AN ARTIFICIAL NEURAL NETWORK APPROACH TO DYNAMIC PORTFOLIO OPTIMIZATION

XAVIER HUIJTS

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

STUDENT NUMBER

2108520

COMMITTEE

dr. Ç. Güven
dr. B. Özgöde Yigin

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

January 15, 2023

WORD COUNT

8972

ACKNOWLEDGMENTS

A word of gratitude to dr. Güven for the guidance and support during my thesis. In addition, I sincerely want to thank dr. Lucy Tepla (INSEAD), Josh Logan (L&G Investment Management), and Jean-Paul Jaegers (Barclays) for their help and advice.

AN ARTIFICIAL NEURAL NETWORK APPROACH TO DYNAMIC PORTFOLIO OPTIMIZATION

XAVIER HUIJTS

Abstract

In this experimental study, we explore the potential benefits of employing *Multi-Layer Perceptron* (MLP) and *Simple Recurrent Neural Network* (SRNN) architectures in predicting one-month-ahead portfolio weights. Conventionally, one-month-ahead portfolio weights are indirectly derived by firstly predicting returns (and volatility in fewer cases) of the considered assets which are subsequently used for portfolio construction. This approach imposes downsides when considering multiple asset classes (e.g., stocks and bonds). Therefore, we propose a methodological approach to directly predict portfolio weights. This method involves modeling the relationship between the US economic state (at time t) and historically well-performing portfolios (at time $t+1$). Two assets are considered for building a portfolio: a large US stock and bond index. The dataset comprises 31 years of data (1990-2021) and includes indicators related to the US economy and financial markets. To incorporate recent information into the training set, a rolling-window is used. The MLP and SRNN models are evaluated from a model and strategic portfolio perspective. From a model perspective, the SRNN demonstrates superior performance compared to MLP, showing lower mean absolute error and greater consistency across predictions. For portfolio evaluation, the machine learning portfolios are compared to the *equally weighted benchmark* portfolio. Both the MLP and SRNN portfolios outperform the benchmark in terms of cumulative returns. However, the MLP exhibits a relatively low risk-adjusted return (Sharpe ratio), caused by high variance across returns. In contrast, the SRNN portfolio produces solid returns, while maintaining relatively low variance, resulting in a superior Sharpe ratio compared to the benchmark portfolio. Despite this, the lack of statistical significance leaves us with insufficient evidence to deem SRNN as a valuable portfolio prediction tool for investors. Hence, further model improvement, evidence, and robustness testing are imperative to test its practical applicability.

1 DATA SOURCE, ETHICS, CODE, AND TECHNOLOGY STATEMENT

For the purpose of this study, 18 different time-series datasets are extracted and used. Except for the stock index (S&P500), all datasets are extracted from the open-source database of the *St. Louis Federal Reserve Bank* (FRED). For each dataset extracted from FRED, a URL is provided in Appendix E (page 45). The stock index (S&P500) dataset is extracted from the *Center for Research in Security Prices* (CRSP). CRSP is a closed source database and is accessed under the license of the *Wharton Research Data Services* (WRDS), facilitated by Tilburg University. This data will not be disclosed. Both FRED and CRSP retain ownership of the respective datasets during and after the thesis. This study has not been subject to data collection from human or animal participants.

ChatGPT (version: January, 2022) is used in minor parts of the thesis to re-phrase text written by us. All code and figures are created by us. Visit Appendix A (page 41) for a brief description of the Python notebooks and Appendix B (page 42) for all the libraries used. Visit GitHub to access the Python notebooks.

2 INTRODUCTION

Recent studies have demonstrated the effectiveness of machine learning models in predicting out-of-sample portfolios, outperforming the selected benchmarks. In the portfolio optimization literature, it is common the use a monthly time-horizon. This means that features are used to predict the one-month-ahead optimal portfolio. In the realm of investing, a portfolio can be described as collection of financial assets (Branke et al., 2009).

Existing studies predominantly concentrate on the integration of machine learning applications in the stock market (Santos et al., 2022), while adding alternative asset classes to a portfolio significantly ameliorates risk-adjusted return (Arouri et al., 2014). Few studies also include additional asset classes to their investment universe, such as bonds and commodities. These studies address the fundamental *asset allocation* problem in portfolio management: finding the optimal distribution of capital across asset classes (Tütüncü & Koenig, 2004). For instance, an individual may choose to allocate 60% to stocks and the remaining 40% to bonds.
As presented in equation 1, a portfolio (denoted as $\boldsymbol{p}$) is essentially a vector of portfolio weights. Portfolio $\boldsymbol{p}$ includes N different asset classes, each assigned a weight (w_i) quantifying the percentage portfolio contribution. In this study, portfolio $\boldsymbol{p}$ adheres to two constraints. Firstly, each asset class

i possesses a positive weight (w_i) within the range of [0, 1], allowing for long-positions[1] only. Secondly, the sum of all weight (w_i) equals 1.

$$\boldsymbol{p} = \{w_1,\ w_i,\ ...,\ w_N\} \tag{1}$$

$$\text{where } 0 \leq w_i \leq 1 \qquad \text{and} \qquad \sum_{i=1}^{N} w_i = 1$$

Harry Markowitz (1952) developed the building blocks of portfolio optimization during the 1950s, suggesting that assets should be selected based on the assets' expected return and risk (measured by combined variance across asset returns, influenced by inter-asset correlation). In the decades following, research intensified on future asset return and risk prediction. Traditionally, linear models such as linear regression and ARIMA models are used to predict asset returns (J. Yu & Chang, 2020). More recently, machine learning models gained popularity in addressing this task, consistently outperforming the linear model (Götze et al., 2023). Superior relative performances of the machine learning models may be justified by its ability to model complex and non-linear relationships between the features and target (Liang et al., 2022; J. Yu & Chang, 2020). This is in harmony with the financial literature, which suggests that financial markets (e.g., stock and bond markets) are complex and non-linear systems (Hinich & Patterson, 1985; Hommes, 2001; Inglada-Perez, 2020).

In studies which consider more than one asset class, it is customary to utilize an index for representing each asset class, as opposed to individual assets (such as a single stock like Apple). An index monitors the collective return of similar assets; for instance, the S&P500 index tracks hundreds of US stocks. In studies involving multiple asset classes, the following steps are generally included:

(i) For each index, a model is trained with monthly economic features at time t and the index return (target) at $t + 1$.

(ii) Feed the economic features (at time t) into the trained model to (out-of-sample) predict the index return (at time $t + 1$). Follow this step for each month in the test set.

(iii) Use the predicted returns to construct a portfolio for time $t + 1$. Most studies include simple strategies which do not require modeling techniques. For example, equal weights are assigned to each asset with a positive return prediction.

[1] The usage of leverage and short-positions are out of scope in this study.

(iv) Finally, the performances of the predicted portfolios are compared to a benchmark portfolio over an extended time-period, and evaluated by the Sharpe ratio (Sharpe, 1966), assessing the portfolio's risk-adjust return.

This approach has been proven to work but has an inherent drawback. When multiple asset classes are considered, this process becomes complex and (potentially) computationally expensive. This complexity arises due to the necessity of constructing distinct separate return prediction models for each considered asset class. Moreover, in case expected volatility and inter-asset correlation are also predicted (in addition to return), the required number of models further escalates. In this study, we suggest an alternative approach that may mitigate this drawback. Our objective is to directly predict the optimal portfolio weights in one model, including the following steps:

(i) For each month in the selected time-period, retrieve the best performing portfolio that maximized the Sharpe ratio – this is a simple optimization problem (no machine learning techniques are involved).
Example: we consider an investment universe of two assets: (i) a stock and (ii) bond index. In December 2021, the portfolio maximizing thee Sharpe ratio consisted of 67% stock and 33% bond index, denoted as {0.67, 0.33}.

(ii) Train a model with economic features at time t and the best-performing portfolios (target) at $t + 1$.
Example: the target for the month December 2021 is {0.67, 0.33} - a vector of two portfolio weights for the stock and bond indices.

(iii) Feed the economic features (at time t) into the trained model to (out-of-sample) predict optimal portfolio weights (at time $t + 1$). Follow this step for each month in the test set.

(iv) Finally, akin to the conventional approach, the predicted portfolios are compared to a benchmark portfolio over an extended time-period, employing the Sharpe ratio metric.

By adopting this methodology, the model learns which portfolios performed historically well corresponding to certain economic states. Diverging from the conventional method, our approach directly predicts the one-month-ahead portfolio, with the target encompassing the weights for both stock and bond indices. The more accurate the model can predict

the target, the closer it approaches the optimally defined portfolio weights. Equation 2 illustrates our approach. X represents the features (at time t), inputted into model f, to generate the one-month-ahead optimal portfolio $\boldsymbol{p}$ (at time $t+1$). As denoted in equation 3, the output is a vector of two weights; each weight signifying the portfolio contribution of the respective indices.

$$\boldsymbol{p}_{t+1} = \mathbf{f}(\mathbf{X}_t) \tag{2}$$

$$\boldsymbol{p}_{t+1} = \{W_{\text{Stock Index}}, \quad W_{\text{Bond Index}}\} \tag{3}$$

The social relevance is in hands of institutional and retail investors, who may enhance their market positions with machine learning techniques. Despite the prevailing use of traditional linear methods in portfolio management (Mirete-Ferrer et al., 2022), potential benefits arise for institutional portfolio managers, like pension funds, managing crucial capital. If more successful research on machine learning applications in portfolio optimization arise, pension funds may integrate machine learning techniques into their asset allocation processes. Additionally, retail investors may also benefit from machine learning methods. Typically having limited access to professional financial advisory, retail investors may rely on false or ineffective financial market information. Moreover, financial data and computational resources became more accessible to retail investors in recent years. These developments open avenues for to systematically leverage machine learning tools, potentially enhancing investment decisions. These motivations and developments lead to the following research question:

> *To what extend can investors benefit from Multi-Layer Perceptron and Simple Recurrent Neural Network models in predicting one-month-ahead optimal portfolio weights?*

As described by Bouwmans (2022), prior to developing these models, four decisions must be made. (i) The investment universe: the considered assets during a certain time-period, (ii) the time-frequency for predicting the portfolio weights, (iii) the selected indicators for feature extraction, (iv) the machine learning algorithms. The selected investment universe consists of a stock and bond index spanning the period from 1990 to 2021 (see table 1), tracking a large number of US stocks and bonds respectively. An extra low-risk bond index is selected as stocks are considered as relatively risky assets, which becomes evident in Appendix C (page 43). The portfolio weights are predicted on a monthly basis. The selected features comprise macro-economic and financial market related indicators, detailed in Appendix D (page 44). Lastly, the selected machine learning algorithms

are the *Multi-Layer Perceptron* (a *feedforward neural network* architecture) and *Simple Recurrent Neural Network* architectures.

Table 1: Investment universe

Asset Class	**Index**
Stocks	S&P 500
Bonds	ICE BofA AAA US Corporate Index Total Return Index Value

* Note: "AAA" is the best credit rate a bond may receive, suggesting a relatively safe investment.

The selected models serve the purpose of predicting out-of-sample portfolio weights, where each weight can be a number between 0 and 1. Hence, this study adopts a *regression* approach. As the model incorporates multiple features and produces two output values, our approach is termed a *multivariate multiple regression*. These predicted portfolio weights provide direct guidance for investment decisions, determining the proportional allocation of the portfolio across the indices.
The assessment of the portfolios generated by the models entails a comparison with a benchmark portfolio, evaluated by the *Sharpe ratio*. Consequently, this leads to the exploration of the following sub-questions:

RQ1 *To what extend does the Simple Recurrent Neural Network model improve or deteriorate the performances compared to the Multi-Layer Perceptron (based on MAE and R^2)?*

RQ2 *To what extend do the predicted machine learning portfolios out- or underperform the benchmark portfolio in terms of Sharpe ratio?*

To answer both sub-questions, we train a *Multi-Layer Perceptron* (MLP) and *Simple Recurrent Neural Network* (SRNN) to predict out-of-sample portfolio weights. The data set is separated into 6 rolling-windows, to enhance recency in the training set.

Both sub-questions evaluate the models' output, but from a different perspective. The first sub-question is focused on comparing the performances of the ANN models using conventional model evaluation metrics. These metrics provide information on how accurate the models' predictions are compared to the ground truth (MAE), and its ability to draw relationships between the features and target (R^2). The distribution of *absolute errors* is also considered.
The primary focus of this study involves utilizing the complete feature set. However, we also partition the full feature set into two subsets for model evaluation. This allows us to assess the incremental advantages of adding macro-economic and financial market features in addition to the

more straightforward approach of solely relying on historical stock and bond index data.

The second sub-question evaluates the predictions from a strategic investment perspective. The portfolio undergoes dynamic re-balancing each month based on the updated predictions from the ANN models. The out-of-sample performances are subsequently compared to a widely used benchmark portfolio, the *equally weighted* (EW) portfolio. The EW portfolio maintains a fixed allocation of 50% to both the stock and bond indices (0.5, 0.5), making it a static portfolio with constant weights. This comparison serves to evaluate whether machine learning techniques offer advantages over the simpler and computationally cheap EW benchmark portfolio.

