Norwegian School of Economics
Bergen, Fall 2018

Employing Deep Learning for Stock Return Prediction on the Oslo Stock Exchange

A comprehensive test of deep learning models for predictive purposes and applicability to trading

Henrik Lund and Jonas Løvås

Supervisor: Thore Johnsen

Master thesis

MSc in Economics and Business Administration, Finance

Norwegian School of Economics

Abstract

We predict daily out-of sample directional movements of the constituent stocks of the Oslo Stock Exchange Benchmark Index (OSEBX) using Long Short-Term Memory (LSTM) networks, benchmarked against other machine learning and econometric techniques. Our results unambiguously show that the LSTM model outperforms all benchmark models in terms of predictive performance. When testing simple long trading strategies utilizing the predictions, we find that the LSTM model outperform all other methods with a Sharpe ratio of 3.25 prior to transaction costs from 1999 - 2017. In comparison, the OSEBX had a Sharpe ratio of 0.30 over the same period. We find that the LSTM model seems to follow a short-term mean-reversion strategy. While seeing somewhat diminishing excess returns in the last years, the excess returns are still present in the three latest years, which differs from similar studies on other indexes where the excess returns have been found to be absent in recent years. When total transaction costs are implemented we see that the excess returns are lost in the bid-ask spread. Training the model on spread-adjusted returns and imposing advanced strategies leads to a modest Sharpe ratio of 0.37 over the whole period. Even though the trading performance after total transaction costs is not statistically significant better than the OSEBX, we see that LSTM networks have predictive properties that can make it a great tool and complement to other trading strategies.

Acknowledgements

We would like to thank our supervisor Thore Johnsen for his excellent guidance and valuable input. We would also like to thank Ole Jakob Wold from DNB Asset Management for his valuable insights into a practitioners approach and technical pointers on our methodology which lifted the quality of our thesis, and John Erik Sloper from Quantfolio for discussing different approaches and challenges within financial machine learning.

Contents

1. Introduction

The field of Artificial Intelligence (AI) has gone through a rapid development over the last decades, hand in hand with the growth in available computing power. The recent development in AI has been driven by Deep Learning techniques, already excelling in areas such as autonomous vehicles, speech recognition and computer vision (Marr, 2018). Deep Learning techniques are able to interpret complex data on a deeper level than humans. Google DeepMind's AlphaGo beat the world champion in the game of Go in 2016, a game that is extremely complex and largely based on human intuition (Hassabis, 2016). The most promising use cases of Deep Learning within finance include prediction, algorithmic trading and decision-support. However, the Efficient Market Hypothesis (Fama, 1970) states that it is not possible to predict future stock returns based on publicly available information. The hypothesis has been challenged since its conception by evidence of inefficiencies such as momentum and mean-reversion, and faces an even larger challenge with the rise of Deep Learning.

Long Short-Term Memory (LSTM) networks, a Deep Learning technique, are inherently suitable for financial time series predictions. Despite the technology having the potential to bring promising contributions to the field, academic work on financial time series prediction is limited compared to other sciences like physics or chemistry. This might be explained by two factors. Firstly, it is notoriously difficult to backtest trading strategies and be able to separate between strategies that will perform well in the future and those that will not (Lopez de Prado, 2018). Secondly, if you find an anomaly in the financial markets you are much better served financially keeping it to yourself than publishing it.

In the current literature on the topic, there is a gap between the research results and the applicability to real-world trading. We will therefore contribute to bridge this gap and give insights into how well such predictive models perform when actual transaction costs and real-life considerations in the financial markets are taken into account.

1.1 Problem definition

Based on the potential for Deep Learning in finance and the fast pace in the field, the scope of this thesis will be to test predictive models on the constituent stocks of the Oslo Stock Exchange Benchmark Index (OSEBX), which previously has not been explored in terms of Deep Learning. We will evaluate the predictive performance of different models and see how it translates into trading performance. By also putting emphasis on transaction costs and implementational challenges, we aim at making the paper appealing to both academics and decision-makers in the financial industry. Based on our motivation and choice of scope, our research can be divided into two research questions:

1. How well does an LSTM network predict daily out-of-sample directional movements of the constituent stocks of the OSEBX, and how does simple long trading strategies based on LSTM predictions perform?

2. How does these results hold up when total transaction costs are taken into account and more advanced trading strategies are implemented?

By exploring these topics we hope to give insights into how this promising technology can improve predictive performance and be used by professionals in the financial industry in the future. How our model would be implemented for real trading, from fetching live data to executing trades, is out of scope for this thesis as the implementation would make the thesis too extensive. Short strategies are also out of scope due to implementational issues regarding liquidity, lending availability and transaction costs. Moreover, we want this thesis to be interesting for both the technically advanced as well as those with financial knowledge. The implication of this is that we elaborate on both financial theory and Deep Learning, while adding more detailed technical explanations in our appendix.

1.2 Literature review

Employing Deep Learning for stock return prediction is a relatively new topic, and to the best of our knowledge it does not exist any literature on this topic on the Oslo Stock Exchange (OSE). However, some research do exist on the use of machine learning and Deep Learning for predicting stock prices and returns primarily on the US market. However, most of the academic research only implements simple strategies and estimates performance based on rough estimates of transaction costs. In this section we will sum up the main findings and research done previously in this field and elaborate on research predicting stocks on the OSE.

We can find research back to 1993 of predictions of stock prices and returns using Deep Learning, where Galler and Kryzanowski (1993) correctly classified 72 percent of the one-year-ahead stock returns either being positive or negative using financial ratios and macroeconomic variables. Olson (2003) used Deep Learning to forecast one-year-ahead Canadian stock returns using accounting ratios as input values, and reported that their model outperformed traditional regression techniques.

More recent, Huck and Krauss (2017) did a comparison of different machine learning techniques for stock price prediction on the constituents of the S&P 500, where Deep Learning did not outperform more traditional techniques. More interestingly the models performed particularly good in situations of high market turmoil, e.g., the dot-com bubble and the global financial crisis. In addition they found that returns after 2000 were significantly lower, and their opinion was that this might be due to increasing use of machine learning following increased computing power.

Fischer and Krauss came with more relevant literature in 2018 on Deep Learning, using LSTM networks to predict out-of sample directional movements for the constituents of the S&P 500. This was the first application of LSTM networks on stock return prediction and it is therefore the most relevant literature for our thesis. The paper used an LSTM network benchmarked against more traditional techniques, and found that their networks (LSTM) exhibited the highest predictional accuracy and that trading strategies based on the predictions outperformed the other models with a Sharpe ratio before transaction costs of 5.8 from 1992-2015. However, the LSTM network did not outperform after 2010, showing similar patterns as Huck and Krauss (2017). The paper also aimed towards giving a glimpse into the black-box of Deep Learning by shedding light on common patterns in traded stocks that were identified, and found that a

large part of the predictions were based on mean-reversion.

Research on the performance of prediction models on the Oslo Stock Exchange (OSE) has been very limited. Andersen and Mikelsen (2012) used a combination of machine learning techniques to predict stock returns of the Oslo Benchmark Index (OBX) constituents and the Dow Jones. They found that while the Dow Jones was highly efficient, the results on the OBX showed inefficiencies with potential for exploitation. However, no trading system was found to ouperform an efficient market portfolio after transaction costs.

Based on the motivation outlined previously and the research done within the field, our contribution to the literature will be two-folded:

1. Firstly, we will add on previous research on LSTM networks as a prediction method by testing the robustness on a new, smaller and less liquid stock universe (OSEBX). It is thus interesting to see if there are differences in performance, as well as seeing whether there are differences in the persistence of performance. Moreover, we will add on previous work in terms of using deeper models, present data differently for the models and include additional independent variables to see if it can improve performance.

2. Secondly, we will go further than previous research by taking a practitioners approach and look at how total transaction costs will affect performance, as well as testing more complex strategies to reduce transaction cost impact. Most academic research uses simple long/short strategies and assumes a flat transaction cost ranging from 1.95 basis points to 10 basis points, not taking bid-ask spreads into account. We will account for the bid-ask spread that represent a substantial part of the total transaction costs when trading on the Oslo Stock Exchange, as well as compose more advanced strategies to counter the high transaction costs. Hence, we will make the thesis more relevant for practitioners and real-world application.

1.3 Thesis structure

The thesis will start by briefly elaborating on the relevant financial and technical theory. Moving on, chapter 3 will explain our choice of data, while chapter 4 outlines the theoretical methodology behind the predictive models and trading strategies. Chapter 5 will include the analysis and results of the prediction models and backtests of our trading strategies. In chapter 6 we take a practitioners approach, taking total transaction costs into account and implementing advanced strategies. Furthermore, a pragmatic discussion of the robustness of our models and challenges associated with implementing this in practice will be done in chapter 7.

2. Theory

In this chapter we will start by discussing the Efficient Market Hypothesis, before we will look deeper into challenges related to implementation of trading strategies in terms of backtesting and transaction costs. Lastly, we will give a brief introduction to Deep Learning, focusing on Neural Networks and Long Short-Term Neural networks.

2.1 The Efficient Market Hypothesis

The Efficient Market Hypothesis (EMH) is an economic theory presented by Eugene Fama (1970), suggesting that markets are efficient and all available information are taken into account for all prices, implying that all prices are "fair". In 1991 Fama rephrased the theory to:

"Prices reflect information to the point where the marginal benefits of acting on information (the profits to be made) do not exceed the marginal costs"

It is common to distinguish between three forms of the Efficient Market Hypothesis. The weak form suggest that prices reflects all information from past prices, and thus implies that technical analysis will not work. Semi-strong form suggest that prices reflects all publicly available information, thus implying that fundamental analysis will not work. Hence, active management is largely a waste of time and effort and it is hard to justify the costs associated with it. Strong form suggest that prices reflect all available information relevant to the firm, including information available only to company insiders (Fama, 1970).

There is a large body of evidence in the literature suggesting that the weak form and semi-strong form of the theory holds. A strong case for the semi-strong from holding, is that the majority of studies that analyze the performance of mutual funds find that mutual funds under-perform their benchmarks. Malkiel (1995) looked at US equity mutual funds adjusted for survivorship bias and found that on the aggregate level mutual funds underperformed index even gross of expenses. Looking to the Oslo Stock Exchange, Fure (2014) found that 21 out of 22 mutual funds did not generate a risk-adjusted return over the benchmark.

On the other hand, there is also a substantial amount of literature advocating why the EMH does

not hold. Poterba and Summers (1987) found evidence of mean-reversion in stock prices, that underperforming stocks yielded substantially higher returns than overperforming stocks. They concluded that the mean-reversion behaviour was due to time-varying returns and "price fads" that caused stock prices to deviate from fundamental values. Jegadeesh and Titman (1993) found evidence of stock prices exhibiting momentum, that strategies buying past winners and selling past losers generated abnormal returns. Brodin and Abusdal (2008) found evidence that momentum strategies generated an excess return not explained by systematic risk factors on the Oslo Stock Exchange.

If the weak-form of the efficient market hypothesis holds, our technical approach will not be able to generate a risk-adjusted return above the market portfolio. Hence, this paper will implicitly test whether the weak-form EMH holds for the OSEBX constituents. We believe that certain inefficiencies exist, as shown in the paragraph above, and that Deep Learning models can be a suitable tool to uncover potential inefficiencies and exploit them.

2.2 Trading implementation

2.2.1 Backtesting

Backtesting is the process of assessing the viability and performance of a trading strategy by the performance on historical data. Backtesting is notoriously difficult, and a large subject in and of itself. In 2014, a team of researchers at Deutsche Bank published a study identifying the following seven basic errors frequenting backtests (Luo et al., 2014):

1. **Survivorship bias:** Using an investment universe only including current stocks.
2. **Look-ahead bias:** Using information that was not public at the moment that the simulated decision would be made.
3. **Storytelling:** Making up a story ex-post to justify some random pattern.
4. **Data mining and data snooping:** Training the model on the testing set.
5. **Transaction costs:** Simulating transaction costs correctly is hard because you would have to perform the trade to know them for sure.
6. **Outliers:** Basing a trading strategy on a few extreme outcomes that may never happen again.
7. **Shorting:** Typically requires finding a lender, and the cost of lending and amount available is generally unknown.

Even though the backtest avoids these basic errors, the case is still that the flawless backtest does not exist. A key issue is the fact that backtesting is done ex-post, and the issue of selection bias therefore becomes prominent. If the researcher tests enough different strategies on the past, some of them are likely to yield good results due to overfitting to the past, and the past will not repeat itself (Lopez de Prado, 2018) . Hence, good performing backtests does not guarantee good performance in the future.

2.2.2 Transaction costs

Of the seven mentioned basic errors, one of the most prominent is the simulation of transaction costs. In order to know certainly which transaction costs would have occurred, one would

have to travel back in time and actually perform the trades. Total transaction costs can be decomposed into (Arne Ødegaard, 2009):

- Direct transaction costs: Commissions
- Indirect transaction costs: Price impact and implementation shortfall

Where direct transaction costs represents the commissions paid to the broker. Price impact is the difference between the last traded price when you execute the trade, and the actual price that is paid. Implementation shortfall involves the opportunity cost if you place an order that does not immediately go through, and the market starts drifting. Thus, you might end up not getting the stock you want and your portfolio composition and returns suffer. It is therefore a trade-off between price impact and implementation shortfall, you can choose to buy the stock at the decision time and fill the order at the current ask orders and avoid implementation shortfall, or you can place an order in between the bid-ask spread to reduce the price impact while being subject to implementation shortfall (Arne Ødegaard, 2009).

