ERASMUS UNIVERSITY ROTTERDAM

ERASMUS SCHOOL OF ECONOMICS

BACHELOR THESIS ECONOMETRICS AND OPERATIONAL RESEARCH

Deep learning methods with attention mechanism for financial market prediction

Author:
M.H.A.M. van der Meulen
Student ID: 505560

Supervisor:
S.H.L.C.G. Vermeulen
Second Assessor:
A. Tetereva

July 4, 2021

Abstract

This thesis evaluates if adding an attention mechanism to a Long Short-Term Memory (LSTM) network and a Gated Recurrent Units (GRU) network results in a more stable long-short portfolio with higher returns. The goal of this research paper is to predict the probability for each stock to out-/underperform the cross-sectional median for the next day for the companies that are in the S&P 500 from January 1990 until December 2020. We found that adding an attention mechanism does not result in a portfolio with higher returns. The portfolios with the highest daily returns, as well as risk-adjusted returns are the portfolios with the random forest, with a daily return of 0.20%, and LSTM network with a daily return of 0.19%, before transaction costs. Moreover, the LSTM network with attention mechanism, GRU network, and GRU network with attention mechanism portfolios have lower exposure to risk compared to the portfolio with the random forest and LSTM network. Specifically, we found that the LSTM network achieved the highest returns and Sharpe ratio in the more volatile periods. The portfolios for the random forest and the LSTM network outperformed the LSTM network with attention mechanism, GRU network, and GRU network with attention mechanism both on out-of-sample return prediction as well as risk-adjusted returns.

Contents

1 Introduction

Predicting stock price movements is one of the most challenging tasks in the financial world because the stock markets are nonlinear, dynamic, complex, and incomplete (Abu-Mostafa and Atiya 1996; Booth et al. 2014; L. Zhang et al. 2018). Every single day millions of dollars worth of trading occur over the world, and every trader wants their investment to be worth money. Investors who sell or buy at the right time will end up with a profit. Investors analyze the companies to predict the stock price movement for that company. However, it is difficult for investors to handle all the publicly available information from the companies and analyze it.

The Efficient Market Hypothesis (EMH) is a popular theorem that states that all the publicly available information is already included in the price so that it is impossible to outperform the market (Fama 1965). This hypothesis is highly accepted among the financial academia, but in reality this hypothesis is often debated and highly questionable. Nowadays, big companies spend a lot of money on hiring experts to build statistical prediction models. These analysts are trying to uncover the mysteries of the financial stock market data. Because of the improvements of the ability in computers, deep learning achieved great progress (Song 2018). That is why researchers focus more on deep learning to identify the stock trends because these techniques can handle a large amount of data and are able to discover non-linear and chaotic patterns (G. P. Zhang 2001). To this day, an algorithm or model which can consistently predict the price of future stock values correctly does not exist (Tipirisetty 2018). Since neural networks are able to discover nonlinear and chaotic patterns (G. P. Zhang 2001), they may be able to forecast stock price movements more accurately than linear models. Long Short-Term Memory (LSTM) network is a variant of recurrent neural networks. LSTM networks have proven to perform well in time series learning because they can maintain long-term memory (Zhuge et al. 2017). Gated Recurrent Units (GRU) are similar to LSTM, but simpler to compute and may lead to better results when using a smaller dataset (Fu et al. 2016; Zhao et al. 2019). These models are also suitable to process the non-linear, non-stationary, and complicated time series (Y. Zhang et al. 2021). Recently, more research focused on the attention mechanism in deep neural networks. The attention mechanism is introduced to help the network to locate the informative data segments in the sequence. Further, it helps to extract the discriminative features of inputs, and it helps to visualize the learned diagnosis information (X. Li et al. 2019). Attention mechanism has been successfully used in different fields of deep learning applications and should help the network to gain more predictive power (Niu et al. 2021).

This thesis evaluates different deep learning models with attention mechanisms in a large-scale time series prediction problem. The research question that is answered in this work can

be formulated as follows: Can a LSTM network or a GRU network with attention mechanism contribute to more stable portfolios with higher returns compared to the random forest and the LSTM network? To be able to answer this question a few sub-questions need to be answered; Which trading strategy results in the most stable portfolio with the highest returns?; What are the optimal hyperparameters for LSTM network with attention mechanism, GRU network, and GRU network with attention mechanism?; Which model results in more stable portfolios with higher returns in the global financial crisis (2008) and in the COVID-19 (2020) period? This research will contribute to the understanding of the predictive power of deep learning models, also to the best of our knowledge, there has been no previous attempt to deploy a GRU network with an attention mechanism on large-scale financial market prediction tasks.

To provide an answer to our research question, we use data obtained from Yahoo Finance. The dataset contains the daily total return indices for all the companies that are in the S&P 500 from January 1990 until December 2020. This period is divided into rolling windows of four years. The first three years are used to train the model and the last year is used for predicting. This results in 28 rolling blocks.

First, we evaluate the predictive accuracy for all the models. Second, we look at the financial performance of the portfolio for each model. We evaluate the return characteristics, the risk characteristics, and the annualized risk-return metrics. Finally, we can answer the research question by analyzing the portfolios for each model in a quantitative way by comparing all the different portfolios, to check which portfolio is more stable and results in higher returns.

Our results show that the portfolio which contains ten stocks results in the highest returns and return per unit of risk, compared with the portfolios which contain 20 or 30 stocks. The Diebold-Mariano test confirms that all the forecasts exhibit similar predictive accuracy. The results show that adding an attention mechanism to the LSTM network and GRU network does not result in a portfolio that gives higher returns. The portfolio that gives the highest mean return over the whole sample period is the random forest and LSTM network, which gives daily returns of 0.20% and 0.19% respectively. When looking at the annualized return, the LSTM network gives the highest return of 87.88%. However, the portfolios with the LSTM network and random forest are more than twice as risky as the portfolios with the LSTM network with attention mechanism and GRU network. Moreover, the portfolio with the GRU network with attention mechanism shows the lowest downside risk. Further, in the more volatile periods, in the global financial crisis and the COVID-19 period, the LSTM network gained the highest daily returns in both periods. So the portfolio with the LSTM network resulted in the portfolio with the highest daily returns, also in more volatile periods, but is also the riskiest portfolio.

The remainder of this thesis is organized as follows. Section 2 provides a literature review of the related studies. Section 3 discusses the dataset that is used in this thesis. Section 4 provides an in-depth discussion of our methodology, i.e., the feature generations for the models, the model architectures, model training, trading strategy, and the output analysis. Section 5 presents the results and discusses the most relevant findings. Lastly, Section 6 concludes the results.

2 Literature Review

Predicting stock prices has never been easy, due to the abnormality and volatility of the financial market (Adam et al. 2016). That is why simple models cannot predict future asset values with higher accuracy. For decades, a large body of research has emerged to find determinants of financial returns in the time series by implementing linear regressions. More recent work focused on nonlinear methods and machine learning techniques to predict the movement of the stock prices.

The Efficient Market Hypothesis (EMH), introduced by Fama (1965), asserts that the stock market is an efficient market. They say that the prices of stocks are a reflection of all publicly known information and it is impossible to outperform the market consistently. A variant of the Efficient Market Hypothesis is the semi-strong form, introduced by Fama (1970), which states that the prices of the stocks reflect all the publicly available information so that investors cannot retrieve abnormal returns from publicly released information. However, this theory is often discussed and highly questionable because there are cases where investors generate continuous, abnormal returns using mostly publicly available information (Coval et al. 2005). Therefore it is important for investors to analyze and monitor publicly released company information.

Much research has been devoted to revealing the determinants of financial returns in the time series, how average returns change over time. First linear regression models, such as Moving Average (MA) and Auto-regression (AR) were used to predict stock prices. Next, the Auto-regression Moving Average (ARMA) model was introduced by Box and Jenkins (1976). These models can deal with sequences. An extension of the ARMA model is the Auto-regression Integrated Moving Average (ARIMA) model, which can deal with non-stationary time series. After these models, the Conditional Heteroskedasticity (ARCH) model was introduced by Engle (1982). This model can simulate the variation of the time series variable. Another popular model in the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model (Bollerslev 1986), which is used for predicting and analyzing the volatility of stock prices. These

methods are used very often in the field of forecasting time series. Furthermore, the state space models (Kalman 1960), such as hidden Markov models and linear dynamical systems have also been very popular in the past decades for efficient message passing algorithms (Zheng et al. 2017). These models are used for modeling and learning time series patterns such as trend and seasonality (Hyndman et al. 2008). The goal of the state space model is, given the observations, to infer information about the states, as new information enters. These models have a compact closed-form due to their simple Markov structure and their main benefit is that these models make use of state space variables that evolve over time, which depends on the input variables. However, their simplicity is also a disadvantage, because these models cannot capture the long-range dependencies as they are in the real world and are not able to understand the shared patterns from a dataset with similar time series (Rahman et al. 2019). Therefore, these models are not able to predict the stock prices accurately. Researchers have focused more on deep neural networks because these models have the capability to extract higher-order features and can identify complex patterns across and within time series with less human effort (Yu et al. 2016). This paper essentially lays out the methodology upon which further research on stock price prediction is based.

2.1 Deep Neural Networks

Neural networks have been popular in the last decade because they could outperform linear models with a simple neural network (Refenes et al. 1994). That is the reason why researchers focus more on neural networks because these models are able to approximate any nonlinear function (Cybenko 1989). Another advantage of neural networks is that the network is generated through training and does not need to be specified (Livingstone et al. 1997). However, neural networks also have disadvantages. They are likely to overfit, which means that the model learns too well on the training model and whereby the model cannot recognize the pattern of the testing data and greatly loses its predictive power (Hawkins 2004).

In the paper of Fischer and Krauss (2018), where they tried to forecast the predicting out-of-sample daily directional movements for companies in the S&P 500, with a Long Short-Term Memory (LSTM) network for the period from 1992 until 2015. In this paper, they found that the LSTM network achieves the highest annualized returns of 82.29% after transaction costs. Also, the LSTM network achieves the highest Sharpe ratio of 2.34, which can be interpreted as a single-to-noise ratio. Interestingly, Fischer and Krauss (2018) discovered that the LSTM network is not performing better than the random forest after 2009 because since then, the returns from the LSTM network fluctuate around 0%.