To our knowledge, novelty lies in the methodological approach of predicting out-of-sample portfolios. Unlike conventional methods that indirectly predict future portfolio weights based on return predictions for multiple assets, our approach stands out by directly leveraging features to predict one-month-ahead portfolio weights. This approach may reduce the complexity of predicting optimal portfolio weights and potentially required computational resources by minimizing the number of models needed. The experimental and unorthodox nature of this study requires creative solutions for developing the target, model architecture, and evaluation metrics.

3 RELATED WORK

Machine learning is the latest iteration in the "longstanding quantitative investing paradigm" and an emerging topic in the field of portfolio management (Israel et al., 2020, p.1). Unsurprisingly, decision-making in portfolio choice is still dominated by conventional methods (Mirete-Ferrer et al., 2022). The staggering modernization of portfolio optimization methods can partly be explained by the lack of available data in finance (Chakravorty et al., 2018).

To our knowledge, this study's methodology includes novel elements. Therefore, multiple related studies are described. Additionally, we introduce the features used in the current literature.

3.1 *Investment universe and feature selection*

Santos et al. (2022) reviewed the academic applications of artificial intelligence (AI) in the domain of portfolio optimization and found abundant research on stock market applications, however, a deficiency of alternative asset classes. This is surprising, as adding alternative asset classes to a traditional portfolio significantly ameliorates risk-adjusted return (Arouri et al., 2014). This can be explained by the lower correlation across asset classes, resulting in a more diversified portfolio.

The assets considered in this study are a stock and bond index. The target represents the portfolios that historically maximized the Sharpe ratio. Therefore, understanding what factors affect the portfolio Sharpe ratio is essential for selecting predictive features. The portfolio Sharpe ratio is essentially a function of the stock and bond index return and its combined volatility.

Stock returns are influenced by a wide range of factors, such as developments in political events, commodity prices, and interest rates. More specifically, abundant literature exists on the effect of macro-economic factors on asset returns (Gunasekarage et al., 2004). Early studies addressed several macro-economic and financial market indicators significantly affecting stock prices such as inflation and interest rates (Chen et al., 1986; Fama, 1981). Likewise, bond returns can partly be explained by macro-economic and financial market factors, such as unemployment, economic output, and government bond rates (Fama & Bliss, 1987; Fricke & Menkhoff, 2015). According to Abouseir et al. (2020) and Chang and J. Yu and Chang (2020), the utilization of macro-economic features to realize access portfolio returns

is not fully exploited yet, certainly in the realm of machine learning. In their studies, they demonstrated the predictive power of macro-economic indicators in constructing out-of-sample portfolios. In addition to macro-economic data, Abouseir et al. (2020) also extract features from financial market indicators such as government bond and currency rates.

3.2 *Portfolio prediction*

The purpose of optimizing portfolio weights is either to increase return and/or reduce risk (Habbab & Kampouridis, 2024). Unsurprisingly, most of the observed literature related to machine learning applications in portfolio optimization are to predict expected return and/or volatility.

Research on machine learning applications in predicting stock prices is rich. For example, Kaczmarek and Perez (2022) leveraged the Random Forrest (RF) algorithm using stock-level and macro-economic features to predict the one-month-ahead returns of individual stocks in the S&P500 index. Each month, the top 10% stocks with the highest predicted return were selected to construct a portfolio. The portfolios leveraging RF outperformed the selected benchmark portfolio with 16.5% to 18% increases in *Sharpe ratio*. Gu et al. (2020) performed a similar study, however, also employed a FNN model (next to the RF). In their study, both the RF and FNN outperformed the linear models. In addition, the FNN model outperformed the RF model.

A selective number of studies consider, besides stocks, additionally asset classes. These studies generally apply index investing strategies. Indices track a large number of similar assets, reducing security-specific risk, becoming an approximation for the wider economy.

Abouseir et al. (2020) incorporated both linear (LR-Ridge and LR-Lasso) and non-linear (RF and XGBoost) models to predict one-month-ahead portfolios. They considered a stock, bond, and commodity index respectively. The study includes a regression (predicting returns) and binary classification (predicting positive or negative return sign) approach. Both macro-economic and financial market indicators were considered for feature extraction. Different to other studies, they used both monthly and daily data for extracting monthly features. For indicators with daily data, 6 different statistical features were extracted based on the daily (intra-month) returns: (i) mean, (ii) standard deviation, (iii) Kurtosis, (iv) Skewness, (v) 25th percentile, and (vi) 75th percentile. This approach proves attractive for our study, as it enables the extraction of a large number of potentially predictive features.

For each month, they construct a portfolio of exclusively the indices with a positive predicted return. Results on the linear models were poor but the non-linear models significantly outperformed the benchmarks used in terms of Sharpe ratio.

Pinelis and Ruppert (2022) predicted both one-month-ahead return and risk using RF and Elastic Nets models, using macro-economic features. Subsequently, the return and volatility (akin to variance) predictions were used to retrieve the optimal combination between the market index (a mix of US Stock indices) and the risk-free asset (zero return and risk). Likewise, their results benefited from non-linear models, and outperformed the benchmark portfolio.

Studies employing ANN models and simultaneously considering multiple asset classes are scarce. The works of J. Yu and Chang (2020) and Obeidat et al. (2018) are particularly relevant to our research, as they meet both conditions.

J. Yu and Chang (2020) used a FNN to model the relationship between macro-economic indicators and one-month-ahead prices, considering a stock, bond, and real-estate index. Firstly, they use the statistical (GARCH) feature engineering technique to transform the time-series of the macro-economic indicators. Subsequently, after model training, the (transformed) features are fed into the FNN model to predict one-month-ahead returns of the selected assets. The predicted returns are subsequently used as input in a mean-variance model to derive optimal portfolio weights. The mean-variance model is a portfolio optimization technique developed by Harry Markowitz, and retrieves optimal portfolio weights based on the expected return, volatility, and inter-asset (class) correlation. Likewise, the out-of-sample results of the FNN outperformed all benchmarks.

Other popular ANN's for time-series are *Recurrent Neural Networks* (RNN), which are developed to learn sequential patterns. This characteristic is ideal for time-series problems. A popular RNN architecture is the *Long Short-Term Memory* (LSTM) model. Obeidat et al. (2018) applied the LSTM model to predict out-of-sample portfolios considering various asset classes (stock, bond, real-estate and commodity indices). In addition to future return, they also predicted future volatility, and inter-index correlation. These three components are all necessary inputs for the mean-variance optimization technique. In this study, *Principal Component Analysis* (PCA) was performed to reduce the feature dimensionality (from 387 to 70 features),

while maintaining most of the information. The LSTM model significantly outperformed the benchmark portfolios in terms of return and *Sharpe ratio*.

3.3 *Model evaluation metrics*

As described, the reviewed studies use machine learning models to predict out-of-sample returns and in fewer cases, volatility, and inter-asset correlation. Most studies only consider *mean squared error* (MSE) or *mean absolute error* (MAE) for model evaluation. Opposed to these studies, we employ machine learning techniques to predict out-of-sample portfolio weights. Given this difference, utilizing results from these studies as the baseline for our models is not possible.

3.4 *Portfolio evaluation metrics*

The Sharpe ratio (Sharpe, 1966) serves as a widely employed metric for assessing and comparing portfolio performances. The *Sharpe ratio* measures the risk-adjusted return in excess of the risk-free rate, often represented by the *US 3-month Treasury bill* - conventionally perceived a 'risk-free' (Morningstar, 2023).

To measure out-of-sample predicted portfolios, relying solely on absolute *Sharpe ratios* is insufficient. While predicted portfolios may exhibit an impressive *Sharpe ratio*, this may stem from favorable market conditions rather than the inherent quality of the model. Likewise, low *Sharpe ratios* do not necessarily indicate a sub-optimal model. Hence, the *Sharpe ratio* of the predicted portfolios is compared to a benchmark portfolio, over an extended time-frame.
The reviewed related papers have one overlapping benchmark portfolio, which is the *equally weighted* (EW) portfolio. According to Maillard et al. (2010), the EW portfolio is also frequently used in practice. Characterized as a passive investment strategy, the EW portfolio systematically allocates equal weights to each asset in the investment universe, thus, does not necessitate any modeling techniques.

4 METHOD

In this section, the underlying data and methods used are described. This includes a brief description of the data. After, information is provided on how we calculate the target, transform the indicators, and extract features from the transformed indicators. Finally, information on the feature subsets, selected models, hyper-parameter tuning, and evaluation metrics is provided. In figure 1, the methodical steps are ordered in logical sequence.

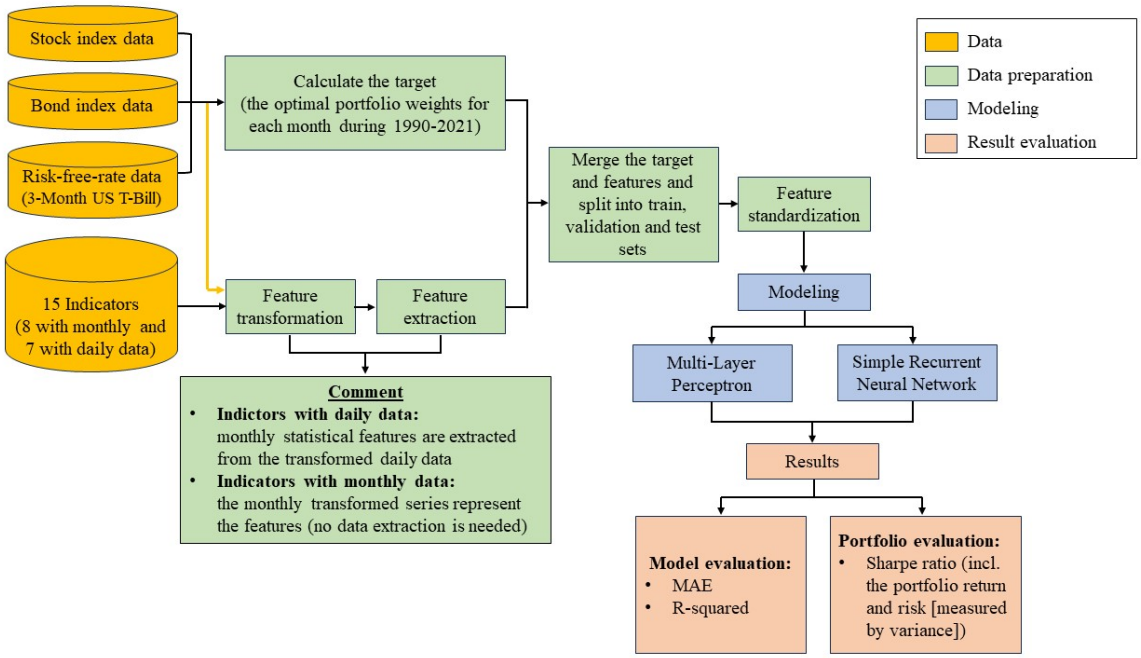

Figure 1: Flowchart. The yellow arrow is there because historical data on the stock index, bond index, and risk-free rate also function as predictive features.

4.1 *Data set description*

In this study, we exclusively use time-series data during the period 1990-2021. To calculate the target (historical maximum Sharpe portfolios), we use return data from the stock index, bond index, and risk-free rate.

In total, data from 18 indicators are collected for feature extraction. Not all related papers are transparent with regard to the selected features. However, J. Yu and Chang (2020) and Abouseir et al. (2020) provided a list of macro-economic and financial market indicators respectively, which

together provide a holistic view on the US economy and financial markets. To limit the number of features (considering the small data set), only indicators related to the US are selected. For 8 indicators, we have monthly data (one observation per month) and for 10 indicators daily data (one observation per weekday). If available, daily data was preferred over monthly data, as it allows for the extraction of a greater number of features per month.

The monthly data is complete and clean. For the daily indicator data extracted from FRED, there are some missing values on a consistent basis (approximately 10 days for each year). As the missing values are evenly distributed across months (non-clustered), and the daily series are solely utilized to derive monthly statistical features, we anticipate that the missing values will not significantly impact results. Thus, the missing instances are removed instead of applying imputation techniques.

In section 4.3, the monthly and daily time-series are transformed accordingly. In section 4.4, we explain how features are extracted from the transformed series.

4.2 *Calculating the target*

Earlier, we introduced an alternative approach to predict the one-month-ahead portfolio. Conventionally, the target comprises the historical returns of an asset, whereas our target comprises the optimal historical portfolio for each month during 1990-2021. We define the optimal portfolio as the portfolio that maximized the *Sharpe ratio* for the given month.

