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’s thesis*

for the degree "Machine Learning and High-Load Systems"

01.04.02 Applied Mathematics and Computer Science

Moscow, 2023

## TABLE OF CONTENT

[TABLE OF CONTENT 2](#_Toc136297834)

[ABSTRACT 3](#_Toc136297835)

[CHAPTER 1. INTRODUCTION 3](#_Toc136297836)

[CHAPTER 2. LITERATURE REVIEW 4](#_Toc136297837)

[CHAPTER 3. METHODOLOGY 5](#_Toc136297838)

[Section 3.1 Prioritized experience replay 7](#_Toc136297839)

[Section 3.2 Relation Graph Attention Networks 7](#_Toc136297840)

[CHAPTER 4. MODEL ARCHITECTURE 9](#_Toc136297841)

[Section 4.1 End to end portfolio construction 9](#_Toc136297842)

[Section. 4.1.1 Prediction layer 11](#_Toc136297843)

[Section 4.1.2 Optimization layer 12](#_Toc136297844)

[CHAPTER 5. EXPERIMENTS ON MARKET DATA 12](#_Toc136297845)

[Section 5.1 Collecting the data 12](#_Toc136297846)

[Section 5.2 Benchmark 14](#_Toc136297847)

[Section 5.2 Features generation 14](#_Toc136297848)

[Section 5.3 Graph generation 14](#_Toc136297849)

[Section 5.4 Training process 16](#_Toc136297850)

[Section 5.5 Experiments 16](#_Toc136297851)

[Experiment 1. Learning gamma () 16](#_Toc136297852)

[Experiment 2. Learning number of attention heads 18](#_Toc136297853)

[Experiment 3. Model without optimization layer 19](#_Toc136297854)

[Experiment 4. Different constraints on portfolio weights 19](#_Toc136297855)

[Experiment 5. Different size of Prioritized Experienced Replay (PER) buffer 21](#_Toc136297856)

[Experiment 5. Different prediction windows 23](#_Toc136297857)

[CHAPTER 5. CONCLUSION 24](#_Toc136297858)

[References 25](#_Toc136297859)

[Appendix A. ETF Description 26](#_Toc136297860)

[Appendix B. Technical indicators 28](#_Toc136297861)

[Appendix C. Graphs generation 29](#_Toc136297862)

[Relative Rotation Graph 29](#_Toc136297863)

## ABSTRACT

This research focuses on addressing the stock 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 novel concept of Prioritized Experience Replay (PER) is proposed and implemented.

The evaluation of the approach is conducted on a dataset comprising major US Exchange-Traded Funds (ETFs), which represent a diverse universe of assets. The performance of the model is assessed using various metrics, including the Sharpe ratio, cumulative returns, etc.

The results highlight the effectiveness of the graph neural network-based approach in capturing the complex relationships and dependencies among assets. The incorporation of PER enhances the model's ability to learn from exceptional cases and optimize portfolio allocations accordingly. Notably, the findings suggest that the introduction of PER leads to improved performance in terms of risk-adjusted returns and outperformance across different market conditions.

These empirical insights contribute to the existing body of knowledge on stock portfolio optimization and underscore the importance of incorporating advanced machine learning techniques and innovative strategies, such as PER, to enhance the accuracy and reliability of portfolio allocation decisions. The research opens avenues for further exploration of graph neural networks in the context of portfolio optimization problem.

## CHAPTER 1. INTRODUCTION

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 market conditions. Traditional approaches have tended to focus solely on one aspect, limiting the breadth of the problem and potential degrees of freedom. 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 beauty of end-to-end optimization firstly proposed by (Donti, et al., 2019) is that end-to-end optimization allows to train models on the same criteria (e.g., Sharpe Ratio) that used to evaluate them, thus overcoming the problem of setting intermediate goals or finding the best approximation to them. The development of machine learning methods allows to create robust frameworks utilizing stochastic optimization to achieve this task.

The study is structured into sections. The first section offers a literature review, situating the current study within the existing body of knowledge. The second section 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 section provides an overview of the model architecture, highlighting the structural components of both the prediction and optimization layers. In the fourth section, 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 section.

Finally, 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. The insights gained from this study contribute to the advancement of portfolio allocation strategies and provide a foundation for further exploration in this domain.

## 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 individuals 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. Markowitz's 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). seeks to establish a relationship between fundamental factors and stock performance. The future 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]](#footnote-1).

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 forest (Mokhtari, et al., 2018).

The emergence of graph neural networks has also 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. They 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. They introduced a dot product of node embeddings in the computations of convolutional layers to capture the dynamic nature of stock co-dependence over time (Temporal Graph Convolution).

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 and effectiveness in forecasting stock performance.

## CHAPTER 3. METHODOLOGY

In this chapter I outline main assumptions of end-to-end optimization process. Following the previous work on this theme (Costa & Iyengar, 2022) I use similar approach to portfolio optimization. The optimization problem requires both the point prediction and the past prediction errors as inputs to quantify and control the model risk. In the optimization layer, 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. Task loss is the ultimate goal to optimize, in this paper I have two version of task goals: maximizing Sharpe ratio or future return.

