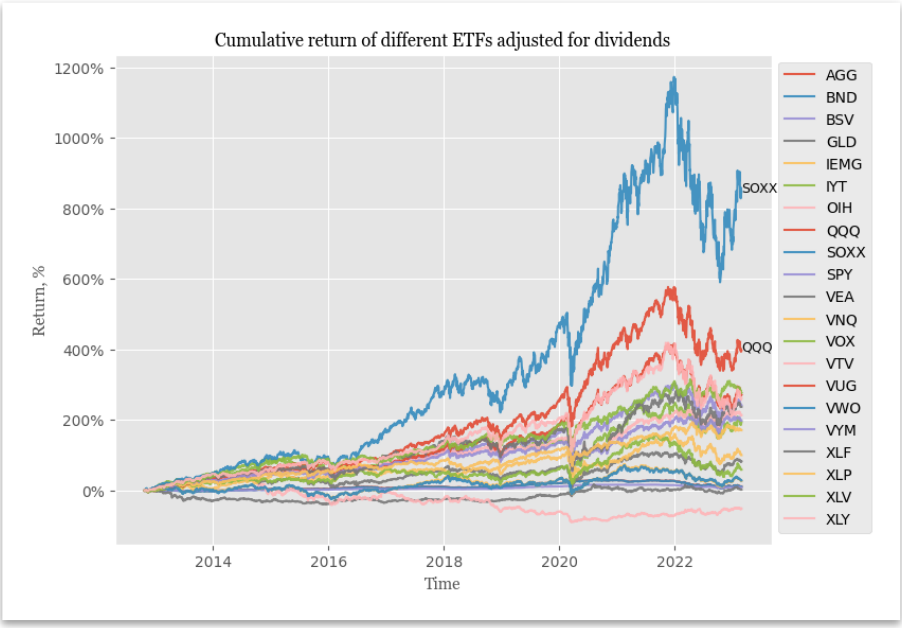

Figure 7. Cumulative performance of ETFs over the studied period

On the other hand, from a correlation table (see Figure 8), it can clearly be seen that all ETFs can be divided into 2 clusters based on their correlation, those with higher correlation are stocks and real estate, with lower or below the average correlation – bonds and commodities.

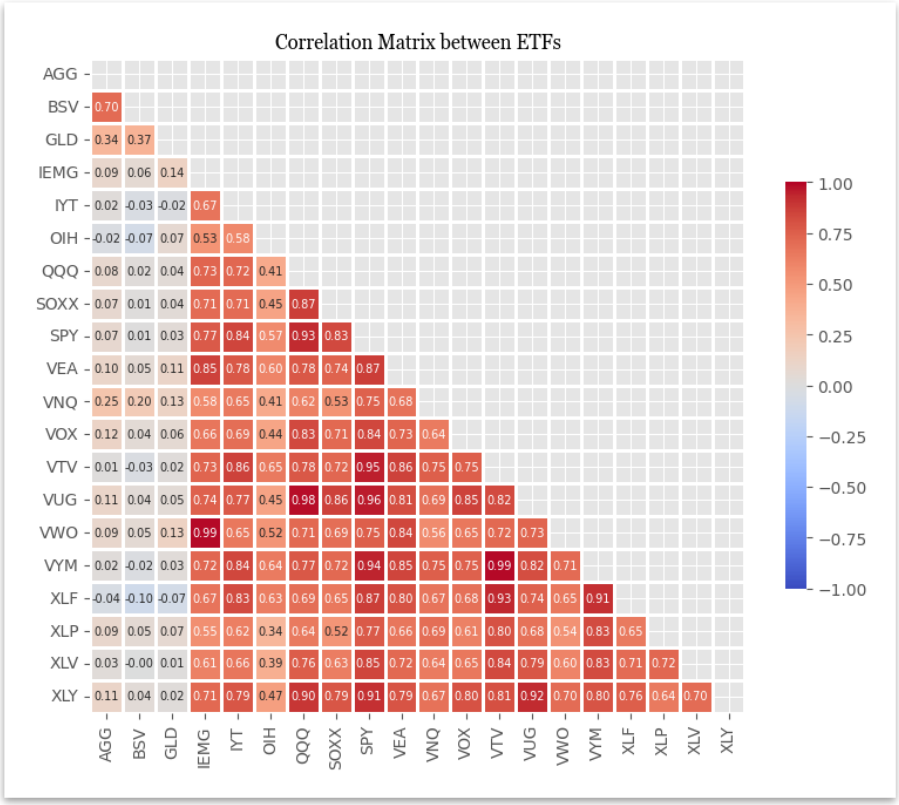

Figure 8. Correlation heatmap for ETFs

Section 5.2 Benchmark

For the benchmark I use simple average of ETFs. Since the ETF universe described in the study includes several asset classes with vastly divergent risk-return profiles: stocks, bonds, real estate, commodities, it is not relevant to measure only against specific index, for instance, S&P 500 because it is applicable only to the part of the universe that is connected to the US stock market. Because of lack of information on past tickers capitalization, I was unable to compute the weighted-counterpart of the benchmark. While, the introduction of such a benchmark could have potentially made comparison more practical in the eyes of a potential investor, this was not feasible at the time of this work being prepared. Furthermore, as it will be shown in later sections, simple average is not an easy benchmark to beat with conventional methods.

Section 5.3 Features generation

To capture relevant information from the ETF price-volume data, I derived a comprehensive set of features based on widely recognized technical indicators. These indicators included the Relative Strength Index (RSI), Moving Average Convergence-Divergence (MACD), Stochastic Oscillators, and several others. The

specific selection of indicators and their respective parameters can be found in Appendix B.

