ERASMUS UNIVERSITY ROTTERDAM

ERASMUS SCHOOL OF ECONOMICS

BACHELOR THESIS IN QUANTITATIVE FINANCE

Does Attention Improve Memory?

A Comparison of an LSTM network with and without
Multihead Attention for Stock Prediction.

Author:	EMMA LIGTHART (500511)
Supervisor:	S.H.L.C.G VERMEULEN
Second Assesor:	A. TETEREVA
Contact:	500511EL@STUDENT.EUR.NL

July 4, 2021

Abstract

This thesis shows that Multihead Attention adds predictive power to a Long Short Term Memory model, for the purpose of financial time series prediction. Recently, Long Short Term Memory models have shown to achieve superior stock prediction performance, compared to traditional methods, which is why this paper focuses solely on Long Short Term Memory models. I use return index data to calculate one-day standardized returns for companies that are constituents of the S&P 500 in the period 1990-2015. I present two Long Short Term Memory models, one of which implements Multihead Attention, and use them to predict directional movement of stock returns. Several portfolios are constructed from these predictions. The Long Short Term Memory model with Multihead Attention is significantly more accurate and shows favorable return and risk characteristics for most of the portfolios, compared to the regular Long Short Term Memory model.

Contents

1 Introduction

Prediction of time series on the financial stock market is an intricate field of research, due to noise and volatility (Liu 2018). According to the efficient market hypothesis, stock price prediction is impossible because information is instantly digested by stock prices (Malkiel and Fama 1970). However, numerous market anomalies have been found that contradict market efficiency[1], which leads to the conjecture that stock prices can be predicted after all. Financial time series prediction is relevant because techniques that accurately predict stock returns may yield considerable profits to actors on the stock market. Apart from financial rewards, prediction of financial time series provides further insights into the financial market (P. Yu and Yan 2020). Furthermore, insights from financial time series prediction are potentially useful for prediction of other time series. This thesis develops a new technique to predict stock return movement, that implements Multihead Attention in combination with Long Short Term Memory (from hereinafter referred to as LSTM). I show that this technique is superior to using only LSTM in terms of accuracy, return and risk metrics.

Contemporary stock prediction techniques can be divided into two categories: statistical learning techniques, such as ARIMA, and machine learning techniques, such as neural networks. Several studies[2] about stock price prediction show the prevalence of machine learning techniques over statistical learning techniques. Consequently, research has shifted its focus to machine learning over the past decades. A machine learning technique that outperforms statistical techniques in numerous papers[3] regarding stock prediction is Long Short Term Memory (LSTM), which is a non-linear state space model. Supported by the outperformance of LSTM compared to statistical techniques, this paper focuses solely on Long Short Term Memory models. Recently, Abbasimehr and Paki (2021) have combined such an LSTM network with a Multihead Attention mechanism, a non-linear multi-layer model, and compared it to several benchmarks. Their Multihead Attention LSTM model shows potential for time series prediction: it outperforms all benchmarks, including regular LSTM, in terms of error.

Motivated by the favorable results of LSTM for stock prediction and LSTM with Multihead Attention for time series prediction, this thesis compares the stock prediction performance of an LSTM network with and without Multihead Attention. I answer the question: does an LSTM network with Multihead Attention outperform a regular LSTM network, for the purpose of stock movement prediction? Hence, the aim of this work is to determine whether adding

[1]See Jacobs (2015), Coval et al. (2005), Chevalier and Ellison (1999), Daniel and Titman (1999) and Hendricks et al. (1993).

[2]See Bhattacharjee and Bhattacharja (2019), Adebiyi et al. (2014), Wijaya et al. (2010) and Kohzadi et al. (1996).

[3]See Ma (2020), Weytjens et al. (2019), Siami-Namini et al. (2019), Siami-Namini et al. (2018), Karakoyun and Cibikdiken (2018).

an attention mechanism improves the performance of an LSTM network for stock movement prediction.

The experimental set up of this paper is as follows. I use one-day simple returns data, calculated from the return indices of companies in the S&P 500 between 1990-2015. I retrieve the return index data from WRDS and Thomson Reuters. The returns are labeled with either a 1, if the stock outperforms the cross-sectional median of that day, or a 0, if it does not. I build two models to predict the labels of the returns, using 240 day normalized return history of the corrresponding stock. The first model is based on the LSTM network of Fischer and Krauss (2018), in view of their strong results. The second model is novel and combines LSTM and Multihead Attention sequentially. I construct several portfolios from these predictions to analyze accuracy and return characteristics.

The results show that the accuracy of the LSTM model with attention mechanism is generally higher than the accuracy of the regular LSTM model. The LSTM model with attention attains a maximum accuracy of 54.20%, whereas the regular LSTM model achieves a maximum accuracy of 53.84%. The Diebold Mariano test shows that the differences in accuracy are significant for part of the portfolios. Both models achieve significantly higher accuracy for the prediction of *undervalued* stocks, compared to the prediction of *overvalued* stocks. Hence, the models are more adequate at predicting undervalued stocks than at predicting overvalued stocks.

Moreover, the portfolios constructed by the LSTM model with Multihead Attention attain superior return and risk characteristics, compared to regular LSTM. The daily average return is higher for portfolios constructed by the LSTM model with attention. Particularly, a trading strategy that takes long positions in the 5 stocks that are predicted to be undervalued by the LSTM model with Multihead Attention, yields daily average return before transaction cost of 0.11%, compared to a maximum of 0.046% for LSTM. Additionally, the LSTM model with attention achieves higher levels of cumulative return, compared to the LSTM model.

The contributions of this thesis are threefold. Firstly, this paper shows that adding Multihead Attention after LSTM improves stock prediction performance, compared to regular LSTM. Secondly, to the best of my knowledge, this thesis is the first to use LSTM and Multihead Attention sequentially for this purpose. While Abbasimehr and Paki (2021) use LSTM and Multihead Attention in combination for time series prediction, they apply LSTM and Multihead Attention in parallel. Thirdly, to the best of my knowledge, this paper is the first to apply a combination of LSTM and Multihead Attention to a large financial dataset for time series prediction.

The paper is organized as follows. Section 2 describes the background and literature. Section 3 and 4 provide the data description and the methodology. Section 5 presents and discusses the results and Section 6 provides the conclusion.

2 Background and Literature

This section provides background on artificial neural networks and presents literature on LSTM and Multihead Attention, from which I develop my hypothesis.

2.1 Artificial Neural Networks

I elaborate on neural networks briefly and point to Braspenning et al. (1995) for a more in-depth introduction. Artificial neural networks consist of interconnected layers of elementary computational units called neurons (Denning 1992). For this thesis, I train in a supervised manner, which means that my data is labeled. Labeled data consists of features together with an independent data point: a target. The features are fed into the input layer of the network and transferred to hidden layers, where the features are transformed to a hidden state. These computations include a weighted summation, which is inserted into an activation function, the outcome of which determines whether the neuron transfers its hidden state to the next layer. The weights in the weighted summation are trainable parameters: they are initiated randomly and optimized during training. After transforming the features through a number of hidden layers, the data is transferred to the output layer, which constitutes the output of the neural network. This output is compared to the label and the error is calculated by a loss function (Goodfellow et al. 2016). During the next training iteration, the network's trainable parameters are updated to minimize the loss.

2.2 Using Long Short Term Memory networks for Stock Prediction

I now provide motivation for using Long Short Term Memory (LSTM). In previous literature, LSTM networks have been used for a variety of purposes, amongst which are speech recognition (Graves et al. 2013), natural language processing (Wang et al. 2016) and handwriting recognition (Greff et al. 2016). More recently, LSTM models have been applied for the purpose of time series prediction and have displayed their potential for this task (Hua et al. 2019). When applied specifically to financial time series prediction, namely stock prediction, LSTM networks show learning behavior, outperforming traditional models such as GARCH and ARIMA (Jia (2016), Siami-Namini et al. (2018), Ma (2020)). Compared to contemporary techniques, machine learning models such as Random Forests, Deep Neural Networks and Multi-level Perceptrons, the LSTM model also performs significantly better at stock prediction (Fischer and Krauss (2018), Nelson et al. (2017),). Specifically, Fischer and Krauss (2018) present the following admirable results: "The LSTM network outperforms the memory-free methods with statistically and economically significant returns of 0.46 percent per day – compared to 0.43 percent for the RAF, 0.32 percent for the standard DNN, and 0.26 percent for the logistic regression" (Fischer and Krauss (2018), p. 2). Motivated by both their favorable results and the other mentioned literature, this thesis continues the work of Fischer and Krauss (2018), by reconstructing their LSTM model and implementing Multihead Attention.

2.3 Using Multihead Attention for Stock Prediction

I now provide motivation for comparison of the LSTM model with an LSTM model that implements Multihead Attention. Attention mechanisms have proven useful in previous literature in a number of applications, such as sequential learning and image and speech recognition (LeCun et al. (2015), Chorowski et al. (2015)). Various studies show the potential of attention mechanisms implemented in LSTM networks for stock price prediction (Kim and Kang (2019), H. Li et al. (2018), Zhang et al. (2019)). Qiu et al. (2020) compare the performance of an LSTM network with attention mechanism with a regular LSTM network. They find that compared to regular LSTM, the LSTM model with attention mechanism performs best at predicting stock prices. These findings lead to the surmise that adding an attention mechanism to an LSTM network improves prediction power.

Motivated by the success of regular attention with LSTM to predict stock prices, I suspect that Multihead Attention also improves predictive power when used in combination with LSTM for the purpose of stock price prediction. To the best of my knowledge, Multihead Attention in combination with LSTM has been used to predict time series only once and only recently by Abbasimehr and Paki (2021). They apply Multihead Attention LSTM to 16 different datasets and show overall outperformance of LSTM with Multihead Attention, compared to all other benchmarks, including regular LSTM.

Based on the suggested potential of Multihead Attention LSTM found in the literature, I compare Multihead Attention LSTM with regular LSTM to predict directional movements of stocks. I synthesize the findings above in the following hypothesis: an LSTM model with Multihead Attention mechanism achieves better performance, characterized by favorable accuracy, return and risk metrics, compared to a regular LSTM model.

3 Data

The data for this paper consists of one-day simple returns, calculated from return index data of all companies in the S&P 500 over the period January 1990 to September 2015. This data period is the same as in Fischer and Krauss (2018), for the purpose of comparability. The data consists of two parts: constituency data of the S&P 500 index and return index data. I first describe how the data is retrieved, after which I present the preprocessing of the data.

3.1 S&P 500 Constituents and Return Index Data

I retrieve the monthly S&P500 index constituency information from the Center for Research in Security Prices via Wharton Research Data Services, under the DSP500LIST. The DSP500 list contains the enter and leave dates of all companies that have been listed on the S&P500 since the origin date. I use this list to determine whether a company was part of the index some time in the period January 1990 to September 2015.