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

* represents features variables.
* represents target variables,
* represent weights that are chosen during the optimization process.
* represents realized returns used to indicate

The goal is to find weights of a neural network that helps to minimize the task loss function subject to a nominal cost function :

(1)

Nominal loss function:

(2)

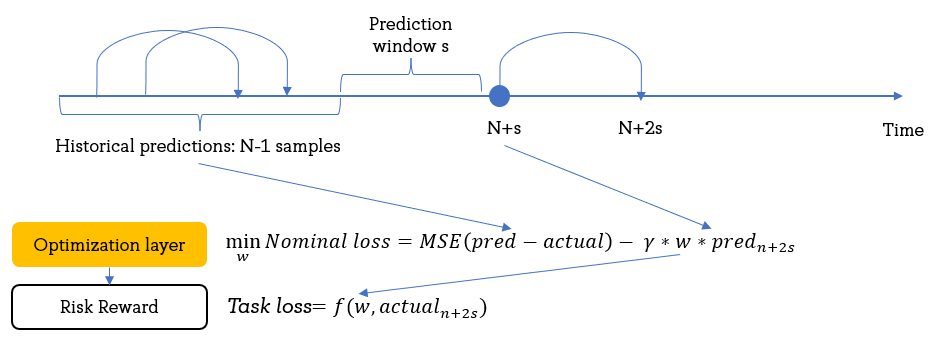
Here, represents the size of the past prediction sample, and denotes the prediction window. The prediction window signifies the duration over which return is predicted. The rationale behind employing a longer prediction window is to make predictions less stochastic as over longer time horizon stocks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

Figure 1: Schema of calculating *nominal* and *task losses*

### Section 3.1 Prioritized experience replay

Similar to the approach adopted in the original study (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 (Schaul, et al., 2016). The fundamental principle behind PER is to train an agent using the specific cases where the agent obtains suboptimal results. By storing these "exceptional" cases in a buffer and retrieving them during training, the agent is expected to learn more efficiently.

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 specific case, I chose to employ PER to store the indices of time periods when the losses were most substantial, 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.

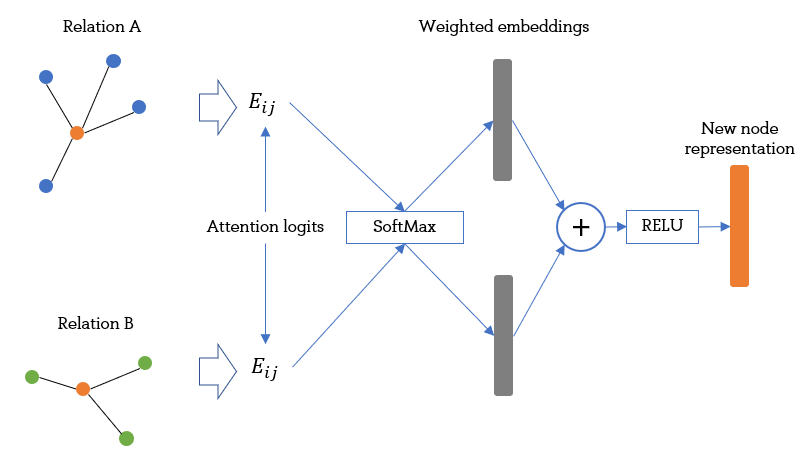
### Section 3.2 Relation Graph Attention Networks

In contrast to conventional approaches, I opted to utilize Graph Neural Networks (GNN) for the prediction layer in my study. 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, and recommendation systems.

The emergence of GNNs took a leap forward in 2018 when graph models with attention mechanisms were introduced (Veličković, et al., 2018). This breakthrough opened new avenues for enhanced graph-based modeling. Subsequently, Relational Graph Attention Networks (R-GAT) (Busbridge, et al., 2019) emerged as the next stage in the development of neural networks. 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.

The input to an R-GAT is a graph, defined as a set of nodes and edges. Each node and edge in the graph is associated with a feature vector. These vectors serve as the initial representations for nodes and edges. The key component of R-GATs (and GATs 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 neighbor's features should have on the node's new feature representation. In R-GATs, the attention score is also dependent on the edge features, allowing it to incorporate relational reasoning. Once the attention scores are computed, the model updates each node's features by taking a weighted sum of its neighbors' features, where the weights are the attention scores. 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' features, and the outputs from all heads are concatenated to form the final updated node features.

There are two types of normalizing attention logits: 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 both to compare their relative performance.

Figure 2. Schematic work of RGAT.

Attention logits are computed for each relation type with the help of both query and key kernels:

Then the additive attention is applied, since I want to take into account edge features as well, I add then to the attention logits)

After that, I reassign attention weights using *across-relation attention mechanism*:

After that, the new node embedding is obtained through propagation:

In case of multiple-head attention, I aggregate individual embeddings by concatenation.

## 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 systematic process. First, N-1 samples are obtained, predictions are made, and the corresponding errors are calculated. Subsequently, I take a time gap of prediction window 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*  **do**

**for** each sample in training sample **do**

predict future return

calculate the error

aggregate with previous errors to the sum

Predict future return for

Calculate Mean Squared Error of past prediction errors

Find optimal weights

Use optimal weights to calculate Risk-Reward

Back-propagate

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 our perspective, this inclusion enhances 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.

Another idea to explore the effectiveness of the optimization layer by comparing it with a model-free approach as opposed to the established model-based one. Model-free approach is the one without the optimization layer. I take the outputs of the prediction layer and consider them to be the logits of weights. By taking a SoftMax, I obtain weights that are included in the risk reward function. The difference is depicted in Figure 3 and Figure 4.