Most academic papers on the subject of deep learning predictions trade day-to-day on adjusted closing prices, only accounting for the direct transaction costs typically assumed to be 5 basis points (Fischer & Krauss, 2018) (Huck & Krauss, 2017) (Avellanda & Lee, 2008). However, doing this underestimates the total transaction costs as a result of a substantial price impact in terms of the bid-ask spread. Ødegaard (2009) found the median relative bid-ask spread on the Oslo Stock Exchange (OSE) to be 200 basis points for the period 2000-2008. If a trading strategy rebalances the portfolio daily, it is clear that the price impact taking the bid-ask spread into account will reduce the profitability substantially.

2.3 Deep Learning: Neural networks

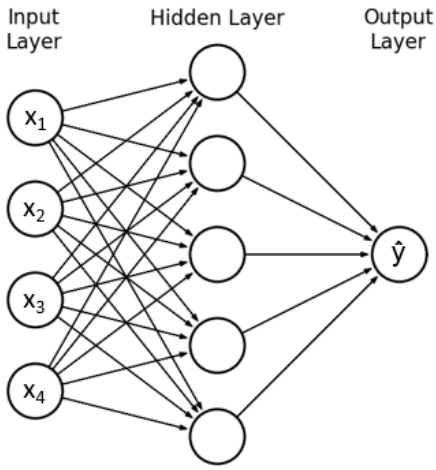
Deep Learning is a subset of machine learning, which in turn is a subset within artificial intelligence (AI). Arthur Samuel coined the phrase machine learning in 1959, and defined it as "the ability to learn without being explicitly programmed". The main purpose of machine learning is to learn the general structure of the underlying data, so that we can produce predictions about future unseen data. Deep Learning is a subset of machine learning that mainly employs neural networks that are able to learn complex data structures. Neural networks have been around for a long time, but due to being very computer intensive their potential has only recently been

realized. They are inspired by the biological neural networks that constitute the human brain, hence the name.

2.3.1 Network structure

Similar to a linear regression, data is structured into independent variables x, explaining the dependent variable y. The dependent variable might be continuous (regression) or categorical (classification). One example might be to predict housing prices (continuous dependent variable) using independent variables such as size, neighborhood, number of bedrooms and so on. Where a linear regression would go directly from the independent variables (and estimated coefficients) to predicted dependent variable, a neural network adds complexity and non-linearity to be able to understand complex data structures.

Figure 2.1: Traditional illustration of a neural network.



The basic units of a neural network are called neurons and connections with corresponding weights. Neurons are shown as circles in figure 2.1 and contains a numeric value. All the neurons are attached to each other by connections that have a specific weight. The neurons in the input layer (shown as circles in figure 2.1) contains the values of the independent variables for one observation. Furthermore, the network has l hidden layers where the value of each neuron in a hidden layer is calculated based on all the neurons in the previous layer and weights connecting them (Goodfellow, Bengio, & Courtville, 2016). Finally, the neuron(s) in the output layer takes their value based on the neurons in the last hidden layer and the weights in the connections to the output neuron(s), and these output neuron values will be the predictions

(Chollet, 2017). Hence, a linear regression can be viewed as a neural network with only an input layer and output layer, where the estimated coefficients are the weights.

2.3.2 Components of the network

The data in the network flows through each neuron by a connection (shown as a line in figure 2.1). Every connection has a specific weight (w) by which the flow of data is regulated (Shubham Panchal, 2018). Each neuron ($x_{l,f}$) in the hidden layers and output layer gets their value by taking the product of all the neurons in the previous layer X_{l-1} and the weights connecting them W_{l-1}, as well as adjusting for a bias, b (Chollet, 2017). The process of going from independent variables to calculating the predictions in the output layer is called *forward propagation*. The network learns by adjusting the parameter space Θ consisting of the weights and the biases.

$$x_{l,f} = \sum_{f=1} (w_{l-1,f} * x_{l-1,f}) + b_{l,f} \tag{2.1}$$

The value of $x_{l,f}$ can be anything ranging from minus infinity to infinity. This value is simply a linear combination of weights and neuron values. The input signal $x_{l,f}$ from equation 2.1 is thus converted to an output signal $g(x_{l,f})$ through a non-linear activation function g, allowing the neural network to understand non-linear data.

2.3.3 How the network learns

At the end of a forward propagation, the output layer's predicted value $\hat{y}$ is compared to the real y value, and a loss function L is computed. The loss function determines the distance between the predicted values $\hat{y}$ and the real values y, and thus explains how accurate the predictions are. The network learns by going through the observations of the data set multiple times and minimizing the loss function L by changing the parameters Θ (weights and biases) of the network. Efficient training of a neural network is thus reliant on a large data set with many observations to be able to learn.

$$L(\hat{y}, y) = L(f(x, \Theta), y) \tag{2.2}$$

The partial derivatives of the loss function with respect to the network parameters Θ is calculated between each forward propagation for an observation to find how much each parameter contributed to the loss function. The parameters are then adjusted according to their partial derivatives.

To summarize, we first forward propagate to get predicted values and calculate the loss function, before finding the partial derivative of each parameter with respect to the loss function and adjust the parameters accordingly. Forward propagating through all the observations in the data set and adjusting the weights along the way is characterized as one *epoch*, and in order to find the optimal parameters it is often necessary to perform multiple epochs.

2.3.4 Long Short-Term Memory Networks

Recurrent Neural Networks (RNNs) can be thought of as multiple copies of the same neural network over time where each network passes a message to its successor (Olah, 2015). RNNs are thus able to capture dependencies over time and sequences, and have shown incredible success within areas such as speech recognition, language modelling and translation (Olah, 2015).

Long Short-Term Memory (LSTM) networks are a type of RNN especially suited to learn long-term dependencies in data structures due to what is called memory cells, making them suitable for financial times series such as stock returns. The name Long Short-Term Memory can be decomposed into two parts: the long-term memory is due to the learned parameters (weights and biases) that changes slowly, while the short-term memory refers to the memory cells that changes at each time step (Hochreiter & Schmidhuber, 1997). For a detailed explanation of how RNNs and LSTMs work, see appendix A.

3. Data

We use market data on the constituents of the Oslo Stock Exchange Benchmark Index (OSEBX) from 1 January 1996 until 31 December 2017. Constituents of the index have been determined semiannually for the period, and we obtain historical lists of constituents through correspondence with the Oslo Stock Exchange. We eliminate survivorship bias by combining these constituent lists into a binary matrix indicating whether the stock was a constituent of the index at any given date. We then download the dividend-adjusted daily closing stock prices, the daily closing bid and ask prices and the daily trading volumes of all stocks listed on the Oslo Stock Exchange through Norwegian School of Economics' stock database "Børsprosjektet". All stocks that had never been constituents of the OSEBX was dropped from the data set, using ISIN number as identifier. Detailed data on stocks such as market capitalization, turnover and beta values are obtained through Thomson Reuters Datastream. Additional independent variables such as government bond yields and foreign exchange rates are downloaded from Norges Bank.

Table 3.1: Summary statistics from data set.

OSEBX Summary Statistics	
Minimum number of constituents in period	52 (H1 2004)
Maximum number of constituents in period	81 (H1 2008)
Average number of constituents in period	63
Total number of constituents in period	235
Largest constituent, share of index (2018)	Equinor, 28 %
Smallest constituent, share of index (2018)	Targovax, 0.02 %
Annualized return (1999-2017)	10.1%

In order to add on previous academic work within deep learning, mostly on the highly liquid and efficient S&P 500 Index, we choose the Oslo Stock Exchange as it is less liquid and likely to be less efficient. The OSEBX Index is chosen as it is a broad index consisting of the most liquid equities listed on the Oslo Stock Exchange (Oslo Stock Exchange, 2018). From table 3.1 we see that the OSEBX has contained between 52 and 81 constituents, with a relative high turnover with a total of 235 constituents over the period. We also see that the index is dominated by some large companies, with Equinor representing 28 percent of the total market cap. In order to fully leverage our predictive models, we sought to obtain time series for a period as long as possible. Our time range is thus chosen as this was the maximum time period that data on constituents was accessible. From 1983 to 1996 the broad index of Oslo Stock Exchange was named the Total Index (TOTX), and Oslo Stock Exchange does not have digital records of the historical constituents of this index (correspondence with Oslo Stock Exchange).

Software and hardware

Data preparation, data handling and performance analysis is performed in Python, a programming language for general-purpose programming. Our deep learning networks are developed with the package Keras, on top of Google's Tensorflow library. We also use the package Scikit-learn for all our benchmark models. Our LSTM networks are trained on GPUs through Google Cloud Platform in order to overcome the issues regarding computing power intensity. All benchmark models are trained on CPUs.

4. Methodology

Throughout the chapter we will explain the methodology behind the prediction models, trading strategies and performance evaluation. Our approach is inspired by the work of Fischer and Krauss (2018) which did similar work on the S&P 500 constituents, and we also approach the problem as a binary classification problem. The predictive models predict the probability that a stock will out-perform the cross-sectional median return the following day. Deeper technical explanation can be found in the appendix B.

4.1 Dependent and independent variables

The primary independent variable for the predictions are a sequence of the last 240 stock returns (from t-240 to t), while the dependent variable is a binary variable showing whether the stock outperformed the cross-sectional median return the following day (t+1). The lookback period of 240 days is chosen due to previous literature and has proven to work well with the type of models we are using (Fischer & Krauss, 2018).

Our initial data set is a 2-dimensional matrix where each row shows the stock price for a given stock at a given day. The price of a stock can thus be represented as $P_{t,s}$, where t is the date and s is the stock. Simple returns, our primary independent variable, can thus be calculated as follows:

$$R_{t,s} = \frac{(P_{t,s} - P_{t-1,s})}{P_{t-1,s}} \tag{4.1}$$

Using the returns, $R_{t,s}$, we create our binary dependent variable:

$$B_{t,s} = 1 \quad if \quad R_{t,s} \geq median(R_t) \tag{4.2}$$

$$B_{t,s} = 0 \quad if \quad R_{t,s} < median(R_t) \tag{4.3}$$

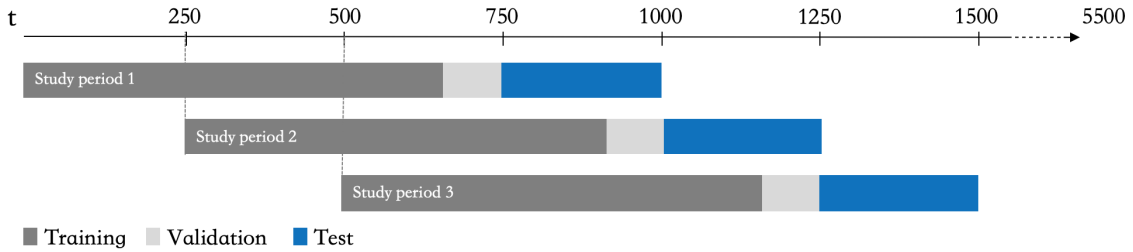
For the more advanced models we add independent variables supplementing the simple returns. These variables are 50-days moving average price, 200-days moving average price, daily traded volume, bid-ask spread of the corresponding stock returns and the macroeconomic variables 3-year government bonds, 10-year government bonds and the USD/NOK foreign exchange rate.

The macroeconomic variables are chosen as they are likely to have impact on the constituent stocks of the OSEBX. For some models, we denoise certain independent variables through a wavelet transformation, which remove much of the noise in the time series and makes it smoother. The goal of denoising is to make it easier for the model to interpret the signals better (Palaniappan, 2018).

4.2 Study periods

We divide our data set into study periods that will be used for training and testing of our models. This is to make sure that the model is trained on selected time periods as the market conditions have been changing over our full time period. When performing deep learning, it is important that the model is trained on specific observations (training set) before it is tested on observations (test set) that it has not yet seen (otherwise the model would know the answers). Each study period contains 1000 trading days of stock returns which is split into 750 training days and 250 test days. The training observations are used to decide the parameters of the LSTM network, while the test observations are used for out-of sample predictions and testing of trading strategies. To avoid testing on the same data set twice, each study period starts 250 days after the previous study period as illustrated in figure 4.1. By doing so we end up with a total of 19 periods as our data set contains 5500 observations and we loose the first 750 observations to training in the very first study period. We thus end up with a non-overlapping prediction period from 29th December 1998 to 27th November 2017.

Figure 4.1: Graphical illustration of how the study periods roll over the complete data set.



A neural network learns certain parameters on it's own, while the researchers determines certain hyper-parameters. The last 20% of the training observations are used as a *validation set*. A validation set is necessary because optimizing hyper-parameters based on the same data we use for predictions would lead to selection bias and much better results than we can expect in the

future (Lopez de Prado, 2018).