Additionally, I incorporated cumulative past returns for various time frames, namely [1, 3, 7, 14, 30, 40, 50, 60, 70, 80, 90] trading days

Overall, leveraging the "ta" Python library, I generated 12 features for each node at every time period under consideration. Therefore, the dimension of a node vector is 23 in total.

Section 5.4 Graph generation

It is common to consider modified correlation table as a prime source for adjacency matrix. Numerous adjustments have been made to make it more stable and robust. For instance, Mantegna showed that *Minimum Spanning Tree (MST)* build on inverse correlation is able to cluster stocks that mimics their industry sectors (Mantegna, 1998). Furthermore, F. Pozzi, et. al (Pozzi, et al., 2013) demonstrated that by constructing *Planar Maximally Filtered Graphs* (PMFG) out of correlation matrices and investing in stocks in the periphery of the graphs, one can achieve better diversification. They have found out that stocks at the center of the graph bear higher financial risks than those on the periphery. Moreover, in the papers by Taylor (Taylor & Cerbo, 2019), and Jaeger (Jaeger & Marinelli, 2022), it was demonstrated that PMFG can better picture market structure than the representation obtained based on correlation matrix. Following the work of Marinelli, *Gowel distance* is applied to Pearson correlation coefficient to measure the linear relationship.

$$d_{i,j}^{\rho} = \sqrt{\frac{1}{2}(1 - \rho_{i,j})} \qquad (3)$$

Then correlation is close to 1, the distance is approaching zero and then correlation is close to -1, the metric is approaching 1, that helps to alleviate the problem of different sign in correlation matrix.

Figures 9 and 10 illustrate the MST (Minimum Spanning Tree) and PMFG (Planar Maximally Filtered Graph) representations of networks constructed from 10-year data of ETF correlations. Notably, the fomer graph clearly demonstrates the distinct separation between Bond ETFs and major stock ETFs, such as SPY, which tracks the S&P 500. Additionally, the MST graph reveals the interconnectedness between VTV (Vanguard Value ETF) and VUG (Vanguard Growth ETF) through their connection to the SPY ticker. This graphical representation further emphasizes the contrasting characteristics of Value and Growth stocks as they occur on the opposite sides of SPY ticker.

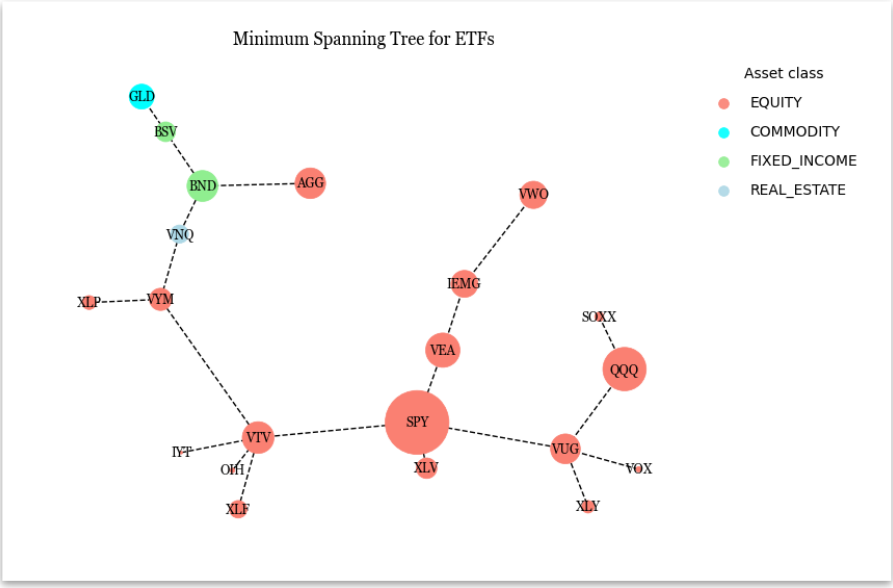

Figure 9. Minimum Spanning Tree on ETFs. *Size of bubble indicates the relative Size of ETF capitalization.*

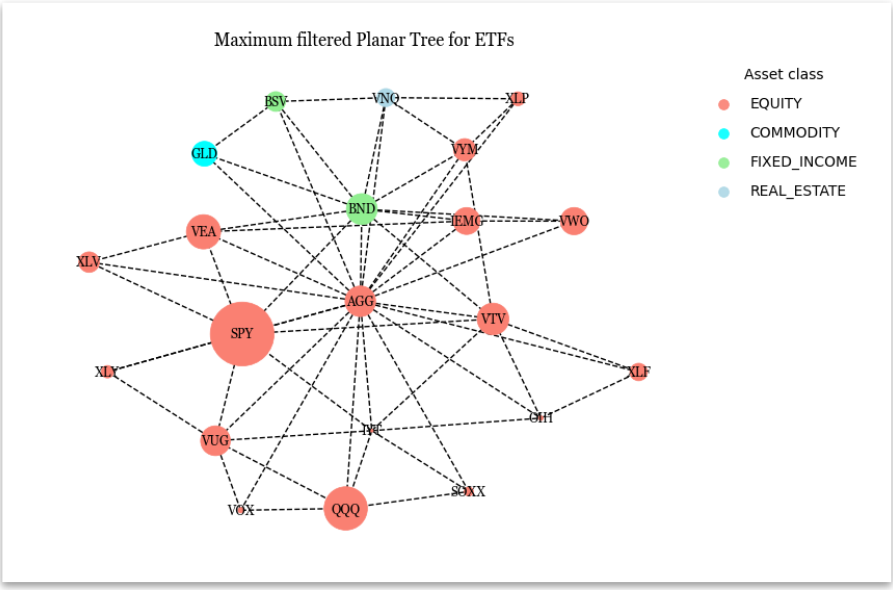

Figure 10. Planar Maximally Filtered Graphs on ETFs. *Size of bubble indicates the relative Size of ETF capitalization.*

In this work, both types of graphs and adjacency matrices are used. They are applied both to ETF return and ETF trading volume changes. I decided to apply this graph generation methods to trading volumes, because it may serve as an additional source of ETFs relatedness. It was shown in in the work by Kaizoji (Kaizoji, 2013) that

elevated trading volumes are consistent with the hypothesis of increased traders' interdependence.