Furthermore, Chung et al. (2014) found that the GRU and the LSTM model are comparable. Choosing between these models depends heavily on the dataset and corresponding task. An advantage of the GRU networks is that they have fewer parameters and may therefore perform better on smaller datasets (Zhao et al. 2019).

Attention mechanisms in neural networks have achieved great success in various machine learning tasks (H. Li et al. 2018). In recent work from X. Wang et al. (2017) they used attention among multiple neurons to deal with the non-stationary data. Their results showed that the models with the attention mechanism performed better in predicting the area under the curve and the F1 score. Furthermore, it is found that implementing the attention mechanism in a recurrent neural network improves the model because the attention mechanism can adaptively select the relevant driving series, and it can capture the long-range temporal information (Qin et al. 2017). In the paper of Qiu et al. (2020) they compared the LSTM and GRU network with the LSTM network with attention mechanism for stock price predicting. Their results showed that the model with an attention mechanism improved the accuracy of the prediction results.

3 Data

The datasets on the total monthly returns of 112 of the Standard & Poor's 500 (S&P 500) index constituents are obtained from Wharton Research Data Services (WRDS). The S&P 500 index constituents have a relatively high degree of market efficiency and liquidity, that is why we focus on these companies in this paper. To eliminate the survivor bias, only the companies that are in the S&P 500 for the whole period from January 1990 until December 2020 are included in the dataset. Also, the companies that have missing pricing data are removed from the dataset, to make sure that the datasets are complete for all the companies. This results in 112 companies in the dataset. The list of stocks used in this thesis can be found in Appendix A. The survivor bias is the incorrectness of measuring the performance of stock or fund in the market as a representative comprehensive sample without removing the stocks or funds that have gone bust in the time periods (Garcia and Gould 1993).

The first step is to divide the data into "study periods" as training-trading sets. The training sets consist of approximately 756 days ($\approx$ three years) and the trading period consists of 252 days ($\approx$ one year). We split the dataset consisting of 31 years starting from January 1990 until December 2020 in 28 of these study periods. This results in 28 non-overlapping trading periods.

The next step is to download the pricing information for the companies that are included in the dataset. The pricing information is retrieved from the Yahoo Finance historical price dataset. This historical price dataset consists of the opening price of that day, the highest price

of that day, the lowest price of that day, the closing price adjusted for splits for that day, the adjusted closing price adjusted for both dividends and splits for that day and the volume that is traded on that day. In this research the adjusted closing price is used, because this price accounts for all relevant corporate actions and stock splits, this makes it the most adequate metric for predicting stock prices (Fischer and Krauss 2018). With the historical adjusted closing price, the simple return indices are calculated for all stocks. First, we focus on the returns of the most recent 20 trading days. Then, we switch to a lower resolution and analyze the returns of the subsequent 11 months. This results in 31[1] features, which corresponds to one trading year and approximately 240 days. Then, we standardize the returns by subtracting the mean and dividing them by the standard deviation. These standardized return indices are used to predict the probability of a stock to outperform the cross-sectional median in the period $t+1$.

4 Methodology

4.1 Feature generation for LSTM network and GRU network

Let T_{study} denote the number of days in a study period and let $R_t^{m,s}$ be defined as the simple return for a stock s at time t over m periods, with $s \in \{1, ..., 112\}$ and $m = \{1, 2, 3, ..., 20\} \cup \{40, 60, ..., 240\}$. First, the focus is on the first 20 trading days and then on the subsequent 11 months. Further, let $\tilde{R}_t^{m,s}$ be defined as the standardized return.

The LSTM and GRU networks need sequences of input features for both the target measure as well as the features for the training. The features in our case are the m standardized returns $\tilde{R}_t^{m,s}$. The sequence length is 240, which gives the information of approximately one trading year. Therefore, we create overlapping sequences of 240 consecutive, standardized returns $\tilde{R}_t^{m,s}$ in a subsequent way: First, we sort the feature matrix V by stock s, date t, and feature m in ascending order. Next, we construct sequences of the following form $\{\tilde{R}_{t-239}^{m,s}, \tilde{R}_{t-238}^{m,s}, ..., \tilde{R}_t^{m,s}\}$ for each stock s and each $t \geq 240$ of the study period. For the first stock $s = 1$ the sequence consists of the standardized returns $\{\tilde{R}_1^{m,1}, \tilde{R}_2^{m,1}, ..., \tilde{R}_{240}^{m,1}\}$. The second sequence consists of $\{\tilde{R}_2^{m,1}, \tilde{R}_3^{m,1}, ..., \tilde{R}_{241}^{m,1}\}$ and so forth.

The first 240 days of the study periods are used for feature generation. This results in a training period starting from $t = 241$ and ending at $t = 756$. The trading period is from $t = 757$ to $t = T_{study}$. The variable T_{study} is the number of trading days in the study period, typically T_{study} is equal to 1008. This results in training data for all the stocks s in an approximate size of 112 x 516 = 57.792 rows and M columns with the features (31), simultaneously with their

[1]The first 20 features are from the first trading month and 11 features from the 11 subsequent months.

corresponding target. For the out-of-sample predictions, a matrix with an approximate size of 112 x 252 = 28.224 rows and M columns with the features (31) is used.

For the target prediction, we follow Fischer and Krauss (2018) and divide each stock at time t into 2 classes of equal length, also referred to as a binary classification problem. Class 0, "sell", is realized if the one-period return $R^{1,s}_{t+1}$ of stock s is smaller than the cross-sectional median return of all stocks in period t. Whereas Class 1, "buy", is realized if the one-period return $R^{1,s}_{t+1}$ of stock s is larger than the cross-sectional median return of all stocks in period t.

4.2 Random Forest

The random decision forest was first suggested by Ho (1995) and later enlarged by Breiman (2001). Random forests consist of a large number of regression trees, each on different samples of the data. The random forest algorithm starts with B trees. From each of the B trees in the committee, a bootstrap sample is drawn from the original training data. This means that a decision tree develops on the bootstrap sample. At each split, a modified decision tree is modified to this sample, whereby only a subset m of the p features is available as a potential split criterion. The growth of the trees stops once the maximum depth J is reached. The final output is a collection of B random forest trees, so that classification can be accomplished as a majority vote. The hyperparameter setting is chosen by Fischer and Krauss (2018) and can be found in Appendix C.

There are two reasons why one would choose the random forest as a benchmark model. Firstly, the random forest performs quite well compared to the LSTM network, according to Fischer and Krauss (2018). Secondly, the random forest is a state-of-art machine learning model that requires practically no tuning and regularly delivers good results.

4.3 Recurrent Neural Network

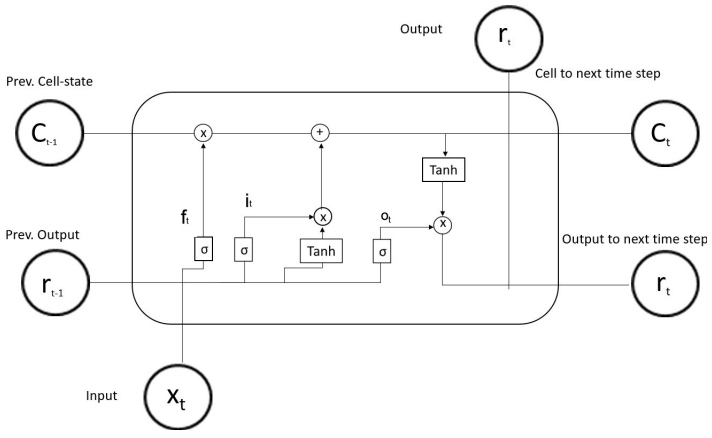
Recurrent Neural Networks (RNN) is a class of neural networks, which can contain information from a sequential input across time steps (Dupond 2019). RNN allows information to persist within a layer over a longer period of time, in a regular neural network the information only flows forward from layer to layer. RNNs store this information in the layer by adding a hidden state in each neuron. The advantage of the RNN is that it is not only influenced by what they learn from the supervised training, but also from what they learn in generating previous outputs. Another advantage is that the gradients of the RNN are cheap to compute (Sutskever et al. 2011). However, the problem with the basic RNN structure is that this network is not able to remember the information from the previous outputs for a long time. In practice, these

networks are only able to remember the information between five and ten steps back in the past (Gers et al. 2000). Another problem is that the gradient of the RNN during training can become vanishing or exploding (Bengio et al. 1994). There are several types of RNN with different structures which are able to overcome these problems, two of these, long short-term memory and gated recurrent unit, are discussed in this paper. These networks introduce a memory cell to overcome the vanishing gradient problem. This cell only remembers the most relevant information from the past inputs and propagate this information to the next point in time.

4.3.1 Long Short-Term Memory Neural Networks

Long Short-Term Memory (LSTM) neural networks are a particular class of the RNN. LSTM networks have been implemented in many research areas but are most popular in the field of language processing. Although the LSTM architecture exists since 1997, introduced by Hochreiter and Schmidhuber (1997), its application in financial market prediction has only been used in recent years. This model was introduced to solve the vanishing gradient problem by implementing memory cells to maintain the time-related information and are able to catch long-term decencies, as in real life (Greff et al. 2016). This memory cell is presented in Figure 1. The memory cell of the LSTM contains an input gate, output gate and a forget gate, these gates are used to control the interactions between the memory cells and these three gates also determine if the information is added or removed from the cell-state C_t. This cell-state is connected in a linear way to ensure strong and stable gradients (Borovkova and Tsiamas 2019). Another difference compared to the RNN is that the LSTM network makes use of multiple sigmoid gates (σ in Figure 1) to control the information that flows through the model. The sigmoid function takes a value between 0 and 1, if the value is close to 1 it lets the information flows through otherwise, it removes the information (Zhuge et al. 2017).

Figure 1: Architecture of the Memory cell of LSTM . In this picture f_t is the forget gate, i_t is the input gate and o_t is the output gate.