Standardizing the data

The returns are standardized based on the mean and standard deviation only from the training set to avoid look-ahead bias. If n denotes the study period, we thus standardize as follows:

$$R^n_{t,s} = \frac{(R^n_{t,s} - \mu^n_{train})}{\sigma^n_{train}} \tag{4.4}$$

The data set contains some missing values arising due to stocks being de-listed or listed during the study period length of 1000 days. Any missing values present in the data set after standardization are replaced by zeros, as the LSTM network is unable to predict in the presence of any missing values among the independent or dependent variables.

4.3 Training the network

The aim of having such an extensive data set is to be able to train our model sufficiently enough to learn possible patterns in the stock returns. We train the model independently on all 19 study periods to generate predictions independently in each study period. We first obtain a list of all stocks that were constituents of the OSEBX at the last training day of the study period - as these are the only stocks that we will perform predictions on.

Our model then classifies each observation as either 1 (outperform median) or 0 (underperform median) and calculates the probability that the observation belongs to class 1. We apply the loss function binary cross-entropy which calculates not only how accurate the predictions are, but *how far the predicted probabilities are from the true value* (1/0). Based on the results, the model will adjust the parameters of the network in order to minimize the loss function. The model will iterate through the process until it has reached an optimal choice of parameters.

Overfitting is when a predictive model learns the detail and noise in the training data to such an extent that it negatively impacts the performance of the model on new data (Brownlee, 2016). Unfortunately, overfitting is an overhanging threat when dealing with financial data (Lopez de Prado, 2018), and we therefore apply several methods to avoid this outlined in section B of the Appendix.

4.4 Selected models

We have a total of seven models that are evaluated in the next chapter. Three are benchmark models and four are LSTM variations. The models differ both between what type of model, training approach and hyper-parameters employed. Overview of hyper-parameters can be found in section B of the Appendix.

Benchmark models

Logistic regression

Logistic regression is a widely used econometric technique that is suited to predict a binary dependent variable (1/0). Due to its simplicity and widespread adoption within econometrics, we use it as a bottom line benchmark for the other models. The model calculates the logarithm of the odds for an observation belonging to class 1 as a linear combination of the independent variables. The logarithm of the odds are converted to the probability of the observation belonging to class 1 using a logistic function, hence the name (Cox, 1958).

Random forest (RAF)

Decision trees are a machine learning technique that splits the data into subsets (branches) based on learned conditions to be able to classify observations. Shortcomings of decision trees include overfitting to the data as well as finding locally optimal solutions. A random forest consist of multiple decision trees, where the average of the predictions of the decision trees are used as the predictions of the random forest. Hence, random forests reduce the risk of overfitting and locally optimal solutions.

Support vector machine (SVM)

Support vector machines plot the observations as points in a multidimensional space and seeks to use a mathematical method to separate observations of the two classes. The model essentially plots a boundary in the multidimensional space that maximizes the distance between the boundary and the closest observation from each class on both sides of the boundary. The vectors from the closest observation on each side to the boundary is called the support vectors, hence the name of the method. One of the key advantages of the model is the ability to create non-linear boundaries, thus being able to classify non-linearly separable classes.

LSTM variations

LSTM individually trained

This model trains on each stock separately before it moves on to the next stock within each study period. The advantage of individual training is that the model learns the patterns specific to the given stock, while the disadvantage is that the number of observations are drastically reduced.

LSTM trained on all stocks

This model trains on all stocks simultaneously within each study period. Hence, it does not learn patterns specific to each stock, but patterns represented by all stocks. A key advantage is that this yields more observations for training.

Stacked LSTM

Same as LSTM trained on all stocks, but this model uses two LSTM layers in the network, making the model deeper and increasing the number of parameters. Advantages are that the model might uncover more complex patterns than the one-layer model.

LSTM with additional independent variables

Same as LSTM trained individually, but we add independent variables that we believe might be able to help further explain the stock return performance. The variables used are simple returns, bid-ask spread, volume, moving averages of stock returns and the USD/NOK foreign exchange rate.

4.5 Predictive performance

To evaluate the performance of the predictive models we apply several performance measurements. The first measurement is the directional accuracy, which is the fraction of observations correctly classified. Directional accuracy can be calculated as follows:

$$Directional\ accuracy = \frac{correct\ predictions}{total\ predictions} \tag{4.5}$$

Moving on we also calculate positive accuracy and negative accuracy, which explains how likely it is that an observation will actually be positive (negative) when we predict that it is positive (negative):

$$Positive\ accuracy = \frac{correct\ positive\ predictions}{total\ positive\ predictions} \tag{4.6}$$

Recall is also a relevant measurement, representing how robust the predictive model is. Recall explains how many of the actual positive observations that we predicted to be positive:

$$Recall = \frac{correct\ positive\ predictions}{total\ positive\ observations} \tag{4.7}$$

The F1 score is the harmonic mean between positive accuracy and recall, e.g. how precise and robust the model is.

$$F1 = 2 * \frac{positive\ accuracy * recall}{positive\ accuracy + recall} \tag{4.8}$$

Lastly we have binary cross-entropy (BCE), which we use as our loss function when training our networks. The key conceptual difference between accuracy and binary cross-entropy is that binary cross-entropy takes into account not only whether our predictions were correct or not, but also how correct they are (penalizes the difference between predicted probability and real outcome). Since our trading strategies are based on the predicted probabilities, this loss function is better suited for our purpose than directional accuracy alone (Lopez de Prado, 2018). Binary cross-entropy can be calculated as follows:

$$BCE = -\frac{1}{N}\sum_{n=0}^{N}(y_n log[p_n] + (1 - y_n) log[1 - p_n]) \tag{4.9}$$

Where N is the number of observations, y_n is the true class of observation n and p_n is the predicted probability of observation n.

4.6 Trading performance

4.6.1 Simple strategies

We apply different long trading strategies based on the predictions and measure the cumulative return over time. Only applying long strategies is a conscious choice, due to short strategy implementation issues such as liquidity, lending availability and transaction costs.

Before transaction costs

We instruct the model to buy K stocks at close-price each day, and adjust the portfolio accordingly. The stocks selected are the K stocks with the highest predicted probabilities to beat the cross-sectional median return the following day. The portfolio value, prior to transaction costs, can therefore be computed as follows:

$$V_t = V_{t-1} + \sum_{k=0}^{K}(V_{t-1} * w_{t-1,k} * r_{t,k}) \tag{4.10}$$

Where V_t is the value of the portfolio at time t, k refers to one out of K stocks, $w_{t-1,k}$ is the weight of each stock k at time $t-1$ such that the product of V_{t-1} and $w_{t-1,k}$ is the monetary value of the capital invested in stock k at close-price on day $t-1$. In general, we use equal weights for all the stocks held in the portfolio. From close-price at day $t-1$ until close-price at day t, the actual return, $r_{t,k}$, for stock k is computed and multiplied with the capital invested at time $t-1$ to get the monetary return for this stock. The sum of these returns added to the portfolio value at time $t-1$ amounts to the portfolio value at time t.

Due to Oslo Stock Exchange updating the constituent lists semi-annually, some of our predictions will be for stocks that are no longer available for trading. If one of the top K stocks are not available for trading, the portfolio will then invest more in the other $K-1$ stocks as opposed to keeping the funds as cash.

After direct transaction costs

The direct transaction cost used is the lowest commission cost offered to private investors at the Oslo Stock Exchange at 2.9 basis points (round-trip of 5.8 basis points) (Nordnet, 2018). In order to qualify for this commission rate you need to perform 40 trades per month, and the minimum cost is 39 NOK - meaning that trades need to be above 134 482 NOK to achieve a commission of 2.9 basis points. Our strategies are likely to fulfill the trading frequency requirement, and we assume that trade sizes would be large enough to make the cost 2.9 basis

points. The portfolio value after direct transaction costs is thus calculated as follows:

$$V_t = V_{t-1} + \sum_{k=0}^{K}(V_{t-1} * w_{t-1,k} * (r_{t,k} - x * p_{t,k}) \tag{4.11}$$

Where x represents the one-way direct transaction cost, and $p_{t,k}$ is a variable deciding how much transaction costs should be applied to stock k at time t. If we did not have to buy the stock (already holding it) at time t-1, and did not sell it at time t, $p_{t,k}$ takes a value of 0. If we had to buy the stock at t-1, but did not have to sell it at time t, $p_{t,k}$ takes a value of 1. Similarly, if we did not have to buy the stock at time t-1 but sold it at time t, $p_{t,k}$ takes a value of 1. If we both had to buy the stock at time $t-1$ and sell it at time t, $p_{t,k}$ takes the value 2 (round-trip transaction).

Accounting for bid-ask spread

Thirdly, we also include indirect transaction costs in the form of the bid-ask spread. We create a matrix of the relative bid-ask spread at market close at time t for all stocks s, and divide it by 2 to get the "one-way" spread ($s_{t,k}$) we would have to cross to either buy or sell a stock. The portfolio value is then calculated as follows:

$$V_t = V_{t-1} + \sum_{k=0}^{K}(V_{t-1} * w_{t-1,k} * (r_{t,k} - (x + s_{t,k}) * p_{t,k}) \tag{4.12}$$

4.6.2 Advanced strategies

When we also include the indirect transaction cost in the form of the bid-ask spread, the transaction costs heavily affect portfolio performance. We thus implement more sophisticated strategies in order to increase profitability, reduce the turnover as well as limiting trading in illiquid stocks with high bid-ask spreads.

Bet sizing

Many of the strategies we produce follows the academic standard of keeping the weights $w_{t-1,k}$ equal for all K stocks. We also follow Lopez de Prado (2018) and implement bet sizing to utilize the differences in the certainty of our predictions to improve profitability. We first rank all the constituent stocks by their predicted probabilities of beating the cross-sectional median the following day. We then standardize the probabilities, select the K stocks with the highest probabilities and calculate their weight by re-scaling the standardized probabilities so that the sum of all weights add up to one. This technique allows us to invest more in the stocks that

we are more certain of outperforming the median, and less in those stocks where the degree of uncertainty is higher.

Constraining strategies

We also add different constraints to reduce the turnover and trading of illiquid stocks. K stocks are first selected based on their predicted probabilities of beating the cross-sectional median return the following day, and thereafter stocks are removed if they do not fulfill the given constraint. We implement five different constraints:

- **Volume**: Limiting the strategy to only invest in stocks that have had an average trading volume by value over the last 10 days that is above a certain percentile of the volume distribution of all stocks in the period.
- **Monthly trading**: In order to reduce turnover, we rebalance the portfolio every 20th trading day.
- **Probability threshold**: A clear weakness of the simple long strategy is that it does not consider general developments in the market. Thus, during a bear-market we still invest even though it may be likely that even the top K stocks will underperform a cash position. Hence, we include a threshold for the predicted probability, so that the probability has to be above a certain threshold before we invest in the stock.
- **Spread threshold**: We remove stocks from the K initially selected if their average bid-ask spread over the last 10 days is above a selected threshold.
- **Turnover**: When rebalancing the portfolio, we check which predicted probabilities our current holdings has for the following day. If our current holdings are part of the top $K * multiplier$ stocks the following day, we do not sell this stock. If it is not part of the top $K * multiplier$ stocks, we replace it with the stock with the highest predicted probability for the following day that is not a part of the current portfolio. Here, K is the chosen number of stocks to hold as a baseline and *multiplier* is a parameter we determine.

5. Analysis and results

In this section, we start by comparing the predictive performance of our models before evaluating the performance of trading strategies relying on our predictive models. Trading performance is first evaluated without transaction costs, before taking direct transaction costs into account. All performance evaluation is based on the performance on the out-of sample test set, consisting of 4 750 consecutive trading days, starting 29 December 1998 and ending on 27 November 2017.

5.1 Predictive models

We have seven predictive models that are considered for test set comparisons. Three of the models are econometric and machine learning models that serve as benchmarks for the other four LSTM variations. See methodology chapter and appendix B for deeper explanation of intuition and architecture of the models:

- **Logistic (benchmark):** Logistic regression, trained on each stock.
- **RAF (benchmark):** Random forest, trained on each stock.
- **SVM (benchmark):** Support vector machine, trained on each stock.
- **LSTM_i:** LSTM, one layer, trained on each stock.
- **LSTM_a:** LSTM, one layer, trained on all stocks.
- **LSTM_d:** Stacked LSTM with two layers, trained on all stocks.
- **LSTM_f:** LSTM, one layer, trained on each stock with six independent variables (returns, bid-ask spread, volume, USD/NOK, moving averages of stock returns)

5.2 Predictive performance

Table 5.1: Different measurements of predictive performance for all models

	Logistic	**RAF**	**SVM**	**LSTM_i**	**LSTM_a**	**LSTM_d**	**LSTM_f**
Positive predictions share	0.5479	0.658	0.6254	0.6725	0.8147	0.8408	0.4867
Directional accuracy	0.5034	0.5232	0.5153	0.5286	0.528	**0.5317**	0.5161
Positive accuracy	0.5087	0.5328	0.5282	0.5361	0.5295	0.5307	**0.5371**
Negative accuracy	0.497	0.5046	0.4937	0.5131	0.5216	**0.5367**	0.4962
Recall[1]	0.5507	0.6742	0.6353	0.6933	0.8296	**0.8581**	0.5027
F1 score[2]	0.5289	0.5952	0.5769	0.6047	0.6464	**0.6559**	0.5194
Binary cross entropy[3]	0.7673	0.7274	0.6958	0.6938	0.6923	**0.6917**	0.7662

[1] The fraction of correct positive predictions among all the actual positive observations.