In addition to four graphs already mentioned, I developed a novel graph representation based on Relative Rotation Graph (RRG). RRG was developed by Julius de Kempenaer and have been available ɯn Bloomberg terminal since 2011.[3] The graph has two axes measuring relative momentum and relative strength respectfully. There are four quadrants on the RRG chart: Leading (strong relative strength and strong momentum), Weakening (strong relative strength but weakening momentum), Lagging (weak relative strength and weak momentum) and Improving (weak relative strength but improving momentum). The idea is to cluster stocks that appears to be in one quadrant, suspecting that similar economic forces lead to over and underperformance of the ETF relative to the benchmark. The general idea behind graph extraction from RRG is depicted in Figure 11. So, at each point in time 4 separate subgraphs are generated based on quadrants where tickers occur.

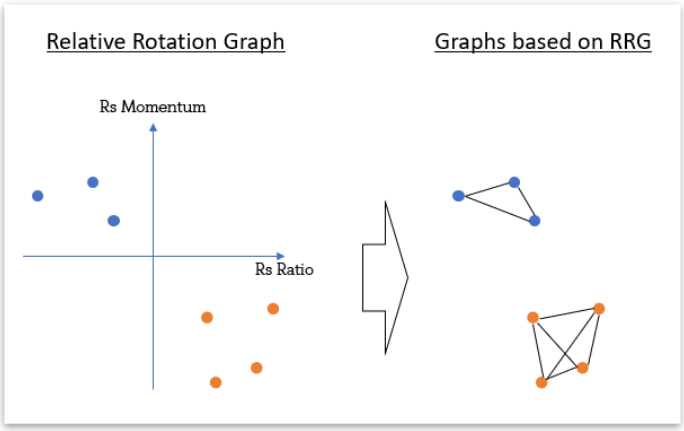

Figure 11. Scheme of graph extraction from RRG

All types of graphs are summarized in Table 1 below.

№	Factor	Type of graph
1	Daily Return	MST
2	Daily Return	PMFG
3	Daily Volume Change	MST
4	Daily Volume Change	PMFG
5	Daily Return	Clustering based on RRG

Table 1. Description of Relation Graphs used in R-GAT

[3] https://www.relativerotationgraphs.com/about/the-company

The whole description of the methods to create graphs can be found in Appendix C.

Section 5.5 Training process

The data consists of 2506 data points. To avoid data leakage and ensure a reliable evaluation process, a separation of 90 days was implemented between the training and test sets. This time window was specifically chosen to accommodate the calculation of the correlation matrix, which required a 90-day period.

Most models unless otherwise stated were trained over a span of two epochs. Most of the time convergence to local minimum was observed during the 2nd epoch. Also, unless mentioned base model include the cap in maximum weight of 25%. In addition, it assumes that parameter gamma (γ) is fixed at 0.05, the forecast window is equal to 5, and sample size is equal to 10, PER buffer size is equal to 100, and task loss is an Information ratio.

Section 5.6 Experiments

To analyze the impact of hyperparameters on the model's performance, a series of experiments were conducted. The first set of experiments focused on investigating the role of the risk-appetite parameter, gamma (γ). The subsequent experiments explored the differences between models with varying numbers of attention heads and assessed the significance of incorporating optimization layers. To closer mimic real word applications, a set of experiments with various maximum caps on weights were conducted. By imposing these constraints, the study aimed to explore the model's ability to optimize under weights concentration limits. Furthermore, the importance of Prioritized Experience Replay (PER) was examined. The experiments evaluated the performance of the model with and without PER.

Experiment 1. Learning gamma (γ)

The first experiment focused on understanding the improvement in performance by learning gamma while keeping all other parameters fixed. For this task a set of 4 evaluation periods were introduced. The exact duration of training and evaluation periods can be found in Section 5.7. For each period 5 models were learned. Further on, 10 different gamma values ranging from 0.01 to 0.1 has been used in evaluation set. So, a total set of 200 observations have been collected. The results from these experiments are summarized in Figure 12. From it, it can be concluded that smaller values tend to lead to greater variations in results, whereas high values lead to lower variations. All in all, mean excess return over benchmark stays relatively unaffected.