As visible in equation 4, the portfolio's *Sharpe ratio* is calculated by dividing the monthly return by the standard deviation. R is simply the realized portfolio return in the given month. σ is calculated by taking the standard deviation over the daily (intra-month) return series r, which is an indication of risk. Due to discrepancies in daily publishing of the stock and bond indices (for some months, bond returns are also published during the weekend), we decided to use the overall monthly portfolio return instead of the average daily return series. Consequently, the discrepancies may lead to minor estimation error in the denominator; the portfolio standard deviation.

$$\text{Sharpe Ratio}_{\text{monthly}} = \frac{R}{\sigma_r} \tag{4}$$

Figure 2 shows the process of finding the optimal portfolio for a given month. The *Sharpe ratio* is calculated for each weight combination of the stock and bond index, with a step-size of 1% (to make it a discrete/finite problem). Finally, the portfolio maximizing the *Sharpe ratio* is selected as the target for the given month.

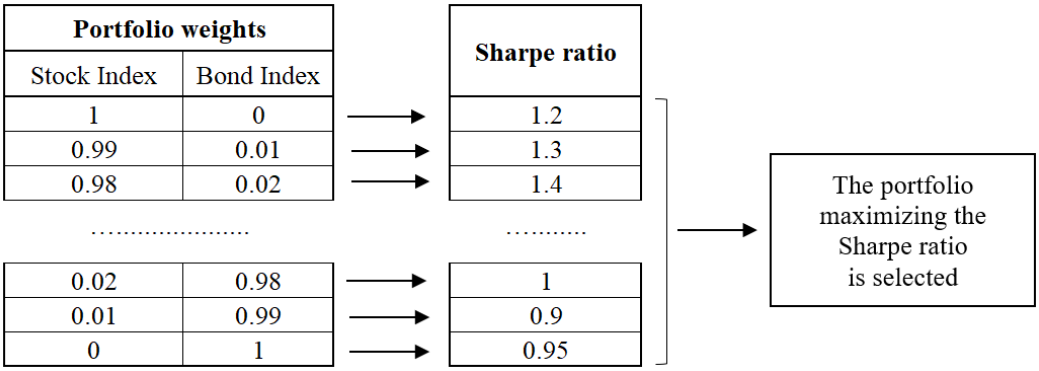

Figure 2: Monthly process of retrieving the historical optimal portfolio weights (target).

The decision to use this *grid-search* approach is taken to enhance accuracy (preventing the use of inaccurate targets for model training). With larger datasets, analytical solutions can be computationally expensive and optimization techniques may be considered. In table 2, the calculated optimal portfolios for the first and last three months are visible.

Table 2: The optimal historical portfolios (target) for the first and last three months

	Target	
Month & year	$w_{Stockindex}$	$w_{Bondindex}$
01-1990	0.45	0.55
02-1990	0.0	1.0
03-1990	0.56	0.44
10-2021	0.64	0.36
11-2021	0.0	1.0
12-2021	0.67	0.33

4.3 *Indicator data transformation*

Stationarity is a time-series property where the parameters are static over time (Lazzeri, 2020). Most economic time-series are non-stationery and models must be employed to estimate potential seasonality (Lazzeri, 2020; R Rosca, 2011). To optimally use our ML models, the features must be stationary. As this study considers similar indicators as Abouseir et al. (2020), the same approach is applied to transform the indicators accordingly and test for stationarity.

Depending on the indicator, we have daily or monthly time-series data. Let $X_t^{(i)}$ be the observation of indicator i at time t. In this case, t can be either a day (for daily data) or a month (for monthly data). Four methods are considered to transform the indicators into stationary series:

- **Method 1:** $X_t^{(i)}$
  The raw (unchanged) time-series of indicator i
- **Method 2:** $\Delta X_t^{(i)} = X_t^{(i)} - X_{t-1}^{(i)}$
  The simple difference between the observation at time t and the previous observation (t-1)
- **Method 3:** $\Delta(X^2)_t^{(i)} = (X_t^{(i)})^2 - (X_{t-1}^{(i)})^2$
  The squared difference between the observation at time t and the previous observation (t-1)
- **Method 4:** $\Delta\ln(X)_t^{(i)} = \ln\left(\frac{X_t^{(i)}}{X_{t-1}^{(i)}}\right)$
  The log-normal change of the observation at time t and the previous observation (t-1)

We have macro-economic and financial market indicators. The indicators can be divided into 6 sub-groups: (i) assets, (ii) currency (FX), (iii) interest rates, (iv) inflation, (v) economic output, and (vi) saving rate.

The method providing the best stationary results over each sub-group is selected, while also considering established literature. To test each generated series for stationarity, the *Dickey-Fuller test* is applied - the *adfuller* function from the *statsmodels* library is leveraged for this purpose. The test provides a t-statistic and p-value; the lower the t-statistic and p-value, the greater the null-hypothesis is rejected. In case the p-value is lower than the selected alpha level of 0.05, we consider the time-series to be stationary. See in the below the results per sub-group summarized (see Appendix F [page 46] for more details):

- **Assets:** *method 4* is selected to transform the *assets* (return) indicators, considering the results and literature (suggesting assets are commonly transformed using log-normal returns) (Black and Scholes, 1973, as cited in Abouseir et al., 2020).
- **Currency (FX):** *method 4* is selected to transform the currency (FX) indicator, despite *method 3* provided best stationary results. This decision is made as *method 4* also provides significant results (p-value<0.05) and is suggested by the literature (Black and Scholes, 1973, as cited in Abouseir et al., 2020).
- **Interest rates and inflation:** *method 2* is selected to transform the interest rate and inflation indicators, providing the best stationary results. These results align with the literature (Hull and White, 1990, as cited in Abouseir et al., 2020; Vasicek, 1977, as cited in Abouseir et al., 2020).
- **Economic output:** *method 2* is selected to transform the economic output indicator, provided the best stationary result.
- **Saving rate:** *method 4* is selected to transform the saving rate indicator, provided the best stationary result.

In figure 3, 4, and 5 (see next page), three examples are provided to show the effect of transforming the raw data into stationary series.

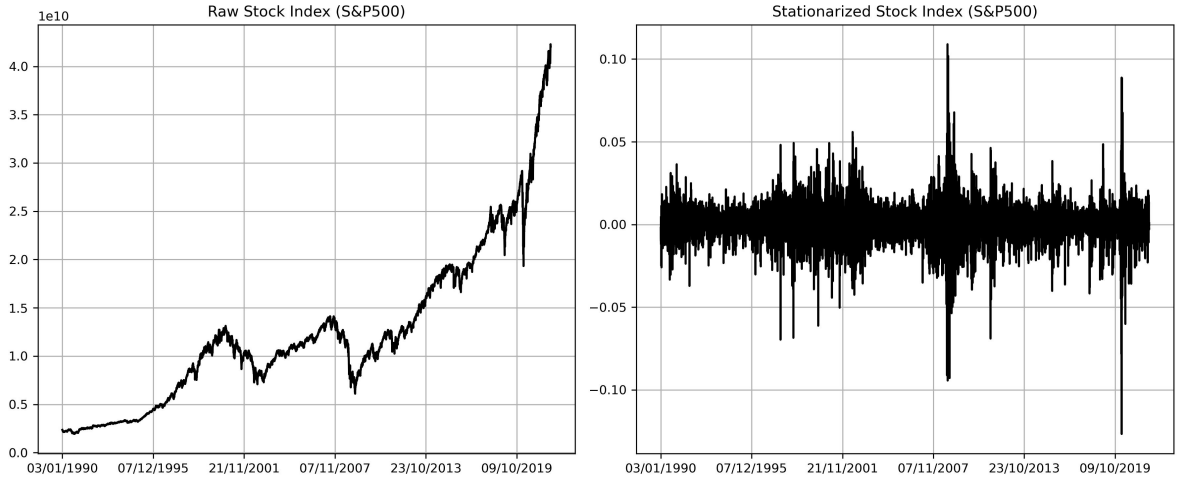

Figure 3: Stock Index: S&P500 (assets: method 4)

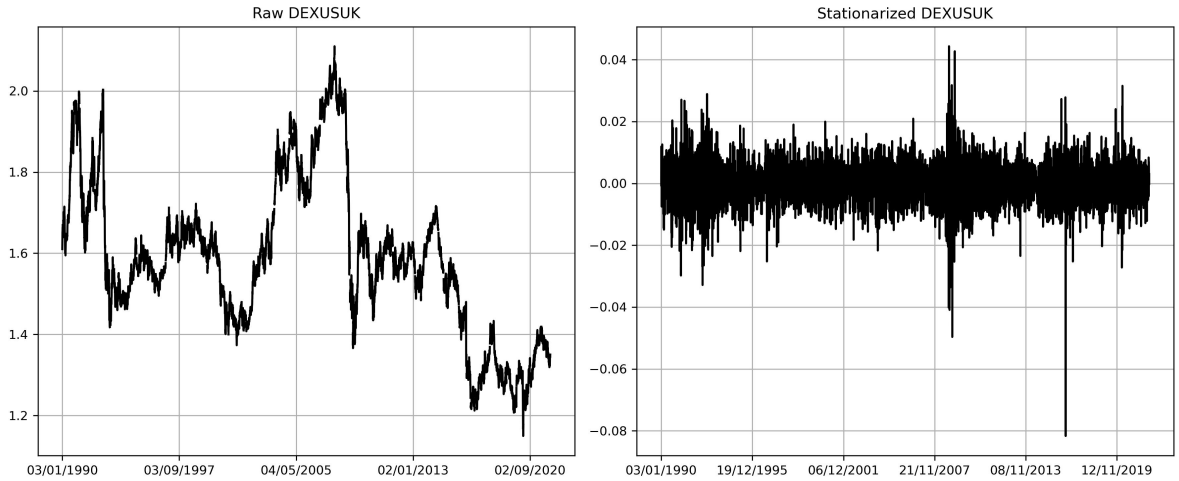

Figure 4: DEXUSUK (currency [FX]: method 4)

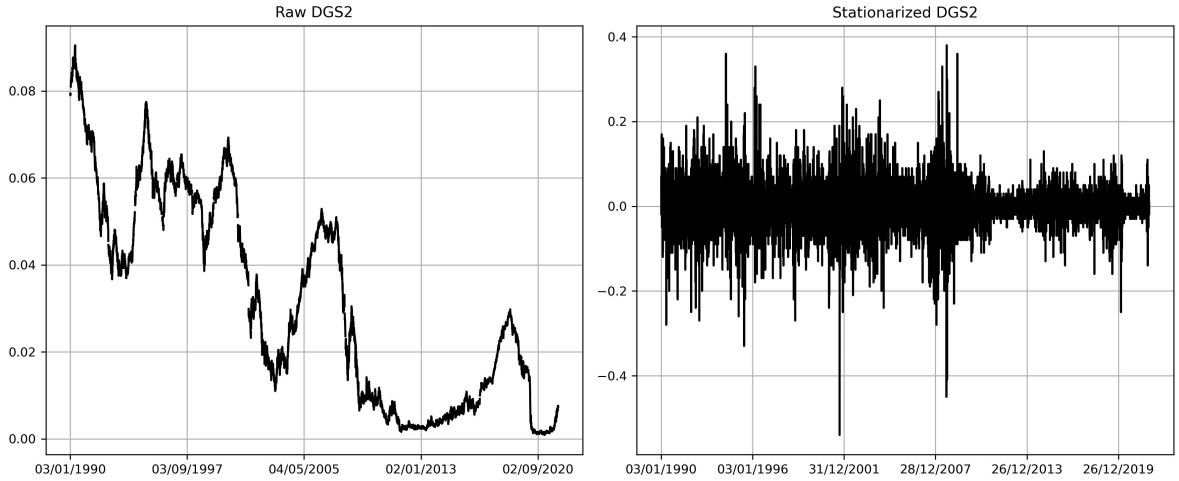

Figure 5: DGS2 (interest rate: method 2)

4.4 *Feature extraction and standardization*

Now that all (monthly and daily) indicators are transformed into stationary series, features can be extracted. Likewise for feature extraction, a similar approach is used as Abouseir et al. (2020). Depending on the publishing frequency of the indicators, we have monthly or daily data. For monthly indicators, there is one observation (feature) for each month, which is simply the observation for the given month in the stationary series. For indicators with daily data, 6 statistical features are extracted from the daily stationary series for each month. These 6 statistics are taken over the intra-month data, providing the following 6 features for each month:

(i) Mean

(ii) Standard deviation

(iii) Skewness

(iv) Kurtosis

(v) 25th percentile

(vi) 75th percentile

Visit Appendix G (page 47) for the libraries and functions used for each statistical feature. In total, 68 monthly features are extracted: 8 from the monthly stationary series and 60 (6 features * 10 indicators with daily data) from the daily stationary series. We perceive this number to be acceptable, considering that similar studies used similar number of features (and have even fewer instances).

The final step involves standardizing the features, allowing for consistent scaling across the features. The necessary statistics for each feature (mean and standard deviation) are calculated based on the training set data of the first window (1990-2010). Equation 5 illustrates the standard scaling formula for feature i, wherein each observation X (at time t) has the feature's mean (μ) subtracted, and is subsequently divided by the feature's standard deviation (σ). This process is applied to all 68 features.

$$X^{(i)}_{\text{scaled}} = \frac{X_t - \mu}{\sigma} \tag{5}$$

4.5 *Feature subsets*

The primary focus of this study involves utilizing the complete feature set. In addition to this, we also partition the full feature set into two subsets:

(i) The stock and bond index features (12 features)

(ii) The macro-economic and financial market features (56 features)

Besides using the full feature sets, model performances will be measured for each feature subset. This approach allows us to assess the incremental advantages of incorporating macro-economic and financial market features compared to the more straightforward approach of solely relying on historical stock and bond index data. To enhance performances, the hyper-parameters are individually tuned for each unique feature subset.