[2] The harmonic mean between positive accuracy and recall.

[3] How far the predicted probabilities are from the true value.

We see that the directional accuracy of all models are above 50%, which is better than a random guess. However, we see that the *LSTM_d* performs the best with accuracy of 53.17% while our least sophisticated benchmark model, the logistic regression, shows an accuracy of 50.34%. Except positive accuracy, the *LSTM_d* model performs the best on all measurements, including binary cross entropy which is an important measurement for trading performance because it is the only measurement taking into account how far the predicted probabilities are from the true outcomes, which are the probabilities used to determine portfolio composition later on. We observe that the LSTM models perform quite similar, with the *LSTM_f* standing out with a much higher binary cross-entropy. Surprisingly small differences in accuracy scoring have enormous effects on the portfolio performance, which we will see later on.

An important question is whether or not our models' predictive performance is statistically significant better than just choosing stocks at random. If that were the case we might as well stop right away and try a different approach. If the accuracy was indeed 50%, we could model the number of correctly classified stocks X as a binomial distribution, approaching a normal distribution due to the large number of observations:

$$X \sim B(n = 273750, p = 0.5, q = 0.5) \quad \longrightarrow \quad X \overset{appr.}{\sim} \mathcal{N}(\mu = n * p, \sigma = \sqrt{npq}) \tag{5.1}$$

We find a probability of 5.8e-240 (≈0) that our *LSTM_d* model has a true accuracy of 50%. We can therefore conclude that this model is statistically significant more accurate than just choosing stocks by random. Doing the same calculation for the logistic regression, we find a p-value of 0.0005, meaning that it is also statistically significant more accurate than a coin toss at a 5% confidence level. All other models are statistically significant better than a coin toss at any reasonable significance level.

5.3 Trading performance

While predictive performance measured by metrics such as accuracy and binary cross entropy are important, what really matters for our intended purpose is how trading strategies perform on the basis of those predictions. Hence, we backtest simple long trading strategies based on the predictions from our considered models. We will start by not considering transactions costs, before we move forward with our best performing model and see how it performs after direct transaction costs.

5.3.1 Before transaction costs

All our predictive models produce a probability that a given stock will have a return greater or equal to the cross-sectional median of all constituent stocks one day ahead. The constituent stocks are then ranked based on their probabilities, and an investment portfolio of the K stocks with the highest probabilities are constructed. At every time t, all funds are invested in K stocks with each stock having the same weight $(1/K)$, which means that the portfolio is rebalanced frequently.

Table 5.2: Trading performance of all models (K = number of stocks in portfolio)

	Annualized	**Logistic**	**RAF**	**SVM**	**LSTM_i**	**LSTM_a**	**LSTM_d**	**LSTM_f**
	Return	-0.0168	-0.064	0.1364	0.6342	**1.0595**	0.8255	0.9453
	Standard deviation	0.3732	0.3604	0.2641	0.2876	0.3163	0.3490	0.3872
K = 5	Sharpe ratio	-0.1321	-0.2678	0.3936	2.0926	**3.2468**	2.2726	2.3573
	Max drawdown[1]	0.9304	0.9661	0.7022	**0.4805**	0.5805	0.5964	0.6221
	VaR 5%[2]	-0.6306	-0.6568	-0.2979	0.1613	**0.5392**	0.2516	0.3084
	Return	-0.0095	-0.0566	0.0822	0.5265	0.7592	0.6173	0.7043
	Standard deviation	0.2968	0.2968	**0.2143**	0.2318	0.2544	0.2801	0.3057
K = 10	Sharpe ratio	-0.1417	-0.3001	0.232	2.1309	2.8562	2.088	2.1972
	Max drawdown	0.8878	0.9312	0.6584	0.5032	0.6144	0.6253	0.6419
	VaR 5%	-0.4977	-0.5447	-0.2702	0.1452	0.3407	0.1566	0.2014

OSEBX had an annualized return of 10.10%, std of 22.7%, Sharpe ratio of 0.30 and maximum drawdown of 64.1%.

Sharpe ratio calculated using average risk-free rate of 3.25% annually (3-year Norwegian Government Bond).

[1]The maximum loss from a peak to a trough for the portfolio, before a new peak is attained.

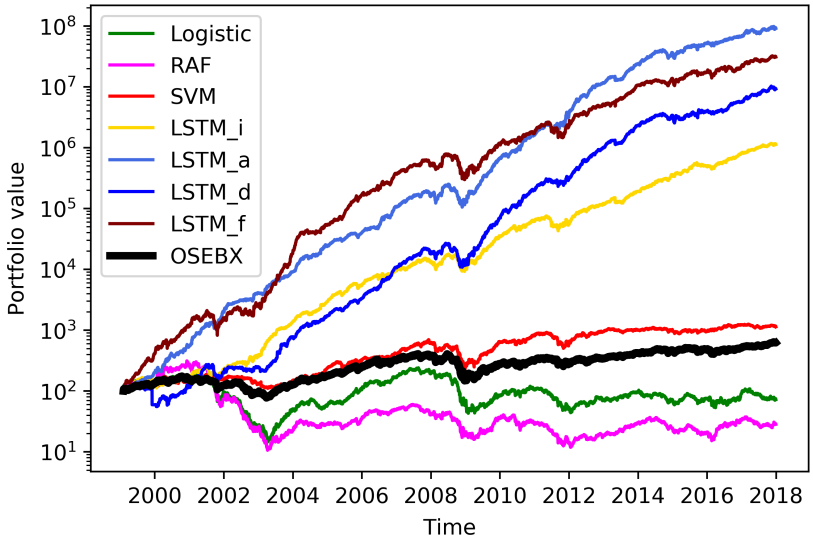
[2]Annual value at risk. There is a 5% probability of realizing a return lower than this number during a year, computed using annualized return and standard deviation.

From table 5.2 we can conclude that the simple LSTM model trained on all stocks (*LSTM_a*) seems to generally outperform all the other models. We see that the model has the highest return and Sharpe ratio for all portfolio sizes. When looking at the risk characteristics, *LSTM_a* also performs well against the other LSTM models except showing higher risk in terms of maximum drawdown and standard deviation than the *LSTM_i*.

Returns are generally higher for the low K portfolio, which seems reasonable as we then invest where our predictive models are most confident. When looking at the Sharpe ratio, we see that this is highest for the $K = 5$ portfolio for all Deep Learning techniques employed (except for the *LSTM_i* model). The higher return of the low K portfolio comes at a cost in terms of increased risk. We get a higher standard deviation due to the lack of diversification and the high impact that occurs every time we miss on an investment. This pattern is true across all models and the standard deviation range from 21.4% for the SVM model with 10 stocks, up to 38.7% for the *LSTM_f* model with 5 stocks.

An interesting finding is the fact that the LSTM models does not show a higher maximum drawdown than the OSEBX (64.1%), which we might have expected for a more volatile portfolio. For this metric we can not observe the same pattern across portfolio sizes. Despite portfolios of 10 stocks being more diversified, the maximum drawdown does not seem to be systematically lower for these portfolios. The same applies for the annual value at risk, where we cannot observe systematic differences across portfolio sizes. However, we do observe that the LSTM models have significantly better value at risk, much due to the fact that the return is extremely high for these models before accounting for transaction costs.

Figure 5.1: Trading performance before transaction costs (K = 5)



A graphical inspection of figure 5.1 tells the same story in terms of portfolio return as discussed. We observe that all models have been highly correlated to each other and the OSEBX index itself as we would expect when trading with the same selection of stocks. The LSTM model with additional independent variables outperformed all others from 1999 until 2011, while the simple LSTM model trained on all stocks outperformed from 1999 to 2012 and going forward. We can also note that all strategies based on the LSTM models have outperformed the OSEBX overall, while the other machine learning and econometric technique based strategies have underperformed or performed marginally better than the benchmark.

An interesting finding is that strategies relying on the LSTM model trained with additional independent variables (*LSTM_f*) perform much better than we would expect based on the accuracy scoring in table 5.1. The model performs better in terms of Sharpe ratio than the

regular LSTM network ($LSTM\_i$) and the model with two layers ($LSTM\_d$), despite having a lower accuracy and higher binary cross entropy. Furthermore, it is interesting to note that the model showing the best trading performance ($LSTM\_a$) was not among the best predictive performers in the preceding section.

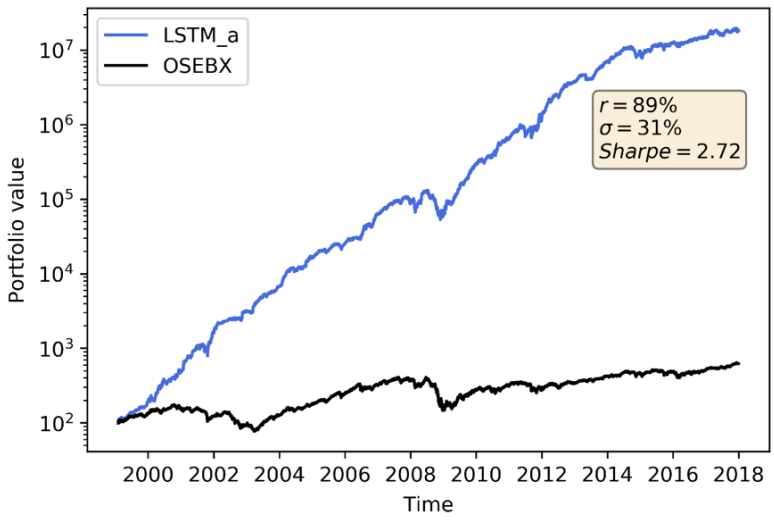
5.3.2 After direct transaction costs

In the preceding sections we saw that the model trained on all stocks ($LSTM\_a$) with K = 5 outperformed all the other models before transaction costs measured by the Sharpe ratio. In this section, we therefore evaluate the performance of this model after direct transaction costs. Whereas it might be easy to obtain good results prior to transaction costs, it is the results after transaction costs that really matters. We therefore apply a direct transaction cost of 2.9 basis points (round-trip cost of 5.8 basis points) to every trade performed, whenever we would need to buy or sell a stock.

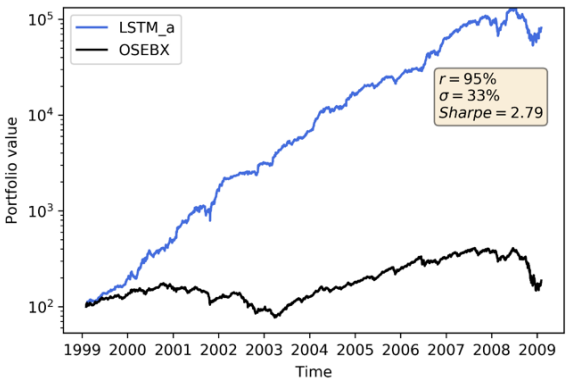
From figure 5.2 we see that annualized return is reduced by 27 percentage points, which might seem lower than expected from the transaction cost of 2.9 basis points. This is because transaction costs are only applied when buying or selling stocks, and the portfolio does not rebalance all the stocks at every time step. The transaction cost will have the effect of reducing the expected return, but will not affect the analysis from the previous section further due to the fact that this is subtracted after the model have been trained, and does not change its behavior.

As the stock markets are likely to have changed after the financial crisis in 2008, an important step of evaluating the model is to analyze the performance for the period before and after the financial crisis.

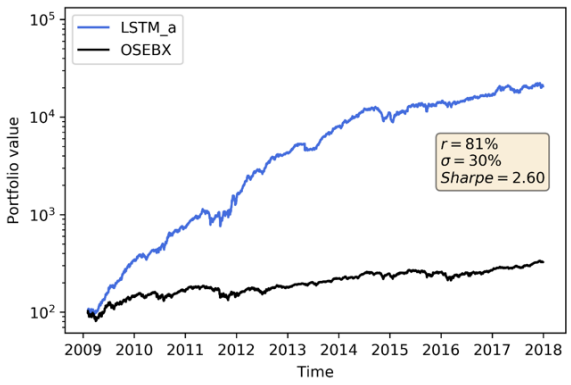
Figure 5.2: Trading performance after direct transaction costs (K = 5)



(a) Full period



(b) 1999-2008



(c) 2009-2017

	Full period		**1999-2008**		**2009-2017**		**2015-2017**	
Annualized	LSTM_a	OSEBX	LSTM_a	OSEBX	LSTM_a	OSEBX	LSTM_a	OSEBX
Return	0.8914	0.1010	0.9488	0.0644	0.8141	0.1414	0.2601	0.1233
Standard deviation	0.3159	0.2266	0.3286	0.2416	0.3001	0.2084	0.2486	0.1671
Sharpe Ratio	2.7189	0.3022	2.7888	0.1319	2.6045	0.5226	0.9157	0.5434
Max drawdown	0.5972	0.6410	0.5972	0.641	0.3317	0.2879	0.1806	0.2209
VaR 5%	0.3718	-0.2717	0.4084	-0.3331	0.3205	-0.2013	-0.1488	-0.1516

For the period from 1999 to the end of 2008, we observe a clear out-performance of the LSTM model over the OSEBX benchmark. The performance from 2009 until the end of our testing period is still strong, while not performing as strongly as in the preceding period.