The forget gate regulates, with the sigmoid layer, if the memory cell needs to remember the information from the previous state or if this information needs to be removed. If it gets a value of 1, the information needs to be remembered and if it gets a value of 0, the information needs to be forgotten. This gate defines which information is removed from the cell state. The formula of the forget gate can be shown in Equation 1. Next, the input is put through the input gate, which regulates the information of the input x_t and the output from the previous cell state r_{t-1}. The input gate uses the sigmoid function to decide which information should be added to the cell-state originating from the forget section, see Equation 4. Then, a tanh-layer decides how much information of the inputs is added to the cell-state by creating a vector of new candidate values, see Equation 2. Subsequently, the cell-state is updated by removing information, if any, from the previous cell-state through the forget gate also through the input gate. New and updated values are added to the cell-state memory vector, see Equation 5. The last gate in the memory cell is the output gate, see Equation 3. This gate determines through a sigmoid layer, which elements of the cell-state will be converted to output. Next, the output gate decides which of these candidate's values of the cell-state will be converted to the output. Then the cell-state C_t is transformed by the tanh-layer, which scales the values between [-1,1] (Sagar Sharma and Simone Sharma 2017). Lastly, the output r_t is a combination of the output gate and the transformed cell-state, see Equation 6. The output gate decides which of the values of the transformed cell-state will flow through to the output r_t.

The formulas used for the LSTM network can be shown in the equations below (Jia et al. 2019; Zhuge et al. 2017). The variables used in the equations represent: f_t for the forget gate, i_t for the input gate, σ is the sigmoid activation function, which is an element-wise multiplication, W_j are the weights for the neurons in the respective gate, r_{t-1} is the output from the previous time step of the LSTM memory cell, x_t is the input from the previous time step and b_j is the bias for the respective gate. The $\odot$ refers to an element-wise product, while a $\cdot$ refers to an element-wise multiplication.

$$\mathrm{f_t} = \sigma(W_f \cdot [r_{t-1}, x_t] + b_f) \quad (1)$$

$$\Delta\,\mathrm{C_t} = tanh(W_C \cdot [r_{t-1}, x_t] + b_C) \quad (2)$$

$$\mathrm{o_t} = \sigma(W_o \cdot [r_{t-1}, x_t] + b_o) \quad (3)$$

$$\mathrm{i_t} = \sigma(W_i \cdot [r_{t-1}, x_t] + b_i) \quad (4)$$

$$\mathrm{C_t} = f_t \odot C_{t-1} + i_t \odot \Delta C_t \quad (5)$$

$$\mathrm{r_t} = o_t \odot tanh(C_t) \quad (6)$$

4.3.2 Gated Recurrent Units

The Gated Recurrent Units (GRU) is introduced by Cho et al. (2014) and is a special kind of RNN. The GRU network is very much like the LSTM network. They both make use of gates,

but the main difference is that the GRU network has two gates, reset gate and update gate. The LSTM network has three gates, which makes the GRU network less complex (Dey and Salem 2017). The input and forget gate in the LSTM network are replaced by the update gate in the GRU network, and thereby the reset gate is directly connected to the previously hidden gates. The GRU network uses these two gates to solve the vanishing gradient problem. With these two vectors, the cell decides which information should be passed through to the output and which information should be removed. The most important factor is that they are able to train the model to keep information from long ago, without removing the information that is relevant for the prediction (Rahman et al. 2019). One feature of the LSTM unit that is missing in the GRU unit is that the GRU network does not have the controlled exposure of the memory content. The LSTM unit controls the amount of memory content that has flowed through other LSTM units by the output gate. The GRU network exposes its full content without any control (Cho et al. 2014).

GRU networks and LSTM networks show similar performance in many machine learning applications. The GRU network is less complex compared to the LSTM network and there is no theoretical guidance that tells which model to use for which application, but the final decision depends massively on the size of the dataset (Dey and Salem 2017).

Figure 2: Architecture of the GRU cell. In this picture z_t is the update gate and q_t is the reset gate.

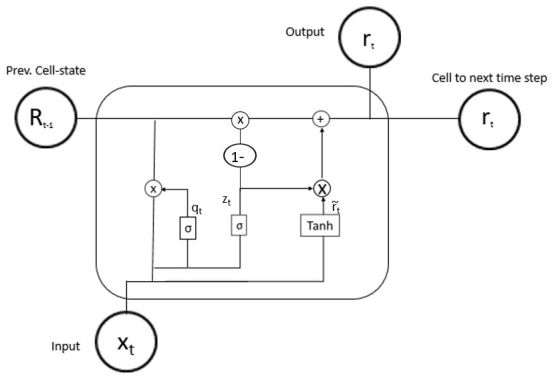


Figure 2 shows the architecture of the GRU cell. The update gate, see Equation 7, determines through a sigmoid layer how much of the previous hidden state will be remembered. The larger the value of the update gate, the more information of the previous hidden state will be remembered. This gate also removes the risk of the vanishing gradient problem, because it helps the network to control how much of the previous information needs to be remembered. Next, the reset gate decides through a sigmoid layer how much of the previous information to forget, see Equation 9. The smaller the value of the reset gate, the more the previous information will be ignored. Then, a tanh layer creates a vector of candidate values which decides

which of the features, inputs, and elements from the previous hidden state should be added to the hidden state, see Equation 8. Lastly, the hidden state, the output r_t, is calculated by means of the update gate and the elements of the vector of the candidate values, see Equation 10.

The equations used by the GRU network are shown in the equations below (Chung et al. 2014; Rahman et al. 2019). The variables used in the equations represent: z_t for the update gate, q_t for the reset gate, U_j are the weights for the previous hidden state of the respective gate and $\tilde{r}_{t-1}$ is the vector of candidate values for the hidden state. The $\odot$ refers to an element-wise product, while a $\cdot$ refers to an element-wise multiplication.

$$z_t = \sigma(w_z \cdot x_t + U_z \cdot r_{t-1}) \quad (7) \qquad q_t = \sigma(w_q \cdot x_t + U_q \cdot r_{t-1}) \quad (9)$$

$$\tilde{r}_t = tanh(w_r \cdot x_t + q_t \odot U_q r_{t-1}) \quad (8) \qquad r_t = (1 - z_t) \odot r_{t-1} + z_t \odot \tilde{r}_t \quad (10)$$

4.4 Self-Attention Mechanism

Attention mechanisms are a well-known technique for analyzing time series data and lead to better results when combined with a LSTM network (Xu et al. 2015). It is also shown that it improves the performance of RNN (Qin et al. 2017). Self-attention difference from the general attention mechanism. The self-attention mechanism, also called intra-attention, is a mechanism that relates different positions of a sequence to model dependencies between different parts of the sequence (Vaswani et al. 2017). The general attention mechanism tries to discover the important parts of the sequences relating to the network output, but the self-attention mechanism tries to find the important parts of the sequences which relate to each other. This is done to leverage those intra-sequence relationships to improve the predictive power of the network (Cheng et al. 2016). The standard RNN is not able to detect important parts of the information that flow through the network. The attention mechanism is essentially a neural network within a neural network, which teaches the model to weigh portions of a sequence for relative feature importance (Y. Wang et al. 2016). So the attention mechanism can capture the important parts of the sequence in response to the given aspect. The general concept of the attention mechanism is that attention can be adjusted to work with temporary classification problems. Whereby the used sequences are a fixed-length vector that contains a collection of time-series data, in our case stock market returns. The attention that is used in this paper is the self-attention layer and the following formulas are used in this attention mechanism (Y. Wang et al. 2016):

$$h_{t,t\text{-}1} = tanh(x_t^T W_t + x_{t-1}^T W_x + b_t) \quad (11) \qquad a_t = softmax(e_t) \quad (13)$$

$$e_{t,t\text{-}1} = \sigma(W_a h_{t,t-1} + b_a) \quad (12) \qquad l_t = \sum_{t-1} a_{t,t-1} x_{t-1} \quad (14)$$

Equation 11 shows the hidden node output ($h_{t,t-1}$) in a two-dimensional matrix, whereby

W_t and W_x are the weight matrices corresponding to the hidden states x_t and x_{t-1}, which is the output from the LSTM or GRU layer, and b_t is the bias vector, which is the deviation of the attention mechanism. In Equation 12 the sigmoid activation output of the attention two-layer network is shown. In this equation, W_a is the weight matrix corresponding to their non-linear combinations and b_a is the bias vector. The weight matrices indicate, which information should be emphasized. Furthermore, Equation 13 shows the softmax activation of e_t. This produces a vector with weights between 0 and 1. This vector weights the importance of the individual parts of the batched sequences. If the value is close to 1 a lot attention is paid to that value. So the attention mechanism calculates the weighted effect of each input unit in the input unit on each output sample of the output sequence (Wani et al. 2020). Equation 14 shows the attention-focused hidden state representation l_t of a token at timestamp t, which is given by a weighted summation of the hidden state representation x_{t-1}.

4.5 Model Training

Hyperparameters are model parameters that have to be selected before the estimation process (Lago et al. 2018). In the LSTM and GRU network, these are the parameters: batch size, epochs, the number of layers, the number of neurons in each layer, optimizer, activation function, loss function, and dropout regularization. For the LSTM benchmark model, the hyperparameter setting is chosen by Fischer and Krauss (2018) and can be found in Appendix D. For the LSTM network with attention mechanism, the GRU network, and the GRU network with attention mechanism, we perform a grid search to find the optimal hyperparameters.

Before the training of the LSTM with attention mechanism, the GRU network, and the GRU network with attention mechanism, a grid search is performed to find the optimal parameters. Grid search is a trial and error method for every hyperparameter setting (Aszemi and Dominic 2019). The grid search iterates through every possible combination for the different hyperparameters and searches for the combination of hyperparameters with the lowest loss. The different options for all the hyperparameters can be found in Appendix E, F, and G, for the three models. The hyperparameter for the three models for which a grid search is performed are: batch size, number of neurons in each layer, number of LSTM or GRU layers, the dropout rate in each layer, and the number of epochs. The optimizer function used in these networks is the AdamW optimizer. This optimization function is very similar to the Adam optimization function, but the difference is that the AdamW function makes use of the decoupled weight decay from the gradient update. The Adam optimization function performed the best compared to seven different optimization functions including RMSprop and Adagrad (Reimers and

Gurevych 2017). The AdamW optimization function leads to lower training losses and testing errors compared to Adam optimization function, applied in deep neural networks (Loshchilov and Hutter 2017). That is why we chose the AdamW function as our optimization function. The loss function that is used for the networks is the binary crossentropy because this loss function is used in binary classification problems (Ketkar 2017). The activation function used for the LSTM and GRU layers will be the default activation functions as explained in Section 4.3.1 and Section 4.3.2 respectively. The activation function for the dense layers, one dense and one output layer, in all the networks is the Rectified Linear Unit (ReLU) activation function. In the article of Javid et al. (2021), they found that adding a dense layer with the ReLU activation function improves the performance of various types of neural networks. To prevent the model from overfitting we use a dropout function in the LSTM and GRU layers, which randomly drops a fraction of the input units at each update during training to reduce the risk of overfitting and to improve the out-of-sample performance (Srivastava et al. 2014). Also, we make use of early stopping patience of 10, which prevents the model from overfitting.