4.6 *Training and testing*

Using the period 1990-2021, we have 31 years of monthly data. Considering the sequential nature of the data, a sliding-window is used (see figure 6). The sliding-window shifts the training data in each window to ensure recent data being included in the training set. For each window, the hyper-parameters are separately optimized.

The rolling-window works as follows; in the first window, the data from 1990-2015 is used for training (1990-2010) and validation (2010-2015). After hyper-parameter tuning, the training and validation sets are merged for model training. This model is used to perform out-of-sample predictions on the first test set, which is 2016 (for the first window). In the second window, the training, validation, and test sets all move up with one year, continuing until the last year of testing, which is 2021.

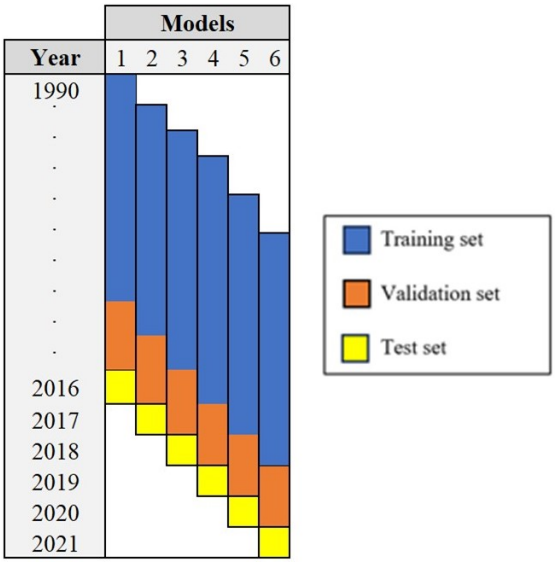

Figure 6: Rolling-window

4.7 *Model selection*

In this study, two *Artificial Neural Network* (ANN) architectures are employed. This decision is motivated by the relative strong performance of ANNs compared to linear models and to a lesser extend, its machine learning counterparts. Due to our novel approach, we decided to use standard ANN architectures. The selected architectures are the MLP and SRNN respectively.
MLPs are one-directional neural networks, as information is only forwardly transmitted through the network (Popescu et al., 2009). Besides the input and output layer, an MLP has at least one hidden layer. The MLP is trained by minimizing the error function with relation to the model's weights and biases (realized through forward- and back-propagation). In figure 7, a sample MLP architecture is presented with two (dense) hidden layers including both 8 neurons. For simplicity, 5 features are presented (in reality we used 68 features). The output layer signifies the predicted optimal portfolio weights (target).

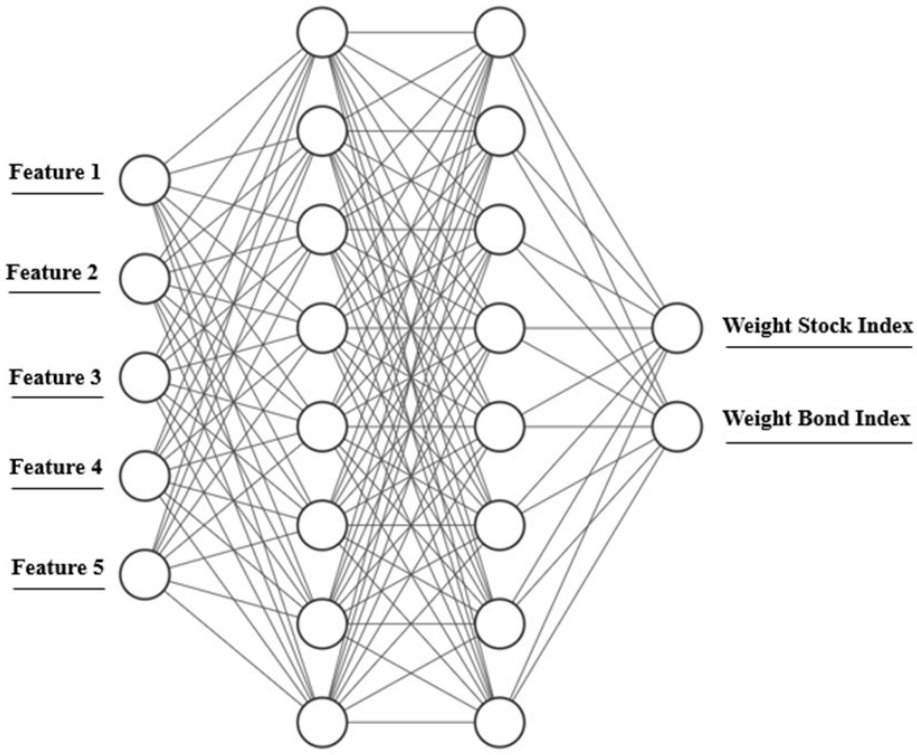

Figure 7: Sample MLP Architecture. The NN-SVG tool created by Alex Lenail (https://alexlenail.me/NN-SVG/) is used to draw the raw dense neural network.

Opposed to MLPs, RNNs are developed to learn sequential patterns, and can learn temporal behavior (Tölö, 2020). This characteristic makes RNNs well-suited for time-series problems, which are sequential by nature. Because RNNs maintain the temporal order of the time-series, it may reduce over-fitting according to Tölö (2020). This is advantageous for our study, considering the limited number of instances. There are multiple RNN architectures, with *Simple* RNN (SRNN), *Gated Recurrent Unit* (GRU), and *Long Short-Term Memory* (LSTM) as popular choices. As previously mentioned, the SRNN is selected in this study.
In figure 8, the underlying architecture of the SRNN is presented. X and y represent model's input and output respectively, while h represents the *hidden state*, which transmits information through the sequence.

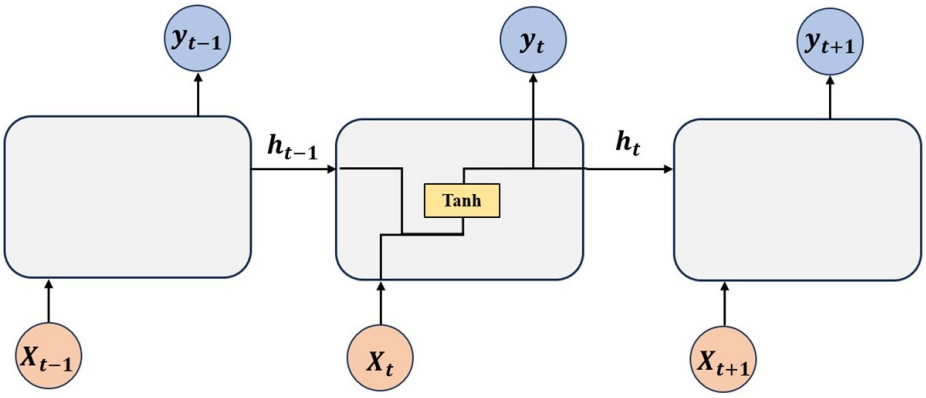

Figure 8: Simple Recurrent Neural Network Architecture. Inspired by Awan (2022).

For both neural network models, the *SoftMax* activation function is used in the output layer. *SoftMax* is ideal for our problem, since its properties match the properties of the target (re-visit equation 1 [page 3] for the portfolio properties). The mathematical notation of *SoftMax* is shown in equation 6. For each element i in vector Z received by the output layer, the exponential of element i is taken and divided by the sum of the exponentials of all elements in vector Z. This normalization process ensures each element to become positive and the entire output vector to sum up to 1 (Banerjee et al., 2020). In this study, the output vector comprises two elements (the portfolio weights), meaning $N = 2$.

$$\text{SoftMax}(Z_i) = \frac{\exp(Z_i)}{\sum_{j=1}^{N} \exp(Z_j)} \tag{6}$$

$$\text{where } 0 \leq \text{SoftMax}(Z_i) \leq 1 \quad \text{and} \quad \sum_{i=1}^{N} \text{SoftMax}(Z_i) = 1$$

4.8 *Hyper-parameter tuning*

Inherent to ANNs are numerous hyper-parameters which can be optimized. In table 3, the hyper-parameter candidates for each model are presented. For consistency purposes, the hyper-parameters tuned for the MLP and SRNN models are identical. The *ReLu* activation function is used in the dense hidden layers, while *Tanh* is used in the simple recurrent layer. The *SoftMax* activation function is used in the output layer. The number of epochs and batch size are static and predefined at 10 (epochs) and 32 (batch size) respectively.

With 31 years of monthly data, we have few instances. Obeidat et al. (2018) and J. Yu and Chang (2020) did not optimize the number of layers, and statically used 3 hidden layers. However, considering the different approach in this study and the size of the data, we consider either 2 or 3 hidden layers: {2, 3}. For the MLP, each hidden layer is a dense hidden layer. For the SRNN, the first hidden layer is a simple recurrent layer, followed by either one or two dense hidden layers.
An insufficient number of neurons in the hidden layer(s) may lead to under-fitting (due to a lack of complexity), while too many result in over-fitting (T. Yu & Zhu, 2020). The reviewed related papers did not provide candidate number of neurons. Therefore, we select several frequently used candidates: {8, 16, 32, 64, 128}. We allowed each hidden layer to have a unique number of neurons.

Certainly in deep learning, small datasets are prone to over-fitting (Feng et al., 2019). In computer vision problems, data can be expended by generating synthetic images (e.g., by image rotation), also called data augmentation. According to Oh et al. (2020), such methods are less popular with time-series problems, given its sensitivity to data modification. Thus, it is essential to consider methods that reduce the risk of over-fitting. A method which may mitigate the risk of over-fitting are dropout layers. A dropout layer randomly deactivates a predefined proportion of neurons during training, which is a form of regularization (Passos & Mishra, 2022). Three different dropout rates are considered: {0.0, 0.25, 0.5}.

We also consider batch-normalization, which normalizes the output of each hidden layer. This can improve computational speed and model performance (Bjorck et al., 2018). In contrast to the other hyper-parameters, batch-normalization is either *true* or *false* and applied after each hidden layer: {True, False}. The batch-normalization layer is located after the dropout layer, to prevent that information from the deactivated neurons

being passed through normalization statistics.

The selected optimization function is *Adam*, which is widely used in training neural network models given its adaptive learning rate and high performance (Bae et al., 2019). The default learning rate of *Adam* (in the *Keras* library) is 0.001 ("Adam - Keras Documentation", n.d.). In addition to this, one level below and above the default learning rate is considered: {0.0001, 0.001, 0.01}.

The *mean absolute error* (MAE) is the selected loss function and aims to minimize the spread between the predicted and actual portfolio weights. The *random optimizer* is used due the large number of possible hyper-parameter combinations (900 possible combinations). For each window, 100 trials are considered. *Grid search* is not selected considering the large hyper-parameter combination space and our limited access to computational resources. The *Bayesian optimizer* is not selected considering its difficulties with tuning the number of layers and neurons according to Wu et al. (2019).

Table 3: Hyper-parameter candidates

Hyper-parameter	**Candidates**
Number of hidden layers	{2, 3}
Number of neurons	{8, 16, 32, 64, 128}
Batch-optimization	{*True*, *False*}
Drop-out rate	{0.0, 0.25, 0.5}
Batch size	{32}
Learning rate	{0.0001, 0.001, 0.01}
Optimizer	{*Adam*}
Epochs	{10}

4.9 *Model evaluation*

The rolling-window provides 6 out-of-sample years, generating monthly predicted portfolio weights. For the purpose of evaluating model performance, two conventional key metrics, namely *mean absolute error* (MAE) and R^2, are selected.

The primary objective of the model is to minimize MAE. Subsequently, MAE is also used to compare the models, as it quantifies how accurate the

predicted portfolio weights are to the actual optimal weights (the *ground truth*). Equation 7 shows the mathematical notation of MAE, adjusted to our problem. While the predicted output comprises two values, MAE consolidates these values into a single metric. The *absolute error* for one observation is the average absolute difference between the actual optimal ($W_i^{Stockindex}$ and W_i^{Bond}) and predicted portfolio weights ($\hat{W}_i^{Stockindex}$ and $\hat{W}_i^{Bond}$). MAE performs this step for each out-of-sample observation and sums the *absolute errors*. N represents the total number of observations (12 [months] for each year), and i signifies the individual data points within the test set.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|W_i^{\text{Stock index}} - \hat{W}_i^{\text{Stock index}}\right| + \left|W_i^{\text{Bond index}} - \hat{W}_i^{\text{Bond index}}\right|}{2} \quad (7)$$

In table 4, results for one sample observation is provided. The actual optimal portfolio weights for the stock and bond indices are 60% and 40%, respectively. The predicted portfolio weights for the same indices are 90% and 10%, respectively.