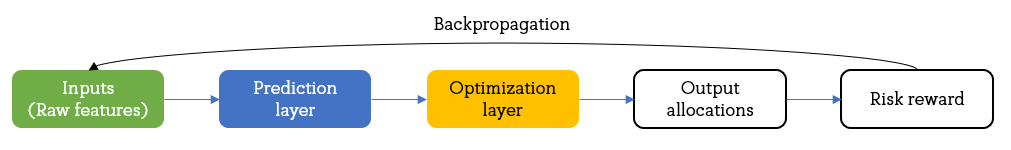
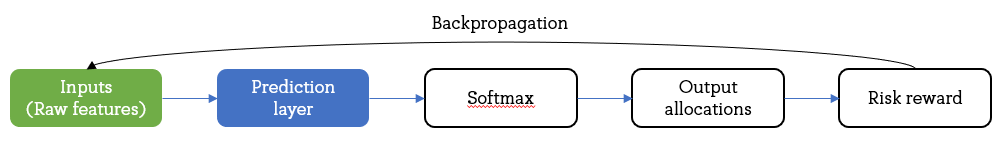


Figure 3: Computational schema of **model-based** approach

Figure 4: Computational schema of **model-free** approach



#### Section. 4.1.1 Prediction layer

Prediction layer consists of 3 layers: 2 relational graph attention layers (RGAT) and one linear layer. RGAT layers have the same number of attention heads: defined in the hyperparameters. In the base scenario: 2 heads. Number of heads is stated per relation: since I have 5 relations: there are 10 attention heads in total. They have across-relation attention mechanism meaning that attentions from all relations are weighted to achieve final numbers.

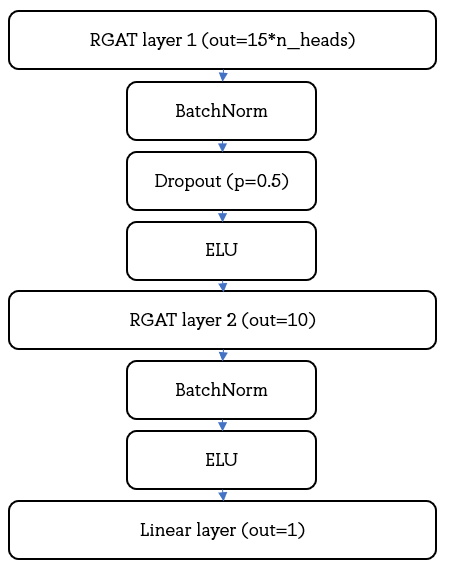


Figure X: Scheme of prediction layer

#### 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 solve the weights optimization problem. By incorporating the convex optimization layer, I can effectively address optimization challenges and achieve more robust and tractable portfolio optimization within the overall model.

## CHAPTER 5. EXPERIMENTS ON MARKET DATA

### Section 5.1 Collecting the data

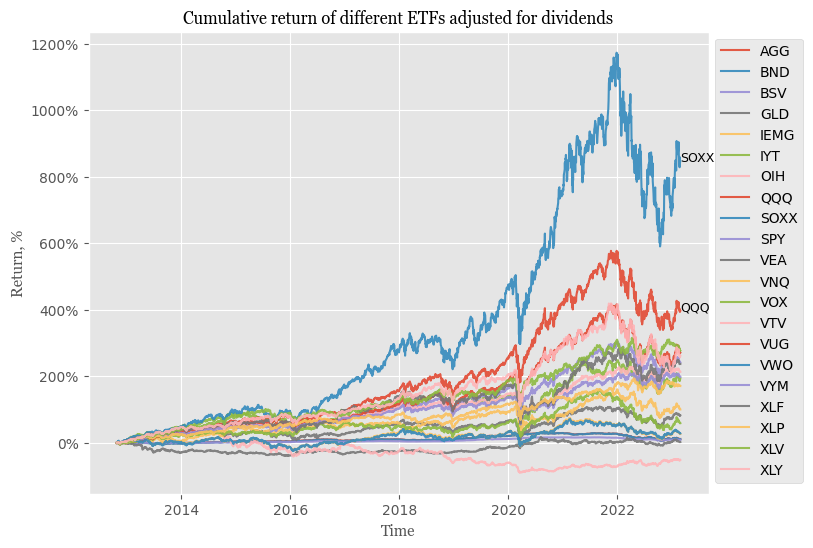
The decision to utilize ETFs as the target universe instead of individual stocks is based on two primary reasons. Firstly, it addresses the concern of survivorship bias prevalent in stock selection. In today's market, selecting stocks solely based on their performance introduces a risk of overfitting, as only the successful companies that have not been acquired or gone bankrupt are observable. Consequently, mitigating this bias requires additional measures, such as obtaining information on ceased tickers, which can be challenging. ETFs offer 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 comprehensive list of these ETFs provided in Appendix A. The selection of ETFs was based on several criteria, including their 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 13th September 2004 to 13th April 2023, providing a comprehensive view of the market dynamics over the considered period.

At figure XX, 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.



Добавить матрицу с корреляцией активов

### Section 5.2 Benchmark

For the benchmark I use simple average of ETF performances. 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 like S&P 500 because it is applicable only to the part of the universe that is connected to US stock market.

### Section 5.2 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, providing transparency and reproducibility to the analysis.

Additionally, I incorporated cumulative past returns for various time frames, namely [1, 3, 7, 14, 30, 40, 50, 60, 70, 80, 90] trading days. Notably, it is crucial to clarify that when referring to days, I specifically mean trading days rather than calendar days. This distinction is necessary to accurately capture the dynamics of the ETF market.

Overall, leveraging the "ta" Python library, I systematically generated 14 features for each node at every time period under consideration.