After determining the 1228 companies that were a constituent of the S&P500 during the observation period, I retrieve the Return Index (RI) data of these companies via Thomson Reuters. The Return Index includes all stock splits, dividends and relevant corporate actions (Fischer and Krauss 2018), rendering it the most suitable price metric for this research. Due to lack of data availability, the number of companies in the final dataset equals 772, with a total number of observations equal to 5,366,644.

3.2 Data Preprocessing

The preprocessing of the data follows Fischer and Krauss (2018). I construct 23 study periods, that include 750 days of observations for training and 250 days of observation for testing. The data set from 1990 to 2015 is divided into rolling blocks of 1000 days, in which the testing periods do not overlap. The stocks included in study period i, are all stocks that were in the S&P 500 at the last day of training period i, as in Fischer and Krauss (2018).

The features fed into the models are normalized one-day simple returns. For all of the observations in a study period i, I calculate the one-day simple return of a stock s, $R_t^{1,s}$, as

$$R_t^{1,s} = \frac{P_t^s}{P_{t-1}^s} - 1, \tag{1}$$

in which P_t is the price index at day t. For each study period i, these one-day returns are stacked into a matrix $\boldsymbol{Z}_i$, in which row t corresponds with day t and column s corresponds with stock s. The summary statistics of matrix $\boldsymbol{Z}_i$ for all study periods are shown in Table 3.1.

Table 3.1: Summary Statistics of the one-day simple return data, calculated from the Return Index data of companies in the S&P 500 over the period December 1989 to September 2015

Statistic	Value
Mean	0.001
Standard Deviation	0.024
Maximum	3.15
Minimum	-0.972

The entries of matrix $\boldsymbol{Z}_i$ are normalized by subtracting the mean of the training one-day returns and dividing by the standard deviation of the training one-day returns. I use the mean and standard deviation of the training sample only, to avoid look-ahead bias. I gather the features for a prediction of stock s in a vector, that contains the 240 day history of normalized one-day returns of stock s. This is the return information of approximately one trading year, similar to Fischer and Krauss (2018).

The targets of the neural networks are constructed as follows. Each return is labelled with either a 1 or a 0, depending on the class to which it belongs. I define class 0 as the class to which all one-period returns belong that perform below the cross-sectional median of that day. I define class 1 as the class to which all one-period returns belong that perform above the cross-sectional median of that day. Stocks with label 1 can be interpreted as undervalued, in the sense that they outperform the cross-sectional median return. Stocks with label 0 can similarly be interpreted as overvalued.

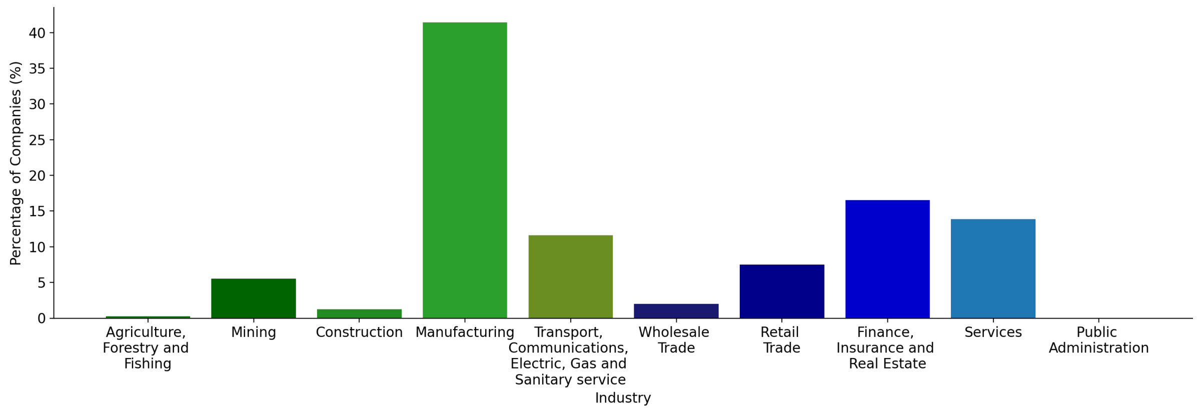


Figure 3.1: Percentage distribution of the companies in the data set over industries of the standard industry classification

The choice and preprocessing of the data introduce two biases. Firstly, by choosing S&P 500 companies, I introduce an S&P 500 bias. All companies in the S&P 500 meet strict requirements, causing them to have similar characteristics, such as their industry classifications. Figure 3.1 shows the percentage distribution of the companies in the dataset over different industries, as classified by the Standard Industry Classification. The distribution is unbalanced: companies in the manufacturing industry comprise 41.4% of the total number of companies, whereas there are little to no companies in the agriculture and public administration industries. Therefore, it is uncertain whether my results are generalizable to outside the S&P 500. Secondly, when preprocessing the data, I include those companies in a study period that are in the S&P 500 at the last day of training. As a consequence, there is a look-ahead bias: on day 1 (to 749) I use those companies for training that are in the S&P 500 on day 750. These biases should be taken into account for the remainder of this thesis.

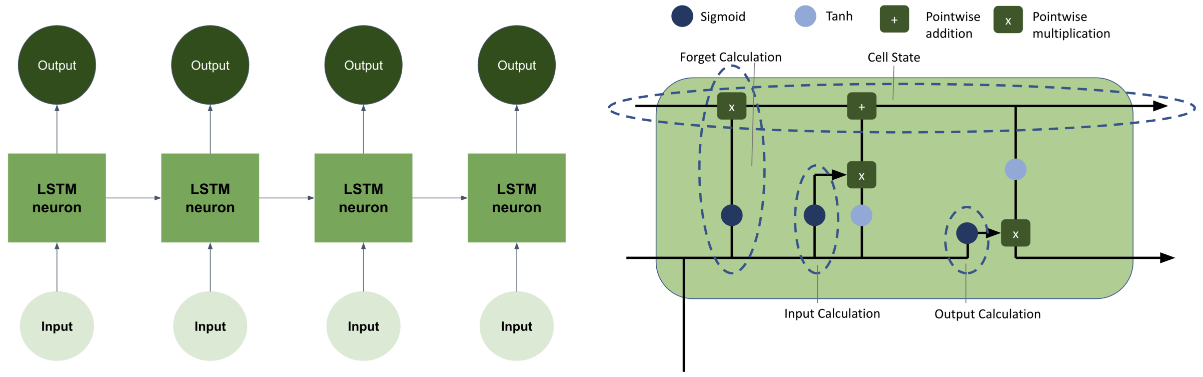
4 Methodology

The methodology consists of five parts. First, I describe the software and hardware used in Section 4.1. Secondly, I present the workings of LSTM and Multihead Attention in Sections 4.2 and 4.3. Thirdly, I elaborate on the model architectures and hyperparameter settings in Section 4.4. Lastly, I describe the training procedure and the portfolio construction in Section 4.5.

4.1 Software and Hardware

For preprocessing the data and building and training the neural network models, I employ Python version 3.8 (Python Software Foundation 2019). Data preprocessing is performed with the packages pandas, numpy and math. For building both models, I rely on the Keras and Tensorflow libraries. The models are trained and tested on an Intel(R) Core i5 CPU. For output analysis, I use both Python and R (R Foundation n.d.).

4.2 Long Short Term Memory



(a) Illustration of an LSTM layer   **(b)** Illustration of an LSTM neuron

Figure 4.1: Illustration of LSTM. Figure (a) illustrates the architecture of an LSTM layer, in which the connections within the layer are shown. Figure (b) shows the inner architecture of an LSTM neuron, where the calculations, activations and other operations are depicted.

The LSTM neural network is a prominent type of neural network, introduced by Hochreiter and Schmidhuber (1997) and further refined by Gers and Schmidhuber (2001). The macro structure of an LSTM layer is presented in Figure 4.1(a), which shows the recurrency inherent in LSTM layers. The interconnections between the neurons within the LSTM layer enable the model to feed data from previous iteration back into the neurons, creating an inner state: the cell state (Jia 2016). Hence, information from previous parts of the time series can be stored and used for the prediction of the current part for a dynamic period of time (Mehtab and Sen 2019). In the context of financial time series predictions, the cell state can be assigned several interpretations depending on the input data, such as the state of the economy. This inner state renders the model especially useful for this thesis. I train the model in order of time such that the cell state of the model contains return information of other stocks at the same time frame. This information, stored in the cell state, gives the model information on the overall state of the stock market, which is probably useful for the prediction.

The cell state is determined by three distinct calculations[4], as shown in Figure 4.1(b). I present the calculations of the cell state $\boldsymbol{c}_t$ and the output of an LSTM neuron, $\boldsymbol{h}_t$, following Qiu et al. (2020), and I use the following notation:

- Vectors $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$ are the input vector of a neuron in a previous layer at the current time step t and the input vector from the LSTM neuron at the previous timestep $t-1$, respectively.
- Matrices $\boldsymbol{U}^f, \boldsymbol{W}^f, \boldsymbol{U}^i, \boldsymbol{W}^i, \boldsymbol{U}^c, \boldsymbol{W}^c, \boldsymbol{U}^o, \boldsymbol{W}^o$ are weight matrices of trainable parameters.
- Function f is the forget calculation, i the input calculation and o the output calculation. The outputs of these functions are the vectors $\boldsymbol{f}, \boldsymbol{i}$ and $\boldsymbol{o}$, respectively.
- Vectors $\hat{\boldsymbol{c}}_t$ and $\boldsymbol{c}_t$ are the candidate cell state and the cell state, respectively.
- Vector $\boldsymbol{h}_t$ is the output vector of the neuron at the current time step.
- The symbol $\odot$ denotes the Hadamard elementwise product.
- The symbol σ is the sigma function and tanh is the tanh function, given by following equations, in which $\boldsymbol{x}$ is a vector:

$$\sigma_i(\boldsymbol{x}) = \frac{1}{1+e^{-x_i}}, \qquad \tanh_i(\boldsymbol{x}) = \frac{e^{x_i} - e^{-x_i}}{e^{x_i} + e^{-x_i}}. \tag{2}$$

An LSTM neuron first inserts its inputs $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$ into the forget calculation, next into the input calculation, and lastly into the output calculation. The forget calculation decides what information is irrelevant for further calculations, by calculating a distribution-like vector over its inputs. The forget calculation is given by the equation:

$$f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) = \sigma(\boldsymbol{U}^f \boldsymbol{x}_t + \boldsymbol{W}^f \boldsymbol{h}_{t-1}). \tag{3}$$

[4]More commonly referred to as gates.

The sigma function, σ, maps its inputs onto the range $[0,1]$. We can thus interpret the elements in the vector output $\boldsymbol{f}$ of the forget calculation f as the likelihoods that datapoints are *not* forgotten, meaning that they are not fed to further LSTM cells (Y. Yu et al. 2019).