Table 4: Example: Prediction results of one instance

	$W^{\textbf{Stock index}}$	$W^{\textbf{Bond index}}$
Actual	0.60	0.40
Predicted	0.90	0.10

The *absolute error* for the single observation in table 4 can be calculated by taking the average over the absolute differences between the actual and predicted weights:

$$\text{Absolute error} = \frac{|0.6 - 0.9| + |0.4 - 0.1|}{2} = \frac{0.3 + 0.3}{2} = 0.3 \quad (8)$$

The *mean squared error* (MSE) is intentionally omitted as it would reduce the overall error and hinder interpretation. This is because the predicted and actual weights are always in the range of [0, 1], and therefore, the error will always be in the range of [-1, 1]. Taking the square of a value this range may result in a diminished error. For instance, in case the error is 0.7, the squared error would be reduced to 0.49 ($0.7^2 = 0.49$).

Finally, the R^2 is used to measure the degree the model its features can explain variances in target. The R^2 will be taken over each out-of-sample

year for both the MLP and SRNN. This metric offers insights into the explanatory capabilities of the models concerning variance in the predicted portfolio weights. The *adjusted* R^2 - which would make sense to use when comparing R^2 results on feature subsets - is intentionally avoided as the number of features exceeds the observations per window, leading to a loss of scale and interpretability in the *adjusted* R^2.

4.10 *Portfolio evaluation*

The MLP and SRNN models both predict one out-of-sample portfolio for each month during 2016-2021. To measure the performance of these predicted portfolios, a frequently used benchmark portfolio is selected:

- Equally weighted portfolio: a static portfolio allocating 50% to the stock- and 50% to the bond index, denoted as {0.5, 0.5}.

The predicted MLP and SRNN portfolios are compared with the EW portfolio based on the annualized Sharpe ratio, which are calculated for each out-of-sample year during 2016-2021. Equation 9 shows the formula used to convert monthly returns into annual *Sharpe ratios*. $\hat{r}$ is the average monthly excess return, while σ is the standard deviation over the monthly returns (r) and is essentially a penalization for variance across returns. This division is then multiplied by the square root of the number of months, resulting in the annualized *Sharpe ratio*.

$$\text{Sharpe ratio}_{\text{annualized}} = \frac{\hat{r}}{\sigma_r} \cdot \sqrt{12} \tag{9}$$

Inspired by Bouwmans (2022), a one-sided *t-test* is performed to test whether the machine learning (MLP and SRNN) portfolios significantly out- or under-perform the EW benchmark portfolio. The selected significance level (alpha) is 0.05. This leads to the following hypothesis:

$$H_0 : \text{Sharpe ratio}^{\text{machine learning}} \leq \text{Sharpe ratio}^{\text{equally weighted}}$$
$$H_1 : \text{Sharpe ratio}^{\text{machine learning}} > \text{Sharpe ratio}^{\text{equally weighted}}$$

5 RESULTS

This section reports the obtained results. Firstly, the hyper-parameter tuning results for both models are presented. Secondly, the model results (MAE and R^2) on the full feature set are reported. After, model results are provided for the feature subsets. Finally, the MLP and SRNN portfolios are analyzed from an investment perspective and compared to the selected benchmark portfolio (this is only performed for the full feature set).

5.1 *Hyper-parameter tuning results (full feature set)*

Table 5 and 6 report the hyper-parameter tuning results (on the full feature set) for the MLP and SRNN respectively. There are no significant differences in results between the two models. Overall, several patterns can be observed. Batch-normalization prevailed in nearly all windows, while dropout layers (dropout > 0%) are favored to a lower extend. Results on the number of layers are divided, with just over half of the windows having 2 layers (instead of 3). The number of neurons is neither concentrated toward one value, however, is skewed towards a lower number of neurons. The learning rate is largely skewed toward the default learning rate of the *Adam* optimizer (0.001) or a larger learning rate (0.01).

Table 5: MLP hyper-parameter tuning results

Hyper-parameters	2016	2017	2018	2019	2020	2021
# layers	3	3	3	2	2	2
# neurons (layer 1)	8	16	64	32	128	32
# neurons (layer 2)	8	16	32	16	128	16
# neurons (layer 3)	64	32	8	/	/	/
Batch-optimization	False	True	False	True	True	True
Dropout rate	0.00	0.00	0.00	0.25	0.25	0.5
Learning rate	0.0001	0.01	0.01	0.01	0.01	0.001

Table 6: SRNN hyper-parameter tuning results

Hyper-parameters	2016	2017	2018	2019	2020	2021
# layers	2	2	3	2	3	2
# neurons (layer 1)	128	64	128	16	8	64
# neurons (layer 2)	64	8	64	8	16	8
# neurons (layer 3)	/	/	64	/	8	/
Batch-optimization	True	True	True	True	True	True
Dropout rate	0.00	0.25	0.25	0.25	0.00	0.25
Learning rate	0.01	0.001	0.001	0.01	0.01	0.001

5.2 *Model evaluation (full feature set)*

Table 7 shows the MAE for each window of both models. In 5 out of 6 windows, SRNN outperforms MLP in terms of annual MAE. This translates into a lower average MAE of 0.34 (SRNN) compared to 0.39 (MLP). Reflecting on these results, SRNN outperformed MLP in predicting exact portfolio weights. The average MAE of both MLP and SRNN are mainly driven upwards by a relatively large error in in 2020 (0.66) and 2018 (0.50) respectively.In case of excluding these years with the largest error, SRNN still outperforms MLP, with an average MAE of 0.31 (SRNN) to 0.34 (MLP).

Table 7: Mean absolute error results

Models	2016	2017	2018	2019	2020	2021	Mean
MLP	0.35	0.29	0.37	0.38	0.66	0.30	0.39
SRNN	0.32	0.27	0.50	0.34	0.34	0.27	0.34

In figure 9, you can observe the distribution of *absolute errors* for both MLP and SRNN. The MLP exhibits more extreme results, and is more often either exceptionally accurate or inaccurate. Conversely, the SRNN consistently produces moderately accurate results, with smaller tails in the *absolute error* distribution. This observation can be quantitatively supported: the standard deviation over the *absolute errors* for MLP stands at 0.30, whereas SRNN demonstrates a lower standard deviation of 0.21.

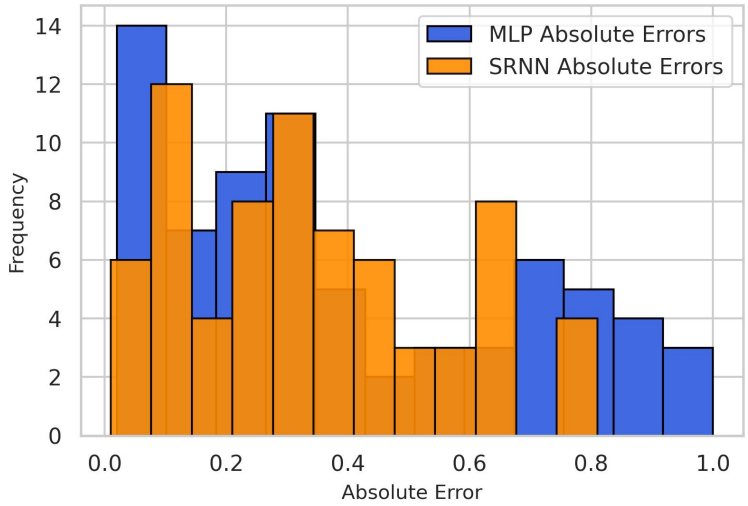

Figure 9: Absolute error distribution of the MLP and SRNN models

Table 8 reports the R^2 for each out-of-sample year. For most years, a negative R^2 can be observed, meaning that the models failed to explain

out-of-sample variance. However, for the year 2021, there are positive R^2 values with 0.035 (MLP), and to a higher extend, 0.343 (SRNN). SRNN also exhibited a positive R^2 of 0.113 during 2016. In these years, the models succeeded to explain (some) out-of-sample variance. Considering the average R^2, the SRNN outperforms MLP with -0.254 to -0.917 respectively.

Table 8: R^2 results

Models	2016	2017	2018	2019	2020	2021	Mean
MLP	-0.151	-0.955	-0.194	-0.979	-3.217	0.035	-0.917
SRNN	0.113	-0.489	-0.939	-0.506	-0.043	0.343	-0.254

5.3 *Model evaluation (feature subsets)*

To understand the feature importance in predicting optimal portfolio weights, we partitioned the full feature set into two subsets. For simplicity, the stock and bond index features are referred to as 'Stock/Bond' and the additional macro-economic and financial market features as 'MACRO'. A concise summary of the results is presented below, while a comprehensive analysis is available in Appendix I (page 49).

The outcomes from the feature subsets reveal different patterns. Notably, the MLP with only Stock/Bond features performed slightly better (average MAE of 0.38) compared to the full feature set (average MAE=0.39), while the MLP with MACRO features slightly worse (average MAE=0.40). In contrast, both feature subsets for SRNN exhibited lower performance compared to the full feature set. However, it is noteworthy that the MACRO features (average MAE=0.35) demonstrated relatively strong performance compared to the Stock/Bond features (average MAE=0.37).

In harmony with the identified patterns in the full feature set, the SRNN consistently produces more stable results across the two feature subsets. Despite minor improvements, the MLP still produces relatively extreme *absolute errors* when using the feature subsets. This observation is quantitatively supported by examining a relatively higher standard deviation across the *absolute errors* for MLP in both subsets (compared to the SRNN results).

Since the results from the feature subsets indicate either marginal or no model improvements compared to the full feature set, we do not further analyse the feature subsets from a portfolio investment perspective.

5.4 *Portfolio evaluation (full feature set)*

The MLP and SRNN both predicted out-of-sample portfolio weights. In table 9, the annual returns on these machine learning portfolios are reported, including the EW benchmark portfolio. Table 9 also shows the average over the annual returns, and the cumulative return over the whole out-of-sample period. In terms of the mean and cumulative return, the machine learning portfolios outperform the EW benchmark portfolio. Comparing the machine learning portfolios, MLP outperforms the SRNN.
This is more clearly illustrated in figure 10, which present the cumulative returns on the MLP, SRNN, and EW (benchmark) portfolios.

Table 9: Annual Returns

Year	**MLP**	**SRNN**	**EW benchmark portfolio**
2016	9.9%	9.7%	7.2%
2017	13.7%	16.1%	13.5%
2018	4.6%	-2.5%	-4.7%
2019	12.2%	18.2%	20.7%
2020	23.8%	17.5%	16.1%
2021	19.4%	16.5%	12.2%
Mean	14.0%	12.6%	10.8%
Cumulative	217.0%	200.9%	182.4%

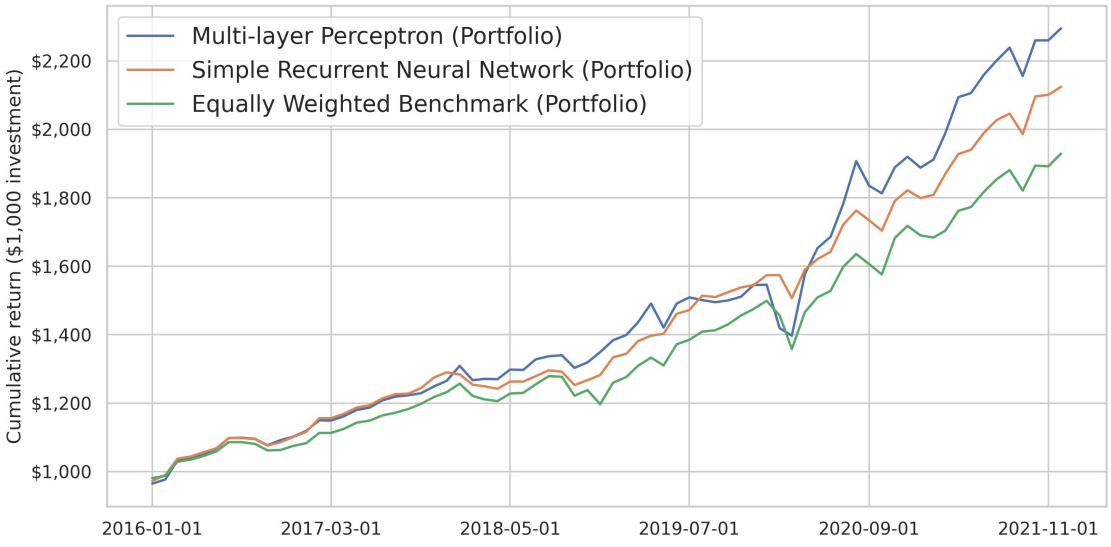

Figure 10: The cumulative returns on the MLP, SRNN, and EW (benchmark) portfolios (2016-2021). Each portfolio started with an initial $1,000 investment.

In addition to return, we also consider variance over monthly returns. In Appendix H (page 48), the standard deviations across returns are reported. MLP exhibits the highest average standard deviation, while SRNN the lowest.
In table 10, the annualized *Sharpe ratios* are visible for the MLP, SRNN, and the EW (benchmark) portfolios. Two negative Sharpe ratio can be observed during 2018 for the SRNN and EW portfolios respectively, which is caused by a negative average monthly return. The MLP portfolio avoided an average loss due to high weights in the bond index during sharp downfalls of the stock index in 2018. In the next paragraph, the Sharpe ratio averages are further analyzed.