4.6 Trading Strategy

For all the models, we forecast the probability $\hat{P}^s_{t=1|t}$ for each stock s to out-/underperform the cross-sectional median in period $t+1$, only making use of information up until time t. We then rank all stocks in each period $t+1$ with the highest probability at the top and the lowest probability at the bottom. The top of the ranking corresponds to the most undervalued stocks, which are expected to outperform the cross-sectional median in $t+1$. The trading strategy is that we go long (first buy and then sell) in the top k and short (first sell and then buy) in the flop k stocks of each ranking, for a long-short portfolio consisting of $2k$ stocks, this trading strategy is suggested by Huck (2009) and Huck (2010). In this paper, we choose $k \in \{5, 10, 15\}$.

4.7 Output analysis

To evaluate the different portfolios and models, we look at the predictive accuracy and the details on financial performance prior to and after transaction costs.

To test the predictive accuracy of the models, we use two tests to compare the models. First, we test with the Diebold-Mariano (DM) test (Diebold and Mariano 2002) to evaluate the null that the forecast of model i have inferior accuracy than the forecasts of model j, for $i, j \in \{RAF, LSTM, LSTMattention, GRU, GRUattention\}$ and $i \neq j$. For each forecast and each model, 0 is assigned if the forecast is correctly specified and 1 otherwise, and we use this vector with classification errors for the DM test. Second, a Pesaran-Timmermann (PT) (Pesaran

and Timmermann 1992) test to evaluate the null hypotheses that prediction and response are independently distributed for each of the forecasting models.

In this paper, we also look at the financial performance of each portfolio. First, we look at the return characteristics: mean return, standard error, minimum, first quartile, median, third quartile, maximum, share of positive return, standard deviation, skewness, and kurtosis. Second, we look at the risk characteristics: daily value at risk (VaR), the daily conditional value at risk (CVaR) both for 1, 2, and 5 percent, and the maximum drawdown. The VaR calculates the maximum expected loss of a portfolio or investment, given normal market conditions, over a given time period and given the probability p, $p \in \{0.01, 0.02, 0.05\}$ (Linsmeier and Pearson 2000). CVaR is also called the mean excess loss because the CVaR is the expected loss if the worst-case threshold is crossed, so when the losses go beyond the VaR breakpoint (Rockafellar, Uryasev, et al. 2000). The maximum drawdown represents the maximum loss from a peak to a trough of a portfolio, it measures how sustained one's losses can be (Magdon-Ismail and Atiya 2004). Lastly, we look at the annualized risk-return metrics: annualized return, annualized standard deviation, annualized downside deviation, annualized Sharpe ratio, and annualized Sortino ratio. The annualized downside deviation is the downside risk. It focuses on the return that falls below a minimum threshold and it is used to calculate the Sortino ratio. The Sharpe ratio calculates the risk-adjusted return, see Equation 15 (Sharpe 1994). The Sortino ratio also calculates the risk-adjusted return of the portfolio, but it is a modification of the Sharpe ratio and uses the annualized downside deviation instead of the annualized standard deviation, see Equation 16 (Mohan et al. 2016). In this formula R_f stands for the annualized return of the portfolio, R_f is the risk-free rate, σ_p is the standard deviation of the annualized portfolio's excess return, and DD stands for the annualized downside deviation. For both the measures hold that the higher these values, the better the portfolio. As a measurement for the risk-free rate, we take the one-month Treasury rate, which is at 0.05%. We take the Treasury bills as the measurement for the risk-free rate because Treasury bills are considered to be nearly free of default risk since they are fully backed by the U.S. government.

$$\text{Sharpe}\, ratio = \frac{R_p - R_f}{\sigma_p} \qquad (15) \qquad\qquad \text{Sortino}\, ratio = \frac{R_p - R_F}{DD} \qquad (16)$$

5 Results

The results are presented for the portfolios with the random forest (RAF), LSTM network, LSTM network with attention mechanism (LSTM AT), GRU network, and GRU network with attention mechanism (GRU AT). We analyze the returns prior to and after transaction costs of

5 bps per half-turn, following Avellaneda and Lee (2010). The used software to maintain the results can be found in Appendix B.

5.1 Grid search

For the LSTM At network, the grid search results in a network with four LSTM layers. The first LSTM layer contains of 256 neurons and a dropout rate of 0.7. The second layer is the attention layer as explained in Section 4.4. The third, fourth, and fifth layers are LSTM layers with 128, 256, and 128 neurons respectively and all these layers have a dropout rate of 0.5. The last two layers are dense layers with 128 and 2 neurons respectively. The batch size is 128 and the epochs are 1000. The options for each hyperparameter can be shown in Appendix E. The GRU network contains three GRU layers. The first and second layers contain 256 neurons and a dropout rate of 0.5 and 0.7 respectively. The third GRU layer contains 64 neurons and a dropout rate of 0.3. The batch size for this network is 512, and the number of epochs is 1000. The options for each hyperparameter can be shown in Appendix F. The last model is the GRU At network. This model has three GRU layers. The first GRU layer contains 28 neurons and a dropout rate of 0.5. The second layer is the attention mechanism as explained in Section 4.4. The third and fourth layers have 512 and 128 neurons respectively and a dropout rate of 0.7 and 0.5 respectively. The model contains two dense layers which have 256 and 2 neurons respectively. This model also has a batch size of 512 and uses 1000 epochs. The options for each hyperparameter can be shown in Appendix G.

5.2 Trading strategy Evaluation

First, we analyze the characteristics of the different portfolios, which consist of $2k$ stocks, for the top k stocks we go long, and for the flop k stocks we go short. We have three trading strategies $k \in \{5, 10, 15\}$ and compare the performance of the portfolios for the daily mean returns, annualized standard deviation, annualized Sharpe ratio, and accuracy, prior to transaction costs.

In Table 1 the daily performance for the long-short portfolios for different sizes are presented. Irrespective of the size k, the RAF and LSTM network show positive daily returns. Specifically, the daily returns prior to transaction costs at $k = 5$ are at 0.20% for the RAF and 0.19% for the LSTM network, compared to 0.00% for the LSTM At network, -0.02% for the GRU network, and -0.02% for the GRU At network. Also for larger portfolio sizes, the LSTM and RAF achieve the highest mean returns per day. With respect to the standard deviation, a risk metric, The LSTM At, GRU, and GRU At network exhibit a much lower standard deviation than the RAF and LSTM network, across all levels of k. For the Sharpe ratio, the return per unit

of risk is highest for the RAF for $k = 5$, and slightly less for the LSTM network. The Sharpe ratios for larger portfolios for the RAF are slightly less, but for the LSTM, LSTM At, GRU, and GRU AT network the Sharpe ratio is higher for the portfolio with $k = 10$. With respect to the accuracy, which is an important machine learning metric, meaning the share of correct classifications, we see a clear advantage for the RAF and LSTM network with 50.58% and 50.59% respectively. When looking at the other three models their accuracy is around 50.00% and becomes slightly higher for the larges portfolios.

We focus on the long-short portfolio with $k = 5$.

Table 1: The daily performance characteristics for the long-short portfolio for k=5, k=10, and k=15 before transaction costs. For the five different methods: the Random forest (RAF), LSTM network, LSTM network with attention layer (LSTM At), GRU network, and GRU network with attention layer (GRU At).

	k=5					k=10					k=15				
	RAF	**LSTM**	**LSTM At**	**GRU**	**GRU At**	**RAF**	**LSTM**	**LSTM At**	**GRU**	**GRU At**	**RAF**	**LSTM**	**LSTM At**	**GRU**	**GRU At**
Return [%]	0.20	0.19	0.00	-0.02	-0.02	0.15	0.19	-0.01	0.01	-0.01	0.12	0.12	-0.02	0.01	-0.01
Standard dev. [%]	1.47	1.76	0.90	0.94	1.07	1.00	1.81	0.92	0.67	0.83	0.82	1.06	0.61	0.72	0.76
Sharpe ratio	3.13	2.91	0.01	-0.33	-0.29	3.03	2.97	0.01	0.19	0.17	2.99	2.63	-0.35	0.18	-0.02
Accuracy[%]	50.58	50.59	50.01	50.00	50.01	50.58	50.49	50.15	50.03	50.13	50.58	50.59	50.02	50.00	50.00

5.2.1 Predictive accuracy

In this section, we benchmark the predictive accuracy of the LSTM network with attention mechanism (LSTM at), GRU network, and the GRU network with attention mechanism (GRU At) against those of the benchmark models. We use the Diebold-Mariano (DM) test and the Pesaran-Timmermann (PT) test as explained in Section 4.7. The results of these two tests are presented in Table 2. In panel A of Table 2, we see the p-values of the DM test. If we test at a five percent significance level, we cannot reject the null hypothesis for all the methods. This means that we cannot infer that for all the model forecasts they outperform the other model forecasts. All methods seem to exhibit similar predictive accuracy. In panel B of Table 2, the p-values of the PT test are presented. If we look at a five percent significant level, we can reject the null hypothesis for all methods. This means that every method that we use in this paper shows statistically significant predictive accuracy.

5.2.2 Predictive characteristics

Table 3 provides insights into the financial performance of the LSTM At network, GRU network, and the GRU At network, compared to the benchmark models, prior and after transaction costs for the portfolio with $k = 5$. The results for the portfolios with $k = 10$ and $k = 15$ can be found in Appendix H and Appendix I.

Table 2: Panel A: P-values of Diebold-Mariano (DM). Panel B: P-values of the Pesaran-Timmermann (PT) test. Both panels are based on the k = 5 portfolio from January 1993 to December 2020.