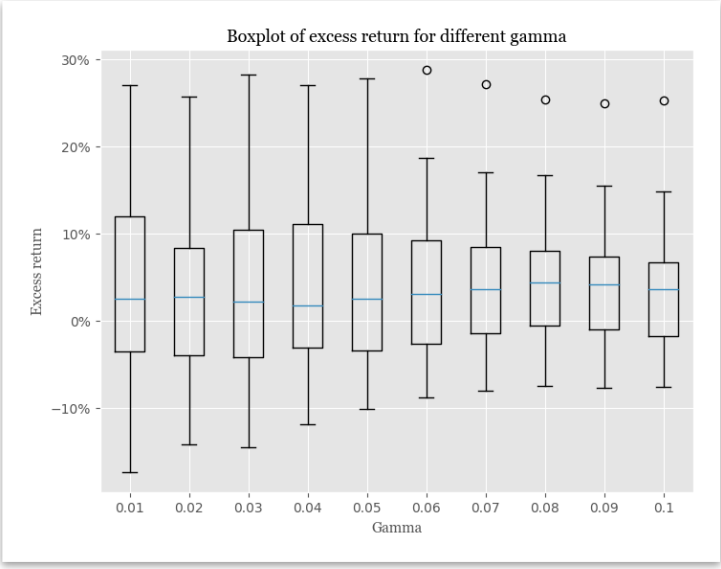

Figure 12. Boxplot of excess return (over benchmark) across different values of gamma.

Experiment 2. Learning number of attention heads

To investigate the impact of the number of heads on the model's performance, three models were implemented with different variations of the number of heads [1, 2, 3] across four evaluation periods. The performance metrics are summarized in Table 2 and Figure 13.

Number of heads	Mean performance	Mean std	Information ratio (IR)
1	9,1%	0,009	0.47
2	**18,3%**	**0,008**	**1,11**
3	17,1%	0,01	0.85

Table 2. results of models' comparison across different number of heads

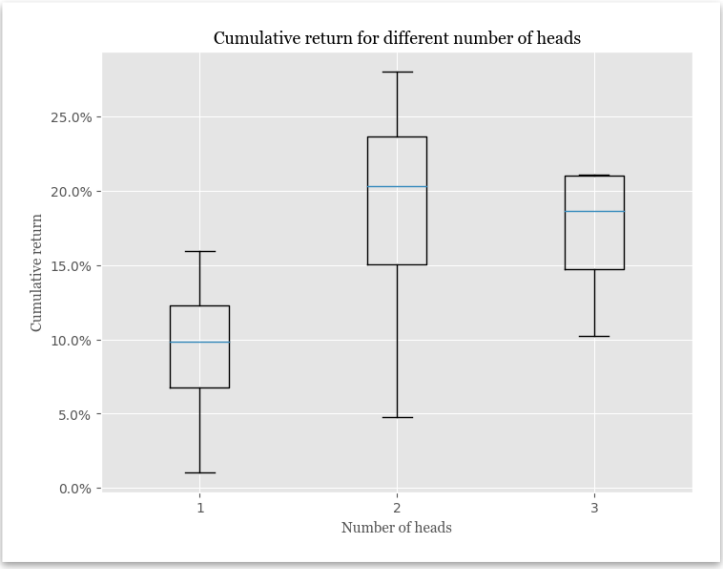

Figure 13. Boxplot of model's performance across 4 evaluation sets.

The results suggest that increasing the number of heads from 1 to 2 yields potential benefits, as evidenced by the improvement in mean performance. However, further increasing the number of heads to 3 does not result in significant performance enhancements and may even lead to overfitting, as indicated by the elevated standard deviation observed in the evaluation sets. However, increasing number of heads may be beneficial for higher dimension embeddings.

Experiment 3. Testing strategies with short positions and concentration limits

In practice, portfolio management often involves the incorporation of short positions to potentially enhance portfolio performance. To examine the effectiveness of such an approach, six strategies were developed with varying constraints on portfolio weights, along with the option to short up to 30% of the total assets. This investment framework, commonly known as the 130/30 portfolio, has gained popularity among hedge funds and portfolio managers. The primary motivation behind investing in 130/30 portfolios is to amplify alpha generation while keeping beta of the portfolio relatively unchanged. Thus, if one believes in the ability of a portfolio manager to generate alpha, the 130/30 structure may provide significant benefits. Investment bank Merrill Lynch reported that over 50 billion USD in assets were currently managed globally utilizing the 130/30 framework.[4]

The specific details of the scenarios employed in the study are outlined in Table 3. The cross-validation of these scenarios followed a similar methodology to previous experiments, utilizing an expanding training window (explained in details in Section

[4] https://www.ipe.com/130/30-strategies-the-us-perspective/22465.article

5.7) and four distinct evaluation sets. The constraints imposed on maximum allocation aim to replicate capital market regulations that limit the concentration ratio of portfolios.

№	Strategy
1	short 30%, max 100%
2	short 30%, max 25%
3	short 30%, max 10%
4	no short 30%, max 100%
5	no short 30%, max 25%
6	no short 30%, max 10%