After the forget calculation, the input calculation determines what information is maintained in the cell state (Y. Yu et al. 2019). Together with the candidate cell state $\hat{\boldsymbol{c}}_t$, the input calculation i determines the cell state $\boldsymbol{c}_t$, via the following equations:

$$i(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) = \sigma(\boldsymbol{U}^i \boldsymbol{x}_t + \boldsymbol{W}^i \boldsymbol{h}_{t-1}), \tag{4}$$

$$\hat{\boldsymbol{c}}_t(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) = \tanh(\boldsymbol{U}^c \boldsymbol{x}_t + \boldsymbol{W}^c \boldsymbol{h}_{t-1}), \tag{5}$$

$$\boldsymbol{c}_t = \boldsymbol{c}_{t-1} \odot \boldsymbol{f} + \hat{\boldsymbol{c}}_t \odot \boldsymbol{i}. \tag{6}$$

The output of the current cell, $\boldsymbol{h}_t$, is jointly determined by the output calculation o and the cell state $\boldsymbol{c}_t$. The output calculation o determines what information from the current cell state $\boldsymbol{c}_t$ is transferred to the next neuron. The output calculation o and the output vector $\boldsymbol{h}_t$, are calculated via:

$$o(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) = \sigma(\boldsymbol{x}_t \boldsymbol{U}^o + \boldsymbol{h}_{t-1} \boldsymbol{W}^o,) \tag{7}$$

$$\boldsymbol{h}_t = \tanh(\boldsymbol{c}_t) \odot \boldsymbol{o}. \tag{8}$$

The choice for an LSTM network is further motivated by the fact that the calculations above avoid common problems of recurrent neural networks: the vanishing gradient problem and the exploding gradient problem (Gers and Schmidhuber (2001), Greff et al. (2016)). The vanishing gradient problem occurs when the gradient decreases to such an extent that it vanishes. This problem results in either a slow network or in the inability of the network to update its trainable parameters (Hochreiter 1998). It is caused by an exponentially fast decaying factor in the derivative of a hidden state, which LSTMs avoid via the cell state $\boldsymbol{c}_t$, whose derivative does not contain such a decaying factor (Bayer 2015). The exploding gradient problem occurs when the gradient increases exponentially (Philipp et al. 2017) and is caused by an exponentially increasing factor in the derivative of the hidden state, similar to the vanishing gradient problem. LSTMs decrease the likelihood of the exploding gradient problem because the derivative of the cell state $\boldsymbol{c}_t$ does not contain an exponentially increasing factor. An LSTM model is unable to avoid the exploding gradient problem completely because it may occur via a different path.

4.3 Multihead Attention

This section elaborates on Multihead Attention, as introduced by Vaswani et al. (2017). Attention mechanisms use vector multiplications to distinguish information that is useful in the sense that it decreases the loss of the prediction, by calculating attention values over the range $[0,1]$ (W. Li et al. 2020). This calculation creates a density-like distribution over all observations for each prediction, in which higher values correspond to a higher likelihood that the value is important for the prediction.

I present the calculations that determine the attention values. Multihead Attention receives input $\boldsymbol{X}$ from the hidden state of the LSTM layer. Matrix $\boldsymbol{X}$ is a matrix of dimension $d \times l$, where d is the feature vector size, in this case 240, and l is the hidden sequence length of the previous layer. This input matrix $\boldsymbol{X}$ is transformed into matrices $\boldsymbol{V}_1$, $\boldsymbol{V}_2$ and $\boldsymbol{V}_3$[5], via matrix multiplication of $\boldsymbol{X}$ with weight matrices $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3$, as presented in the following equations:

$$\boldsymbol{V}_1 = \boldsymbol{X}\boldsymbol{W}_1, \boldsymbol{X}\boldsymbol{V}_2 = \boldsymbol{X}\boldsymbol{W}_2, \boldsymbol{V}_3 = \boldsymbol{X}\boldsymbol{W}_3. \tag{9}$$

The matrices $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3$ are trainable parameters and their dimensions are equal to $l \times d$. The matrices $\boldsymbol{V}_1$, $\boldsymbol{V}_2$ and $\boldsymbol{V}_3$ form the input of the Scaled Dot Product Attention function, which computes the compatibility of the columns of $\boldsymbol{V}_1$ with the columns of $\boldsymbol{V}_2$ via matrix multiplication[6] of $\boldsymbol{V}_1$ with $\boldsymbol{V}_2^\top$. The entries of the resulting matrix are divided by the square root of the feature vector size, d, and inserted into a softmax function to scale to the range $[0, 1]$. The calculation of the matrix of Attention values and the softmax function are given by:

$$\text{Attention}(\boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3) = \text{softmax}(\frac{\boldsymbol{V}_1\boldsymbol{V}_2^\top}{\sqrt{d}})\boldsymbol{V}_3, \tag{10}$$

$$\text{softmax}_{ij}(\boldsymbol{Z}) = \frac{e^{(\boldsymbol{Z})_{ij}}}{\sum_{i=1}^{K} e^{(\boldsymbol{Z})_{ij}}}, \tag{11}$$

in which softmax_{ij} is the i, j-th element of the output matrix of the softmax function and Z is a matrix with K rows. The intuition behind the Attention computation is that columns that are closer to one another in a multidimensional space and, hence, present more important information to one another, have a larger product than vectors that lie further away from each other. I use Scaled Dot Product Attention because it is more rapid and resource efficient, compared to other forms of attention (Vaswani et al. 2017).

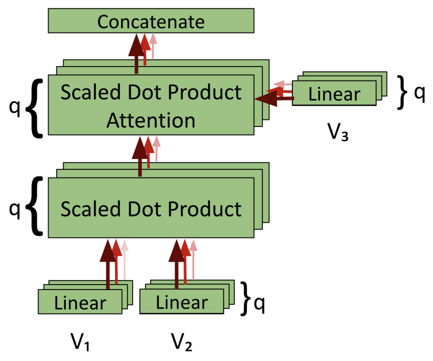


Figure 4.2: Illustration of the Multihead Attention Mechanism. The green rectangles depict different stages of the Attention computation. Matrices $\boldsymbol{V}_1$, $\boldsymbol{V}_2$ and $\boldsymbol{V}_3$ are transformed into Attention values via matrix multiplication and scaling via softmax.

[5]More commonly referred to as query, key and value.

[6]The matrix multiplication of $\boldsymbol{V}_1$ with $\boldsymbol{V}_2^\top$ is equivalent to performing a dot product operation on all columns of $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ separately, hence Dot Product Attention.

The crux of Multihead Attention is that V_1, V_2 and V_3 are used q times to calculate Attention, as illustrated in Figure 4.2. Matrices V_1, V_2 and V_3 are calculated q times by linearly projecting input matrix X with different trainable weight matrices W_1, W_2, W_3, through the use of linear layers (Vaswani et al. 2017). The q different attention outputs are eventually concatenated into a single layer, which forms the final hidden state of the Multihead Attention. Hence, the Multihead Attention model learns from q different data representations V_1, V_2, V_3 of the same input data $\boldsymbol{X}$. Multihead Attention learns more often from the same data compared to regular attention, where V_1, V_2 and V_3 are used only once to calculate Attention.

4.4 Model Architecture and Hyperparameter Settings

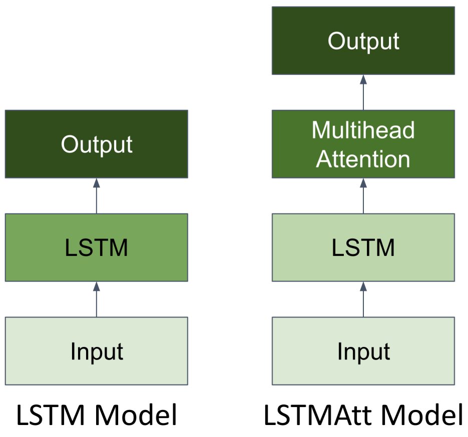


Figure 4.3: Model Architecture. The left part of this figure shows the LSTM model. The right part of this figure, called "LSTMAtt", shows the LSTM model with Multihead Attention.

I build two models, to determine whether adding an attention mechanism to an LSTM model improves model performance. Firstly, I reconstruct the LSTM model from the paper by Fischer and Krauss (2018), which is presented on the left in Figure 4.3. The LSTM model implements the following hyperparameter settings:

- An input layer with 1 feature and 240 timesteps.
- An LSTM layer with 25 hidden neurons and a dropout value of 0.1.
- An output layer with two neurons and a softmax activation function.
- The binary cross-entropy loss function, which is suitable for binary classification problems.
- The RMSprop optimizer, which is one of the most used optimizers (Xu et al. 2021) and is generally an apt method (Fischer and Krauss 2018). RMSprop avoids the vanishing gradient problem and uses an adaptive learning rate, which could potentially help the model escape local minima (Tieleman and Hinton 2012).

For the second model, called "LSTMAtt", I add a Multihead Attention layer to the LSTM model. This model can be viewed in Figure 4.3, on the right. I substantiate my argument that implementing Multihead Attention after LSTM improves model performance as follows. Whereas LSTM layers use all information from previous iterations, Multihead Attention mechanisms distinguish information that is useful for the current prediction. Multihead Attention can attend to previous hidden states directly through positional encoding[7], instead of using a state variable that contains information from all previous hidden states, as in LSTM. LSTMs consider the entire dataset and save all previous information in a state variable. Presumably, not all information from previous iterations and thus not the entire hidden state of the LSTM model is relevant for the prediction of the stock movement at hand. It is likely that return information of the most recent days is more relevant for a prediction than return information of the least recent days. An attention mechanism thus discerns information that is important for the prediction at hand and the mechanism might thereby reduce noise. By implementing Multihead Attention after LSTM, I expect that the Multihead Attention identifies the relevant parts of the hidden state of the LSTM layer.

For reasons of comparability, the hyperparameter settings are the same in the LSTMAtt model as in the LSTM model. The number of heads of the Multihead Attention layer needs to be chosen. In the original paper on Multihead Attention, Vaswani et al. (2017) use 16 heads. In the well-known BERT Multihead Attention model by Devlin et al. (2018), 12 attention heads are used. However, removing some of the heads from these models does not reduce performance for NLP tasks; Michel et al. (2019) suggest to invest parameters efficiently and use less heads than in the BERT and the original Multihead Attention model. Because no specific number is provided, I decide to reduce the number of heads used in these models by trial and error and to use 5 heads.

I implement two methods to avoid overfitting. Firstly, the model uses dropout, which entails that certain neurons are randomly removed during training. I use a dropout value of 0.1, which I find to be optimal via trial and error. Secondly, I implement early stopping, such that the model quits training when the validation loss stops decreasing. For validation, I split the training samples into two parts: 80% of the training sample is used for training and 20% of the training sample is used for validation. I use a patience value of 10, which means that the model quits training when the validation loss does not decrease for 10 epochs. The maximum number of epochs is set to 1000.

[7]See Vaswani et al. (2017).