A: DM test							**B:** PT test	
i	$j =$	RAF	LSTM	LSTM At	GRU	GRU At	Method	Result
RAF		-	0.1254	0.1645	0.1154	0.0961	RAF	0.0258
LSTM		0.8746	-	0.5228	0.7524	0.7389	LSTM	0.0145
LSTM At		0.8355	0.7486	-	0.7776	0.5192	LSTM Attention	0.0495
GRU		0.8846	0.2476	0.2224	-	0.6089	GRU	0.0304
GRU At		0.9039	0.2611	0.4808	0.3991	-	GRU Attention	0.0445

Table 3: Panels A, B, and C illustrate the performance of the k=5 portfolio, before and after transaction costs for the Random forest (RAF), LSTM network, LSTM network with attention layer (LSTM At), GRU network, and GRU network with attention layer (GRU At). Panel A depicts daily return characteristics. Panel B depicts daily risk characteristics. Panel C depicts annualized risk-return metrics.

		Before transaction costs					After transaction costs				
		RAF	**LSTM**	**LSTM At**	**GRU**	**GRU At**	**RAF**	**LSTM**	**LSTM At**	**GRU**	**GRU At**
A	**Mean return [%]**	0.1960	0.1857	-0.0021	-0.0190	-0.0192	0.0960	0.08570	-0.1021	-0.1190	-0.1192
	Standard error	0.0009	0.0011	0.0006	0.0006	0.0007	0.0009	0.0011	0.0006	0.0006	0.0007
	Minimum	-0.0493	-0.0652	-0.0316	-0.0333	-0.0426	-0.0503	-0.0662	-0.0326	-0.0343	-0.0436
	Quartile 1	-0.0063	-0.0082	-0.0055	-0.0058	-0.0062	-0.0073	-0.0092	-0.0065	-0.0068	-0.0072
	Median	0.0019	0.0019	0.0000	-0.0002	0.0000	0.0001	0.0009	-0.0010	-0.0012	-0.0010
	Quartile 3	0.0101	0.0116	0.0056	0.0054	0.0059	0.0091	0.0106	0.0046	0.0044	0.0049
	Maximum	0.0623	0.0745	0.0317	0.0341	0.0413	0.0613	0.0725	0.0307	0.0331	0.0403
	Share > 0	55.4938	53.9053	49.6885	48.8831	49.6887	52.1271	51.0721	44.6327	43.9427	44.5909
	Standard dev.	0.0147	0.0176	0.0090	0.0094	0.0107	0.0127	0.0176	0.0090	0.0094	0.0107
	Skewness	0.1063	0.1201	0.0510	0.0077	-0.0893	0.1063	0.1201	0.0510	0.0077	-0.0893
	Kurtosis	2.7175	2.6115	1.6362	1.4458	2.5944	2.7175	2.6115	1.6362	1.4458	2.5944
B	**1-percent VaR**	-0.0416	-0.0515	-0.0266	-0.0267	-0.0332	-0.0426	-0.0525	-0.0276	-0.0277	-0.0342
	1-percent CVaR	-0.0455	-0.0583	-0.0291	-0.0300	-0.0313	-0.0465	-0.0593	-0.0301	-0.0310	-0.0389
	2-percent VaR	-0.0314	-0.0374	-0.0201	-0.0209	-0.0247	-0.0324	-0.0384	-0.0211	-0.0219	-0.0257
	2-percent CVaR	-0.0387	-0.0477	-0.0248	-0.0256	-0.0313	-0.0397	-0.0487	-0.0258	-0.0266	-0.0323
	5-percent VaR	-0.0220	-0.0261	-0.0149	-0.0162	-0.0180	-0.0230	-0.0271	-0.0159	-0.0172	-0.0190
	5-percent CVaR	-0.0308	-0.0375	-0.0199	-0.0210	-0.0247	-0.0318	-0.0385	-0.0209	-0.0220	-0.0257
	Max. drawdown	0.1550	0.2132	0.1535	0.1735	0.0022	0.2171	0.2899	0.3048	0.3505	0.0037
C	**Return p.a. [%]**	79.8290	87.8769	0.0954	-4.0430	-3.7126	40.0670	46.3428	-22.0656	-25.2886	-25.0310
	Standard dev. p.a.	0.2332	0.2791	0.1430	0.1489	0.1696	0.2332	0.2791	0.1430	0.1489	0.1696
	Downside dev. p.a.	0.1474	0.1804	0.0920	0.0969	0.1158	0.1487	0.1817	0.0940	0.0988	0.1174
	Sharpe ratio	3.1261	2.9053	0.0086	-0.3298	-0.2930	1.3612	1.3448	-1.6413	-1.8361	-1.7024
	Sortino ratio	5.3219	5.3101	0.0397	-0.5050	-0.3893	2.4093	2.6308	-2.5096	-2.8001	-2.4644

Return characteristics: In panel A of Table 3, we see that the returns of the RAF and LSTM network are positive. The returns for the LSTM At, GRU, and GRU At network are negative. The daily mean return for the RAF is 0.20% before and 0.10% after transaction costs. The median for all the methods is the same or only slightly smaller than the mean returns. If we look at the quartiles as well as the minimum and maximum values, we see that the results suggest that these are not caused by outliers. The share of positive return is the highest for the RAF and is 55.49% before and 52.13% after transaction costs. For the LSTM At network, GRU network, and GRU At network the share of positive return is less than 50% before transaction costs. The

second-best model is the LSTM network. This model performs the same as the RAF but the boundaries are wider compared to the RAF. The LSTM At network places third with a daily mean return of -0.1021% after transaction costs. The GRU network and GRU At network place fourth and fifth with a daily mean return of -0.1190% and -0.1192% respectively, after transaction costs. If we take a closer look at the standard error and standard deviation, we see that the LSTM At, GRU, and GRU At networks have lower standard errors and standard deviations. This suggests that in these methods there is less variability within and across the sample.

Risk characteristics: In panel B of Table 3, if we look at the risk characteristics, we observe a mixed picture. For the daily value at risk (VaR) and the daily conditional value at risk (CVaR) for 1-percent, the LSTM At network shows the lowest values -2.76% and -3.01% respectively after transaction costs. The riskiest method is the LSTM network with a 1-percent VaR of -5.25%, which means that in 1% of all the cases the loss is -5.25%. This is almost twice as risky as the LSTM At network and GRU network. However, the GRU At network has the lowest maximum drawdown 0.37%, suggesting lower downside risk, and therefore whenever losses are made they are relatively lower than those of the other portfolios.

Annualized risk-return metrics: In panel C of Table 3, we analyze the annualized risk-return metrics. We observe that the LSTM model achieves the highest annualized return of 46.34% after transaction costs, compared to the random forest (40.07%), LSTM At network (-22.07%), GRU network (-25.29%), and GRU At network (-25.03%). the LSTM At network, GRU, and GRU At network have the lowest annualized standard deviations and downside deviation respectively. If we look at the Sharpe ratio, we observe that the RAF achieves the highest level of 1.36, with the LSTM network coming in second with 1.34, while all the other methods have a Sharpe ratio below 0.0. But the LSTM network achieves the highest level for the Sortino ratio with a value of 2.63 after transaction costs.

5.2.3 A critical review in crisis times

In 2008 there was a financial crisis going on all over the world, also known as the global financial crisis. In 2020 a pandemic, called COVID-19, gripped the entire world. In these two years, the stock market prices show more volatility compared to the years before (Banchit et al. 2016; Uddin et al. 2021). That is why it is interesting to know which portfolio achieves the highest returns in more volatile periods. In Table 4 the performance of all the five portfolios is shown for both the years 2008 and 2020.

If we look at the performance of the five portfolios in Table 4 for the global financial crisis in 2008 and in the COVID-19 period in 2020, we see that the LSTM network achieves the highest

returns in both periods. In 2008 all the methods achieve positive daily returns, but in 2020 only the RAF and the LSTM network achieve positive returns. The Sharpe ratio is also the highest for the LSTM network in both periods. For the standard deviation, we observe that in both periods the LSTM At network has the lowest standard deviation, with the GRU network coming second. If we take a further look at the accuracy rate, we see that in 2008 the RAF has the highest accuracy and the LSTM network has the highest accuracy in 2020. Furthermore, it is interesting that in 2008 the RAF, LSTM At, and GRU At network are comparable to each other that year because their daily returns and Sharpe ratio do not differ much from each other.

Table 4: The performance of the k=5 long-short portfolio before transaction costs in a period of the financial crisis in 2008 and the corona crisis in 2020, for the Random forest (RAF), LSTM network, LSTM network with attention layer (LSTM At), GRU network, and GRU network with attention layer (GRU At).

	2008					2020				
	RAF	**LSTM**	**LSTM At**	**GRU**	**GRU At**	**RAF**	**LSTM**	**LSTM At**	**GRU**	**GRU At**
Return[%]	0.0315	0.1725	0.0273	0.0028	0.0276	0.1733	0.3373	-0.0619	-0.0158	-0.0238
Standard dev.[%]	2.5563	3.4329	1.3621	1.3783	2.0225	1.9310	3.2542	1.1039	1.4656	2.2684
Sharpe ratio	0.2037	0.9990	0.3297	0.0329	0.2246	1.7848	2.5864	-0.8246	-0.1692	-0.1620
Accuracy[%]	50.55	49.97	50.03	50.05	49.95	50.27	51.08	49.96	49.84	50.00

6 Conclusion

This research applies Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks with attention mechanism and long-short portfolios to the forecasting of stock returns of 112 S&P 500 constituent public equities, from January 1990 until December 2020. The research question focuses on whether the attention mechanism results in a more stable portfolio with higher returns compared to the random forest and LSTM network.

The results show that the long-short portfolio with ten stocks gives the overall best results compared to the portfolios with 20 and 30 stocks. Looking at the return characteristics of the portfolio with ten stocks, the LSTM network and random forest achieve the highest returns. The LSTM network with attention mechanism, GRU network, and GRU network with attention mechanism do on average not result in positive returns. Further, if we look at the risk characteristics the results show that LSTM network with attention mechanism and GRU network have the lowest daily value at risk. Moreover, the GRU network has the lowest maximum drawdown, which indicates that the losses from the investment were small. Next, we analyze the annualized risk-return metrics. These results show that for the LSTM network the annualized returns are the highest with a return of 46.34% per year. When comparing the risk-adjusted return, we conclude that this is the highest for the random forest, with a rate of 1.36. Finally,

we look at two periods for which the volatility was higher compared to other years, the global financial crisis (2008) and the COVID-19 period (2020). The results show that in both periods the LSTM network achieves the highest daily returns and risk-adjusted returns. When comparing the standard deviations for both periods, the LSTM network with attention layer and GRU network have the lowest standard deviation and the LSTM network the highest, which corresponds to the results for the whole sample period. Overall, the long-short portfolios with the LSTM network and random forest resulted in the highest daily, annualized, and risk-adjusted returns. Adding an attention mechanism to the LSTM network did not result in higher returns for the portfolio, but it did result in a less risky portfolio. The GRU network and GRU network with attention mechanism showed similar results for the portfolio on average. When comparing the portfolios of the LSTM network with attention mechanism and the GRU network with attention mechanism, it shows that the portfolio with the LSTM network with attention achieves higher daily and risk-adjusted returns and is a less risky portfolio.