### Section 5.3 Graph generation

It is common to consider modified correlation table as a prime source for adjacency matrix. For example, In the paper (Taylor & Cerbo, 2019), (Jaeger & Marinelli, 2022) it was proposed that *minimum spanning tree* *(MST)* and *maximum filtered planar tree (MFPL)* can better picture market structure than the representation obtained based on correlation matrix.

Therefore in the following approach was used:

* Gower distance metric as a distance to measure nodes connectivity:
* (3)

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

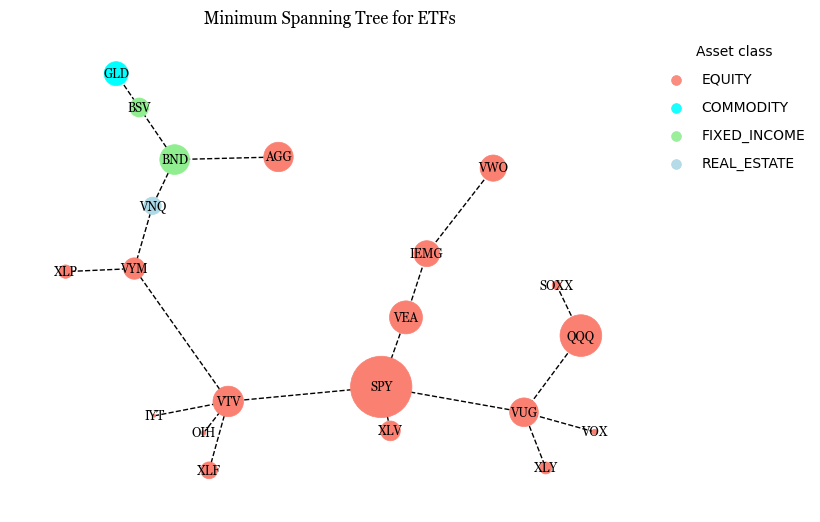
**

Figure X: Example of *minimum spanning tree* on ETFs 90-periods return correlation

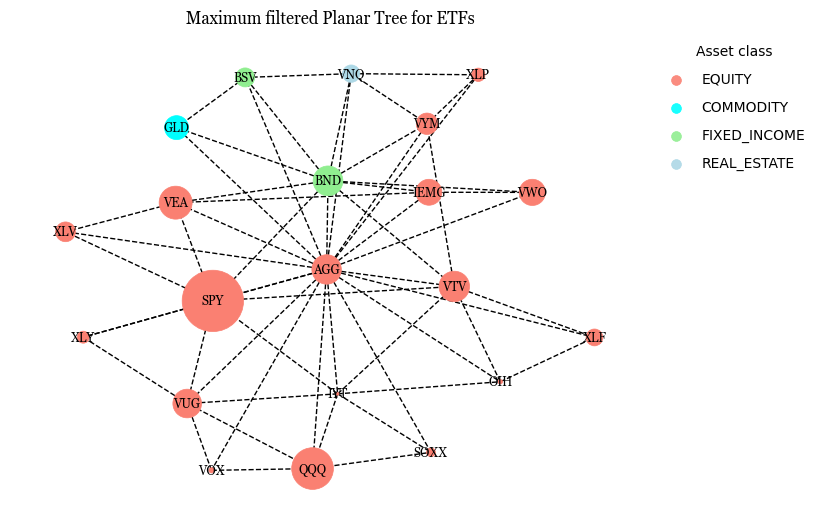
**

Figure X: Example of *maximum filterted planar tree* on ETFs

In this work, both types of graphs and adjacency matrices are used. They are applied both to ETF return and ETF trading volume changes.

In addition to them another novel method has been developed. The first one is Relative Rotation Graph (RRG). These charts were developed by Julius de Kempenaer and have been available on Bloomberg terminals since 2011.[[2]](#footnote-2) 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.

It is natural to cluster stocks by region, industry or secotre, as well, as by their capitalization. It was shown (Marcos de Prado) that clustering approach can help to achieve better portfolio diversification.

|  |  |  |
| --- | --- | --- |
| # | Variable | Type of graph |
| 1 | Daily Return | MST |
| 2 | Daily Return | MFPL |
| 3 | Daily Volume Change | MST |
| 4 | Daily Volume Change | MFPL |
| 5 | Daily Return | Clustering based on RRG |

Table 1: Description of Relation Graphs

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

### Section 5.4 Training process

### Section 5.5 Experiments

The historical data consisted of weekly asset and feature returns from 07Jan2000 to 01Oct2021.

#### Experiment 1. Learning gamma ()

The first experiment focused on understanding the improvement in performance by learning gamma while keeping all other parameters fixed. The data was split into train and eval sets. Train sets consisting of 1800 samples and eval set of 526 samples. Eval set is roughly of 2 years of trading data since every trading year on average consists of 255 trading days. To prevent the feature leakage, there is a 90 trading days interval between training and test sets, so features calculated in test set in no way can intersect with returns from the train set (The window to calc correlation and covariance is 90 trading days). In the basis scenario I use