Exploring this further, wee see from the table in figure 5.2 that the performance over the last three years is much lower than in the preceding years with a Sharpe ratio of 0.92 compared to 2.72 for the whole period. The volatility is somewhat lower for this period, so the Sharpe ratio decrease is driven by the reduced annualized return. Moreover, the maximum drawdown of 59.7% over the whole period occurs during the financial crisis in 2008, which is lower than the OSEBX maximum drawdown of 64.1% for the same period.

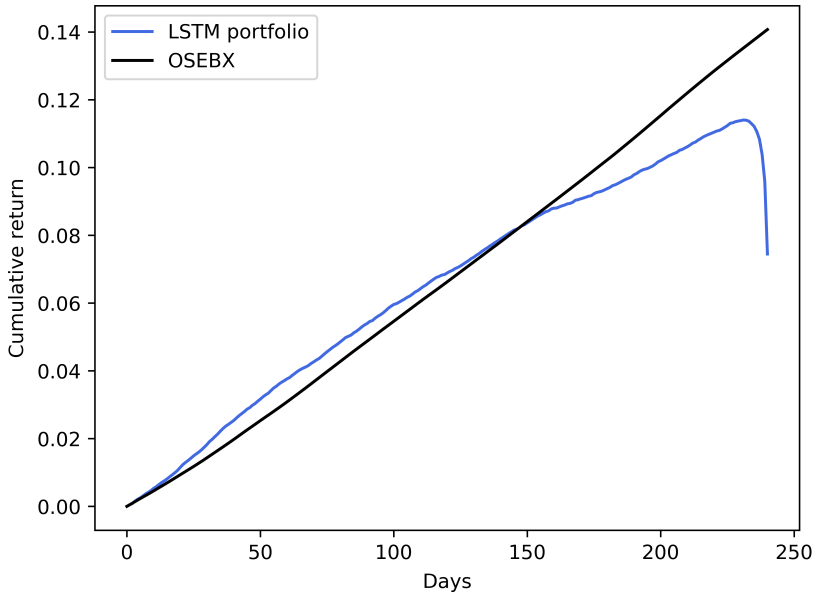
We see that the relatively modest direct transaction cost makes a large impact on the annualized return due to the frequency of the trading. We test how high the transaction cost could have been before the LSTM model would have a Sharpe ratio similar to the benchmark OSEBX (0.30), and find the break-even transaction cost to be approximately 20.5 basis points. The break-even cost might seem very high, but is due to the fact that we do not necessarily rebalance the whole portfolio at every time step.

While only taking the direct transaction cost into account is a shallow assumption for actual trading performance, our opinion is that this is an important part of the thesis due to findings in previous literature based on similar assumptions. Hence, we use the same method for the sake of comparability.

5.3.3 Portfolio analysis and stock selection

It is very interesting to analyze what types of stocks the LSTM model from the previous section holds in its portfolio, and what types of strategies it follows. By doing so we also shed some light into what is often criticized as a black-box model. We start by looking at the pattern in terms of return and volatility exhibited by the selected stocks in the period prior to selection. By analyzing the daily returns for the 240 previous days of when the chosen stocks are picked, we discover some very interesting patterns.

Figure 5.3: Average accumulated return of selected stocks over the 240 days prior to selection. Based on *LSTM_a* with K = 5.



From figure 5.3 we observe that the selected stocks have underperformed the OSEBX average in the period prior to selection. The selected stocks are on average performing at par with the market for the first 150 days, but underperform during the last 90 days. The selected stocks especially underperform in the very last days before selection, loosing on average 3.4% of their value in the last five days.

It might be counterintuitive to invest in stocks doing poorly in the recent past, but it is similar to a widely used strategy called short term mean-reversion (Poterba & Summers, 1987). Apparently, our model has discovered a well known trading strategy on its own, without any input or instructions from us. This might just partly explain the behaviour of the model, but it is still interesting to observe this behavior from a machine without any knowledge beyond daily stock returns.

In addition, we calculated the corresponding standard deviation and beta of the stocks in the preceding 240 days before selection. We find an average portfolio standard deviation of 0.195 and a beta of 0.88, compared to a standard deviation of 0.203 for the market portfolio (OSEBX). In terms of the beta, it is somewhat lower than expected. This might be explained by the fact that despite the portfolio having a similar volatility as the OSEBX, it moves differently than the market due to the choice of under-performing stocks. Another explanation might be how beta

values are calculated in our case. We calculate beta as the co-variance of our average portfolio return with the *average* OSEBX return divided by the variance of the *average* OSEBX portfolio, which equally weights all the constituents, while in reality they should be weighted by market capitalization.

Table 5.3: Descriptive statistics for the stocks selected.

	LSTM Portfolio	OSEBX
Stocks traded total	189	235
Average daily portfolio turnover	28.45%	-
Average cumulative return (t-240 to t)	6.94%	14.07%
Average standard deviation (t-240 to t)	19.47%	20.29%
Average Beta	0.88	1
Median market cap	12 240	11 458
Median turnover by value	64 496	27 333
Median free float %	61	62
Median bid-ask spread (bp)%	82	102

Cumulative return and standard deviation based on equally-weighted average of the OSEBX constituents.

Based on model $LSTM\_a$ with k=5.

Market cap, turnover by value and free float for 20 most traded stocks of LSTM portfolio,

retrieved from Reuters Datastream December 2018.

Further statistics on the specific 20 top stocks can be found in appendix C.

We also analyze how many trades and how many stocks are being held over the whole period. The LSTM model traded in 189 out of 235 stocks available and performed on average 2.85 trades each day, giving a daily portfolio turnover of 28.45% (10 trades would be maximum if we both bought 5 stocks and sold 5 stocks at every time step).

Moreover, we have analyzed our LSTM model's top 20 traded stocks over the whole period to understand further the stocks chosen by the model. We observe a range from small cap, ill-liquid stocks like Odfjell to some of the most traded stocks on OSE like Norsk Hydro and Marine Harvest. Of the top 20 stocks, 14 are listed today. Median daily volume turnover by value for these stocks were 64 496 for 2018, compared to 27 333 for all constituents on OSEBX, which is surprisingly high compared to what we expected. The median market capitalization

for these stocks are about the same as the median for OSEBX. The median percentage of free float of our selected stocks are quite similar to that of the OSEBX. We also see that the median bid-ask spreads of the selected stocks are lower than for the OSEBX, further implying that the stocks are more liquid.

To summarize, the stocks being picked can be characterized as especially bad performers over the last 5 days, with moderate volatility, lower beta than expected and higher turnover by value than the OSEBX median. It seems evident that the model follows a strategy that exploits short-term mean-reversion, while the stocks it trades in seems to be more liquid than expected.

5.3.4 Our findings relative to previous literature

Fischer and Krauss (2018) performed a similar analysis on the constituents of the S&P 500, and found a strong out-performance of the Deep Learning driven strategies from 1992-2015 with a Sharpe ratio of 5.8 before transaction costs. However, they found that there were no excess returns for the period 2010-2015. Our test period starts out in 1999 and considers a different stock universe, so the results are not directly comparable. Since the S&P 500 includes 500 stocks as opposed to the OSEBX averaging 63 stocks, strategies based on the S&P 500 are likely to take advantage of a larger variation between stocks and more certain predictions for the top K stocks, and thus likely to achieve a higher return and Sharpe ratio. We also find returns that are somewhat decreasing the closer we are to the present day, but as our analysis above shows the excess returns are still present over the last three years. This might indicate that the exploitable inefficiencies on the S&P 500 have been arbitraged away, while the potential still exists on the OSEBX.

6. A practitioners approach

In the previous chapter we followed the classic academic approach where we only included simple trading strategies and only took the direct transaction costs into account. However, this ignores the majority of the real-world transaction costs and as such we need to also take those into account to really understand the strategy performance. In this chapter, we include the effects of the bid-ask spread and seek to implement strategies that will overcome the significant cost levels. We also train models on returns adjusted by the bid-ask spread and other independent variables and compare their performance to those only trained on simple returns. Finally, we analyze which stocks are selected and how it differs from those found in the previous chapter.

6.1 Transaction cost analysis

As defined in the theory chapter, transaction costs can be decomposed into direct transaction costs (commissions) and indirect transaction costs (market impact, implementation shortfall). In this chapter, we also include the effects of market impact and implementation shortfall. We avoid implementation shortfall by trading at the bid/ask prices to ensure that our portfolio is the preferred portfolio (it is hard to simulate the implementation shortfall ex-post with our data set). Furthermore, our data set only includes the closing bid and ask prices, and does not include the order depth and volume allocated to the given bid and ask prices. Hence, the true market impact might be higher than what we calculate.

Direct transaction costs are set to 2.9 basis points, and we find the median relative bid-ask spread in our data set to be substantial at 102 basis points. We find a median bid-ask spread of 142 basis points from 1999-2008, and a median bid-ask spread of 75 basis points from 2009-2017. This is somewhat lower than Ødeegard's (2009) findings of a median bid-ask spread at Oslo Stock Exchange of 200 basis points from 2000-2008, likely due to OSEBX representing the most liquid companies listed on the OSE. The spread will lead to a very high median round-trip transaction cost of 107.8 (102 + 2.9 * 2) basis points.

6.2 Trading performance after total transaction costs

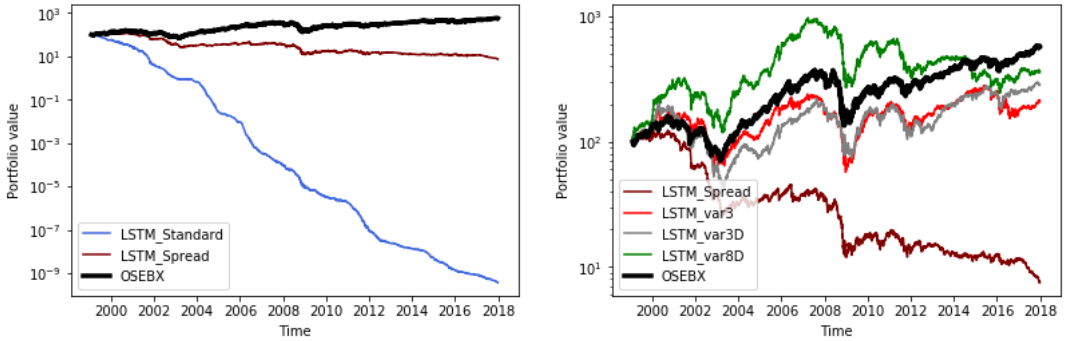
The best-performing model in the previous chapter had an annualized return of 89%, Sharpe ratio of 2.72 after direct transaction costs and a break-even transaction cost of 20.5 basis points. When bid-ask spreads are taken into account, with a median of 102 basis points, it is clear that this strategy is unlikely to be viable. We therefore train different LSTM models on returns adjusted for the actual bid-ask and other independent variables, as well as implementing more advanced strategies to reduce turnover and transaction costs.

6.2.1 Training on spread-adjusted returns

We consider five different LSTM models with the following independent variables:

- **LSTM_Standard:** Simple returns. Best performing model from previous chapter.
- **LSTM_Spread:** Spread-adjusted simple returns.
- **LSTM_Var3:** Spread-adjusted simple returns, volume and bid-ask spreads.
- **LSTM_Var3D:** Same as *Var3*, where the independent variables are de-noised.
- **LSTM_Var8D:** Same as *Var3D*, as well as 50- and 200-day moving average stock price, and de-noised USD/NOK fx rate, 3-year and 10-year government bond yields.

Figure 6.1: Trading performance after total transaction costs (K = 10)



(a) Simple vs spread-adjusted returns

(b) Additional independent variables

From the graph to the left in 6.1 we see that the model trained on the spread-adjusted simple returns by far outperform the model trained on simple returns, however still showing a negative annualized return of -12.7%. From the graph to the right, we see that additional independent variables improves the performance of the models. We see that the model with 8 denoised independent variables outperform the other models with an annualized return of 7%.

Table 6.1: Trading performance after total transaction costs (K = 10)

Annualized	**LSTM_standard**	**LSTM_spread**	**LSTM_var3**	**LSTM_var3D**	**LSTM_var8D**	**OSEBX**
Return	-0.7494	-0.1273	0.0396	0.0566	**0.0698**	0.1010
Standard deviation	0.2701	**0.2308**	0.2510	0.2504	0.2622	0.2266
Sharpe Ratio	-2.8945	-0.6924	0.0283	0.0964	**0.1423**	0.3022
Max drawdown	1.0000	0.9422	0.7621	0.7932	**0.7489**	0.6410
VaR 5%	-1.1937	-0.5070	-0.3732	**-0.3552**	-0.3615	-0.2717

By analysing table 6.1 the most important insight is the fact that none of the models beat the OSEBX in terms of risk-adjusted return on average. The poorest performing models are not worth investigating further. The *LSTM_standard* and *LSTM_spread* are declining over the whole period, and the max drawdown measurement will therefore count this as one long drawdown. On the other side, the LSTM_var8D is performing *relatively* good. Due to lower return and higher volatility compared to the OSEBX, the Sharpe is obviously lower. Despite this, from the graphical inspection on the previous page, we see that this model outperformed the OSEBX from 1999 until 2014.