4.5 Training, Forecasting and Portfolio Construction

The training procedure is organized with the aim to avoid future bias as much as possible. Consider study period i, for which data of S stocks s is available over 1000 days. The first 750 days of this study period are used for training. The first input sequence consists of 240 consecutive standardized one-day returns $R_t^{1,s}$ before date d_{i1} of one specific stock s, where d_{i1} is the 241-st day of the training sample of study period i. I start training from the 241-st day to ensure that the moving window is of equal length during the entire training period. Once the network has worked through this first input sequence, the next sequence is the 240 day return history of the next stock in the sample period, for the same date d_{i1}. This procedure repeats until the model has trained on all S stocks on date d_{i1}. After this iteration, the model continues training with the next date d_{i2}. The model thus receives its input sequences in order of time. This process repeats until the model has trained on the sequences of all 750 days since the 241st day. Testing is performed on the 250 out of sample observations. The model is trained and tested for each of the 23 study periods separately.

In order to construct portfolios and gain insight into the return characteristics of the models, I list the predictions of both models in ascending order. The data used for prediction are the 250 out-of-sample observations of each study period. Each model predicts the probability $P_{t+1|t}^{s}$ that stock s outperforms the cross-sectional median return on day $t+1$ given the 240 previous one-day returns. Hence, the upper k stocks on the list correspond with the stocks that the model predicts to be undervalued on day $t+1$. The lower k stocks correspond with the stocks the model predicts to be overvalued on day $t+1$. I construct 9 different portfolios for each model: the portfolios go either long in the k undervalued stocks, short in the k overvalued stocks or both long and short, where $k \in \{5, 10, 100\}$. I assume round-trip transaction costs of 10 basis points, in line with Avellaneda and Lee (2010). The accuracy of both models is calculated separately for each portfolio to provide insight into the specific capabilities of the models. In addition, I calculate an error vector, from which I derive the Diebold Mariano statistic to assess the significance of the differences in accuracy.

5 Results

This section presents the results in four parts. First, I compare the accuracy of the two models for different portfolios in Section 5.1. Second, I elaborate on the return and risk characteristics of the portfolios in Section 5.2. Finally, I show the development of cumulative return and analyze the training loss of the models in Sections 5.3 and 5.4.

5.1 Accuracy

Table 5.1 shows the accuracy of the models for different portfolios with different values of k. The accuracy of the LSTMAtt model is generally higher than the accuracy of the LSTM model. This indicates that the LSTMAtt model performs better than the LSTM model. The difference between the accuracy of the models is largest for the undervalued predictions for $k = 10$, for which the LSTMAtt model achieves an accuracy that is 0.69% higher than the LSTM model. Furthermore, the accuracy of predicting the *undervalued* companies is universally higher compared to the accuracy of the *overvalued* predictions. Both models appear to perform better at predicting undervalued stocks.

Table 5.1: Accuracy of the forecasts of the LSTM model and LSTMAtt model in percentage, for different portfolios that go long, short or both in k stocks

	k=5		**k=10**		**k=100**	
Predictions	**LSTM**	**LSTMAtt**	**LSTM**	**LSTMAtt**	**LSTM**	**LSTMAtt**
All (in %)	51.16	51.29	50.47	50.77	50.02	50.21
Undervalued (in %)	53.84	54.20	52.66	53.29	52.19	52.41
Overvalued (in %)	48.48	48.38	48.27	48.25	47.85	48.00

The table presents the accuracy for all predictions in portfolio with $k \in \{5, 10, 100\}$ ("All"), predictions of the undervalued stocks ("Undervalued") and predictions of the overvalued stocks ("Overvalued").

I evaluate the significance of the differences in accuracy with a Diebold Mariano test as introduced by Diebold and Mariano (2002). The results are presented in Table 5.2. Overall, the difference between the accuracy of LSTM and the LSTMAtt is significant in part of the cases, for a significance level of 5%. Specifically, the difference is significant for the undervalued predictions of $k = 10$ and both and either the under- and overvalued predictions of $k = 100$. Whilst there are differences in accuracy between LSTM and LSTMAtt for $k = 5$, the difference is not significant. The predictions of the LSTM model are not significantly more accurate than the predictions of the LSTMAtt model for all, the undervalued and the overvalued predictions. Moreover, the predictions of the undervalued stocks are significantly more accurate than the predictions the overvalued stocks, for all values of k and for both models. This confirms the idea that both models are more apt at predicting which stocks are undervalued, compared to predicting which stocks are overvalued. A possible cause for this finding is that the patterns that show that stocks are undervalued are clearer, or at least more perceptible by the models, than the patterns that show that stocks are overvalued.

Table 5.2: Results of the Diebold Mariano test of predictive accuracy for alternative forecasts of returns, for different values of k

k=5					
Alternative Hypothesis			**2 > 1**	**1 $\neq$ 2**	**2 < 1**
Predictions 1	**Predictions 2**	**DM statistic**	**p-value**	**p-value**	**p-value**
LSTM	LSTMAtt	0.48	0.32	0.63	0.68
Underv. LSTM	Underv. LSTMAtt	0.92	0.18	0.36	0.82
Overv. LSTM	Overv. LSTMAtt	-0.24	0.59	0.81	0.41
Overv. LSTM	Underv. LSTM	12.62	**0.00**	**0.00**	1.00
Overv. LSTMAtt	Underv. LSTMAtt	13.69	**0.00**	**0.00**	1.00
k=10					
Alternative Hypothesis			**2 > 1**	**1 $\neq$ 2**	**2 < 1**
Predictions 1	**Predictions 2**	**DM statistic**	**p-value**	**p-value**	**p-value**
LSTM	LSTMAtt	1.53	0.06	0.13	0.94
Underv. LSTM	Underv. LSTMAtt	2.24	**0.01**	**0.03**	0.99
Overv. LSTM	Overv. LSTMAtt	-0.08	0.53	0.94	0.47
Overv. LSTM	Underv. LSTM	14.63	**0.00**	**0.00**	1.00
Overv. LSTMAtt	Underv. LSTMAtt	16.88	**0.00**	**0.00**	1.00
k=100					
Alternative Hypothesis			**2 > 1**	**1 $\neq$ 2**	**2 < 1**
Predictions 1	**Predictions 2**	**DM statistic**	**p-value**	**p-value**	**p-value**
LSTM	LSTMAtt	2.95	**0.00**	**0.00**	1.00
Underv. LSTM	Underv. LSTMAtt	2.52	**0.01**	**0.01**	0.94
Overv. LSTM	Overv. LSTMAtt	1.65	**0.05**	0.10	0.95
Overv. LSTM	Underv. LSTM	46.18	**0.00**	**0.00**	1.00
Overv. LSTMAtt	Underv. LSTMAtt	46.98	**0.00**	**0.00**	1.00

The alternative hypotheses state that predictions i are more accurate (>), less accurate (<), or unequal to ($\neq$) predictions j, where $i, j \in [1, 2]$. The overall predictions (under- and overvalued) are called "LSTM" and "LSTMAtt", whereas the predictions of the undervalued and overvalued stocks are called "Underv." and "Overv." respectively. "DM" statistic stands for the Diebold Mariano statistic. P-values smaller than 0.05 are in bold.

5.2 Return and Risk Characteristics

The return characteristics are presented in Table 5.3. In general, the average daily returns of LSTMAtt portfolios are higher than the returns of the LSTM portfolio. Specifically, the Long & Short average return is negative for LSTM and positive for LSTMAtt for all k. The returns of going long are higher than going short or going long and short, for both models and all k, corresponding with the higher accuracy of the undervalued predictions. Irrespective of k, the returns of the Short portfolios are negative for both models, corresponding with the lower accuracy of overvalued predictions. The highest average return is achieved by the $k = 5$ LSTMAtt Long portfolio, which achieves an average daily return equal to 0.11%. An optimal trading strategy involves going long in the $k = 5$ most undervalued predicted stocks by the LSTMAtt model, holding them one day and selling them the next day. Compared to these results, the LSTM model by Fischer and Krauss (2018) achieves a factor 10 higher average returns of 0.46% for $k = 10$.

Furthermore, I analyze several risk metrics: standard deviation, a measure of risk, Sharpe

Table 5.3: Return and risk characteristics for different portfolios: average daily return before transaction costs, standard deviation (st. dev.), annualized Sharpe ratio and 5-percent daily Value at Risk (VaR)

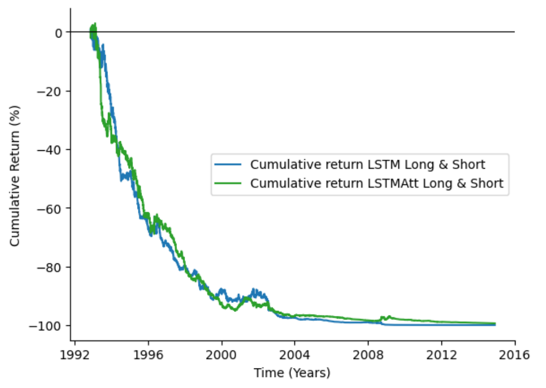
	k=5		**k=10**		**k=100**	
Metric	**LSTM**	**LSTMAtt**	**LSTM**	**LSTMAtt**	**LSTM**	**LSTMAtt**
Return (%)						
Long & Short	-0.045	0.021	-0.039	0.022	-0.014	0.012
Long	0.046	0.11	0.045	0.097	0.045	0.067
Short	-0.137	-0.067	-0.123	-0.053	-0.073	-0.043
St. Dev. (%)						
Long & Short	1.61	1.40	1.16	1.01	0.44	0.38
Long	2.23	2.56	1.94	2.08	1.28	1.35
Short	3.27	2.50	2.55	2.04	1.49	1.33
Sharpe Ratio						
Long & Short	-0.43	0.23	-0.51	0.34	-0.49	0.49
Long	0.32	0.67	0.36	0.72	0.54	0.77
Short	-0.66	-0.42	-0.07	-0.40	-0.76	-0.50
5-percent VaR						
Long & Short	-0.021	-0.019	-0.016	-0.013	-0.006	-0.005
Long	-0.034	-0.031	-0.029	-0.027	-0.020	-0.019
Short	-0.039	-0.033	-0.033	-0.028	-0.020	-0.018

Ratio, a measure of return per risk unit, and VaR, a measure of downside risk, all presented in Table 5.3. Overall, the LSTMAtt portfolios show favorable risk characteristics. The LSTM Long & Short portfolios have higher standard deviation, regardless of k. The LSTMAtt portfolios take more risk while going long, whereas the LSTM portfolios take more risk while going short. The annualized Sharpe Ratio is generally higher for the LSTMAtt portolios compared to the LSTM portfolios. Hence, the risk taken by the LSTMAtt portfolio provides higher rewards, compared to LSTM. The 5-percent daily VaR shows that the downside risk is smaller for the LSTMAtt portfolios, compared to the LSTM portfolios, for all values of k. Hence, not only do the LSTMAtt portfolios outperform LSTM in terms of return characteristics, they also outperform LSTM in terms of risk metrics.