Table 10: Annual Sharpe ratios

Year	**MLP**	**SRNN**	**EW benchmark portfolio**
2016	1.231	1.398	1.236
2017	4.689	4.430	5.470
2018	0.682	-0.479	-0.584
2019	1.400	3.271	2.992
2020	1.210	1.623	1.102
2021	2.049	2.025	1.648
Mean	1.877	2.045	1.977

Table 11 shows the *t-test* results. The mean *Sharpe ratio* of the machine learning portfolios (MLP & SRNN) are compared to the EW (benchmark) portfolio. The mean Sharpe ratio of the MLP portfolio (1.877) is lower compared to the EW (benchmark) portfolio (1.977), resulting in a negative t-statistic and high p-value of 0.539. With a mean Sharpe ratio of 2.045, the SRNN portfolio outperforms the EW (benchmark) portfolio. However, the *t-test* results in a p-value of 0.476, which is not near the alpha level of 0.05.

Table 11: T-test results(Sharpe ratio comparison)

Year	**MLP**	**SRNN**
N	6	6
Mean	1.877	2.045
St. Dev.	1.446	1.683
Mean EW (benchmark)	1.977	1.977
STDEV EW (benchmark)	2.060	2.060
T-statistic	-0.100	0.062
P-value	0.539	0.476

6 DISCUSSION

In this section, hyper-parameter tuning results are discussed. In addition, we discuss the machine learning model and portfolio performances.

6.1 *Hyper-parameter tuning*

Differences in hyper-parameters across windows may be explained by the changing relationships between the target and features over time, requiring different levels of model complexity. Alternatively, each window may not receive the same hyper-parameter combinations, due to the *stochastic* nature of random search.

6.2 *Model and portfolio results results (full feature set)*

The model performances are measured by MAE and R^2. In terms of MAE, SRNN outperforms MLP in 5 out of 6 years, which also results in a lower average MAE, with 0.34 (SRNN) versus 0.39 (MLP). By considering the distribution of *absolute errors* (see figure 9), SRNN demonstrates to have more consistent results and a lower variance across the *absolute errors*. This is an advantageous model characteristic, as large mistakes may have significant negative consequences in the realm of investing. In terms of R^2, a similar pattern can be observed. Only in 2020, MLP exhibited a positive R^2 of 0.035, signifying very limited explanatory capability. SRNN demonstrated to have more explanatory power, with a positive R^2, and a substantially larger average R^2 of -0.254 versus -0.917 for MLP.
In terms of both average and cumulative return, the machine learning portfolios (MLP & SRNN) exhibit superior performance compared to the EW benchmark portfolio (see table 10). MLP particularly performs well in terms of return, also out-performing SRNN. To provide a more holistic assessment on the portfolio performance, the variance across returns is also considered. Despite MLP achieved superior returns, it exhibits the lowest average *Sharpe ratio* (lower than both the SRNN and EW benchmark portfolios). The reason for this is that MLP has the largest variance across returns (see Appendix H [page 48]), which is penalized in the Sharpe ratio calculation. In contrast, SRNN achieves the lowest variance across portfolio returns. Consequently, despite being outperformed by MLP in terms of return, SRNN realized has the highest average Sharpe ratio. To assess the statistical significance of SRNN's out-performance compared to MLP and the EW benchmark, a t-test is conducted. The result indicates no significant difference in the Sharpe ratio between SRNN and the EW benchmark portfolio, with a p-value of 0.476, which is of considerable distance from

the selected alpha level (0.05). It is worth noting that the limited number of out-of-sample years (N=6) poses a challenge in obtaining statistically significant results, given the high penalty in the denominator of the *t-test* formula.

6.3 *Model results (feature subsets)*

The MLP exhibits a slight improvement when using only the Stock/Bond index features compared to the full feature set, while using only MACRO features deteriorates results. This suggests that for MLP, incorporating an extensive set of MACRO features may be disadvantageous, as it could either carry limited predictive power or contribute to over-fitting, (despite attempts to address this during hyper-parameter tuning).

Conversely, the SRNN exhibits a decline in model performance for both feature subsets. Unlike MLP, SRNN performs better with the MACRO feature subset than the Stock/Bond feature subset. This observation suggests that SRNN is either less sensitive to over-fitting or can extract more information from MACRO features, potentially as SRNN can capture temporal behaviour. The addition of MACRO features, alongside Stock/Bond features, appears to be advantageous for SRNN.

6.4 *Explaining model performance disparities*

The differing model results may be attributed to distinct model characteristics. The SRNN's relatively strong results (in terms of accuracy, consistency, and *Sharpe ratio*) suggest there is a temporal and dynamic relationship between portfolio weights and the economic state.
Likewise, MLP's overall under-performance may be due to the absence of a sequential memory, treating the data set as a single state in time and failing to capture temporal behavior. The occasional strong performance of MLP in certain years might indicate the existence of an underlying fundamental relationship between the economic state and portfolio weights, subject to significant changes over time.
Another plausible explanation is that SRNN is less sensitive to over-fitting compared to MLP, a characteristic of SRNN attributed to its ability to maintain temporal order, as discussed by Tölö (2020).

7 CONCLUSION

In this section, answers on the sub- and main research questions are provided. These answers focus on arguments based on the full feature set results. Moreover, multiple research limitations and extensions for future research are considered.

7.1 *Research questions*

RQ1 *To what extend does the Simple Recurrent Neural Network model improve or deteriorate the performances compared to the Multi-Layer Perceptron (based on MAE and R^2)?*

The *Multi-Layer Perceptron* (MLP) and *Simple Recurrent Neural Network* (SRNN) are evaluated and compared based on the *mean average error* (MAE) and R^2. In terms of MAE, the SRNN exhibits superior performance in predicting optimal portfolio weights, showing smaller out-of-sample mean absolute errors. In addition, the SRNN shows more consistency across predictions resulting in lower variance across absolute errors, whereas MLP performances are more volatile. While recognizing the potential for improvement in R^2 for both models, it is noteworthy that the SRNN shows to be more capable of explaining variance in the target. Consequently, these findings support the conclusion that the SRNN outperforms the MLP in terms of overall model performance.

RQ2 *To what extend do the predicted Multi-Layer Perceptron and Simple Recurrent Neural Network portfolios out- or under-perform the benchmark portfolio in terms of Sharpe ratio?*

In terms of average and cumulative return, the SRNN and to greater extend the MLP, outperform the EW benchmark portfolio. However, MLP also realized substantially higher variance across monthly returns, leading to a substantial reduction in *Sharpe ratio*. In contrast, the SRNN exhibits the lowest variance across monthly returns, which together with solid returns, contributes to SRNN having the largest *Sharpe ratio*. However, the result of the *t-test* indicates that there is no statistical evidence that SRNN significantly outperformed the EW benchmark portfolio. Consequently, these findings support that the MLP under-performs the EW benchmark portfolio in terms of *Sharpe ratio*, while SRNN outperforms the EW benchmark portfolio, albeit without statistical significance. Now answers on the sub-questions are formulating, the main research question can be answered:

To what extend can investors benefit from Multi-Layer Perceptron and Simple Recurrent Neural Network models in predicting one-month-ahead optimal portfolio weights?

The MLP exhibits superior out-of-sample returns, however, is also prone to relatively large variance across portfolio returns (like its model evaluation results). Consequently, we judge that at this point, the MLP carries an inherent risk which is too substantial to be a reliable portfolio prediction tool for investors.
On the other hand, the SRNN portfolio performs well in terms of returns while maintaining a relatively low variance across returns, resulting in the portfolio with the largest *Sharpe ratio*. Despite this, the lack of statistical significance leaves us with insufficient evidence to (at this point) deem SRNN as a valuable portfolio prediction tool for investors. Furthermore, the increasing transaction costs associated with employing the SRNN model as investment strategy may outweigh the additional predictive benefits of the model.

7.2 *Scientific and societal contribution*

With this study, our aim is to introduce an innovative methodology for predicting future multi-asset portfolios using machine learning techniques. This contributes to the existing literature (e.g., Abouseir et al. (2020), Obeidat et al. (2018), and J. Yu and Chang (2020)), where (broadly speaking) a similar approaches is used. With this methodology, we hope to provide new insights into predicting future portfolios and potentially offer a more computationally efficient approach. However, further model improvement, evidence, and robustness testing are imperative to the practical applicability of this methodology in a real-life investment context. Consequently, we acknowledge that, at this stage, the study's societal contribution is limited.

The reduction in computational requirements and complexity in this study might result in diminished overall performance compared to conventional methods (applied in existing literature). To objectively evaluate this, it is essential to compare our results with those of the conventional method, both using the datasets employed in this study.

7.3 *Research limitations*

This study faces limitations from both economic and methodological perspectives. A methodological limitation present in this study involves the tuning of a limited number of hyper-parameters. Multiple hyper-

parameters have been statically defined, such as the batch-size (32), number of epochs (10), and optimizer (*Adam*). This decision is made to prevent the hyper-parameter combination space to explode. Nonetheless, optimizing these hyper-parameters may potentially benefit model performances.
Furthermore, despite notable decreases in recent years, transaction costs still remains a factor of consideration. This means that acquiring an index (which can be done through *exchange-traded-funds* [ETFs]) comes with an economic cost, especially when regularly re-allocating funds from one index to another. In the portfolio optimization literature, transaction costs are commonly excluded. However, for practical purposes, transaction costs may be considered in future research to draw a more realistically analysis.

7.4 *Recommendations for future research*

In addition to the solutions offered for the research limitations, five potential research extensions are provided.
(i) Finding large financial time-series datasets is challenging, especially for alternative indices. Therefore, despite being unpopular in time-series, synthetic up-sampling techniques may be considered to reduce the risk of over-fitting. Oh et al. (2020) propose an up-sampling technique for time-series data. (ii) From a feature perspective, we only considered indicators related to the US market. However, US markets are not solely affected by domestic developments, and indicators outside the US may be considered. For instance, the Nikkei index (Japan's largest stock index), which was an important predictor in the study of Abouseir et al. (2020). (iii) The feature transformation techniques used varied substantially across related papers. Obeidat et al. (2018) used PCA techniques to reduce feature dimensionality. This can be considered to reduce the number of features while retaining most feature information. (iv) From a model perspective, the SRNN outperformed the MLP considering multiple metrics. However, the SRNN is the most standard RNN architecture, and may encounter problems in learning long-term trends (Manaswi et al., 2018). Besides, SRNNs are sensitive the phenomenon of a vanishing gradient, which involves the decay of the loss function's gradient. The LSTM architecture provides architectural solutions for these problems. Therefore, LSTM models may be considered in future studies. (v) A significant challenge in our task arises from the dynamic nature of financial markets, leading to shifting trends in historical optimal portfolios (the target) over time (refer to Appendix J [page 51] for more details). For instance, during the training period, a notable proportion of optimal portfolios exhibited a higher allocation to the bond index, due to certain market conditions during those years. This phenomenon may impose challenges when making out-of-sample predictions during

periods when the stock index is more dominant. Attention mechanisms becomes particularly relevant in this context, as they can emphasize attention to parts in the input space (Hollis et al., 2018). Therefore, attention mechanisms can be considered to emphasize the training years where the stock index weight (the minority class) prevailed in the generated optimal portfolios.

REFERENCES

Abouseir, A., Le Manach, A., El Mennaoui, M., & Zheng, B. (2020). Integration of macroeconomic data into multi-asset allocation with machine learning techniques. *Available at SSRN 3586040*.

Adam - keras documentation. (n.d.). https://keras.io/api/optimizers/adam/#:~:text=The%20learning%20rate.,for%20the%201st%20moment%20estimates

Arouri, M., Nguyen, D. K., & Pukthuanthong, K. (2014). Diversification benefits and strategic portfolio allocation across asset classes: The case of the us markets. *Unpublished working paper, IPAG Business School*.

Awan, A. A. (2022). *What are recurrent neural networks (rnn)*. https://www.datacamp.com/tutorial/tutorial-for-recurrent-neural-network

Bae, K., Ryu, H., & Shin, H. (2019). Does adam optimizer keep close to the optimal point? *arXiv preprint arXiv:1911.00289*.

Banerjee, K., Gupta, R. R., Vyas, K., Mishra, B., et al. (2020). Exploring alternatives to softmax function. *arXiv preprint arXiv:2011.11538*.

Bjorck, N., Gomes, C. P., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. *Advances in neural information processing systems*, *31*.

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*, *81*(3), 637–654.

Bouwmans, C. (2022). Empirical asset pricing via feedforward neural networks (master's thesis).

Branke, J., Scheckenbach, B., Stein, M., Deb, K., & Schmeck, H. (2009). Portfolio optimization with an envelope-based multi-objective evolutionary algorithm. *European Journal of Operational Research*, *199*(3), 684–693.

Chakravorty, G., Awasthi, A., & Da Silva, B. (2018). Deep learning for global tactical asset allocation. *Available at SSRN 3242432*.