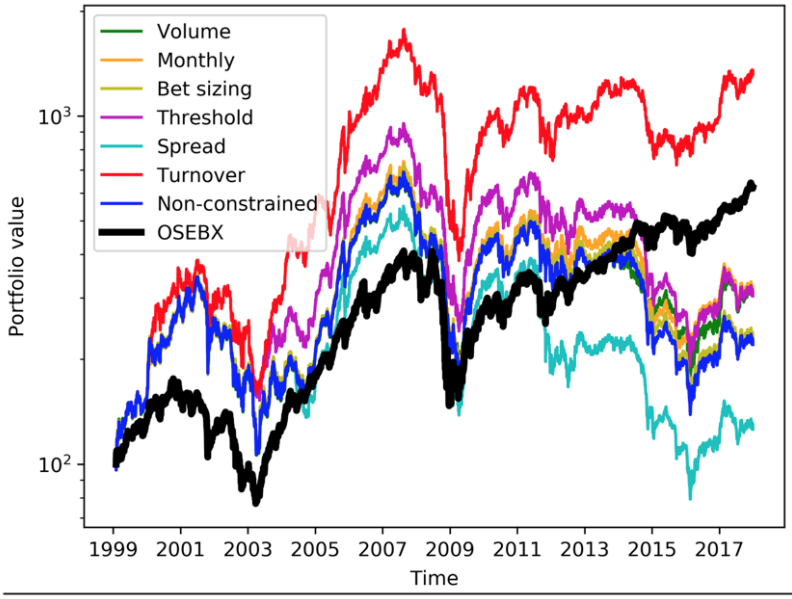
6.2.2 Imposing constraining strategies

We move forward with the model with eight de-noised independent variables (LSTM_var8D) and implement different constraining strategies in order to see if we can improve performance. The parameters for the constraining strategies are chosen as they gave the best performance, and are outlined below:

- **Volume:** only allows trading in the stocks that have had an average trading volume over the last ten days, that is over the 28th percentile of trading volume that day
- **Monthly:** rebalancing is only performed every 20th trading day.
- **Bet sizing:** adjusts the weights based on the standardized predicted probabilities.

- **Threshold:** only invests when the predicted probability is above 77%.
- **Spread:** only invests if the average spread over the last ten trading days is less than 100 basis points.
- **Turnover:** makes sure that stocks are not removed from the portfolio if they are part of the predicted top 15 (K*3) stocks the following day.

Figure 6.2: Trading performance of constraining strategies (K = 5)

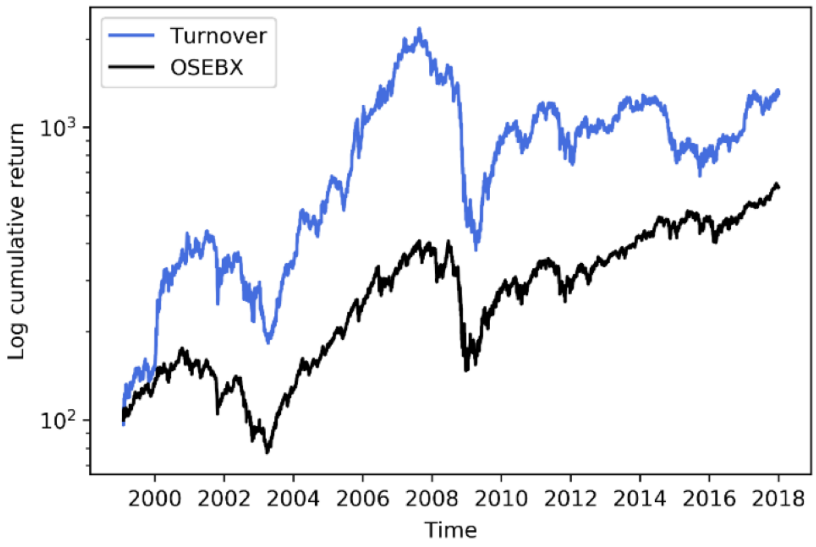


	Return	Std	Sharpe Ratio	Max drawdown	VaR 5%
Volume	-0.0608	0.2957	0.0957	0.7479	-0.4256
Monthly	0.0617	0.2965	0.0986	0.7551	-0.426
Bet sizing	0.0448	0.2922	0.0419	0.7959	-0.4359
Threshold	0.0603	0.2860	0.0972	0.7991	-0.4101
Spread	0.0122	0.2995	-0.0677	0.8562	-0.4804
Turnover	0.1446	0.303	0.3700	0.8252	-0.3538
Non-constrained	0.0425	0.2946	0.0339	0.7991	-0.4421
OSEBX	0.101	0.2266	0.3022	0.6410	-0.2717

Figure 6.2 shows the performance of the different constraining strategies, as well as the benchmark OSEBX and the non-constrained strategy (*LSTM_var8D* from the previous section). We see that all the constraining strategies perform better than the non-constrained strategy, except for the spread-restricted strategy. This is likely due to the the fact that the model is already trained on returns adjusted for the bid-ask spread, and that further constraints of the bid-ask spread thus are redundant. We also tested a constraining strategy that only rebalanced the

portfolio weekly (every 5th trading day), but this did not outperform the monthly strategy. We clearly see that the *Turnover* constrained strategy outperforms all the other constraints, and is the only strategy that outperform the OSEBX benchmark over the whole period. When all constraints (except spreads) isolated improve the performance, it is reasonable to believe that combining them altogether will lead to even better performance. It does not make sense to combine the Turnover and Monthly strategies, as they both restrict turnover and thus essentially does the same. Therefore, we combine the volume, bet sizing, threshold and turnover strategies to one combined strategy. However, we find that the performance of this combined strategy (with different parameter combinations) does not outperform the turnover strategy on its own.

Figure 6.3: Trading performance after total transaction costs (K = 5)



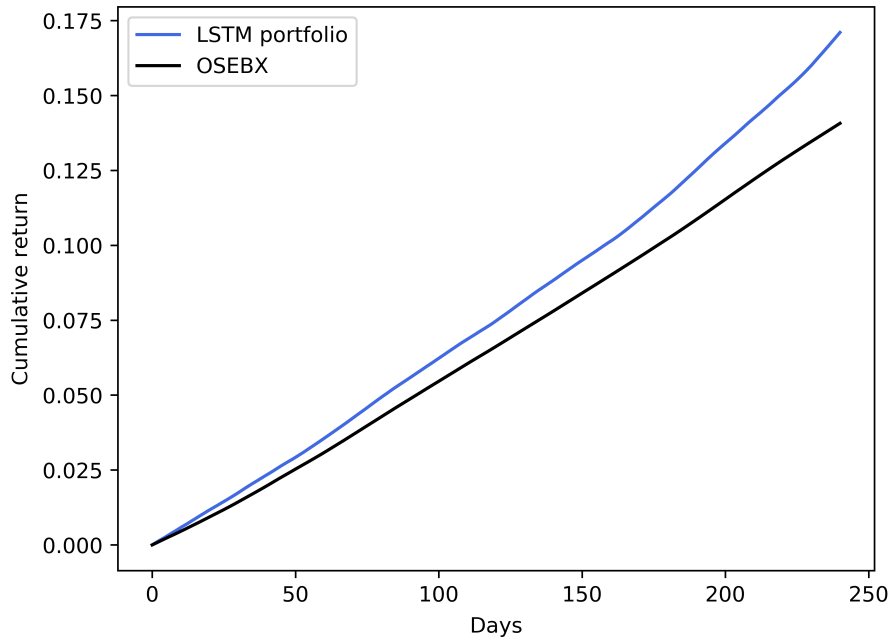
	Return	Std	Sharpe Ratio	Max drawdown	VaR 5%
Turnover	0.1446	0.303	0.3700	0.8252	-0.3538
OSEBX	0.101	0.2266	0.3022	0.6410	-0.2717

With an annualized return of 14.5% and a Sharpe ratio of 0.37, the *Turnover* strategy outperforms the OSEBX with a return of 10.1% and a Sharpe ratio of 0.30. We also see that the strategy seems to under-perform in recent years. Moreover, maximum drawdown and value at risk shows that the risk is higher for our portfolio than just holding the index portfolio (OSEBX). We follow Lopez De Prado (2018) and test whether the excess risk-adjusted returns are statistically significant by calculating the Probabilistic Sharpe ratio. We find that we can not reject the null hypothesis of no significant difference with a p-value of 0.44. Based on our

relatively extensive search of different predictive set-ups and constraining strategies, we can therefore not conclude that LSTM predictions seems to be profitable to trade on alone.

6.2.3 Portfolio analysis and stock selection

Figure 6.4: Accumulated return over the t = 240 lookback days the model used to choose which stocks to hold in portfolio at time t, with K = 5.



It is interesting to see if there are large differences between the choices of this portfolio compared to the portfolio found before total transaction costs where applied. When looking at the performance of the selected stocks prior to inclusion in the portfolio, we see that the stocks we invest in on average have performed better than the (equally-weighted) average of the OSEBX constituents over the last 240 days. More interesting is the fact that we do not observe the strong short-term mean-reversion as we observed in the previous chapter. This is likely due to the fact that the previous model found exploitable and profitable short-term mean-reversion patterns before taking the total transaction costs into account, but when the bid-ask spread is included these patterns are not profitable to act upon. In this case, it might seem as if the model follows a momentum strategy and pick stocks that have outperformed in the last period. When looking at the properties of the returns of the stocks we invest in in the previous 240 days, we see that the average standard deviation is 27.9% and average Beta is 1.17. Compared to the

average standard deviation of the OSEBX constituents at 20.3%, we see that our model now prefers stocks with high volatility where it previously selected stocks with a similar standard deviation of the OSEBX.

Table 6.2: Descriptive statistics for the stocks selected before and after spreads.

	Before spreads	After spreads	OSEBX
Stocks traded total	192	61	235
Average daily portfolio turnover	46.75%	0.3%	-
Average cumulative return (t-240 to t)	7.45%	17.1%	14.07%
Average standard deviation (t-240 to t)	19.52%	27.9%	20.29%
Average Beta	0.8	1.17	1
Median market cap	27 052	33 749	11 458
Median turnover by value	31 914	113 306	27 333
Median free float %	60	63	62
Median bid-ask spread (bp) %	82	65	102

Cumulative return and standard deviation based on equally-weighted average of the OSEBX constituents.

Based on model $LSTM\_a$ and $LSTM\_var8D$ with a Turnover constraining strategy (K = 5).

with a Turnover constraining strategy.

Market cap, turnover and free float based on the top 20 stocks held by LSTM portfolios,

retrieved from Reuters Datastream December 2018.

Further statistics on the specific 20 top stocks can be found in appendix C.

From the table above we see that this model is heavily constrained and only trades in a total of 61 stocks of the 235 possible candidates, and has a significantly reduced turnover with an average daily portfolio turnover of 0.3%. When looking deeper into the stocks selected, we see that the median daily traded volume by value of the top 20 stocks invested in is 113 306, more than four times the median of the OSEBX constituents at 27 333. This is also more than three times higher the median traded volume under the optimal portfolio from the previous chapter. From the top 20 stocks held we observe many liquid stocks with high market capitalization, with a median market capitalization that is far above the OSEBX average. We also see that the model invests in stocks with a lower median bid-ask spread than the model from the previous chapter.

To summarize, we see that our model now has a strongly reduced turnover and does not exploit the same short-term mean-reversion inefficiency as we found in the previous chapter. We also see that the strategy invests in liquid, volatile stocks with high market cap and turnover by value, as well as relatively low bid-ask spreads. This is in line with what we would expect given that the model now have been directly penalized for the bid-ask spread during training, as well as portfolio turnover being explicitly constrained.

7. Discussion

Throughout this thesis we have gone through some technically challenging topics and theories. Despite our effort to account for all possible transaction costs and effects, some questions still stands unanswered. In this chapter we will elaborate on the the general challenges of doing backtesting, robustness of our method, our results in light of the Efficient Market Hypothesis and finally discuss the inherent value of our findings and the future of Deep Learning in financial markets. We will also elaborate on relevant areas of further research.

7.1 Backtesting challenges

7.1.1 Transaction costs

Despite the model taking into account the direct transaction cost associated with each trade and the historical bid-ask spread, it is not possible to account for the actual transaction costs that had occurred if this strategy were implemented - then we would have to travel back in time and execute the strategies. Our data set does not give us the volume available at different bid-ask prices, so we can not test how the portfolio size would affect returns due to price impact. Another issue is the fact that we have used an average direct transaction cost to illustrate the direct commissioning for trades over the whole period. Most likely this cost was higher in the early 2000s than today due to physical brokers compared to today's electronic solutions. This will lead to an overestimation of returns early on and push our annualized return and Sharpe ratio higher than it actually would have been.