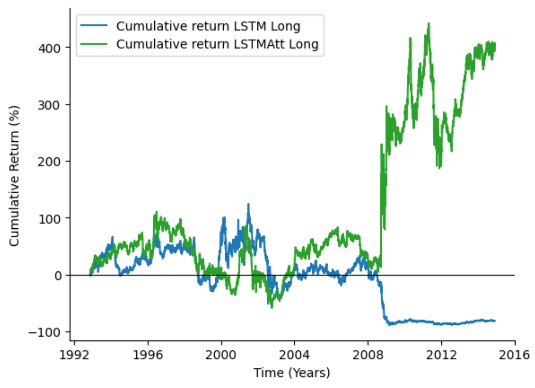
5.3 Development of Cumulative Return

This section focuses on portfolios with k equal to 5 because these portfolios show superior development of cumulative return, compared to portfolios with $k = 10$ and $k = 100$. The portfolios with $k = 10$ and $k = 100$ lose their edge after subtraction of transaction costs, as presented in Appendix A. Figure 5.1 shows the cumulative return after transaction costs over the entire observation period for portfolios with $k = 5$.

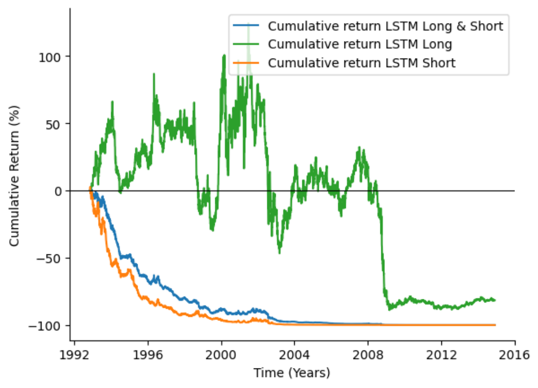
Figure 5.1(a) and (b) show the cumulative returns of the Long & Short portfolios and the Long portfolios as constructed by the LSTM and LSTMAtt models for $k = 5$, respectively. In general, the cumulative return of the *Long* portfolio progresses to much higher levels for the



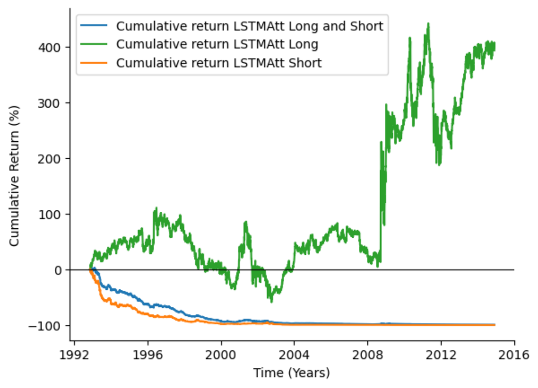
(a) Cum. return LSTM and LSTMAtt Long & Short



(b) Cum. return LSTM and LSTMAtt Long



(c) Cum. return LSTM



(d) Cum. return LSTMAtt

Figure 5.1: Development of cumulative return after transaction costs. In this figure, Cum. denotes cumulative.

LSTMAtt model than for the LSTM model. The development of cumulative return for the *Long & Short* portfolios is quite similar for both models. Specifically, Figure 5.1(a) shows that both Long & Short portfolios move to a negative 100% cumulative return over the observation period, with only small fluctuations. Fluctuations are much clearer for the Long portfolios in Figure 5.1(b).

I analyze the Long portfolios in 5.1(b) more extensively. From 1993 to around 1998, the cumulative return of the long portfolios varies between roughly $0 - 100\%$ and the LSTMAtt portfolio attains slightly higher levels. In 1998, before the dotcom bubble crisis, the cumulative returns of both portfolios decreases to negative values, until just before 2000, the start of the dotcom bubble crisis, when the cumulative return of the LSTM portfolio experiences an increase. The LSTM portfolio appears to profit from the dotcom bubble crisis, whereas the LSTMAtt portfolia suffers from it. The LSTM portfolio reaches its peak cumulative return in 2002, after which it decreases to a level around 0%. In 2008, the start of the Financial Crisis, the LSTMAtt portfolio experiences a sharp increase in cumulative return, after which it increases to levels above 400% in 2011. The LSTMAtt portfolio experiences a decrease in cumulative return in 2011, from which it recovers in the following years. Simultaneously in 2008, the LSTM portfolio experiences a sharp decrease in cumulative return. However the LSTM portfolio does not recover from this contraction during the observation period.

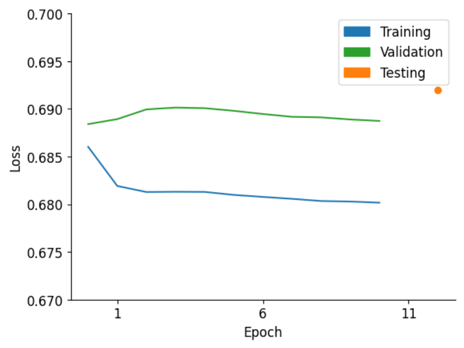
Hence, the largest difference in cumulative return between the models arises during the Financial Crisis, which the LSTMAtt model predicts more accurately than the LSTM model. The large peak of the LSTMAtt portfolio might be caused by the inner workings of the LSTMAtt model. The LSTMAtt model pertains to previous data directly and is able to skip outliers and noise consequently. The LSTM model pertains to previous data via a cell state, which summarizes all data and cannot skip outliers or noise. The LSTM model is therefore more sensitive to outliers and noise. This produces higher rewards during times of high volatility for the LSTMAtt model, such as the Financial Crisis, compared to the LSTM model. Economic literature suggests two related causes for these (extremely) high returns. Firstly, during volatile periods, an information overload might cause numerous relative arbitrage opportunities (Jacobs and Weber 2015). Simultaneously, limits to arbitrage are relatively higher in volatile periods, preventing investors from grasping these arbitrage opportunities (Baker et al. 2011). Because these limits to arbitrage are not modelled, it is questionable whether these high returns in 2008 could realistically be achieved by an investor. Nevertheless, the LSTMAtt model is able to construct a portfolio that achieves persistent positive cumulative return through the observation period, whereas the LSTM model is not.

I now analyze the portfolios for both models seperately, as presented in Figure 5.1(c) and (d). Figure 5.1(c) shows that the Short and Long & Short portfolios converge to a negative 100% over the observation period, whereas the Long portfolio does achieve positive cumulative return for some part of the observation period. Figure 5.1 shows the same for the Short and Long & Short LSTMAtt portfolios, which also converge to -100%. The Long LSTMAtt portfolio is the only portfolio that yields positive returns towards the end of the observation period, proving to be the optimal trading strategy once more. One critical note to the analysis of cumulative return is that LSTM could only be feasibly deployed towards the end of the years 2000. Multihead Attention, was only introduced in 2017 and could thus not have feasibly been deployed during my observation period.

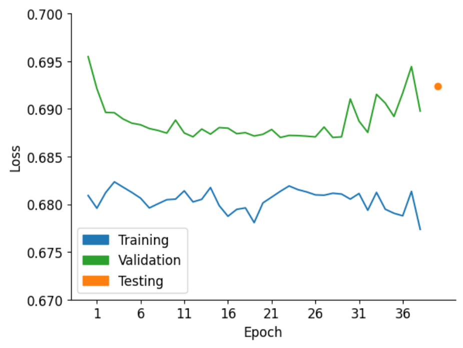
5.4 Loss analysis

This section discusses the learning behavior of both models, by analysing the loss during training, validation and testing. The loss graphs of both models for all 23 training period can be viewed in Appendix B. In general, the LSTMAtt model trains for more epochs, on average

20.17 epochs, than the LSTM model, which trains on average for 15.87 epochs. An explanation is that the LSTMAtt model contains more trainable parameters.



(a) Training, validation and testing loss study period 1993-1996 LSTM



(b) Training, validation and testing loss study period 1993-1996 LSTMAtt

Figure 5.2: Loss graphs study period 1993-1996

Figure 5.2 presents the loss graphs of study period 1993-1996. This study period exemplifies that the LSTM model trains over less epochs than the LSTMAtt model. In this period, and in general, the LSTM model's training loss decreases monotonically, whereas the LSTMAtt model's loss oscillates. Oscillation of the training loss is atypical and can be explained in several ways. Firstly, it could be that there are many saddle points and local minima because of the high dimensionality of the objective function. Secondly, the learning rate could be too large, which disables the model from reaching minima in an efficient way: the model skips the minimum, after which it tries to return to the minima but returns too far, which leads to an oscillation. In this case, the initial learning rate of RMSprop should be chosen smaller. A third explanation is that the training sample is too small for the number of parameters in the LSTMAtt model.

In general, the validation loss of the models decreases very little. For some study periods, the validation loss shows a point of inflection, after which the validation loss increases. This is exemplified in Figure 5.2(b), where from epoch 30, the validation loss appears to increase. In addition, the testing loss (the orange dot in Figure 5.2) of all models and all study periods, is higher than the first training loss. Both these patterns point to overfitting. Although several measures to avoid overfitting have been taken (such as dropout and early stopping), one could increase the dropout value, decrease the patience or implement regularization. Another explanation for these patterns is that the number of observations is too small for the number of trainable parameters.

6 Conclusion

This thesis investigates whether an LSTM network with Multihead Attention outperforms a regular LSTM model, when predicting stock price directional movement. I hypothesize that an LSTM model with Multihead Attention performs better than a regular LSTM model, regarding accuracy, return and risk metrics. I set up two models: a regular LSTM model and an LSTM model with Multihead Attention. Both models predict whether a stock outperforms the cross-sectional median return of that day. I compare the accuracy, return and risk characteristics and analyze the development of the training and testing loss of the models.

The results confirm the hypothesis that an LSTM model with Multihead Attention outperforms a regular LSTM model. The accuracy, return and risk characteristics of the LSTM model with Multihead Attention mechanism are generally better, compared to the LSTM model. The accuracy of the LSTM model with Multihead Attention is generally higher than the accuracy of the regular LSTM model. In addition, the daily and cumulative return and risk of portfolios constructed by an LSTM network with Multihead Attention are preferable to those of portfolios constructed by an LSTM network. An optimal trading strategy involves going long in the top 5 undervalued predicted stocks by the LSTM model with Multihead Attention. Based on the findings above I conclude that the LSTM network with Multihead Attention mechanism outperforms the LSTM network without Multihead Attention.

The results of this thesis are subject to the following limitations. Firstly, the choice and preprocessing of data introduce an S&P 500 bias and a look ahead bias. Therefore, it is uncertain whether these results are generalizable to the entire stock market. Furthermore, the observation period of this thesis runs from 1990 to 2015. Hence, it is uncertain whether the favorable accuracy, return and risk characteristics of the LSTM model with Multihead Attention remain up to today.

The implications of this research are twofold. Firstly, the results suggest that adding a Multihead Attention layer sequentially after an LSTM layer improves performance for time series prediction. Further research is necessary to determine whether the results of this thesis are generalizable to other time series. Secondly, using an LSTM model with Multihead Attention mechanism to construct portfolios yields positive returns, especially when following the trading strategy of going Long in the 5 stocks that are predicted to be undervalued by the LSTM model with Multihead Attention. Whether these positive returns remain to this day, is another area of further research. A final avenue for further research is to investigate whether the learning performance, in terms of training, validation and testing loss, can be improved further, in order to ameliorate both models.