Chen, N.-F., Roll, R., & Ross, S. A. (1986). Economic forces and the stock market. *Journal of business*, 383–403.

Chollet, F., et al. (2015). Keras.

Fama, E. F. (1981). Stock returns, real activity, inflation, and money. *The American economic review*, *71*(4), 545–565.

Fama, E. F., & Bliss, R. R. (1987). The information in long-maturity forward rates. *The American Economic Review*, 680–692.

Feng, S., Zhou, H., & Dong, H. (2019). Using deep neural network with small dataset to predict material defects. *Materials & Design*, *162*, 300–310.

Fricke, C., & Menkhoff, L. (2015). Financial conditions, macroeconomic factors and disaggregated bond excess returns. *Journal of Banking & Finance*, *58*, 80–94.

Götze, T., Gürtler, M., & Witowski, E. (2023). Forecasting accuracy of machine learning and linear regression: Evidence from the secondary cat bond market. *Journal of Business Economics*, 1–32.

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, *33*(5), 2223–2273.

Gunasekarage, A., Pisedtasalasai, A., & Power, D. M. (2004). Macroeconomic influence on the stock market: Evidence from an emerging market in south asia. *Journal of Emerging Market Finance*, *3*(3), 285–304.

Habbab, F. Z., & Kampouridis, M. (2024). An in-depth investigation of five machine learning algorithms for optimizing mixed-asset portfolios including reits. *Expert Systems with Applications*, *235*, 121102.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hinich, M. J., & Patterson, D. M. (1985). Evidence of nonlinearity in daily stock returns. *Journal of Business & Economic Statistics*, *3*(1), 69–77.

Hollis, T., Viscardi, A., & Yi, S. E. (2018). A comparison of lstms and attention mechanisms for forecasting financial time series. *arXiv preprint arXiv:1812.07699*.

Hommes, C. H. (2001). Financial markets as nonlinear adaptive evolutionary systems. *Quantitative Finance*, *1*(1), 149.

Hull, J., & White, A. (1990). Pricing interest-rate-derivative securities. *The review of financial studies*, *3*(4), 573–592.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Inglada-Perez, L. (2020). A comprehensive framework for uncovering nonlinearity and chaos in financial markets: Empirical evidence for four major stock market indices. *Entropy*, *22*(12), 1435.

Israel, R., Kelly, B. T., & Moskowitz, T. J. (2020). Can machines' learn'finance? *Journal of Investment Management*.

Kaczmarek, T., & Perez, K. (2022). Building portfolios based on machine learning predictions. *Economic research-Ekonomska istraživanja*, *35*(1), 19–37.

Lazzeri, F. (2020). *Machine learning for time series forecasting with python*. John Wiley & Sons.

Liang, D., Frederick, D. A., Lledo, E. E., Rosenfield, N., Berardi, V., Linstead, E., & Maoz, U. (2022). Examining the utility of nonlinear machine learning approaches versus linear regression for predicting body image outcomes: The us body project i. *Body Image*, *41*, 32–45.

Maillard, S., Roncalli, T., & Teıletche, J. (2010). The properties of equally weighted risk contribution portfolios. *The Journal of Portfolio Management*, *36*(4), 60–70.

Manaswi, N. K., Manaswi, N. K., & John, S. (2018). *Deep learning with applications using python*. Springer.

Markowitz, H. (1952). Portfolio selection in the journal of finance vol. 7.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, ... Xiaoqiang Zheng. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems [Software available from tensorflow.org]. https://www.tensorflow.org/

McKinney, W., et al. (2010). Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, *445*, 51–56.

Mirete-Ferrer, P. M., Garcia-Garcia, A., Baixauli-Soler, J. S., & Prats, M. A. (2022). A review on machine learning for asset management. *Risks*, *10*(4), 84.

Morningstar. (2023). *3-month treasury bill (index)* [Morningstar. All Rights Reserved.]. https://awgmain.morningstar.com/webhelp/glossary_definitions/indexes/3_Month_Treasury_Bill.html

Obeidat, S., Shapiro, D., Lemay, M., MacPherson, M. K., & Bolic, M. (2018). Adaptive portfolio asset allocation optimization with deep learning. *International Journal on Advances in Intelligent Systems*, *11*(1), 25–34.

Oh, C., Han, S., & Jeong, J. (2020). Time-series data augmentation based on interpolation. *Procedia Computer Science*, *175*, 64–71.

Passos, D., & Mishra, P. (2022). A tutorial on automatic hyperparameter tuning of deep spectral modelling for regression and classification tasks. *Chemometrics and Intelligent Laboratory Systems*, *223*, 104520.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pinelis, M., & Ruppert, D. (2022). Machine learning portfolio allocation. *The Journal of Finance and Data Science*, *8*, 35–54.

Popescu, M.-C., Balas, V. E., Perescu-Popescu, L., & Mastorakis, N. (2009). Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, *8*(7), 579–588.

R Rosca, E. (2011). Stationary and non-stationary time series. *The USV Annals of Economics and Public Administration*, *10*(1), 177–186.

Santos, G. C., Barboza, F., Veiga, A. C. P., & Gomes, K. (2022). Portfolio optimization using artificial intelligence: A systematic literature review. *Exacta*.

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. *9th Python in Science Conference*.

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of business*, *39*(1), 119–138.

Tölö, E. (2020). Predicting systemic financial crises with recurrent neural networks. *Journal of Financial Stability*, *49*, 100746.

Tütüncü, R. H., & Koenig, M. (2004). Robust asset allocation. *Annals of Operations Research*, *132*, 157–187.

van Rossum, G. (1995, May). *Python tutorial* (tech. rep. No. CS-R9526). Centrum voor Wiskunde en Informatica (CWI). Amsterdam.

Vasicek, O. (1977). An equilibrium characterization of the term structure. *Journal of financial economics*, *5*(2), 177–188.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., & Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, *17*(1), 26–40.

Yu, J., & Chang, K.-C. (2020). Neural network predictive modeling on dynamic portfolio management—a simulation-based portfolio optimization approach. *Journal of Risk and Financial Management*, *13*(11), 285.

Yu, T., & Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*.

APPENDIX A: PYTHON NOTEBOOKS

The code developed for this study can by accessed through GitHub. For each stochastic process, a seed (value=42) is used for consistency and reproducibility purposes. The code is separated into 7 notebooks:

- Notebook 1: Calculate the optimal portfolio (target) for each month during 1990-2021
- Notebook 2: Transform the monthly and daily indicator series
- Notebook 3: Dickey-Fuller stationary test on transformed series
- Notebook 4: Feature extraction for daily transformed series
- Notebook 5: Multi-Layer Perceptron models
- Notebook 6: Simple Recurrent Neural Network models
- Notebook 7: T-test/significance testing (Sharpe ratios)
- Notebook 8: Additional graphs

APPENDIX B: LIBRARIES USED

- Pandas (McKinney et al., 2010)
- Matplotlib (Hunter, 2007)
- Numpy (Harris et al., 2020)
- Keras (Chollet et al., 2015)
- Scipy (Virtanen et al., 2020)
- Seaborn (Waskom, 2021)
- Sklearn (Pedregosa et al., 2011)
- Statistics (van Rossum, 1995)
- Statsmodels (Seabold & Perktold, 2010)
- Tensorflow (Martín Abadi et al., 2015)

APPENDIX C: RETURN VOLAILITY/VARIANCE

Figure 11 shows significantly higher variance across monthly stock index returns compared to bond index returns. As variance (or volatility) across return is an indication of risk (or insecurity), this figure partly explains why stocks are perceived to be more risky than bonds.

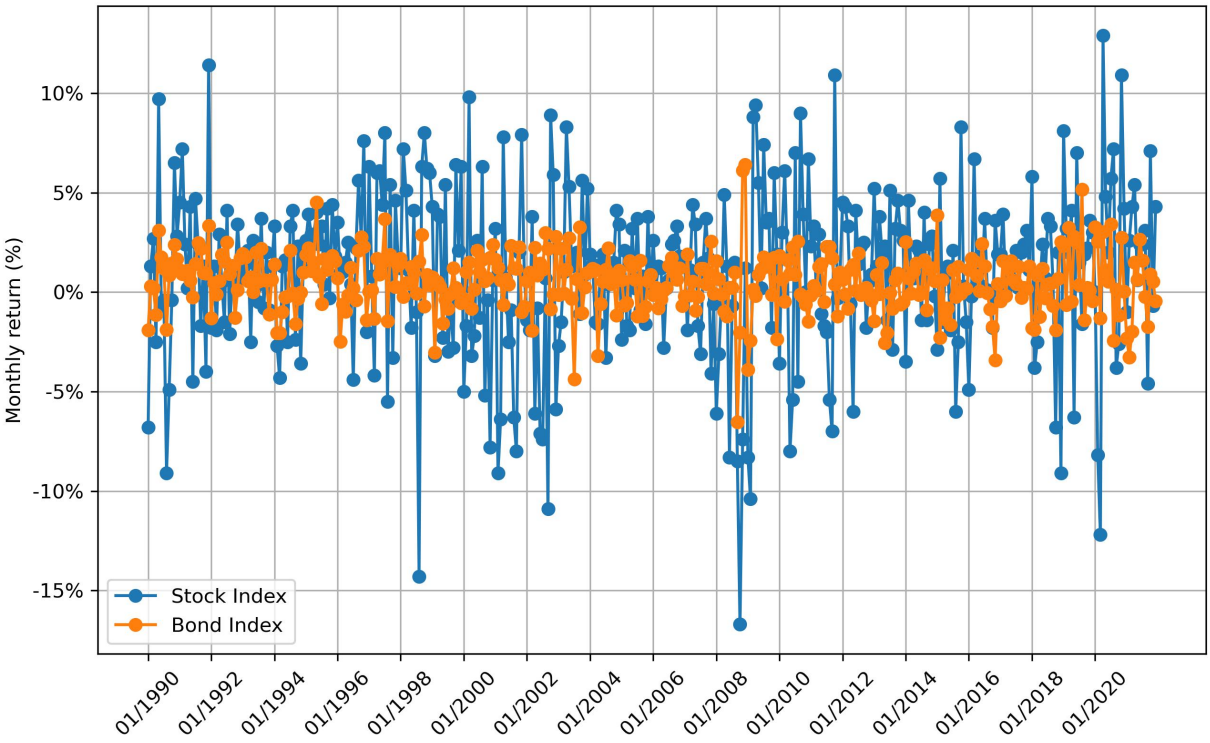

Figure 11: Monthly Stock Index vs. Bond Index Returns (1990-2021)

APPENDIX D: MACRO-ECONOMIC AND FINANCIAL MARKET DATA

Table 12: Macro-economic and financial market data

Ticker	Name	Category	Sub-category	Frequency
CPIAUCNS	Consumer Price Index for All Urban Consumers (YOY)	Macro-economic	Inflation	Monthly
CSUSHPISA	S&P/Case-Shiller U.S. National Home Price Index (YOY)	Macro-economic	Inflation	Monthly
DCOILWTICO	Crude Oil Prices: West Texas Intermediate (WTI)	Macro-economic	Inflation	Weekdays
IR	Import Price Index (End Use): All Commodities (YOY)	Macro-economic	Inflation	Monthly
IQ	Export Price Index (End Use): All Commodities (YOY)	Macro-economic	Inflation	Monthly
MICH	University of Michigan: Inflation Expectation	Macro-economic	Inflation	Monthly
PPIACO	Producer Price Index by Commodity: All Commodities (YOY)	Macro-economic	Inflation	Monthly
INDPRO	Industrial Production: Total Index	Macro-economic	Economic output	Monthly
PASVERT	Personal Saving Rate	Macro-economic	Saving rate	Monthly
S&P500	S&P500	Financial market	Assets	Weekdays
BAMLCC0A1AAATRIV	ICE BofA AAA US Corporate Index Total Return Index Value	Financial market	Assets	Weekdays
DEXUSUK	U.S. Dollars to U.K. Pound Sterling Spot Exchange Rate	Financial market	Currency (FX)	Weekdays
DTB3	3-Month Treasury Bill Secondary Market Rate, Discount Basis	Financial market	Interest rate	Weekdays
DGS2	Market Yield on U.S. Treasury Securities at 2-Year Constant Maturity	Financial market	Interest rate	Weekdays
DGS5	Market Yield on U.S. Treasury Securities at 5-Year Constant Maturity	Financial market	Interest rate	Weekdays
DGS10	Market Yield on U.S. Treasury Securities at 10-Year Constant Maturity	Financial market	Interest rate	Weekdays
TEDRATE	TED Spread	Financial market	Interest rate	Weekdays
T10Y3M	10-Year Treasury Minus 3-Month Treasury Constant Maturity	Financial market	Interest rate	Weekdays

APPENDIX E: URL'S OF USED DATA

Table 13: URL's of used data

Ticker	URL
CPIAUCNS	fred.stlouisfed.org/series/CPIAUCNS
CSUSHPISA	fred.stlouisfed.org/series/CSUSHPISA
DCOILWTICO	fred.stlouisfed.org/series/DCOILWTICO
IR	fred.stlouisfed.org/series/IR
IQ	fred.stlouisfed.org/series/IQ
MICH	fred.stlouisfed.org/series/MICH
PPIACO	fred.stlouisfed.org/series/PPIACO
INDPRO	fred.stlouisfed.org/series/INDPRO
PASVERT	fred.stlouisfed.org/series/PSAVERT
S&P500	Wharton/CRSP/S&P500 Index*
BAMLCC0A1AAATRIV	fred.stlouisfed.org/series/BAMLCC0A1AAATRIV
DEXUSUK	fred.stlouisfed.org/series/DEXUSUK
DTB3	fred.stlouisfed.org/series/DTB3
DGS2	fred.stlouisfed.org/series/DGS2
DGS5	fred.stlouisfed.org/series/DGS5
DGS10	fred.stlouisfed.org/series/DGS10
TEDRATE	fred.stlouisfed.org/series/TEDRATE
T10Y3M	fred.stlouisfed.org/series/T10Y3M

*Note: To access the S&P500 data you must be in possession of the Wharton Research Data Services license.