7.1.2 Problem of false discoveries

We have to be very careful when evaluating how good our strategies and models are. Say we try 100 different predictive models, at least one of those is doomed to show very good results, just by chance. This makes it really hard to determine whether or not good backtesting results are pure luck or actually promising strategies that will perform well in the future. How to address this challenge is perhaps the most fundamental question in quantitative finance (Lopez de Prado,

2018). If there were a quick solution, hedge-funds, investment firms and individuals would yield high returns with certainty, as they would know that a good backtested strategy would yield good results in the future.

7.2 Efficient Market Hypothesis in light of our results

Our model would not be able to generate an excess risk-adjusted return if the weak form of the Efficient Market Hypothesis holds. We see that the LSTM model has significant predictive properties with a directional accuracy of 53.17%, and that strategies utilizing these predictions with and without accounting for direct transaction costs generate significant excess returns. However, when taking total transaction costs into account the performance in terms of Sharpe ratio is *not* significantly better than OSEBX. Hence, we can state that our model is able to extract patterns from historical prices and predict future returns with a certain confidence, but it does not seem to be profitable to trade on the predictions alone due the substantial transaction costs on the OSEBX. Therefore, we can not conclude that the weak-form EMH does not hold on the OSEBX.

We see that the performance stagnates somewhat in recent years. This can indicate that the efficiency of the OSEBX constituents have increased over time, and might be due to increased trading that implement similar predictive models and strategies. Another more likely reason is that macroeconomic (systematic) risk have dominated from 2008 going forward and have somewhat reduced the performance of our models as they leverage the firm-specific (unsystematic) risk. An interesting finding is the fact that Fischer and Krauss 2018 found the same pattern for the S&P 500, only that the diminishing returns started much earlier than for the OSEBX. This seems to strengthen our initial assumption that the OSEBX is less efficient than the S&P 500, which consists of more liquid stocks.

7.3 The inherent value of our predictive model

While it does not seem to be profitable to trade on the predictions alone, we see that the predictive performance of our model can make it suited to complement other, more complex strategies in order to increase their profitability. Quantitative funds are often driven by optimization models that leverage signals generated by different types of models. A typical use-case for our prediction model could therefore be as a signal generator for a larger trading strategy optimization model. It might also be used as a supporting tool for portfolio managers and professionals to guide them in their decision-making process, where other predictive models might currently be used.

7.4 How Deep Learning predictions might affect financial markets

We ask ourselves whether Deep Learning predictions might improve market quality in terms of price efficiency and liquidity. There is one very appealing reason for why it might improve market quality. One of the most fundamental assumptions in finance is that investors act rational. This assumption has proven over time to not always hold, people are affected by feelings and execute trades based on it. An algorithm on the other hand can be seen as perfectly rational since it will never act on feelings.

To understand why it might lower market quality, a thought experiment might be practical. Imagine a stock exchange were absolutely all trades were executed by algorithms using the same perfect prediction model. If markets had a predictive model that could predict the price tomorrow with 100% certainty, all investors would act on it today and the prediction would turn out to be wrong. Hence, one might say that predicting the predictable causes the predictable to become unpredictable (Wright, 2018). A model that can predict the future with 100% certainty is of course not possible, but the dynamics of the thought experiment stands. Moving on, a danger with similar algorithms doing most trades is that they are likely to perform the same trades at the same time. This will in turn lead to reduced liquidity if every market participant wants to perform the same trade (e.g. buy), and no market participant want to be on the opposing side of the trade (e.g. sell).

7.5 Further research

We see that the emerging field of financial Deep Learning holds much promise, and see many areas of further research within the field. Firstly, one can do more research on refining the predictive models used. Other Deep Learning algorithms and models might be implemented, and deeper LSTM models with more layers and more complex structures might be tested. One of the key limitations of LSTM networks are that they are very computer intensive, and hyper-parameter optimization is thus a very time-consuming task. Further research might perform a more thorough hyper-parameter optimization that go further than our ad-hoc experimental approach.

Secondly, within financial time series there is a large space of possible assets to predict and different independent variables to be used for predictions. One might include other independent variables outside historical stock prices, as well as perform different de-noising transformations to the independent variables to reduce the noise that is fed to the predictor. Deep Learning predictions might also be applied to other financial markets which are expected to be less efficient than the OSEBX, or to markets that have a lower bid-ask spread and thus lower transaction costs.

The third, and most directed towards the actual trading viability is to implement a model where turnover, liquidity and transaction costs are being punished within the model instead of reducing the signal as we do. One way to do this is through reinforcement learning, where an agent iteratively makes trading decisions in a market environment and are rewarded based on the performance of those decisions.

8. Conclusion

Long Short-Term Memory (LSTM) networks are a promising Deep Learning technique for sequence learning inherently suitable for financial time series predictions. We therefore predict daily out-of sample directional movements of the constituent stocks of the Oslo Stock Exchange Benchmark Index (OSEBX) from 1999-2017 using LSTM networks, benchmarked against other machine learning and econometric techniques. Our primary independent variable is historical stock returns, and our approach thus implicitly tests the weak form of the Efficient Market Hypothesis on the OSEBX.

We test the predictive performance of the different models, before evaluating the performance of trading strategies based on the predictions. First we follow what seems to be the academic standard within the field of evaluating performance prior to transaction costs and after direct transaction costs. Secondly, we take a practitioners approach and seek to account for total transaction costs through taking the bid-ask spread into account, as well as implement advanced strategies to limit the impact of the high transaction costs. We also shed light into the black-box nature of our models to understand which stocks are selected and what patterns are exploited.

Our results unambiguously show that the LSTM model outperforms all benchmark models in terms of predictive performance with a directional accuracy of 53.17% . When testing simple long trading strategies on top of the predictions, we find that the LSTM model outperforms all other models with an annualized return of 106% and a Sharpe ratio of 3.25 prior to transaction costs from 1999 - 2017. From a risk perspective, our model inherits higher volatility and has about the same risk in terms of maximum drawdown as the OSEBX. After direct transaction costs of 2.9 basis points, we find an annualized return of 89% and a Sharpe ratio of 2.72. In comparison, the OSEBX had a Sharpe ratio of 0.30 over the same period. While seeing somewhat diminishing excess returns in the last years, the excess returns are still present in the last three years, which differs from similar studies on other stock universes where the excess returns seems to be arbitraged away in recent years. We see that the model invest in stocks with a diminishing return in the 90 days prior to investing, with a volatility similar to the OSEBX constituents and average beta level of 0.88. In the last days prior to stock selection, the selected stocks on average have a negative return of -3.4%. Hence it seems that the model

follow a short-term mean-reversion strategy.

When total transaction costs are taken into account we see that the excess return is lost in the bid-ask spread. We therefore train an LSTM model on spread-adjusted returns and other independent variables, as well as implementing advanced strategies. This leads to a modest Sharpe ratio of 0.37 over the whole period, which is not statistically significant higher than that of the OSEBX. The risk of our model is also higher than the OSEBX with both higher volatility, maximum drawdown and value at risk. Moreover, when analyzing which stocks this model selects we can not observe the same short-term mean reversion as we did for the previous model. The model now prefers historically over-performing stocks with higher volatility, higher liquidity and lower bid-ask spreads compared to earlier. This is in line with what we would expect given that the model now have been directly penalized for the bid-ask spread during training, as well as portfolio turnover being explicitly constrained.

Even though the trading performance after total transaction costs is not significantly better than the OSEBX, we see that LSTM networks have significant predictive properties far beyond the predictive methods we compare it to. We believe that while it might not be profitable to trade on the predictions alone, it can make a great tool for investment professionals to complement other trading strategies. Based on our findings we can conclude that our model finds inefficiencies on the OSEBX, but that it is not possible to generate an excess risk-adjusted return by exploiting those inefficiencies. Hence, we can not conclude that the weak form of the Efficient Market Hypothesis does not hold for the OSEBX constituents.

Bibliography

Arne Ødegaard, B. (2009, 1). Hva koster det å handle aksjer på Oslo Børs? *Praktisk økonomi & finans*. Retrieved from http://www.idunn.no/pof/2009/01/art01AU

Brodin, M., & Abusdal, Ø. (2008). *An empirical study of serial correlation in stock returns : cause-effect relationship for excess returns from momentum trading in the Norwegian market* (Tech. Rep.). Retrieved from https://brage.bibsys.no/xmlui/handle/11250/168153

Brownlee, J. (2016). *Overfitting and Underfitting With Machine Learning Algorithms.* Retrieved from https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

Chollet, F. (2017). *Deep learning with Python.* Manning Publications.

Fama, E. F. (1970, 5). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, *25*(2), 383. Retrieved from https://www.jstor.org/stable/2325486?origin=crossref doi: 10.2307/2325486

Fischer, T., & Krauss, C. (2018, 10). 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

Gal, Y., & Ghahramani, Z. (2016). *A Theoretically Grounded Application of Dropout in Recurrent Neural Networks* (Tech. Rep.). University of Cambridge.

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

Hassabis, D. (2016). *What we learned in seoul with AlphaGO.* Retrieved from https://blog.google/technology/ai/what-we-learned-in-seoul-with-alphago/

Hochreiter, S., & Schmidhuber, J. (1997, 11). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Krauss, C., Do, X. A., & Huck, N. (2017, 6). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, *259*(2), 689–702. doi: 10.1016/j.ejor.2016.10.031

Kryzanowski, L., Galler, M., & Wright, D. W. (1993, 7). Using Artificial Neural Networks to Pick Stocks. *Financial Analysts Journal*, *49*(4), 21–27. doi: 10.2469/faj.v49.n4.21

Lopez de Prado, M. M. (2018). *Advances in financial machine learning.* John Wiley & Sons, Incorporated.

Luo, Y., Wang, S., Alvarez, M., Jussa, J., Wang, A., Rohal, G., ... Zhao, Z. (2014). *Seven Sins of Quantitative Investing* (Tech. Rep.). Deutsche Bank. Retrieved from `http://eqindex.db.com/gqs`

Malkiel, B. G. (1995). Returns from Investing in Equity Mutual Funds 1971 to 1991. *Journal of Finance*, *50*(2), 549–72.

Marr, B. (2018). *10 Amazing Examples Of How Deep Learning AI Is Used In Practice?* Retrieved from `https://www.forbes.com/sites/bernardmarr/2018/08/20/10-amazing-examples-of-how-deep-learning-ai-is-used-in-practice/#18ea0887f98a`

Mikelsen, S., & Andersen, A. C. (2012). *A Novel Algorithmic Trading Framework Applying Evolution and Machine Learning for Portfolio Optimization* (Tech. Rep.). Retrieved from `https://brage.bibsys.no/xmlui/handle/11250/266347`

Nordnet. (2018). *Active Trader - Nordnet.* Retrieved from `https://www.nordnet.no/tjenester/kontotyper/active-trader.html`

Olah, C. (2015). *Understanding LSTM Networks – colah's blog.* Retrieved from `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

Oslo Stock Exchange. (2018). *Hovedindeksen - Kursoversikt.* Retrieved from `https://www.oslobors.no/markedsaktivitet/#/details/OSEBX.OSE/overview`

Palaniappan, V. (2018). *Using Machine Learning to Predict Stock Prices.* Retrieved from `https://medium.com/analytics-vidhya/using-machine-learning-to-predict-stock-prices-c4d0b23b029a`

Poterba, J., & Summers, L. (1987, 8). *Mean Reversion in Stock Prices: Evidence and Implications* (Tech. Rep.). Cambridge, MA: National Bureau of Economic Research. doi: 10.3386/w2343

Shubham Panchal. (2018). *Artificial Neural Networks — Mapping the Human Brain.* Retrieved from `https://medium.com/predict/artificial-neural-networks-mapping-the-human-brain-2e0bd4a93160`

Skymind. (2018). *A Beginner's Guide to LSTMs and Recurrent Neural Networks.* Retrieved from `https://skymind.ai/wiki/lstm#vanishing`

Wright, M. (2018). *Why Stock Predicting AI Will Never Take Over the World.* Retrieved from `https://blog.usejournal.com/why-stock-predicting-ai-will-never-take-over-the-world-b1b411decc21`

List of Figures

List of Tables

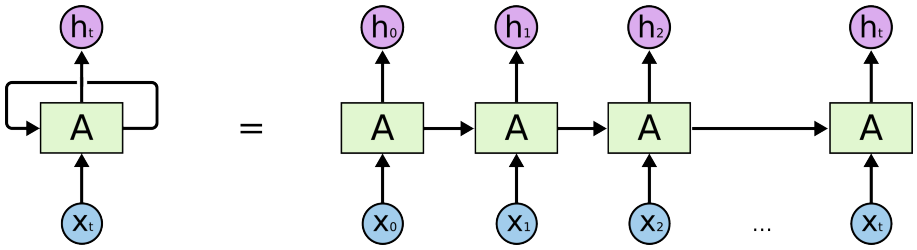
9. Appendix

9.1 Appendix A: Theory

Recurrent neural networks and Long Short-Term Memory networks

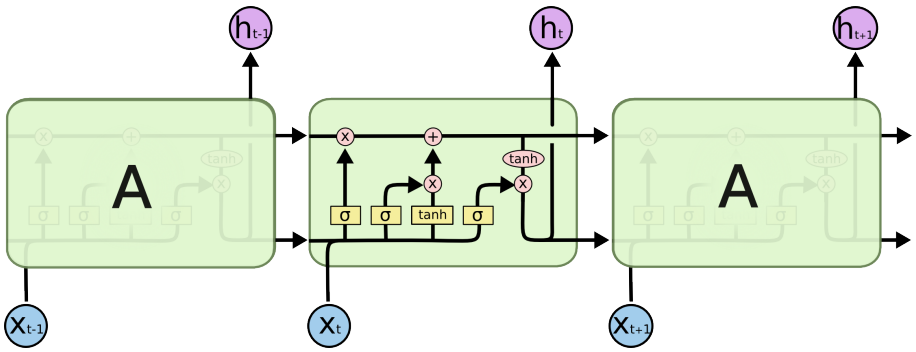
In this section, we go deeper into the intuition behind recurrent neural networks and long short-term memory networks. Recurrent Neural Networks (RNNs) can be thought of as multiple copies of the same neural network over time where each network passes a message to its successor (Olah, 2015). RNNs are thus able to capture dependencies over time and sequences, and have shown incredible success within areas such as speech recognition, language modelling and translation (Olah, 2015).