7 Acknowledgements

I thank Sebastiaan Vermeulen for his guidance, helpful feedback and his overall involvement in the entire process of the development of this thesis. I am grateful for the comments and suggestions of anonymous referees and I thank the Erasmus Data Centre for providing the data resources necessary for this thesis.

References

Abbasimehr, H., & Paki, R. (2021). Improving time series forecasting using LSTM and attention models. *Journal of Ambient Intelligence and Humanized Computing*, *12*(1), 1–19. DOI: 10.1007/s12652-020-02761-x.

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics*, *2014*, 1–7. DOI: 10.1155/2014/614342.

Avellaneda, M., & Lee, J.-H. (2010). Statistical arbitrage in the US equities market. *Quantitative Finance*, *10*(7), 761–782. DOI: 10.1080/14697680903124632.

Baker, M., Bradley, B., & Wurgler, J. (2011). Benchmarks as limits to arbitrage: Understanding the low-volatility anomaly. *Financial Analysts Journal*, *67*(1), 40–54. DOI: 10.2469/faj.v67.n1.4.

Bayer, J. S. (2015). *Learning sequence representations* (Doctoral dissertation). Technische Universität München.

Bhattacharjee, I., & Bhattacharja, P. (2019). Stock price prediction: A comparative study between traditional statistical approach and machine learning approach. *2019 4th International Conference on Electrical Information and Communication Technology (EICT)*, 1–6. DOI: 10.1109/EICT48899.2019.9068850.

Braspenning, P. J., Thuijsman, F., & Weijters, A. J. M. M. (1995). *Artificial neural networks: An introduction to ANN theory and practice* (Vol. 931). Germany, Berlin: Springer Science & Business Media.

Chevalier, J., & Ellison, G. (1999). Are some mutual fund managers better than others? Cross-sectional patterns in behavior and performance. *The journal of finance*, *54*(3), 875–899. DOI: 10.1111/0022-1082.00130.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-based models for speech recognition. arXiv:1506.07503.

Coval, J. D., Hirshleifer, D. A., & Shumway, T. (2005). Can individual investors beat the market? HBS finance working paper No. 04-025. DOI: 10.2139/ssrn.364000.

Daniel, K., & Titman, S. (1999). Market efficiency in an irrational world. *Financial Analysts Journal*, *55*(6), 28–40. DOI: 10.2469/faj.v55.n6.2312.

Denning, P. J. (1992). The science of computing: Neural networks. *American Scientist*, *80*(5), 426–429.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & economic statistics*, *20*(1), 134–144. DOI: 10.1198/073500102753410444.

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. DOI: 10.1016/j.ejor.2017.11.054.

Gers, F. A., & Schmidhuber, J. (2001). Long short-term memory learns context free and context sensitive languages. *Artificial Neural Nets and Genetic Algorithms*, 134–137. US, New York: Springer. DOI: 10.1007/978-3-7091-6230-9$_3$2..

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge, Massachusetts: MIT Press.

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. *2013 IEEE international conference on acoustics, speech and signal processing*, 6645–6649. DOI: 10.1109/ICASSP.2013.6638947.

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*(10), 2222–2232. DOI: 10.1109/TNNLS.2016.2582924.

Hendricks, D., Patel, J., & Zeckhauser, R. (1993). Hot hands in mutual funds: Short-run persistence of relative performance. *The Journal of finance*, *48*(1), 93–130. DOI: 10.1111/j.1540-6261.1993.tb04703.x.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107–116. DOI: 10.1142/S0218488598000094.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Hua, Y., Zhao, Z., Li, R., Chen, X., Liu, Z., & Zhang, H. (2019). Deep learning with long short-term memory for time series prediction. *IEEE Communications Magazine*, *57*(6), 114–119. DOI: 10.1109/MCOM.2019.1800155.

Jacobs, H. (2015). What explains the dynamics of 100 anomalies? *Journal of Banking & Finance*, *57*, 65–85. DOI: 10.1016/j.jbankfin.2015.03.006.

Jacobs, H., & Weber, M. (2015). On the determinants of pairs trading profitability. *Journal of Financial Markets*, *23*, 75–97. DOI: 10.1016/j.finmar.2014.12.001.

Jia, H. (2016). Investigation into the effectiveness of long short term memory networks for stock price prediction. arXiv:1603.07893.

Karakoyun, E., & Cibikdiken, A. (2018). Comparison of ARIMA time series model and LSTM deep learning algorithm for bitcoin price forecasting. *The 13th Multidisciplinary Academic Conference in Prague, 2018*, 171–180.

Kim, S., & Kang, M. (2019). Financial series prediction using attention LSTM. arXiv:1902.10877.

Kohzadi, N., Boyd, M. S., Kermanshahi, B., & Kaastra, I. (1996). A comparison of artificial neural network and time series models for forecasting commodity prices. *Neurocomputing*, *10*(2), 169–181. DOI: 10.1016/0925-2312(95)00020-8.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. DOI: 10.1038/nature14539.

Li, H., Shen, Y., & Zhu, Y. (2018). Stock price prediction using attention-based multi-input LSTM. *Asian Conference on Machine Learning*, 454–469.

Li, W., Qi, F., Tang, M., & Yu, Z. (2020). Bidirectional LSTM with self-attention mechanism and multichannel features for sentiment classification. *Neurocomputing*, *387*, 63–77. DOI: 10.1016/j.neucom.2020.01.006.

Liu, H. (2018). Leveraging financial news for stock trend prediction with attention-based recurrent neural network. arXiv:1811.06173.

Ma, Q. (2020). Comparison of ARIMA, ANN and LSTM for stock price prediction. *E3S Web of Conferences*, *218*. DOI: 10.1051/e3sconf/202021801026.

Malkiel, B. G., & Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, *25*(2), 383–417. DOI: 10.2307/2325487.

Mehtab, S., & Sen, J. (2019). A robust predictive model for stock price prediction using deep learning and natural language processing. DOI: 10.2139/ssrn.3502624.

Michel, P., Levy, O., & Neubig, G. (2019). Are sixteen heads really better than one? arXiv:1905.10650.

Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017). Stock market's price movement prediction with lstm neural networks. *2017 International joint conference on neural networks (IJCNN)*, 1419–1426. DOI: 10.1109/IJCNN.2017.7966019.

Philipp, G., Song, D., & Carbonell, J. G. (2017). The exploding gradient problem demystified. arXiv:1712.05577.

Python Software Foundation. (2019). Python 3.8.10 documentation. Available at: https://docs.python.org/3.8/.

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), 222–227. DOI: 10.1371/journal.pone.0227222.

R Foundation. (n.d.). The R project for statistical computing. Available at: https://www.r-project.org/.

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2018). A comparison of ARIMA and LSTM in forecasting time series. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1394–1401. DOI: 10.1109/ICMLA.2018.00227.

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019). A comparative analysis of forecasting financial time series using ARIMA, LSTM, and biLSTM. arXiv:1911.09512.

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, *4*(2), 26–31.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. arXiv:1706.03762.

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*, 606–615. DOI: 10.18653/v1/D16-1058.

Weytjens, H., Lohmann, E., & Kleinsteuber, M. (2019). Cash flow prediction: Mlp and LSTM compared to ARIMA and prophet. *Electronic Commerce Research*, 1–21. DOI: 10.1007/s10660-019-09362-7.

Wijaya, Y. B., Kom, S., & Napitupulu, T. A. (2010). Stock price prediction: Comparison of ARIMA and artificial neural network methods-an Indonesia stock's case. *2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies*, 176–179. DOI: 10.1109/ACT.2010.45.

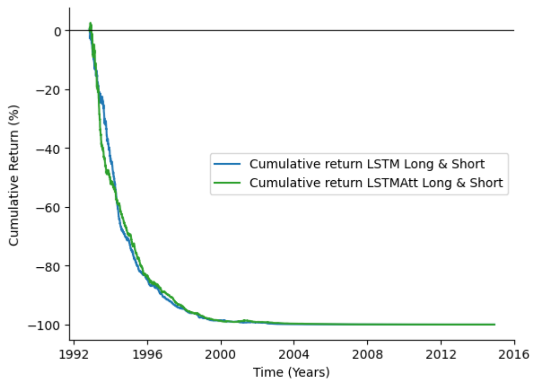
Xu, D., Zhang, S., Zhang, H., & Mandic, D. P. (2021). Convergence of the RMSProp deep learning method with penalty for nonconvex optimization. *Neural Networks*, *139*, 17–23. DOI: 10.1016/j.neunet.2021.02.011.

Yu, P., & Yan, X. (2020). Stock price prediction based on deep neural networks. *Neural Computing and Applications*, *32*(6), 1609–1628. DOI: 10.1007/s00521-019-04212-x.

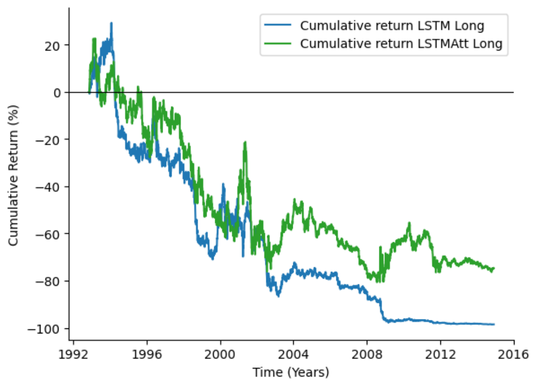
Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation*, *31*(7), 1235–1270. DOI: 10.1162/neco_a_01199.

Zhang, X., Liang, X., Zhiyuli, A., Zhang, S., Xu, R., & Wu, B. (2019). AT-LSTM: An attention-based LSTM model for financial time series prediction. *IOP Conference Series: Materials Science and Engineering*, *569*(5), 052037.