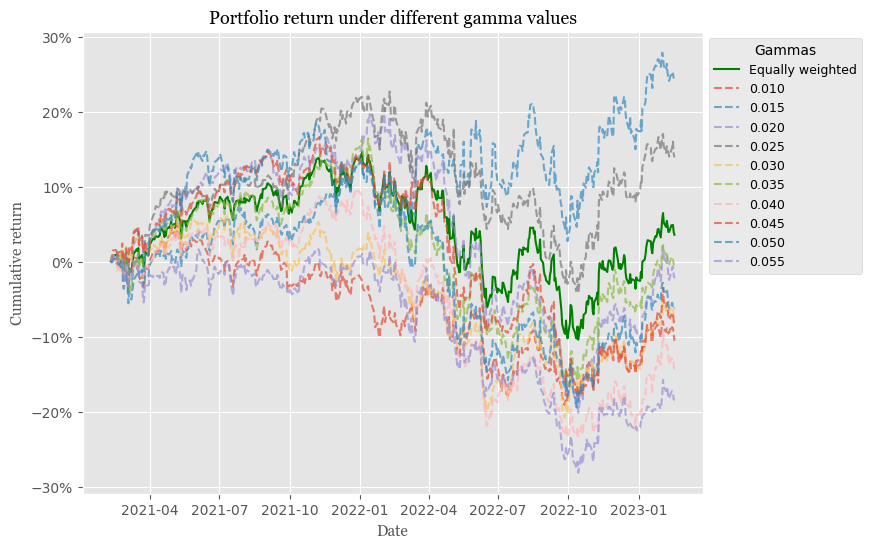
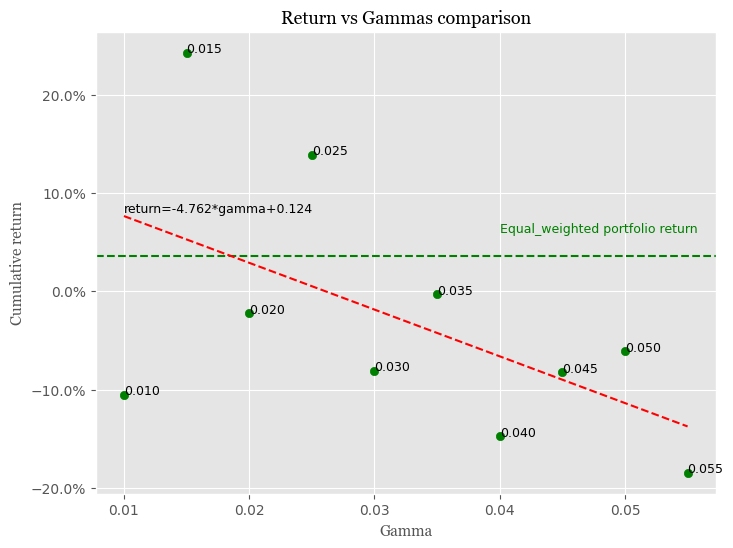
At first experiments I kept gamma as a learnable parameter, as it is difficult to come up with an optimum value beforehand. However, during the experiments I have found out that smaller values of gamma, tend to produce better results. My initial guess was that optimal gamma is in the range from 0.01 to 0.02. It is important to mention that these values apply to the case of sample size equal to 9. All in all, there is a tendency for higher values of gamma to produce worser results all other things being equal. It may be attributed to the fact, that If we place too much weight on gamma, we thus increase the ‘risk-appetite’ of the model, thus leading to worser asset allocation.

Figure: Portfolio return under different gammas

Figure: Dependency between portfolio return and gamma

#### Experiment 2. Learning number of attention heads

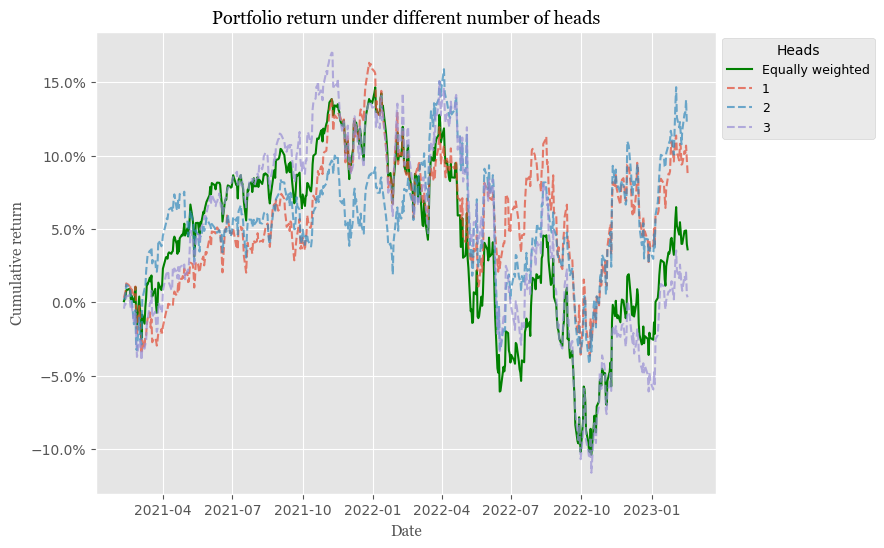
Second experiment was to change number of heads to see if it can meaningfully affect performance. The experiment was done with constant gamma . The results are mixed. From one point of view results indicate that fewer heads are preferable, as the results are better, on the other hand the number of days of 3-heads model leading is bigger than from the counterparts.

Figure X: performance of the model

In addition to evaluating the cumulative return over a specific period, a broader metric was employed to assess the overall performance of each model. This metric involved determining the number of days in which each model outperformed the others throughout the entire testing period. The results yielded from this analysis do not lead to a single definitive conclusion.