Table 3. Description of strategies

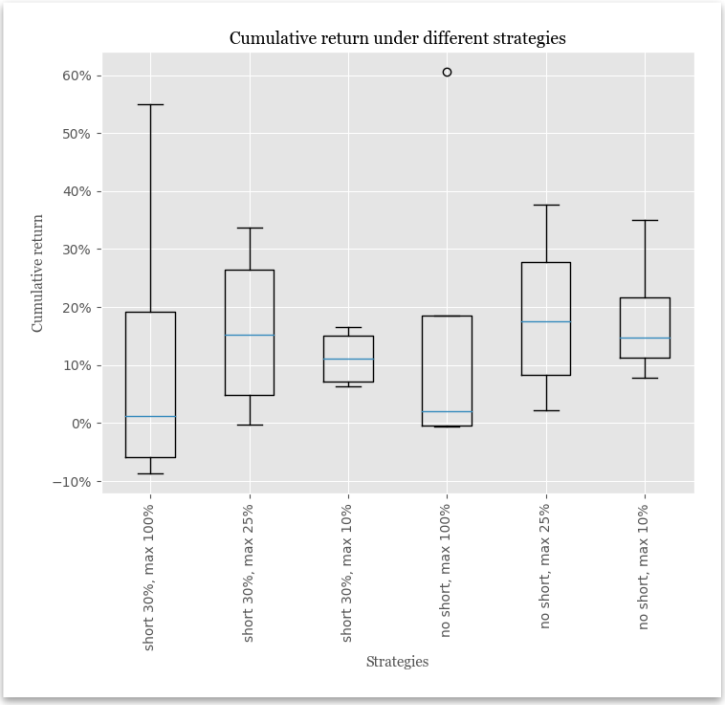

Figure 14. Comparison of returns between different scenarios

Regarding the obtained results, an analysis of variance (ANOVA) was conducted to examine whether there were significant differences in the overall performance of the different strategies. The ANOVA test yielded a statistic of 0.09 with a corresponding p-value of 0.99. This suggests that there is no statistically significant difference in performance. This finding indicates that the model may not have the ability to generate alpha through the application of leverage.

Another question of interest is to study, whether the ability to short, makes some stocks more *"attractive"* for shorting. The difference in mean weights between corresponding scenarios of "short" and "no short": is displayed in Figure 15. The corresponding ANOVA statistics yielded 0.92 with p-value of 0.54. So, the hypothesis of significant differences between stocks allocation between scenarios is rejected, as well.

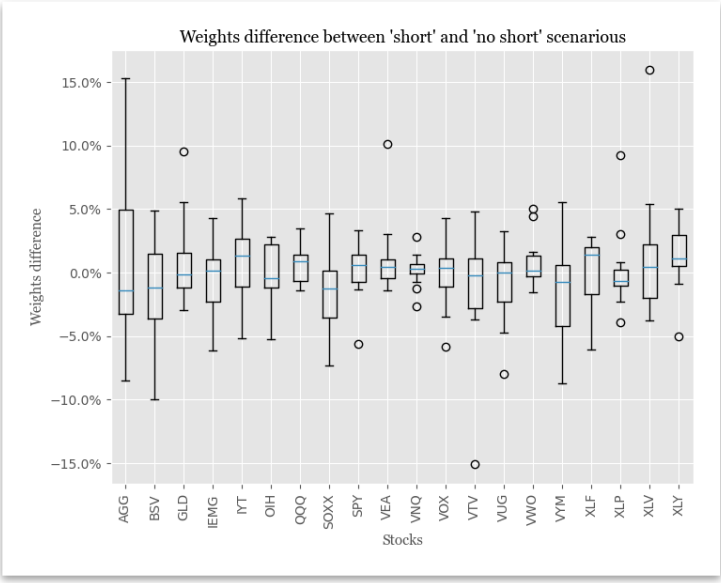

Figure 15. difference in distribution of weights between "short" and "no-short" scenarios.

Experiment 4. Prioritized Experienced Replay

To investigate the impact of the size of PER on the model's performance, three models were implemented with different variations of the PER size [0, 100, 300] across four evaluation periods. The performance is summarized in Figure 16. The gamma parameter was fixed at 0.05.

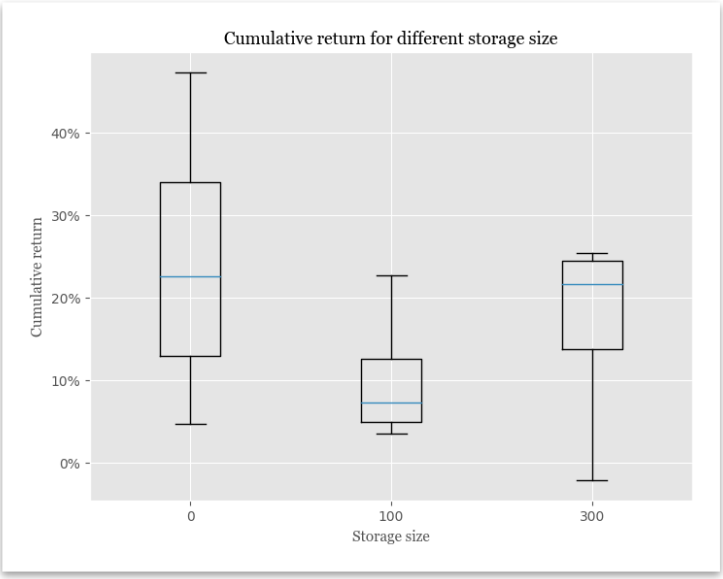

Figure 16. Boxplot of model's performance for different storage sizes

It can be seen that increasing PER size has a muted effect on model's performance but seems to decrease variability in results as compared to the case without PER (storage size is 0). On the other side, models with largest storage sizes beats benchmark in first two evaluation periods with shortest training period as can be seen in Table 4. this may indicate that over longer timespans the distant negative experience is not so important as the most recent history. So, further investigation is needed to access the relevant time span to gather samples for PER buffer.

Evaluation Period	Best model			Benchmark return
	Storage size	Return	Information ratio	
1	300	24%	1,7	20%
2	300	25%	1,6	7%
3	0	29%	0,9	15%
4	0	47%	2,9	33%

Table 4. Best models across different evaluation datasets.

Experiment 5. Comparison with traditional approaches

While the benchmark of equal-weight portfolio may be seen as a relatively simple, I also calculated three other models based on well-known approaches: maximizing Sharpe Ratio on mean-variance frontier, Risk-Parity portfolio and Hierarchical Risk Parity portfolio popularized by Marcos Lopez de Prado (Prado, 2016). I evaluated all of them on the same evaluation datasets I used to evaluate

GNN model. The results are depicted in Figure 17. To make predictions I utilized the portfolio analyzing library *riskfolio*[5].