APPENDIX F: DICKEY-FULLER TEST RESULTS

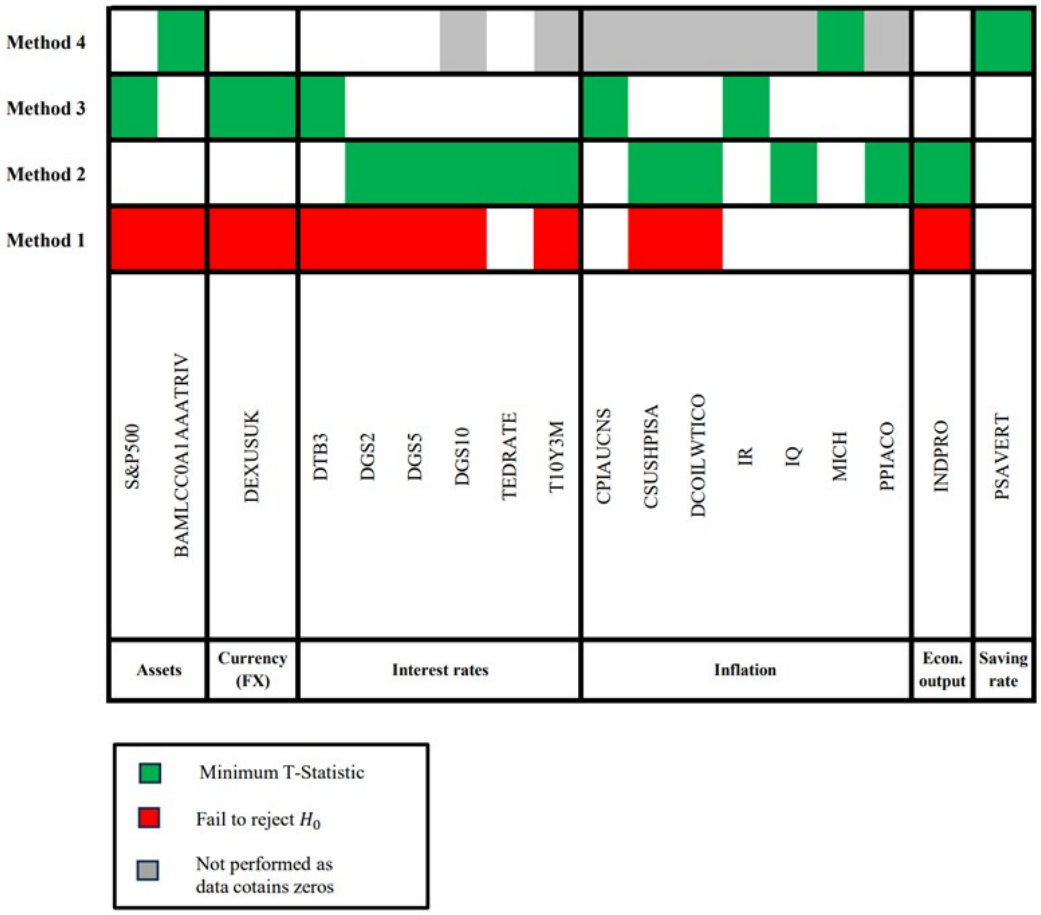

Figure 12: Dickey-fuller test results

APPENDIX G: EXTRACTED FEATURES

For each indicator with daily data, 6 monthly features are extracted from the transformed daily series. This means that each daily indicator provides 6 features per month. Table 14 reports the features, libraries, and functions used for extracting statistical features.

Table 14: Extracted Features

Feature	**Library**	**Function**
Mean	Statistics	mean()
Standard Deviation	Statistics	stdev()
Skewness	SciPy	skew()
Kurtosis	SciPy	kurtosis()
25th Percentile	Numpy	percentile(25)
75th Percentile	Numpy	percentile(75)

APPENDIX H: ANNUAL STANDARD DEVIATIONS OF RETURNS

Table 15: Annual standard deviations of returns

Year	MLP	SRNN	EW benchmark portfolio
2016	2.284%	1.972%	1.664%
2017	0.794%	0.976%	0.673%
2018	2.008%	1.468%	2.231%
2019	2.463%	1.504%	1.853%
2020	5.515%	2.966%	4.719%
2021	2.573%	2.230%	2.065%
Mean	2.606%	1.853%	2.111%

APPENDIX I: FEATURE SUBSETS RESULTS

For simplicity, the stock and bond index features are referred to as 'Stock/Bond', while the additional macro-economic and financial market features are denoted as 'MACRO'.

Table 16 illustrates the MAE results for both models and feature subsets, revealing substantial variations across the models. Surprisingly, the MLP with only Stock/Bond features achieved a slightly superior average MAE of 0.38 compared to the full feature set (average MAE = 0.39). Conversely, the MLP with only MACRO features exhibited a slightly higher average MAE of 0.40. For the SRNN, both feature subsets performed less effectively than the full feature set (MAE=0.34). However, the MACRO features demonstrated a performance close to the full feature set, with an average MAE of 0.35, while the Stock/Bond features exhibited a relatively poor performance, with an average MAE of 0.37.

Table 16: Feature subsets: Mean absolute error results

Models	2016	2017	2018	2019	2020	2021	Mean
MLP + Stock/Bond	0.26	0.26	0.43	0.38	0.66	0.30	0.38
MLP + MACRO	0.40	0.24	0.51	0.33	0.48	0.44	0.40
SRNN + Stock/Bond	0.38	0.25	0.41	0.42	0.31	0.43	0.37
SRNN + MACRO	0.39	0.28	0.40	0.34	0.34	0.33	0.35

Aligned with the patterns observed in the full feature set, the SRNN consistently delivers more stable predictions across both feature subsets. This becomes evident in figure 13, depicting the distribution of *absolute errors*, where MLP feature subsets exhibit a tendency toward more extreme results. Quantitatively, this observation finds support in the standard deviation values over the *absolute errors*, with the MLP (Stock/Bond) at 0.29 and MLP (MACRO) at 0.26. In contrast, the SRNN exhibits lower variability, with standard deviation values of 0.23 (SRNN + Stock/Bond) and 0.21 (SRNN + MACRO).

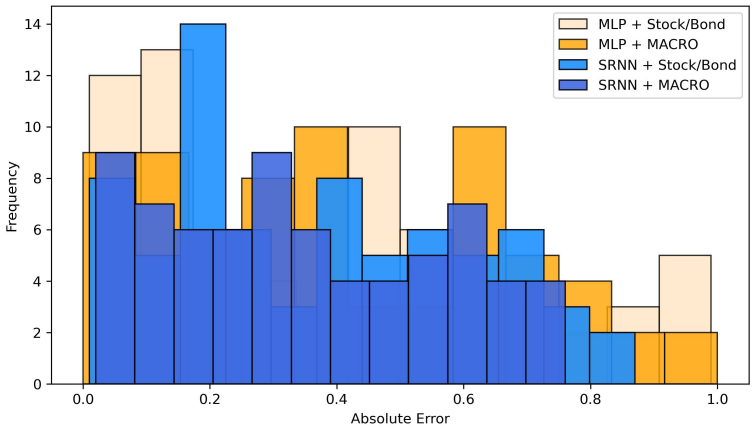

Figure 13: Feature subsets: absolute error distribution of the MLP and SRNN

Table 17 presents the out-of-sample R^2 results for both feature subsets and models. While the Stock/Bond features exhibit improvements in average R^2 compared to the full feature set, they remain notably negative. Notably, the MLP Stock/Bond features surprisingly yield a positive R^2 during 2016. Table 17 shows the out-of-sample results for both feature subsets and models. Compared to the full feature set, the Stock/Bond features both show improvements in the average R^2, however, still significantly negative. Surprisingly, the MLP Stock/Bond features now results in a appositive R^2 during 2016.
In line with the average MAE results of the SRNN, there is a reduction in R^2 for both feature subsets. The full feature set, which initially achieved an average of -0.254, is now reduced to -0.451 (Stock/Bond) and -0.312 (MACRO). Of particular significance is the substantial decrease in positive R^2 values observed in the feature subsets for the years 2016 and 2021.

Table 17: Subsets: R^2 results

Models	2016	2017	2018	2019	2020	2021	Mean
MLP + Stock/Bond	0.177	-0.319	-0.406	-1.222	2.968	0.020	-0.786
MLP + MACRO	-0643	0.047	-0.927	-0.320	-1.793	-0.554	-0.698
SRNN + Stock/Bond	-0.644	-0.075	-0.289	-1.168	0.076	-0.604	-0.451
SRNN + MACRO	-0.600	-0.593	-0.122	-0.496	-0.074	0.013	-0.312

APPENDIX J: ANNUAL AVERAGES OF TARGET

The target consists of the monthly optimal stock and bond index weights for each month during 1990-2021. By taking the annual averages of the monthly weights, structural movements can be identified, as visible in figure 14. During most years, the bond index has a larger contribution in the average optimal portfolio. This can become problematic when predicting out-of-sample portfolio weights during periods where the stock index is more prevailing. Therefore, we introduced attention mechanisms as a potential extension for future research (see page 35 for more details).

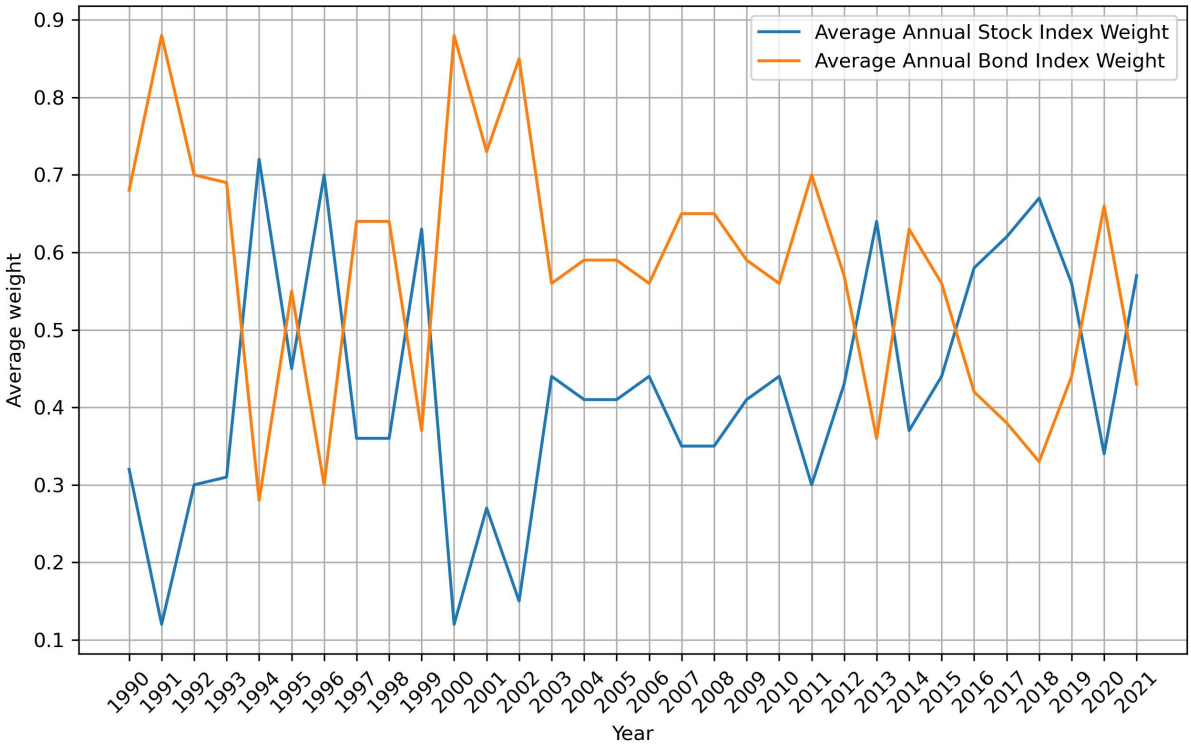

Figure 14: Average annual optimal portfolio weights for the stock and bond indices (target).