When maximizing the Sharpe Ratio, the performance of the models appeared to be relatively unaffected by the number of heads. Each model demonstrated leadership approximately one-third of the time. However, in the context of maximizing return, the model with fewer heads exhibited a clear dominance, as indicated in Table 1.

|  |  |  |
| --- | --- | --- |
| Number of heads | Task loss | |
| Sharpe ratio | Maximum return |
| 1 | 168 | **448** |
| 2 | 158 | 48 |
| 3 | **185** | 16 |

Table 1: Number of days of model leading performance

#### Experiment 3. Model without optimization layer

In order to evaluate the effectiveness of the optimization layer, an experiment was conducted using a model composed solely of the prediction layer and SoftMax afterwards. This allowed to treat outputs as predicted weights. However, without the support of the optimization layer, the model struggled to differentiate between assets, resulting in a tendency to utilize an equal-weighted distribution approach most of the time (Figure X). Training was done on 4 epochs. (add description)

Overall, the analysis suggests that the bare model, lacking an optimization layer, did not provide any additional value compared to the benchmark. The absence of the optimization layer hindered the model's ability to effectively allocate weights among assets and failed to improve upon the baseline approach.

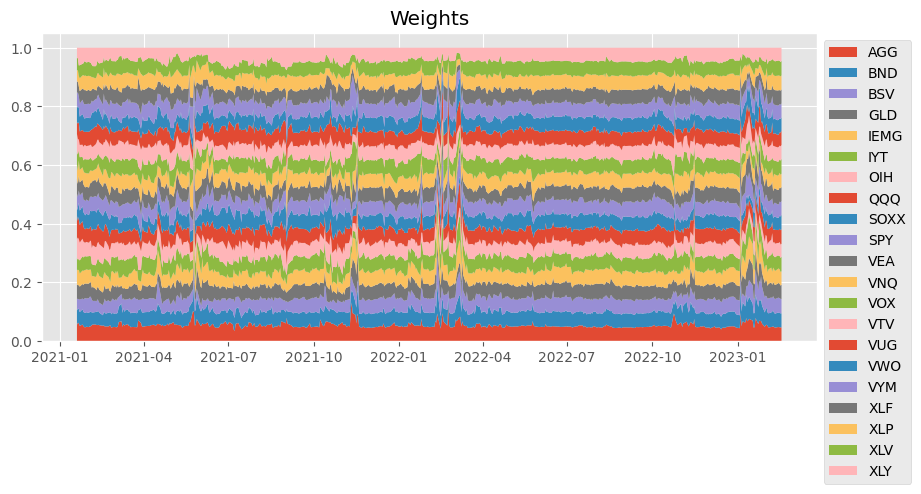


Figure X

#### Experiment 4. Different constraints on portfolio weights

(**Add experiments description**)

In order to examine the impact of different scenarios and constraints on the performance of the model and optimization layer, a series of experiments were conducted. Six distinct scenarios were created, including variations in short selling and maximum concentration constraints. For each set of cases, three scenarios allowed short selling, while the other three did not. Additionally, three maximum weight constraints were evaluated: 10%, 25%, and no restrictions ("no restrictions" indicates no maximum weight constraint). The results of these experiments are presented in Table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Strategy | Cum. return | Sharpe ratio | Average leverage |
| 1 | short 30%, max 100% | **80%** | **1,43** | **7.5** |
| 2 | no short, max 100% | 38% | 1,3 | 0 |
| 3 | short 30%, max 25% | -1% | -0,01 | 3,91 |
| 4 | no short, max 10% | -7% | -0,07 | 0 |
| 5 | no short, max 25% | -9% | -0,42 | 0 |
| 6 | short 30%, max 10% | -23% | -0,84 | 1,4 |
| *-* | *Benchmark, equal weighted* | *3%* | *0,11* | *0* |

Table 1: comparison of the different scenarios

Several noteworthy conclusions can be drawn from the results. Firstly, when short selling is permitted, the optimizer tends to apply excessive risk, particularly when there are no restrictions on weights. For instance, in the "Short 30%, Max 100%" strategy, the average leverage reached 7.5x, while the results only marginally surpassed those of the strategies without leverage.

Furthermore, it is important to highlight that weight constraints significantly impact model performance. The restrictions on weights severely limit the model's ability to generate returns, likely due to the fact that the model tends to select the fewest possible number of stocks given the opportunity, resulting in high turnover and reduced performance.

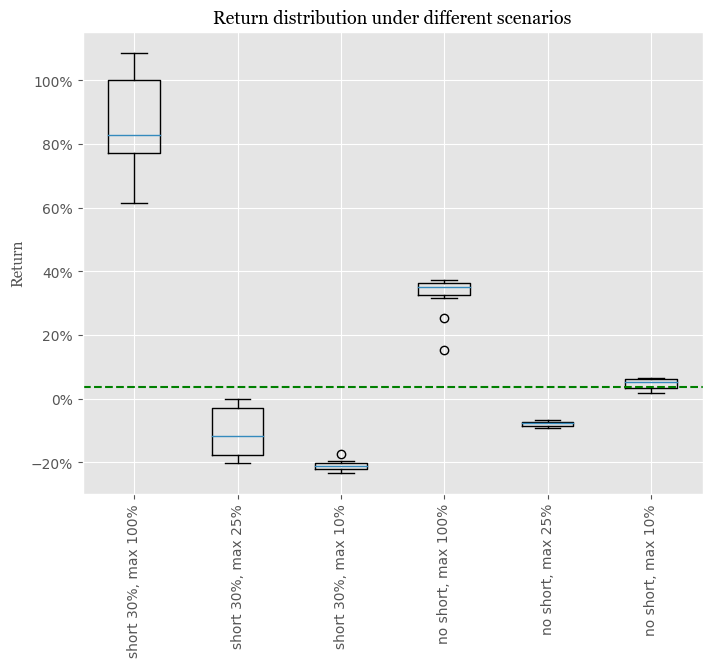
To explore the relationship between the results and the provided gamma value, sub-scenarios were calculated for 10 different gammas ranging from 0.01 to 0.1 for all six models. The results of this analysis can be observed in Table 2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Strategy | Best gamma | Best return | Worst gamma | Worst return |
| 1 | Short 30%, max 100% | **0.1** | **109%** | 0.01 | **61%** |
| 2 | Short 30%, max 25% | 0.02 | 0% | **0.1** | -20% |
| 3 | Short 30%, max 10% | 0.01 | -17% | 0.03 | -23% |
| 4 | No short, max 100% | 0.05 | 37% | 0.01 | 15% |
| 5 | No short, max 25% | 0.03 | -6% | **0.1** | -9% |
| 6 | No short, max 10% | **0.1** | 7% | 0.01 | 2% |