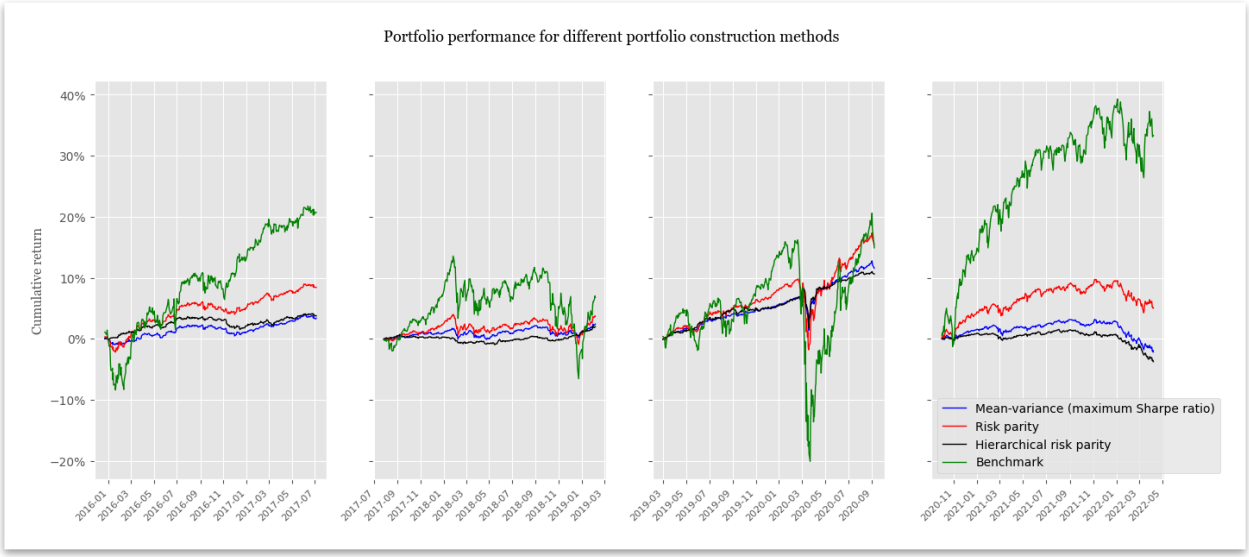

Figure 17. Comparison of popular portfolio optimization approaches *(Equal-weight (Benchmark), Mean-variance (max Sharpe Ratio), Risk-parity and Hierarchical Risk Parity (HRP) approaches)*

From here it can be seen, that all of these approaches stay behind the simple equal-weight portfolio which served as a benchmark for the GNN model on 4 evaluation sets. However, their returns can be considered less volatile as compared to the benchmark.

Section 5.7 Cross validation

In order to evaluate the robustness of the proposed approach, a sensitivity analysis is conducted by simulating 5 models across 4 different time sets with 10 different gammas ranging from 0.01 to 0.09 thus giving 200 estimations. The evaluation is made up with Expanding Train Window method (see Figure 18). Between training and evaluation phases there is a gap equal to the window needed to calculate features, in order to prevent potential leakages.

[5] https://github.com/dcajasn/Riskfolio-Lib

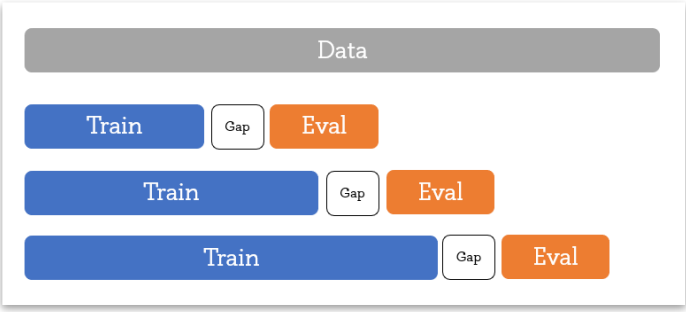

Figure 18. Expanding window scheme for cross-evaluation

Furthermore, the results indicate variability in performance across different evaluation periods. For instance, the first evaluation period from 2nd December 2015 to 5th July 2017 achieved an excess return of 0.9%, while the third evaluation period from 7th February 2019 to 8th September 2020 yielded a stellar average excess return of 10,3%. This variability among other factors may be attributed to the difference in size of training periods as well as to the fact that 3rd Period includes one of the fastest market rebounding after COVID-related drop. Details on performance can be found in Table 5 below.

Train period	Evaluation period	Average return	Benchmark return	Excess return
2013-03-08 : 2015-07-24	2015-12-02 : 2017-07-05	21,7%	20,7%	0,9%
2013-03-08 : 2017-02-24	2017-07-06 : 2019-02-06	8,5%	6,8%	1,7%
2013-03-08 : 2018-09-26	2019-02-07 : 2020-09-08	**25,2%**	14,9%	**10,3%**
2013-03-08 : 2020-04-30	2020-09-09 : 2022-04-08	35,6%	33,3%	2,3%

Table 5: Cross-validation results on model's performance against benchmark.

In order to assess the statistical significance of the findings, a one-sided t-test was employed to evaluate the excess return. The alternative hypothesis considered was that the excess return is greater than zero. The obtained results, as presented in Table 6, indicate statistical significance at the 5% level for all evaluation periods, except for the initial period with the shortest training interval. Notably, a higher gamma value appears to yield more significant results compared to lower values (see Table 7). A threshold can be observed between 0.04 and 0.05, beyond which all results exhibit significance at the 5% level, with higher gamma values tending to produce greater levels of significance.