The main contribution to the current body of literature on financial market prediction is an enhanced insight into the influence of the attention mechanism on the GRU and LSTM networks. Furthermore, this research contributes to the understanding of the main contribution of the attention mechanism on the GRU network.

There are several avenues for extending this research. Firstly, we would suggest to extend the study periods for the networks. LSTM and GRU networks are able to handle a large amount of data (C. Wang et al. 2020). Extending the study periods, for example, not three but five years would give the models more information to learn from. If they have more to learn from, they could possibly predict more accurately. Another interesting extension for this research is adding textual data to the dataset. Our results show that adding an attention mechanism does not result in higher daily returns for the portfolio. The attention mechanism is mostly used in the field of natural language processing (G. Liu and Guo 2019). In this research, the input for the models is the returns from the past 240 days. It could be that the attention mechanism was not able to locate the informative data segments in the sequence. In recent studies (Jin et al. 2019; H. Liu 2018), they found that adding textual information to the dataset of the neural network with attention mechanism improved prediction accuracy. Further research could focus on the performance of the LSTM network and GRU network with attention mechanism with textual information as input. Lastly, it is interesting to evaluate if adding more companies and a larger portfolio, results in higher returns for the portfolios. The results show that for larger portfolios the models with more layers result in higher return per unit of risk and a higher accuracy rate.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv e-prints*, Article arXiv:1603.04467, arXiv:1603.04467.

Abu-Mostafa, Y. S., & Atiya, A. F. (1996). Introduction to financial forecasting. *Applied intelligence*, *6*(3), 205–213.

Adam, K., Marcet, A., & Nicolini, J. P. (2016). Stock market volatility and learning. *The Journal of Finance*, *71*(1), 33–82.

Aszemi, N. M., & Dominic, P. (2019). Hyperparameter optimization in convolutional neural network using genetic algorithms. *Int. J. Adv. Comput. Sci. Appl.*, *10*(6), 269–278.

Avellaneda, M., & Lee, J.-H. (2010). Statistical arbitrage in the us equities market. *Quantitative Finance*, *10*(7), 761–782.

Banchit, A., Abidin, S., & Wu, J. (2016). Are shares more volatile during the global financial crisis? *Procedia-Social and Behavioral Sciences*, *224*, 221–229.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, *31*(3), 307–327.

Booth, A., Gerding, E., & McGroarty, F. (2014). Automated trading with performance weighted random forests and seasonality. *Expert Systems with Applications*, *41*(8), 3651–3661.

Borovkova, S., & Tsiamas, I. (2019). An ensemble of LSTM neural networks for high-frequency stock market classification. *Journal of Forecasting*, *38*(6), 600–619.

Box, G. E., & Jenkins, G. M. (1976). *Time series analysis: Forecasting and control*. Calif: Holden-Day, San Francisco.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv e-prints*, Article arXiv:1601.06733, arXiv:1601.06733.

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv e-prints*, Article arXiv:1409.1259, arXiv:1409.1259.

Chollet, F. et al. (2015). *Keras*. Retrieved June 23, 2021, from `https://github.com/fchollet/keras`

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv e-prints*, Article arXiv:1412.3555, arXiv:1412.3555.

Coval, J. D., Hirshleifer, D. A., & Shumway, T. (2005). Can individual investors beat the market? *Working paper, Harvard University*.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303–314.

Dey, R., & Salem, F. M. (2017). Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. *arXiv e-prints*, Article arXiv:1701.05923, arXiv:1701.05923.

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & economic statistics*, *20*(1), 134–144.

Dupond, S. (2019). A thorough review on the current advance of neural network structures. *Annual Reviews in Control*, *14*, 200–230.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica: Journal of the econometric society*, *50*(4), 987–1007.

Fama, E. F. (1965). The behavior of stock-market prices. *The journal of Business*, *38*(1), 34–105.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical works. *Journal of Finance*, *25*(2), 383–417.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669.

Fu, R., Zhang, Z., & Li, L. (2016). Using LSTM and GRU neural network methods for traffic flow prediction. *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)* (pp. 324–328). IEEE.

Garcia, C., & Gould, F. (1993). Survivorship bias. *Journal of Portfolio Management*, *19*(3), 52.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural computation*, *12*(10), 2451–2471.

Greff, K., Srivastava, R. K., Koutnık, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*(10), 2222–2232.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, *44*(1), 1–12.

Ho, T. K. (1995). Random decision forests. *Third International Conference on Document Analysis and Recognition* (pp. 832–844). IEEE.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Huck, N. (2009). Pairs selection and outranking: An application to the S&P 100 index. *European Journal of Operational Research*, *196*(2), 819–825.

Huck, N. (2010). Pairs trading and outranking: The multi-step-ahead forecasting case. *European Journal of Operational Research*, *207*(3), 1702–1716.

Hyndman, R., Koehler, A. B., Ord, J. K., & Snyder, R. D. (2008). *Forecasting with exponential smoothing: The state space approach*. Springer Science & Business Media.

Javid, A. M., Das, S., Skoglund, M., & Chatterjee, S. (2021). A ReLU dense layer to improve the performance of neural networks. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2810–2814). IEEE.

Jia, M., Huang, J., Pang, L., & Zhao, Q. (2019). Analysis and research on stock price of LSTM and bidirectional LSTM neural network. *Advances in Computer Science Research (ACSR)*, *90*(1), 467–473.

Jin, Z., Yang, Y., & Liu, Y. (2019). Stock closing price prediction based on sentiment analysis and lstm. *Neural Computing and Applications*, 1–17.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *81*(1), 35–45.

Ketkar, N. (2017). Introduction to keras. *Deep learning with python* (pp. 97–111). Apress, Berkeley, CA.

Lago, J., De Ridder, F., & De Schutter, B. (2018). Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms. *Applied Energy*, *221*, 386–405.

Li, H., Shen, Y., & Zhu, Y. (2018). Stock price prediction using attention-based multi-input lstm. *Proceedings of Machine Learning Research*, *95*(1), 454–469.

Li, X., Zhang, W., & Ding, Q. (2019). Understanding and improving deep learning-based rolling bearing fault diagnosis with attention mechanism. *Signal processing*, *161*(1), 136–154.

Linsmeier, T. J., & Pearson, N. D. (2000). Value at risk. *Financial Analysts Journal*, *56*(2), 47–67.

Liu, G., & Guo, J. (2019). Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing*, *337*, 325–338.

Liu, H. (2018). Leveraging Financial News for Stock Trend Prediction with Attention-Based Recurrent Neural Network. *arXiv e-prints*, Article arXiv:1811.06173, arXiv:1811.06173.

Livingstone, D. J., Manallack, D. T., & Tetko, I. V. (1997). Data modelling with neural networks: Advantages and limitations. *Journal of computer-aided molecular design*, *11*(2), 135–142.

Loshchilov, I., & Hutter, F. (2017). Decoupled Weight Decay Regularization. *arXiv e-prints*, Article arXiv:1711.05101, arXiv:1711.05101.

Magdon-Ismail, M., & Atiya, A. F. (2004). Maximum drawdown. *Risk Magazine*, *17*(10), 99–102.

McKinney, W. et al. (2011). Pandas: A foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, *14*(9), 1–9.

Mohan, V., Singh, J. G., & Ongsakul, W. (2016). Sortino ratio based portfolio optimization considering evs and renewable energy in microgrid power market. *IEEE Transactions on Sustainable Energy*, *8*(1), 219–229.

Niu, Z., Zhong, G., & Yu, H. (2021). A review on the attention mechanism of deep learning. *Neurocomputing*, *452*(1), 48–62.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, *12*, 2825–2830.

Pesaran, M. H., & Timmermann, A. (1992). A simple nonparametric test of predictive performance. *Journal of Business & Economic Statistics*, *10*(4), 461–465.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., & Cottrell, G. (2017). A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. *arXiv e-prints*, Article arXiv:1704.02971, arXiv:1704.02971.

Qiu, J., Wang, B., & Zhou, C. (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PloS one*, *15*(1), e0227222.

Rahman, M. O., Hossain, M. S., Junaid, T.-S., Forhad, M. S. A., & Hossen, M. K. (2019). Predicting prices of stock market using gated recurrent units (GRUs) neural networks. *Int. J. Comput. Sci. Netw. Secur*, *19*(1), 213–222.

Refenes, A. N., Zapranis, A., & Francis, G. (1994). Stock performance modeling using neural networks: A comparative study with regression models. *Neural networks*, *7*(2), 375–388.

Reimers, N., & Gurevych, I. (2017). Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. *arXiv e-prints*, Article arXiv:1707.06799, arXiv:1707.06799.

Rockafellar, R. T., Uryasev, S. et al. (2000). Optimization of conditional value-at-risk. *Journal of risk*, *2*, 21–42.

Sharma, S. [Sagar], & Sharma, S. [Simone]. (2017). Activation functions in neural networks. *Towards Data Science*, *6*(12), 310–316.

Sharpe, W. F. (1994). The sharpe ratio. *Journal of portfolio management*, *21*(1), 49–58.

Song, Y. (2018). *Stock trend prediction: Based on machine learning methods* (Doctoral dissertation). Doctoral dissertation, UCLA.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Sutskever, I., Martens, J., & Hinton, G. E. (2011). Generating text with recurrent neural networks. *Proceedings of the 28th International Conference on Machine Learning (ICML)* (pp. 1033–1040).

Tipirisetty, A. (2018). Stock price prediction using deep learning.

Uddin, M., Chowdhury, A., Anderson, K., & Chaudhuri, K. (2021). The effect of covid–19 pandemic on global stock market volatility: Can economic strength help to manage the uncertainty? *Journal of Business Research*, *128*, 31–44.

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in science & engineering*, *13*(2), 22–30.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *arXiv e-prints*, Article arXiv:1706.03762, arXiv:1706.03762.