Table 2: The summary of experiments

The results demonstrate the dependence of strategy performance on the gamma value. Different strategies exhibit varying levels of sensitivity to the gamma parameter. For example, the "Short 30%, Max 100%" strategy achieved the best return with a gamma of 0.1 (109%), while the worst return was observed with a gamma of 0.01 (61%). These findings underscore the importance of selecting an appropriate gamma value to optimize the performance of the model.

In the context of portfolio optimization, gamma (γ) can be considered a "risk-appetite parameter" (Costa & Iyengar, 2022), with higher gamma values indicating a greater confidence in the model's predictions, thus greater risk-appetite. However, the relationship between gamma values and the model's self-assertion requires further investigation to draw meaningful conclusions. This aspect provides an avenue for future research, aiming to delve deeper into the implications and significance of gamma values in assessing the model's confidence in its predictions.



#### Experiment 5. Different size of Prioritized Experienced Replay (PER) buffer

To assess the impact of Prioritized Experience Replay (PER) on model stability, I conducted experiments using three different strategies, each with a varying buffer size. The experiments maintained consistent conditions, including a gamma value of 0.01, default weight constraints of 25%, no shorting allowed, and a training duration of four epochs.

The results (see Table 3) clearly indicate that the introduction of a buffer significantly enhances the model's ability to learn the distribution of returns. Notably, the buffer initially aids in reducing losses during periods of declining benchmarks. Moreover, as the buffer size increases, the gap between outperformance when the benchmark is decreasing or increasing diminishes, illustrating the buffer's effectiveness in mitigating both scenarios.

Additionally, the data reveals interesting insights regarding the relationship between buffer size and model performance. With a buffer size of 100, the model achieved a Sharpe ratio of 0.72 and a return of 17%. It demonstrated outperformance on 435 days, accounting for approximately 85% of the total testing period. Notably, the model exhibited greater outperformance while the benchmark was decreasing (211 days) compared to when it was increasing (224 days).

In contrast, when the buffer size was increased to 500, the model's performance further improved. The Sharpe ratio increased to 1.02, while the return climbed to 25%. The model showcased even more significant outperformance, with 487 days of outperformance, representing approximately 95% of the testing period. Remarkably, the model demonstrated almost equal outperformance during both decreasing and increasing benchmark conditions, with 245 and 242 days, respectively.

Additionally, the absence of a buffer resulted in suboptimal model performance. The model achieved a considerably lower Sharpe ratio of 0.14 and a modest return of 3%. It demonstrated outperformance on 395 days, accounting for approximately 77% of the testing period. Similar to the cases with a buffer, the model exhibited slightly higher outperformance while the benchmark was decreasing (201 days) compared to when it was increasing (194 days).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Buffer size | Sharpe Ratio | Return | Days of outperformance | Outperformance while benchmark decreasing | Outperformance while benchmark increasing |
| 100 | 0.72 | 17% | 435 (85%) | 211 (81%) | 224 (89%) |
| 500 | 1.02 | 25% | 487 (95%) | 245 (95%) | 242 (96%) |
| No buffer | 0.14 | 3% | 395 (77%) | 201 (78%) | 194 (77%) |

Table 3: performance under different PER buffer sizes.

#### Experiment 5. Different prediction windows

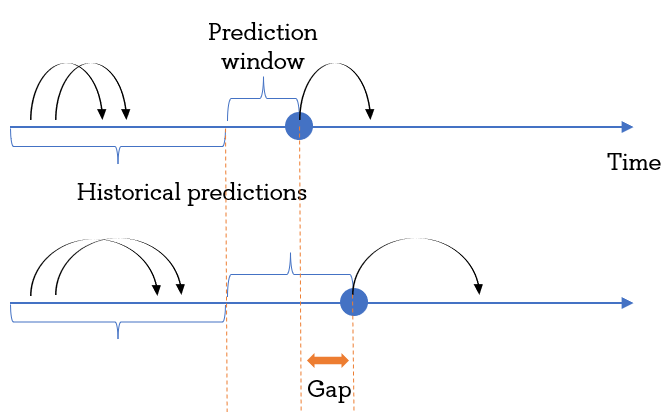
Furthermore, an evaluation was conducted to assess the impact of different prediction windows on the algorithm's efficacy. The rationale behind exploring various window sizes lies in the understanding that next-day returns exhibit high stochasticity, making accurate predictions challenging. Conversely, longer-term returns tend to be less noisy. However, an inherent drawback of this approach is the increasing disparity between past and current forecasts, potentially resulting in less accurate predictions (refer to Figure X).