Evaluation period	T-statistics	P-value
2015-12-02 - 2017-07-05	0,75	0,237
2017-07-06 - 2019-02-06	2,77	**0,003**
2019-02-07 - 2020-09-08	6,92	**<0,001**
2020-09-09 - 2022-04-08	1,92	**0,03**

Table 6. T-statistics across different evaluation periods

Gamma	T-statistics	P-value
0.01	1.18	0.12
0.02	1.32	0.10
0.03	1.36	0.09
0.04	1.62	0.06
0.05	1.89	**0.03**
0.06	2.13	**0.02**
0.07	2.36	**0.01**
0.08	2.40	**0.01**
0.09	2.27	**0.01**
0.1	2.16	**0.02**

Table 7. T-statistics across different values of gamma.

Overall t-statistics for all samples is 2.17 with p-value of 0,021 making it significant at 5% significance level.

CHAPTER 5. CONCLUSION

In conclusion, this study introduced a novel approach to portfolio optimization using Graph Neural Networks (GNN) combined with an optimization layer. The findings demonstrated the potential benefits of this approach, despite the simplicity of the network architecture. Specifically, the integration of Relational Graph Attention Networks into an end-to-end optimization framework showed a promise in enhancing model performance.

The experiments conducted in this study provided insights into the impact of different hyperparameters on the model's performance. The use of Prioritized Experience Replay (PER) was found to be partially beneficial, at least, under short training periods, while the gamma parameter did not significantly enhance the model's behavior in the experiments.

All in all, this work contributes to the growing literature on the application of GNN in Finance, expanding beyond traditional tasks of individual stock forecasting. Meanwhile a number of limitations if this study should be addressed in future research. Particularly, the question of transaction costs and control of turnover rate should be investigated. Another possible way of improvement is to use deeper network structures utilizing more layers and different nodes and edges representations. Moreover, the incorporation of news sources, liquidity and macroeconomic indicators may greatly affect the overall performance by providing relevant graph representations to GNN.

From the literature overview and this study, it is clear that the incorporation of GNN into financial analysis can deepen the understanding of financial markets interconnectedness, while the practical implementations of those ideas are still a work in progress.

References

1. Agrawal, A. и др., 2019. Differentiable convex optimization layers.
2. Black, F. & Litterman, R., 1992. Global Portfolio Optimization. *Financial Analysts Journal*, Том 48, pp. 28-43.
3. Bogle, J. C., 2017. The Little Book of Common Sense Investing: The Only Way to Guarantee Your Fair Share of Stock Market Returns.
4. Busbridge, D., Sherburn, D., Cavallo, P. & Hammerla, N. Y., 2019. Relational Graph Attention Networks.
5. Chen, Y. & Wei, Z., 2018. Incorporating Corporation Relationship via Graph Convolutional Neural Networks for Stock Price Prediction. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*.
6. Clemente, G. P., Grassi, R. & Hitaj, A., 2021. Asset allocation: new evidence through network approaches. *Advances in Complex Systems*.
7. Costa, G. & Iyengar, G. N., 2022. Distributionally Robust End-to-End Portfolio Construction.
8. Donti, P. L., Amos, B. & Kolter, J. Z., 2019. Task-based end-to-end model learning in stochastic optimization.
9. Elton, E. J., Gruber, M. J. & Blake, C. R., 1995. Survivorship Bias and Mutual Fund Performance. *NYU Working Paper*.
10. Eugene F. Fama, K. R. F., 1993. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, Том 33, pp. 3-56.
11. Huang, Y., Capretz, L. F. & Ho, D., 2021. Machine Learning for Stock Prediction Based on Fundamental Analysis. *IEEE Symposium Series on Computational Intelligence (SSCI)*.
12. Jaeger, M. & Marinelli, D., 2022. Network diversification for a robust portfolio allocation.
13. Kaizoji, T., 2013. MODELING OF STOCK RETURNS AND TRADING VOLUME.
14. Mantegna, R., 1998. Hierarchical Structure in Financial Markets. *The European Physical Journal B*.
15. Markowitz, H., 1952. Portfolio Selection. *The Journal of Finance*, Том 7, pp. 77-91.

16. Matsunaga, D., Suzumura, T. & Takahashi, T., 2019. Exploring Graph Neural Networks for Stock Market Predictions with Rolling Window Analysis. *arXiv*.
17. Mokhtari, S., Yen, K. K. & Liu, J., 2018. Effectiveness of Artificial Intelligence in Stock Market.
18. Pozzi, F., Matteo, T. D. & Aste, T., 2013. Spread of risk across financial markets:. *Nature*, Issue 1665.
19. Prado, M. L. d., 2016. Building Diversified Portfolios that Outperform Out-of-Sample. *Journal of Portfolio Management*.
20. Pressacco, F. & Serafini, P., 2007. The origins of the mean-variance approach in finance:. *Decisions Econ Finan*, Том 30, p. 19–49.
21. Scarselli, F. и др., 2009. The graph neural network model.
22. Schaul, T., Quan, J., Antonoglou, I. & Silver, D., 2016. Prioritized Experience Replay.
23. Taylor, S. M. & Cerbo, L. D., 2019. Graph Theoretical Representations of Equity Indices and their Centrality Measures.
24. Thomas, K. & Welling, M., 2016. Semi-Supervised Classification with Graph Convolutional Networks.
25. Vargas, M. R., Anjos, C. E. M. d., Bichara, G. L. G. & Evsukoff, A. G., 2018. Deep Leaming for Stock Market Prediction Using Technical Indicators and Financial News Articles. *2018 International Joint Conference on Neural Networks (IJCNN)*.
26. Veličković, P. и др., 2018. Graph Attention Networks.