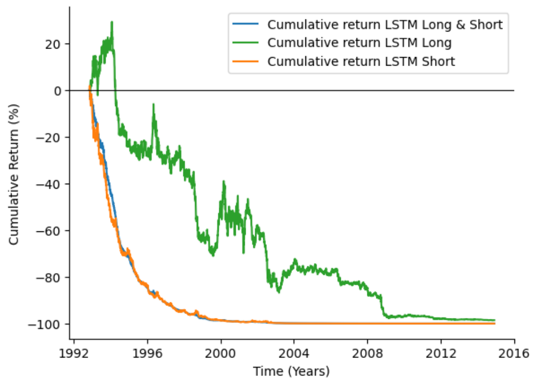
Appendix A: Cumulative Returns



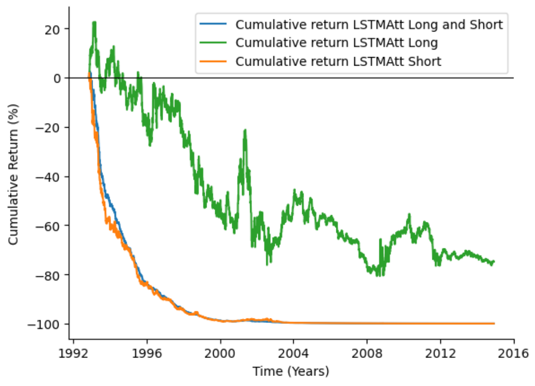
(a) Cum. return LSTM and LSTMAtt Long & Short



(b) Cum. return LSTM and LSTMAtt Long

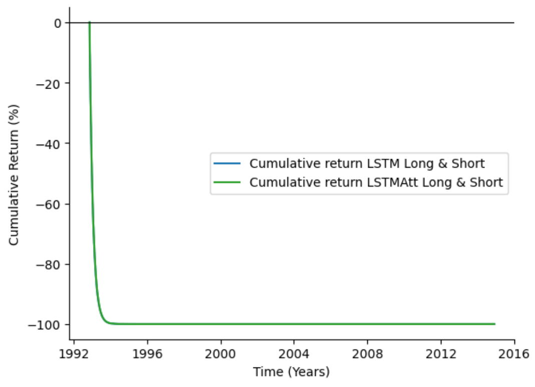


(c) Cum. return LSTM

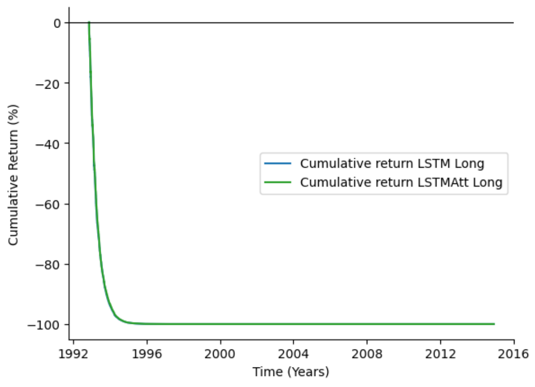


(d) Cum. return LSTMAtt

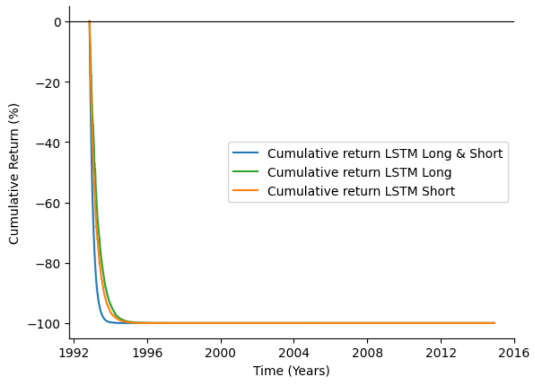
Figure 7.1: Development of cumulative return for portfolios with $k = 10$, after transaction costs. In this figure, cum. stands for cumulative.



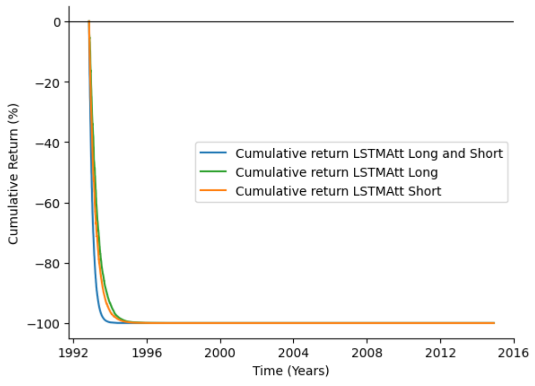
(a) Cum. return LSTM and LSTMAtt Long & Short



(b) Cum. return LSTM and LSTMAtt Long



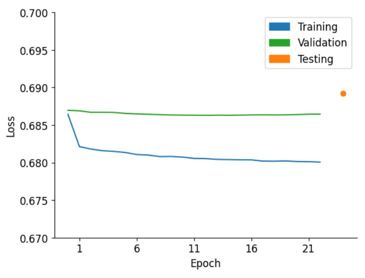
(c) Cum. return LSTM



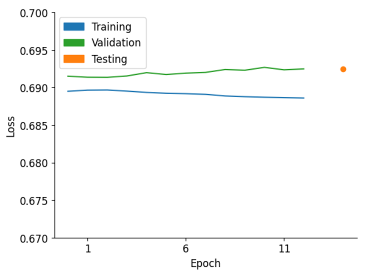
(d) Cum. return LSTMAtt

Figure 7.2: Development of cumulative return for portfolios with $k = 100$, after transaction costs. In this figure, cum. stands for cumulative.

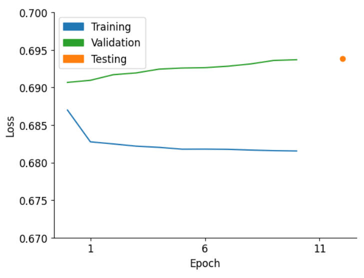
Appendix B: Loss Graphs



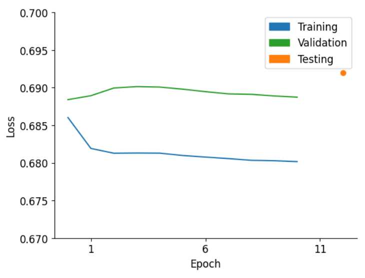
(a) Loss Graph LSTM study period 1990-1993



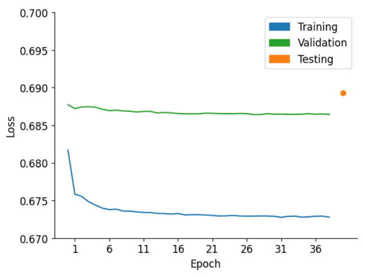
(b) Loss Graph LSTM study period 1991-1994



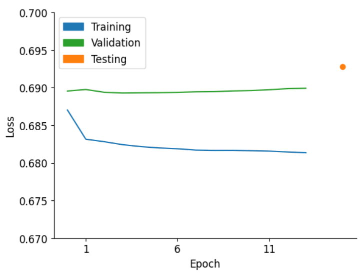
(c) Loss Graph LSTM study period 1992-1995



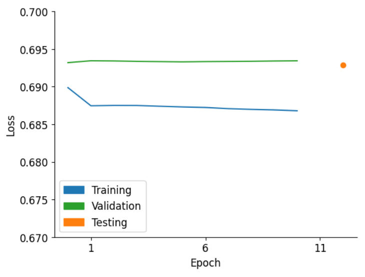
(d) Loss Graph LSTM study period 1993-1996



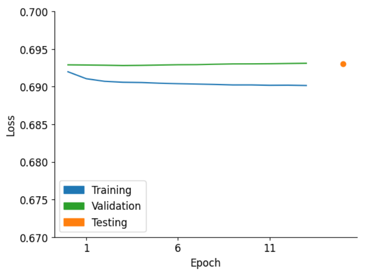
(e) Loss Graph LSTM study period 1994-1997



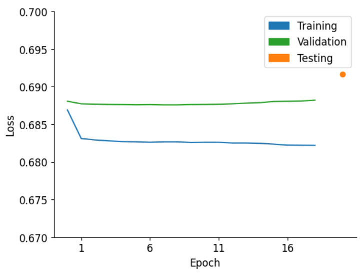
(f) Loss Graph LSTM study period 1995-1998



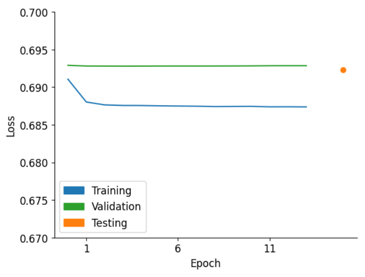
(g) Loss Graph LSTM study period 1996-1999



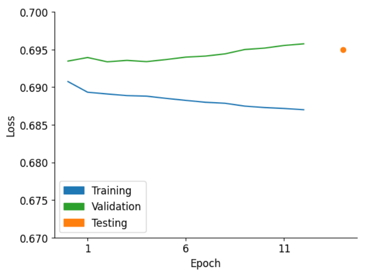
(h) Loss Graph LSTM study period 1997-2000



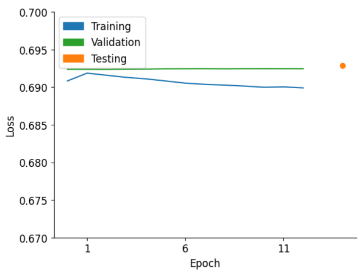
(i) Loss Graph LSTM study period 1998-2001



(j) Loss Graph LSTM study period 1999-2002

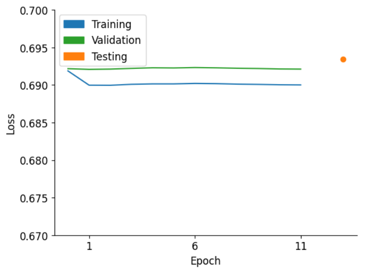


(k) Loss Graph LSTM study period 2000-2003

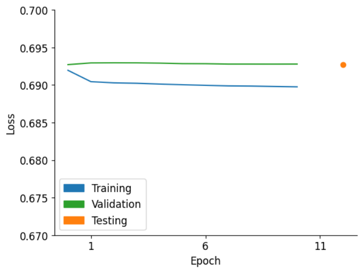


(l) Loss Graph LSTM study period 2001-2004

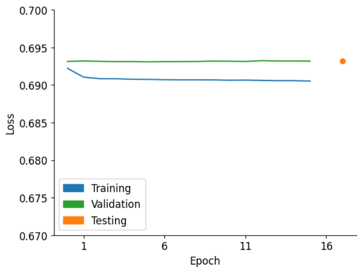
Figure 7.3: Loss Graphs LSTM study periods 1990-2004



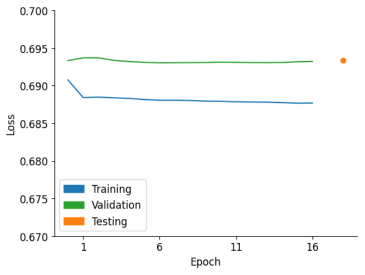
(a) Loss Graph LSTM study period 2002-2005



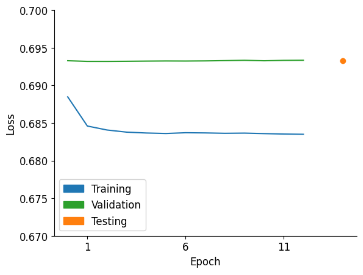
(b) Loss Graph LSTM study period 2003-2006



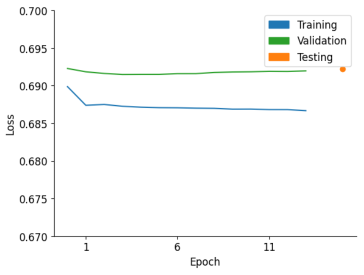
(c) Loss Graph LSTM study period 2004-2007



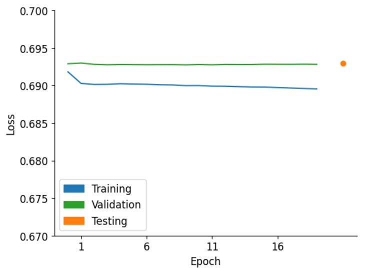
(d) Loss Graph LSTM study period 2005-2008