Wang, C., Du, W., Zhu, Z., & Yue, Z. (2020). The real-time big data processing method based on lstm or gru for the smart job shop production process. *Journal of Algorithms & Computational Technology*, *14*, 1–9.

Wang, X., Yu, L., Ren, K., Tao, G., Zhang, W., Yu, Y., & Wang, J. (2017). Dynamic attention deep model for article recommendation by learning human editors' demonstration. *Proceedings of the 23rd ACM SIGKKD International Conference on Knowledge Discovery and Data Mining* (pp. 2051–2059).

Wang, Y., Huang, M., Zhu, X., & Zhao, L. (2016). Attention-based lstm for aspect-level sentiment classification. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 606–615).

Wani, M. A., Khoshgoftaar, T. M., & Palade, V. (2020). *Deep learning applications, volume 2*. Springer Singapore PTE. Limited, 2020.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *Proceedings of the 32 nd International Conference on Machine Learning, Lille, France, 2015.* (pp. 2048–2057). PMLR.

Yu, H.-F., Rao, N., & Dhillon, I. S. (2016). Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. *Advances in Neural Information Processing Systems NIPS* (pp. 847–855).

Zhang, G. P. (2001). An investigation of neural networks for linear time-series forecasting. *Computers & Operations Research*, *28*(12), 1183–1202.

Zhang, L., Wang, F., Xu, B., Chi, W., Wang, Q., & Sun, T. (2018). Prediction of stock prices based on lm-bp neural network and the estimation of overfitting point by rdci. *Neural Computing and Applications*, *30*(5), 1425–1444.

Zhang, Y., Chu, G., & Shen, D. (2021). The role of investor attention in predicting stock prices: The long short-term memory networks perspective. *Finance Research Letters*, *38*, 101484.

Zhao, M., Wang, H., Guo, J., Liu, D., Xie, C., Liu, Q., & Cheng, Z. (2019). Construction of an industrial knowledge graph for unstructured Chinese text learning. *Applied Sciences*, *9*(13), 2720.

Zheng, X., Zaheer, M., Ahmed, A., Wang, Y., Xing, E. P., & Smola, A. J. (2017). State Space LSTM Models with Particle MCMC Inference. *arXiv e-prints*, Article arXiv:1711.11179, arXiv:1711.11179.

Zhou, H. (2015). *Keras-self-attention*. Retrieved June 23, 2021, from `https://github.com/CyberZHG/keras-self-attention`

Zhuge, Q., Xu, L., & Zhang, G. (2017). LSTM neural network with emotional analysis for prediction of stock price. *Engineering letters*, *25*(2).

7 Appendices

Appendix A

Table 5: Companies included in the dataset. List of S&P 500 constituent stocks used for equity return prediction

Company name	Ticker	Industry	Company name	Ticker	Industry
3M Company	MMM	Industrials	International Business Machines	IBM	Information Technology
AT&T Inc.	T	Communication Services	International Flavors & Fragrances	IFF	Materials
Abbott Laboratories	ABT	Health Care	International Paper	IP	Materials
Advanced Micro Devices	AMD	Information Technology	JPMorgan Chase & Co.	JPM	Financials
Air Products & Chemicals	APD	Materials	Johnson & Johnson	JNJ	Health Care
Altria Group Inc	MO	Industrials	Johnson Controls International	JCI	Industrials
American Electric Power	AEP	Utilities	Kellogg Co	K	Consumer Staples
American Express	AXP	Financials	Kimberly-Clark	KMB	Consumer Staples
American International Group	AIG	Financials	Kroger Co.	KR	Consumer Staples
Apple Inc.	AAPL	Information Technology	Lilly (Eli) & Co.	LLY	Health Care
Archer-Daniels-Midland Co	ADM	Consumer Staples	Lincoln National	LNC	Financials
Avery Dennison Corp	AVY	Materials	Lowe's Cos.	LOW	Consumer Discretionary
Ball Corp	BLL	Materials	Marsh & McLennan	MMC	Financials
Bank of America Corp	BAC	Financials	Masco Corp.	MAS	Industrials
Baxter International Inc.	BAX	Health Care	McDonald's Corp.	MCD	Consumer Discretionary
Becton Dickinson	BDX	Health Care	Medtronic plc	MDT	Health Care
Boeing Company	BA	Industrials	Merck & Co.	MRK	Health Care
Bristol-Myers Squibb	BMY	Health Care	Newell Brands	NWL	Consumer Discretionary
Brown-Forman Corp.	BF.B	Consumer Staples	Newmont Corporation	NEM	Materials
CSX Corp.	CSX	Industrials	Norfolk Southern Corp.	NSC	Industrials
Campbell Soup	CPB	Consumer Staples	Northrop Grumman	NOC	Industrials
Caterpillar INC.	CAT	Industrials	Nucor Corp.	NUE	Materials
Cigna	CI	Health Care	Occidental Petroleum	OXY	Energy
Citigroup Inc.	C	Financials	Oneok	OKE	Energy
Coca-Cola Company	KO	Consumer Staples	Oracle Corp.	ORCL	Information Technology
Colgate-Palmolive	CL	Consumer Staples	PNC Financial Services	PNC	Financials
Comcast Corp.	CMCSA	Communication Services	PPG Industries	PPG	Materials
Conagra Brands	CAG	Consumer Staples	Paccar	PCAR	Industrials
Consolidated Edison	ED	Utilities	Parker-Hannifin	PH	Industrials
Corning Inc.	GLW	Information Technology	PepsiCo Inc.	PEP	Consumer Staples
DTE Energy Co.	DTE	Utilities	Pfizer Inc.	PFE	Health Care
Deere & Co.	DE	Industrials	Procter & Gamble	PG	Consumer Staples
Dominion Energy	D	Utilities	Public Service Enterprise Group (PSEG)	PEG	Utilities
Dover Corporation	DOV	Industrials	PulteGroup	PHM	Consumer Discretionary
DuPont de Nemours Inc	DD	Specialty Chemicals	Rockwell Automation Inc.	ROK	Industrials
Duke Energy	DUK	Utilities	Schlumberger Ltd.	SLB	Energy
Eaton Corporation	ETN	Industrials	Sherwin-Williams	SHW	Materials
Ecolab Inc.	ECL	Materials	Snap-on	SNA	Industrials
Emerson Electric Company	EMR	Industrials	Southern Company	SO	Utilities
Entergy Corp.	ETR	Utilities	Stanley Black & Decker	SWK	Industrials
FMC Corporation	FMC	Materials	Sysco Corp.	SYY	Consumer Staples
FedEx Corporation	FDX	Industrials	TJX Companies Inc.	TJX	Consumer Discretionary
Ford Motor Company	F	Consumer Discretionary	Target Corp.	TGT	Consumer Discretionary
Gap Inc.	GPS	Consumer Discretionary	Texas Instruments	TXN	Information Technology
General Dynamics	GD	Industrials	Textron Inc.	TXT	Industrials
General Electric	GE	Industrials	The Clorox Company	CLX	Consumer Staples
General Mills	Gis	Consumer Staples	The Hershey Company	HSY	Consumer Staples
Genuine Parts	GPC	Consumer Discretionary	The Walt Disney Company	DIS	Communication Services
Grainger (W.W.) Inc.	GWW	Industrials	Union Pacific Corp	UNP	Industrials
Halliburton Co.	HAL	Energy	VF Corporation	VFC	Consumer Discretionary
Hasbro Inc.	HAS	Consumer Discretionary	Walmart	WMT	Consumer Staples
Home Depot	HD	Consumer Discretionary	Wells Fargo	WFC	Financials
Honeywell Int'l Inc.	HON	Industrials	Weyerhaeuser	WY	Real Estate
Humana Inc.	HUM	Health Care	Whirlpool Corp.	WHR	Consumer Discretionary
Illinois Tool Works	ITW	Industrials	Williams Companies	WMB	Energy
Intel Corp.	INTC	Information Technology	Xerox	XRX	Information Technology height

Appendix B

Collecting data, preparing data, grid search, model training, and model validation are conducted in Python 3.7. To do this, the following Python software libraries are used: NumPy (Van Der Walt et al. 2011), pandas (McKinney et al. 2011), sci-kit learn (Pedregosa et al. 2011), Keras (Chollet et al. 2015), Keras-self-attention (Zhou 2015), and Tensorflow (Abadi et al. 2016).

Appendix C

Table 6: Hyperparameters used in the Random Forest. As mentioned in Section 4.2, the hyperparameters for the random forest are presented in this table. For all the other hyperparameters, they choose to set it at their default value.

Hyperparameters	Hyperparameter setting
Number of trees	1000
Maximum depth	20
Feature subsampling	$m = \sqrt{p}$
Seed	1

Appendix D

Table 7: Hyperparameters used in the Long Short-Term Memory network. As mentioned in Section 4.5, the hyperparameters for the LSTM network are presented in this table. For all the other hyperparameters, they choose to set it at their default value.

Hyperparameters	Hyperparameter setting
Neurons LSTM layer	25
Dropout value	0.1
Neurons output layer	2
Activation function output layer	Softmax
Epochs	1000
Early stopping patience	10
Batch size	512
Loss function	binary crossentropy function

Appendix E

Table 8: Hyperparameters used in the Grid Search for the LSTM network with attention layer. As mentioned in Section 4.5, a grid search was performed on a number of hyperparameters in the LSTM network with attention mechanism. These hyperparameters and the options over which a grid search is performed are presented in this table.

Hyperparameters	Grid search options	Optimal choices
Batch size	32, 64, 128, 512	128
Units first LSTM layer	64, 128, 256, 512	256
Dropout first LSTM layer	0.1, 0.3, 0.5, 0.7	0.7
Units second LSTM layer	64, 128, 256, 512	128
Dropout second LSTM layer	0.1, 0.3, 0.5, 0.7	0.5
LSTM layers	2, 3 , 4	4
Units third LSTM layer	64, 128, 256, 512	256
Dropout third LSTM layer	0.1, 0.3, 0.5, 0.7	0.5
Units fourth LSTM layer	128, 256, 512	128
Dropout fourth LSTM layer	0.1, 0.3, 0.5, 0.7	0.5
Units dense layer	64, 128, 256, 512	128
Epochs	100, 1000	1000

Appendix F

Table 9: Hyperparameters used in the Grid Search for the GRU network. As mentioned in Section 4.5, a grid search was performed on a number of hyperparameters in the GRU network. These hyperparameters and the options over which a grid search is performed are presented in this table.