Figure X: Visualization of widening prediction window gap

The experiment involved running six models with a fixed setup, including a Prioritized Experience Replay (PER) of size 100, a gamma value of 0.01, and training for a duration of four epochs. The aim was to investigate the impact of widening the prediction window on model performance. The results obtained from these experiments are presented in Table X.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Widening window | Sharpe ratio | Return |
| 1 | 1 | -0.48 | -10% |
| 2 | 3 | -0.16 | -4% |
| 3 | 5 | 0.48 | 11% |
| 4 | 7 | 0.37 | 8% |
| 5 | 10 | -0.42 | -10% |
| 6 | 14 | **0.73** | **14%** |

Table X: Prediction strategies results

Upon analyzing the results, it is evident that there is no clear evidence to support the hypothesis that a longer prediction window leads to superior performance. In fact, the outcomes demonstrate significant variability across different window sizes. For instance, with a widening window of one day, the model achieved a negative Sharpe ratio of -0.48, resulting in a loss of 10%. Similarly, a window size of 10 days yielded a negative Sharpe ratio of -0.42 and a loss of 10%. On the other hand, a window size of five days exhibited a positive Sharpe ratio of 0.48, generating a return of 11%.

These divergent results indicate that the optimal prediction window for this particular model is not necessarily longer. It underscores the importance of carefully selecting the appropriate window size, considering factors such as the level of noise in short-term predictions versus capturing longer-term trends. Moreover, these findings highlight the complexity of predicting asset returns and the necessity of further research to elucidate the relationships between prediction window size, model performance, and market dynamics.

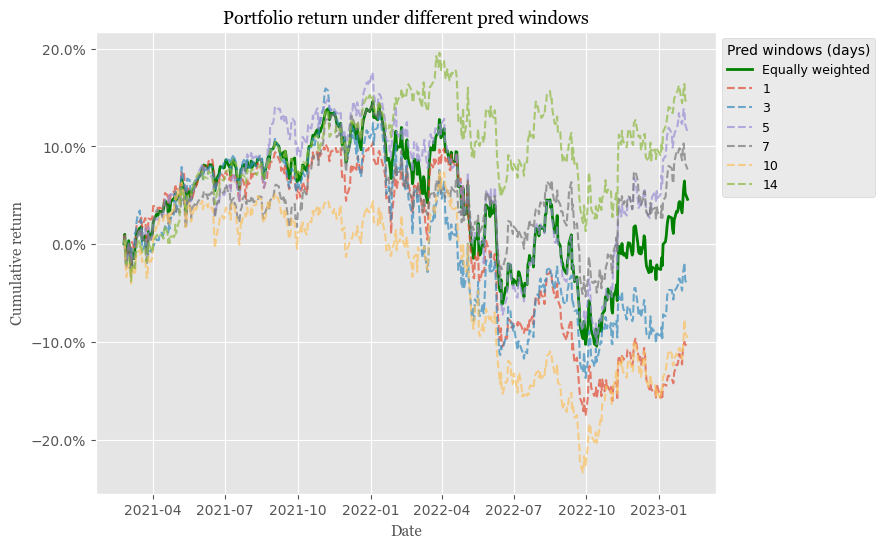


Figure X: Portfolio return under different prediction windows

## CHAPTER 5. CONCLUSION

# References

Agrawal, A. и др., 2019. Differentiable convex optimization layers.

Black, F. & Litterman, R., 1992. Global Portfolio Optimization. *Financial Analysts Journal*, Том 48, pp. 28-43.

Busbridge, D., Sherburn, D., Cavallo, P. & Hammerla, N. Y., 2019. Relational Graph Attention Networks.

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*.

Costa, G. & Iyengar, G. N., 2022. Distributionally Robust End-to-End Portfolio Construction.

Donti, P. L., Amos, B. & Kolter, J. Z., 2019. Task-based end-to-end model learning in stochastic optimization.

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.

Huang, Y., Capretz, L. F. & Ho, D., 2021. Machine Learning for Stock Prediction Based on Fundamental Analysis. *IEEE Symposium Series on Computational Intelligence (SSCI)*.

Jaeger, M. & Marinelli, D., 2022. Network diversification for a robust portfolio allocation.

Markowitz, H., 1952. Portfolio Selection. *The Journal of Finance*, Том 7, pp. 77-91.

Matsunaga, D., Suzumura, T. & Takahashi, T., 2019. Exploring Graph Neural Networks for Stock Market Predictions with Rolling Window Analysis. *arXiv*.

Mokhtari, S., Yen, K. K. & Liu, J., 2018. Effectiveness of Artificial Intelligence in Stock Market.

Schaul, T., Quan, J., Antonoglou, I. & Silver, D., 2016. Prioritized Experience Replay.

Taylor, S. M. & Cerbo, L. D., 2019. Graph Theoretical Representations of Equity Indices and their Centrality Measures.

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)*.

Veličković, P. и др., 2018. Graph Attention Networks.

## Appendix A. ETF Description

| **Ticker** | **Name** | **Capitalization (bn)[[3]](#footnote-3)** | **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 |
| 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 |
| 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[[4]](#footnote-4): employed for the relative rotation graph is as follows:

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.
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.

This algorithm allows for the creation of a relative rotation graph, which provides a visual representation of the relative strength and momentum of various stocks compared to a benchmark.

1. https://www.man.com/maninstitute/factors [↑](#footnote-ref-1)
2. https://www.relativerotationgraphs.com/about/the-company [↑](#footnote-ref-2)
3. As of 25.05.2023 [↑](#footnote-ref-3)
4. https://msw.flxn.de/tool/sector/ [↑](#footnote-ref-4)