Appendix A. ETF Description

Ticker	Name	Capitalization (bn)[6]	Description	Asset class
AGG	iShares Core U.S. Aggregate Bond ETF	89,94	Tracks an index of US investment-grade bonds. The market-weighted index includes Treasuries, agencies, CMBS, ABS and investment-grade corporates.	Bonds
BND	Vanguard Total Bond Market Index Fund	91,82	Tracks a broad, market-value-weighted index of US dollar-denominated, investment-grade, taxable, fixed-income securities with maturities of at least one year.	Bonds
BSV	Vanguard Short-Term Bond Index Fund	37,42	Tracks a market-weighted index of US-government bonds, investment-grade corporate and investment-grade international dollar-denominated bonds with maturities of 1-5 years.	Bonds
GLD	SPDR Gold Shares	59,61	Tracks the gold spot price, less expenses and liabilities, using gold bars held in London vaults.	Commodity
IEMG	iShares Core MSCI Emerging Markets ETF	69,81	Tracks a market-cap-weighted index of emerging-market firms, covering 99% of market capitalization.	Equity
IYT	iShares Transportation Average ETF	0,78	Tracks a broad-based, modified market-cap-weighted index of US stocks in the transportation industry.	Equity
OIH	VanEck Oil Services ETF	2,11	Tracks a market-cap-weighted index of 25 of the largest US-listed, publicly traded oil services companies.	Equity
QQQ	Invesco QQQ Trust	180,33	Tracks a modified-market-cap-weighted index of 100 NASDAQ-listed stocks.	Equity
SOXX	iShares Semiconductor ETF	8	Tracks a modified market-cap-weighted index of 30 US-listed semiconductor companies.	Equity

[6] As of 25.05.2023

Ticker	Name	Capitalization (bn)[6]	Description	Asset class
SPY	SPDR S&P 500 ETF Trust	390,67	Tracks a market cap-weighted index of US large- and mid-cap stocks selected by the S&P Committee.	Equity
VEA	Vanguard Developed Markets Index Fund	112,83	Passively managed to provide exposure to the developed markets ex-US equity space. It holds stocks of any market capitalization.	Equity
VOX	Vanguard Communication Services Index Fund	2,85	Seeks to track the performance of a benchmark index that measures.	Equity
VTV	Vanguard Value Index Fund	96,5	Tracks an index of large-cap stocks in the US. Holdings are selected and weighed based on five value factors.	Equity
VUG	Vanguard Growth Index Fund	85,14	Tracks an index of large-cap stocks in the US. Holdings are selected and weighed based on growth factors.	Equity
VWO	Vanguard Emerging Markets Stock Index Fund	71,36	Passively managed to provide exposure to the emerging markets equity space. It holds stocks of any market capitalization.	Equity
VYM	Vanguard High Dividend Yield Index Fund	47,47	Tracks the FTSE High Dividend Yield Index. The index selects high-dividend-paying US companies, excluding REITS, and weights them by market cap.	Equity
VNQ	Vanguard Real Estate Index Fund	31,35	Tracks a market-cap-weighted index of companies involved in the ownership and operation of real estate in the United States.	Real Estate
XLF	Financial Select Sector SPDR Fund	29,44	Tracks an index of S&P 500 financial stocks, weighted by market cap.	Equity
XLP	Consumer Staples Select Sector SPDR Fund	18,55	Tracks a market-cap-weighted index of consumer-staples stocks drawn from the S&P 500.	Equity

Ticker	Name	Capitalization (bn)[6]	Description	Asset class
XLV	Health Care Select Sector SPDR Fund	40,46	Tracks health care stocks from within the S&P 500 Index, weighted by market cap.	Equity
XLY	Consumer Discretionary Select Sector SPDR Fund	14,93	Tracks a market-cap-weighted index of consumer-discretionary stocks drawn from the S&P 500.	Equity

Appendix B. Technical indicators

Name	Type	Parameters Specification
RSI	Momentum	Periods: 14, 28
MACD	Trend	Periods_slow: 26, 36; Periods_fast: 12, 18; periods_sign: 9, 12
Vortex Indicator	Trend	Periods: 14, 28
Stochastic Oscillator	Momentum	Periods: 14, 28
Williams Indicator	Momentum	Periods: 14, 28
Ulcer Index	Volatility	Periods: 14, 28

Appendix C. Graphs generation

Relative Rotation Graph

The algorithm[7]: employed for the relative rotation graph:

1. Prices of the stocks are loaded and normalized relative to the first value within the specified period.
2. The relative price of each stock is calculated with respect to the benchmark.
3. A rolling mean and standard deviation are computed for this relative price, using a window size of 50 days.
4. The RS ratio is then determined by applying the following formula: 100 + ((relative price - rolling mean) / rolling standard deviation).
5. To compute the momentum, the price of each day is compared to the price from 10 days ago.

[7] https://msw.flxn.de/tool/sector/

6. The resulting momentum values are smoothed with a rolling mean of 50 days.
7. Similarly, the RS momentum is calculated using the formula: 100 + ((momentum - momentum rolling mean) / momentum rolling standard deviation).
8. Finally, the RS ratio and RS momentum are further smoothed using a rolling mean with a window size of 10 days.