Hyperparameters	Grid search options	optimal choices
Batch size	32, 64, 128, 512	512
Units first GRU layer	64, 128, 256, 512	256
Dropout first GRU layer	0.1, 0.3, 0.5, 0.7	0.5
Units second GRU layer	64, 128, 256, 512	256
Dropout second GRU layer	0.1, 0.3, 0.5, 0.7	0.7
LSTM layers	2, 3 , 4	3
Units third GRU layer	64, 128, 256, 512	64
Dropout third GRU layer	0.1, 0.3, 0.5, 0.7	0.3
Units fourth GRU layer	64, 128, 256, 512	/
Dropout fourth GRU layer	0.1, 0.3, 0.5, 0.7	/
Units dense layer	64, 128, 256, 512	64
Epochs	100, 1000	1000

Appendix G

Table 10: Hyperparameters used in the Grid Search for the GRU network with attention mechanism. As mentioned in Section 4.5, a grid search was performed on a number of hyperparameters in the GRU network with attention mechanism. These hyperparameters and the options over which a grid search is performed are presented in this table.

Hyperparameters	Grid search options	optimal choices
Batch size	32, 64, 128, 512	512
Units first GRU layer	64, 128, 256, 512	128
Dropout first GRU layer	0.1, 0.3, 0.5, 0.7	0.5
Units second GRU layer	64, 128, 256, 512	512
Dropout second GRU layer	0.1, 0.3, 0.5, 0.7	0.7
LSTM layers	2, 3 , 4	3
Units third GRU layer	64, 128, 256, 512	128
Dropout third GRU layer	0.1, 0.3, 0.5, 0.7	0.5
Units fourth GRU layer	64, 128, 256, 512	/
Dropout fourth GRU layer	0.1, 0.3, 0.5, 0.7	/
Units dense layer	64, 128, 256, 512	256
Epochs	100, 1000	1000

Appendix H

Table 11: Panels A, B, and C illustrate the performance of the k=10 portfolio, before and after transaction costs for the Random forest (RAF), LSTM network, LSTM network with attention layer, GRU network, and GRU network with attention layer. Panel A depicts daily return characteristics. Panel B depicts daily risk characteristics. Panel C depicts annualized risk-return metrics.

		Before transaction costs					After transaction costs				
		RAF	**LSTM**	**LSTM Attention**	**GRU**	**GRU Attention**	**RAF**	**LSTM**	**LSTM Attention**	**GRU**	**GRU Attention**
A	**Mean return[%]**	0.1452	0.1941	-0.0001	0.0062	-0.0001	-0.0548	-0.0059	-0.2114	-0.1938	-0.1922
	Standard error	0.0006	0.0011	0.0006	0.0004	0.0005	0.0006	0.0011	0.0006	0.0004	0.0005
	Minimum	-0.0332	-0.0675	-0.0363	-0.0223	-0.0275	-0.0352	-0.0695	-0.0383	-0.0243	-0.0294
	Quartile 1	-0.0043	-0.0078	-0.0053	-0.0039	-0.0050	-0.0063	-0.0098	-0.0073	-0.0059	-0.0068
	Median	0.0012	0.0020	0.0001	-0.0001	-0.001	-0.0008	0.0000	-0.0019	-0.0019	-0.0019
	Quartile 3	0.0071	0.0118	0.0053	0.0040	0.0049	0.0051	0.0098	0.0033	0.0020	0.0003
	Maximum	0.0417	0.0744	0.0317	0.0249	0.0212	0.0289	0.0724	0.0297	0.0229	0.0271
	Share > 0	55.6929	54.6505	51.7809	50.4645	50.7326	45.2239	48.2953	38.5807	36.5772	38.1817
	Standard dev.	0.0100	0.0181	0.0092	0.0067	0.0083	0.0100	0.0181	0.0092	0.0067	0.0083
	Skewness	0.1626	0.0546	-0.2538	0.1152	0.0151	0.1626	0.0546	-0.2538	0.1152	0.0151
	Kurtosis	2.3128	2.3010	1.8393	2.0564	1.0850	2.3138	2.3010	1.8393	2.0565	1.0850
B	**1-percent VaR**	-0.0264	-0.0543	-0.0280	-0.0192	-0.0237	-0.0284	-0.0563	-0.0300	-0.0212	-0.0255
	1-percent CVaR	-0.0230	-0.0609	-0.0322	-0.0207	-0.0256	-0.0318	-0.0629	-0.0341	-0.0227	-0.0275
	2-percent VaR	-0.0207	-0.0388	-0.0225	-0.0152	-0.0193	-0.0227	-0.0408	-0.0245	-0.0172	-0.0212
	2-percent CVaR	-0.0253	-0.0499	-0.0273	-0.0180	-0.0226	-0.0273	-0.0519	-0.0293	-0.0200	-0.0245
	5-percent VaR	-0.0150	-0.0277	-0.0157	-0.0112	-0.0144	-0.0170	-0.0297	-0.0177	-0.0132	-0.0163
	5-percent CVaR	-0.0204	-0.0393	-0.0216	-0.0148	-0.0186	-0.0224	-0.0413	-0.0236	-0.0168	-0.0204
	Max. drawdown	0.1080	0.2041	0.0019	0.1044	0.0016	0.3212	0.3986	0.0057	0.4934	0.0052
C	**Return p.a. [%]**	51.0769	90.1120	-2.1465	1.8434	-1.0590	-8.3920	15.3046	-40.5351	-38.2828	-36.6629
	Standard dev. p.a.	0.1586	0.2869	0.1454	0.1057	0.1314	0.1586	0.2869	0.1455	0.1057	0.1314
	Downside dev. p.a.	0.0983	0.1906	0.1003	0.0687	0.0847	0.1013	0.1931	0.1036	0.0729	0.0885
	Sharpe ratio p.a.	3.0259	2.9746	0.0076	0.1864	0.1687	-0.9826	0.1667	-3.4804	-3.9551	-3.3519
	Sortino ratio p.a.	5.2590	5.1136	-0.0557	0.3267	0.2779	-1.3142	0.5009	-4.8451	-5.7764	-4.8496

Appendix I

Table 12: Panels A, B, and C illustrate the performance of the k=15 portfolio, before and after transaction costs for the Random forest (RAF), LSTM network, LSTM network with attention layer, GRU network, and GRU network with attention layer. Panel A depicts daily return characteristics. Panel B depicts daily risk characteristics. Panel C depicts annualized risk-return metrics.

		Before transaction costs					After transaction costs				
		RAF	**LSTM**	**LSTM Attention**	**GRU**	**GRU Attention**	**RAF**	**LSTM**	**LSTM Attention**	**GRU**	**GRU Attention**
A	**Mean return [%]**	0.1211	0.1194	-0.0219	0.0052	-0.0133	-0.1789	-0.1806	-0.3219	-0.2948	-0.3133
	Standard error	0.0005	0.0007	0.0004	0.0005	0.0005	0.0005	0.0007	0.0004	0.0005	0.005
	Minimum	-0.0272	-0.0381	-0.0202	-0.0241	-0.0277	-0.0302	-0.0411	-0.0232	-0.0271	-0.0307
	Quartile 1	-0.0035	-0.0045	-0.0038	-0.0042	-0.0048	-0.0065	-0.0075	-0.0068	-0.0072	-0.0076
	Median	0.0011	0.0011	-0.0003	0.0000	-0.0001	-0.0019	-0.0019	-0.0033	-0.0030	-0.0031
	Quartile 3	0.0057	0.0069	0.0033	0.0041	0.0043	0.0027	0.0039	0.0003	0.0011	0.0013
	Maximum	0.0354	0.0424	0.0231	0.0268	0.0272	0.0334	0.0404	0.0201	0.0238	0.0242
	Share > 0	56.5165	55.4383	47.8422	50.2709	50.2665	37.4001	39.7406	25.0735	29.5647	29.9319
	Standard dev.	0.0082	0.0106	0.0061	0.0072	0.0076	0.0082	0.0106	0.0061	0.0072	0.0076
	Skewness	0.1664	0.0727	0.2054	0.1058	-0.0287	0.1664	0.0727	0.2054	0.1056	-0.0287
	Kurtosis	2.2563	2.2380	1.5400	1.2768	1.9045	2.2563	2.2380	1.5400	1.2768	1.9045
B	**1-percent VaR**	-0.0226	-0.0299	-0.0167	-0.0202	-0.0210	-0.0246	-0.0329	-0.0197	-0.0231	-0.0240
	1-percent CVaR	-0.0249	-0.0340	-0.0185	-0.0221	-0.0243	-0.0279	-0.0370	-0.0215	-0.0251	-0.0273
	2-percent VaR	-0.0165	-0.0222	-0.0129	-0.0156	-0.0165	-0.0195	-0.0252	-0.0159	-0.0186	-0.0195
	2-percent CVaR	-0.0208	-0.0284	-0.0156	-0.0189	-0.0203	-0.0238	-0.0314	-0.0186	-0.0219	-0.0233
	5-percent VaR	-0.0122	-0.0166	-0.0104	-0.0122	-0.0123	-0.0152	-0.0196	-0.0134	-0.0152	-0.0159
	5-percent CVaR	-0.0166	-0.0226	-0.0131	-0.0157	0.0166	-0.0196	-0.0256	-0.0161	-0.0186	-0.0196
	Max. drawdown	0.0863	0.1332	0.0013	0.0014	0.0016	0.5072	0.5401	0.0080	0.0074	0.0078
C	**Return p.a. [%]**	40.1593	43.9136	-4.8328	1.9132	-2.2116	-33.8585	-32.0757	-54.4931	-51.9248	-53.6338
	Standard dev. p.a.	0.1296	0.1674	0.0968	0.1142	0.1199	0.1296	0.1674	0.0968	0.1142	0.1199
	Downside dev. p.a.	0.0806	0.1097	0.0605	0.0723	0.0775	0.0858	0.1140	0.0670	0.0790	0.0840
	Sharpe ratio	2.9854	2.6270	-0.3485	0.1810	-0.0200	-3.2092	-2.4708	-6.5630	-5.2070	-4.9732
	Sortino ratio	5.2551	4.4609	-0.5426	0.3190	-0.0470	-4.5669	-3.4599	-9.1728	-7.3545	-6.9009