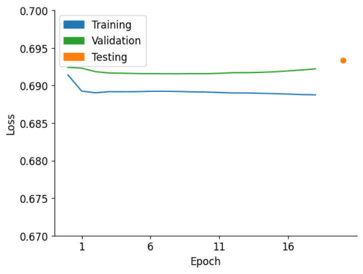
(e) Loss Graph LSTM study period 2006-2009



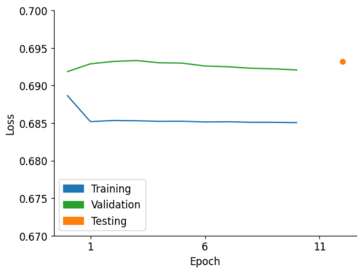
(f) Loss Graph LSTM study period 2007-2010



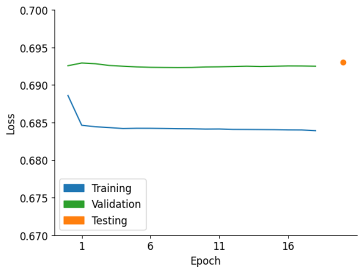
(g) Loss Graph LSTM study period 2008-2011



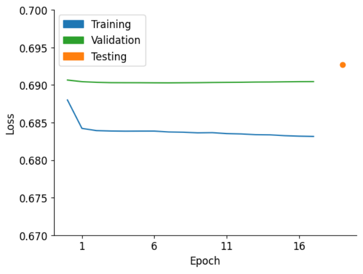
(h) Loss Graph LSTM study period 2009-2012



(i) Loss Graph LSTM study period 2010-2013

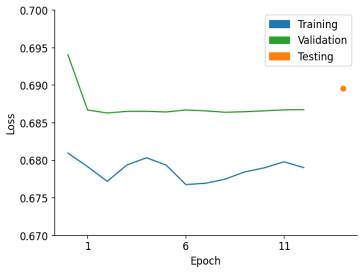


(j) Loss Graph LSTM study period 2011-2014

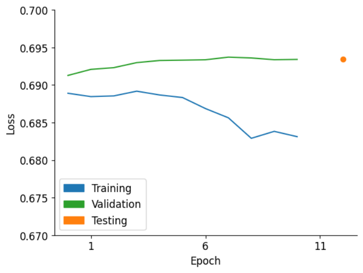


(k) Loss Graph LSTM study period 2012-2015

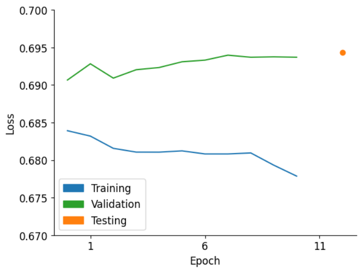
Figure 7.4: Loss Graphs LSTM study period 2002-2015



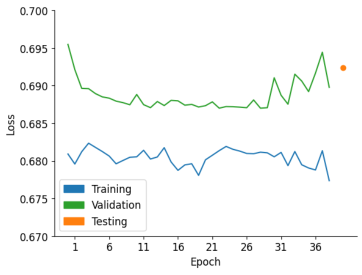
(a) Loss graph LSTMAtt study period 1990-1993



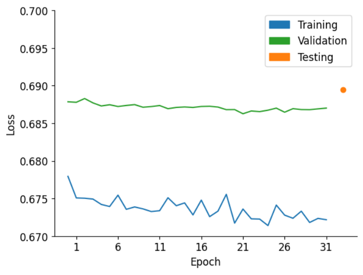
(b) Loss graph LSTMAtt study period 1991-1994



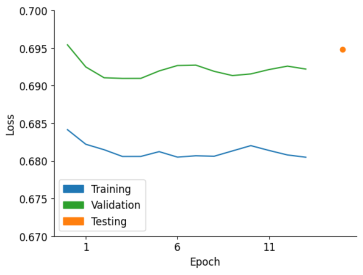
(c) Loss graph LSTMAtt study period 1992-1995



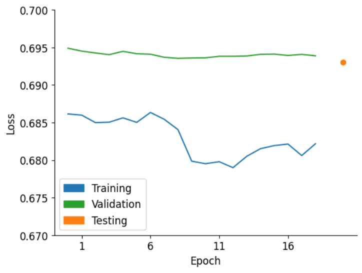
(d) Loss graph LSTMAtt study period 1993-1996



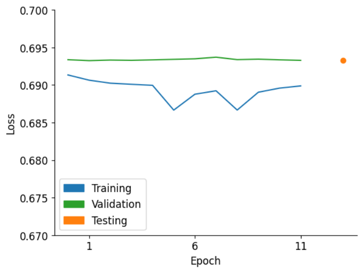
(e) Loss graph LSTMAtt study period 1994-1997



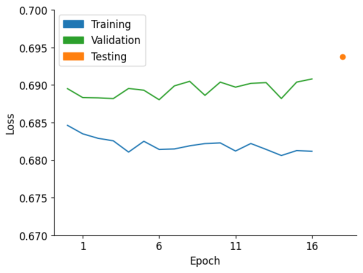
(f) Loss graph LSTMAtt study period 1995-1998



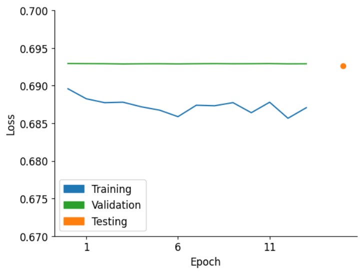
(g) Loss graph LSTMAtt study period 1996-1999



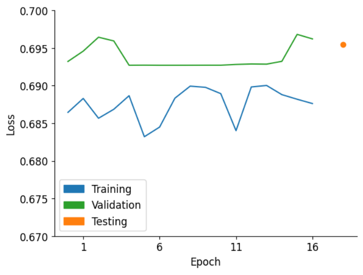
(h) Loss graph LSTMAtt study period 1997-2000



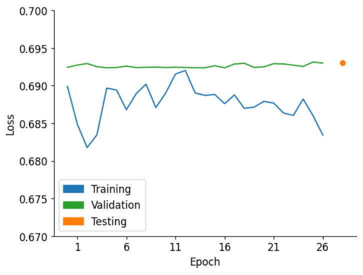
(i) Loss graph LSTMAtt study period 1998-2001



(j) Loss graph LSTMAtt study period 1999-2002

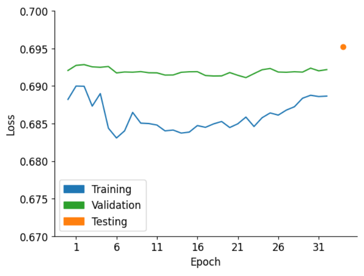


(k) Loss graph LSTMAtt study period 2000-2003

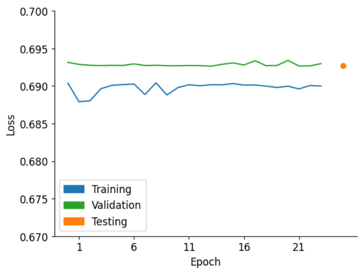


(l) Loss graph LSTMAtt study period 2001-2004

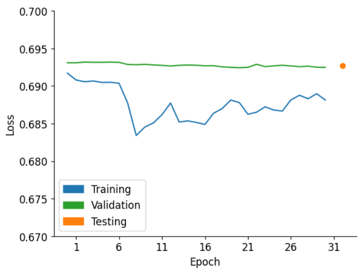
Figure 7.5: Loss graphs LSTMAtt study period 1990 - 2004



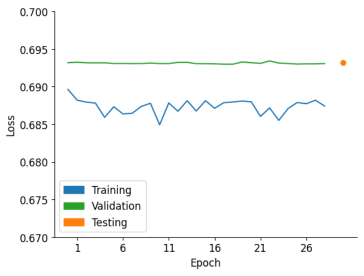
(a) Loss graph LSTMAtt study period 2002-2005



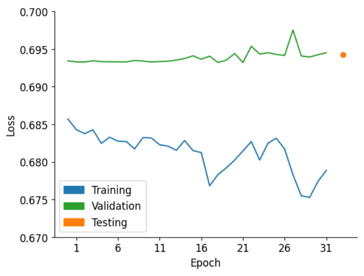
(b) Loss graph LSTMAtt study period 2003-2006



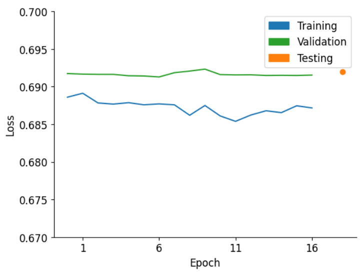
(c) Loss graph LSTMAtt study period 2004-2007



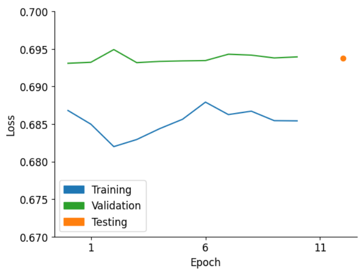
(d) Loss graph LSTMAtt study period 2005-2008



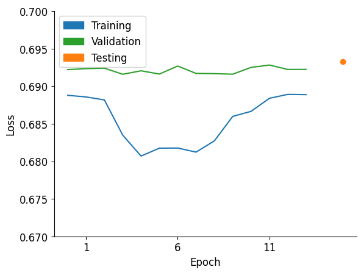
(e) Loss graph LSTMAtt study period 2006-2009



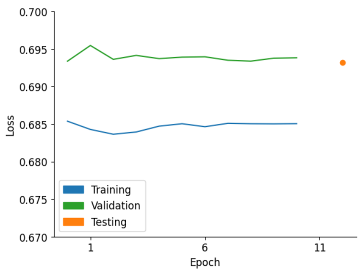
(f) Loss graph LSTMAtt study period 2007-2010



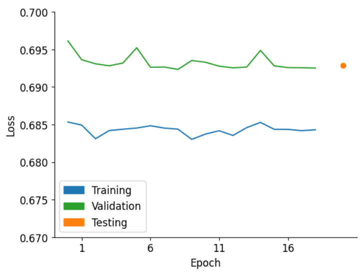
(g) Loss graph LSTMAtt study period 2008-2011



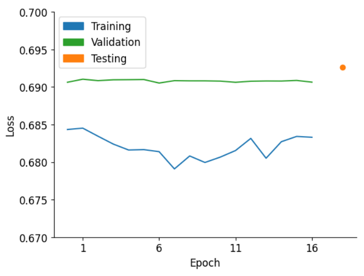
(h) Loss graph LSTMAtt study period 2009-2012



(i) Loss graph LSTMAtt study period 2010-2013



(j) Loss graph LSTMAtt study period 2011-2014



(k) Loss graph LSTMAtt study period 2012-2015

Figure 7.6: Loss graphs LSTMAtt study period 2002 - 2015