Figure 9.1: Illustration of a standard RNN. Source: Olah, 2015



Standard RNNs work very well for short-term dependencies, but struggle with long-term dependencies as the gradients are multiplied backwards through the different looped network resulting in exploding gradients or vanishing gradients (Skymind, 2018). An exploding gradient will lead to oscillating weights and unstable learning, while a vanishing gradient will imply that nothing can be learned in acceptable time (Hochreiter & Schmidhuber, 1997). Long Short-Term Memory networks were therefore proposed by Hochreiter and Schmidhuber in 1997 as a solution to the vanishing/exploding gradient problem, and they are therefore superior in learning long-term dependencies.

Figure 9.2: Flow through an LSTM Network. Source: Colah, 2015



Where the standard RNN cell contains a single neural network layer, an LSTM network cell has four neural networks interacting together. The cell contains a cell state C_t, a forget gate f_t, an input gate i_t and an output gate o_t. A gate is the combination of a neural network layer and a point wise multiplication operation, and a way to determine what information to let through (Olah, 2015).

The cell state C_t runs as a conveyor belt through the cells, and contains the data that is memorized from previous cells. The memory in the cell state is updated through each cell through the forget gate and input gate (Olah, 2015). The forget gate layer f_t determines which part of the cell state (memory) to forget:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{9.1}$$

The input gate layer consists of two neural network gate layers, one sigmoid layer i_t that determines which values in the cell state we want to update, as well as a tanh layer $\widetilde{C}_t$ that produces candidate values to add to the cell state (Olah, 2015):

$$\begin{aligned} i_t &= \sigma(W_i * [h_{t-1}, x_t] + b_i) \\ \widetilde{C}_t &= tanh(W_C * [h_{t-1}, x_t] + b_C) \end{aligned} \tag{9.2}$$

The gates have now decided how much to keep from memory f_t (forget gate layer), as well as how much we will add to the memory i_t (sigmoid input gate layer) and which candidate values to be added $\widetilde{C}_t$ (tanh input gate layer). The cell state C_t is therefore update as follows, with $\circ$ denoting the Hadamard product:

$$C_t = f_t \circ C_{t-1} + i_t \circ \ C_t) \tag{9.3}$$

The last step involves determining what values to output as our $\hat{y}$. Our output will be determined by our current input values x_t, the previous output value $\hat{y}_{t-1}$ as well as the current cell

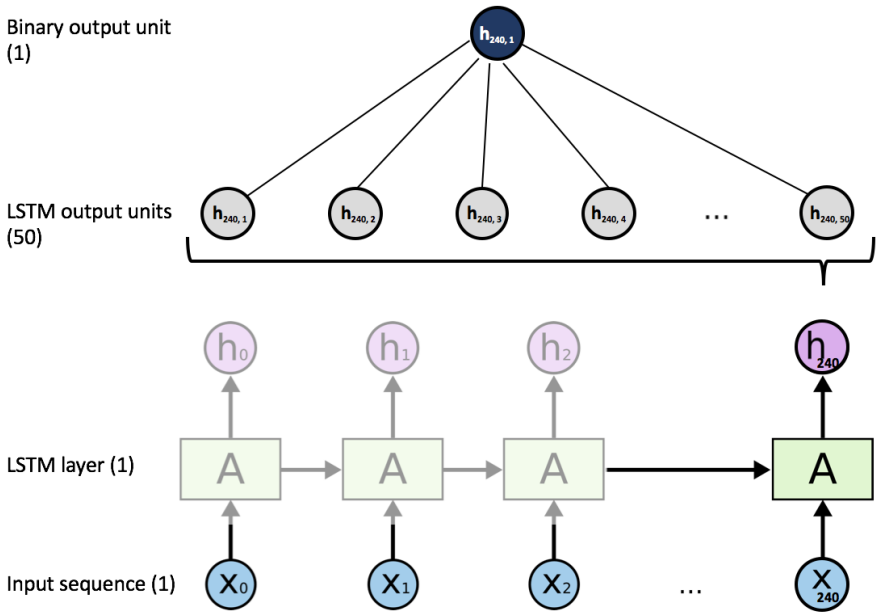
state C_t. The sigmoid (σ) layer determines which parts of the cell state we will output, while the cell state goes through a tanh transformation to get values between -1 and 1 (Colah, 2015):

$$\begin{aligned} o_t &= \sigma(W_o * [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \circ tanh(C_t) \end{aligned} \tag{9.4}$$

9.2 Appendix B: Methodology

Our LSTM architecture

Figure 9.3: LSTM Network architecture.



Our primary independent variable are sequences consisting of 240 stock returns. The LSTM network will then create 240 cells (one for each individual stock return) that each go through the aforementioned LSTM operations, and sends the memory cell state and output forward to the next cell. We use 50 hidden units, meaning that each cell produces 50 output values. At the end of the sequence, the final cell produces the final output (50 values). Both since our output is a binary classification problem, and to add depth to the model, these 50 output values is then forwarded through a normal neural network layer that has one output neuron with a sigmoid activation function, predicting either 1 or 0. The loss function is then calculated as the binary cross-entropy, and all the parameters in the network are updated according to their individual contribution to the loss function.

Measures to reduce overfitting of our models

Firstly, we split between training, validation and test set to avoid overfitting to the training set or overfitting to the test set. Secondly, after each epoch, training the model on all observations while adjusting parameters, we compute the loss function on the training set and the validation set. Once the loss function calculated on the validation set predictions stops decreasing, the iteration stops and no more epochs are run. This process is called early stopping, and makes sure that we do not overfit to the training observations, and improves the performance on unseen data. We use an early stopping patience of 10, meaning that when validation loss has not improved over 10 epochs, training is stopped and the best parameters (of those 10) are used for test set prediction. Thirdly, we use dropout regularization of 10% within the recurrent layer as recommended by Gal and Ghahramani (2016), meaning that a random fraction of the inputs are being dropped, reducing the risk of over-fitting further.

Another way of reducing the risk of overfitting is controlling the relationship between the number of observations and number of parameters in the network. The number of parameters trained in the LSTM layer can be calculated as follows, where h denotes the number of hidden units and i denotes the number of independent variables (Fischer & Krauss, 2018):

$$4hi + 4h + 4h^2 = 4(h(i+1) + h^2) \tag{9.5}$$

In our case we have one independent variable and 50 hidden units, yielding 10 400 parameters. We also have a normal neural network layer connected to the LSTM layer with 50 weights and one bias term, yielding a total of 10 451 parameters to be trained. We have on average 32 000 training observations per study period (average 63 constituents * (750 training days - 240 days lookback) Hence we end up with roughly 3 observations per parameter (32 000 / 10 451) to be trained.

Hyper-parameters of all our models

Here we outline the hyper-parameter values for all the predictive models we have presented in the thesis. All models used in chapter 6 have the same hyper-parameters as the *LSTM_a* model shown, and the only difference is which independent variables are used. All hyper-parameter values that are not mentioned below are set to default values. Benchmark models follow the same underlying methodology as the LSTM models, outlined in our methodology chapter. The

LSTM_i model has such a low number of hidden units (10) in order to maintain a reasonable relationship between the number of observations and trained parameters. Since it fits to stocks individually it only has 510 training observations. The *LSTM_f* has a slightly higher number of hidden units (15) since it also includes additional independent variables.

Table 9.1: Hyper-parameters for the LSTM models

	LSTM_i	**LSTM_a**	LSTM_d	LSTM_f
Number of layers	1	1	2	1
Hidden units layer 1	10	50	30	15
Hidden units layer 2	-	-	20	-
Output neurons	1	1	1	1
Activation function (output neuron)	sigmoid	sigmoid	sigmoid	sigmoid
Optimizer	RMSprop	RMSprop	RMSprop	RMSprop
Epochs	1000	1000	1000	1000
Batch size	32	32	32	32
Learning rate	0.001	0.001	0.001	0.001
Dropout regularization	0.05	0.1	0.1	0.1

Table 9.2: Hyper-parameters for the benchmark models

	Logistic	RAF	SVM
C	0.01	-	100
Solver / Kernel	LBFGS	-	RBF
Number of estimators	-	1000	-
Max depth	-	8	-
Criterion	-	Entropy	-
Gamma	-	-	0.01

9.3 Appendix C: Analysis and results

Statistics top 20 traded stocks - optimal model before transaction costs

	Ticker	Number of trades	Market value (MNOK)	Turnover by value (000s NOK)	Percentage of free float (%)	Beta
Royal Carribean Cruises[1]	RCL	471				
Olav Thon Eiendomsselskap	OLT	445	15 349	3 496	28	0.458
Odfjell[2]	ODF	445	2 004	575	55	0.151
Ekornes[1]	EKO	416				
Orkla	ORK	379	72 507	134 746	68	0.195
Storebrand	STB	373	31 821	120 325	88	0.467
Avantor[1]	AVA	359				
Profdoc	PRO	351				
Norsk Hydro	NHY	341	86 422	326 269	59	0.742
Wilh. Wilhelmsen Holding ser. B	WWIB	332	1 946	2 024	46	0.616
Hafslund	HNB	331				
REC Silicon	REC	330	1 820	32 599	77	0.683
Steen & Strøm[1]	SST	326				
Marine Harvest	MHG	318	104 653	325 722	79	0.392
Kongsberg Automotive[2]	KOA	313				
EVRY	EVRY	312	11 458	6 825	41	-
Norwegian	NAS	311	10 042	126 997	70	0.742
Petroleum Geo-Services	PGS	305	6 601	96 394	81	1.521
DNO	DNO	302	16 615	106 600	60	1.023
ATEA	ATEA	300	13 021	13 499	61	0.356
Average			**26 733**	**92 577**	**58**	**0.525**
Median			**12 240**	**64 496**	**61**	**0.463**
OSE[3]			1 451	1 969	63	
Median OSEBX[4]			11 458	27 333	62	

Data retrieved from Reuters Datastream 04.12.2018

1) Not on OSE as of 04.12.2018

2) On OSE, but not constituent of OSEBX

3) Average of all constituent stocks as of 04.12.2018

4) Average of all constituent stocks as of 04.12.2018

Table 9.3: Data for 20 top traded stocks over period from 1999-2017. Turnover is calculated as average daily turnover over the last year.

Statistics top 20 traded stocks - optimal model trained on spreads

	Ticker	Number of trades	Market value (MNOK)	Turnover by value (000s NOK)	Percentage of free float (%)	Beta
Norsk Hydro	NHY	1500	86 422	326 269	59	0.742
Tomra Systems	TOM	1000	33 749	37 156	66	0.552
Norwegian Air Shuttle	NAS	1000	10 042	126 997	70	0.742
Prosafe	PRS	1000	1 451	162	63	0.845
Telenor	TEL	750	242 055	303 415	41	0.499
Tandberg[1]	TAA	750				
BWG Homes[1]	BWG	750				
ATEA	ATEA	750	13 021	13 499	61	0.356
StepStone[1]	STP	728				
Orkla	ORK	500	72 507	134 746	68	0.195
Equinor	EQNR	500	690 768	713 919	33	1.547
Opticom[1]	OPC	500				
Schibsted ser. A	SCHA	500	33 805	65 225	87	0.376
Marine Harvest	MHG	500	104 653	325 722	79	0,392
Austevoll Seafood	AUSS	500	23 799	38 555	44	0,171
TGS	TGS	500	25 8474	113 306	90	1.507
SensoNor[1]	SEN	500				
DSND Subsea[1]	SFJ	479				
Profdoc[1]	PRO	386				
Apptix[2]	APP	250	98	3 851	38	0.845
Average			**108 701**	**178 133**	**59**	**0.699**
Median			**29 798**	**120 151**	**62**	**0.647**
Median OSE[3]			1 451	1 969	63	
Median OSEBX[4]			11 458	27 333	62	

Data retrieved from Reuters Datastream 04.12.2018

1) On OSE, but not constituent of OSEBX

2) Not on OSE as of 04.12.2018

3) Average of all constituent stocks as of 04.12.2018

4) Average of all constituent stocks as of 04.12.2018

Table 9.4: Data for 20 top traded stocks over period from 1999-2017. Turnover is calculated as average daily turnover over the last year.