Long Short-Term Memory Neural Network for Financial Time Series

Carmina Fjellström
Department of Mathematics
Uppsala University
S-751 06 Uppsala, Sweden
carmina.fjellstrom@math.uu.se

2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10020784

***Abstract*—Performance forecasting is an age-old problem in economics and finance. Recently, developments in machine learning and neural networks have given rise to non-linear time series models that provide modern and promising alternatives to traditional methods of analysis. In this paper, we present an ensemble of independent and parallel long short-term memory (LSTM) neural networks for the prediction of stock price movement. LSTMs have been shown to be especially suited for time series data due to their ability to incorporate past information, while neural network ensembles have been found to reduce variability in results and improve generalization. A binary classification problem based on the median of returns is used, and the ensemble's forecast depends on a threshold value, which is the minimum number of LSTMs required to agree upon the result. The model is applied to the constituents of the smaller, less efficient Stockholm OMX30 instead of other major market indices such as the DJIA and S&P500 commonly found in literature. With a straightforward trading strategy, comparisons with a randomly chosen portfolio and a portfolio containing all the stocks in the index show that the portfolio resulting from the LSTM ensemble provides better average daily returns and higher cumulative returns over time. Moreover, the LSTM portfolio also exhibits less volatility, leading to higher risk-return ratios.**

***Index Terms*—neural networks, LSTM, financial forecasting, time series analysis**

I. Introduction

Prediction of asset prices has long been a central endeavor in mathematical finance and econometrics. Financial time series, however, are notoriously challenging to analyze because of their nonstationarity, nonlinearity, and noise, resulting from the irrational human behavior that drive the data. In the past, methods used are those of traditional nature such as ones based on Autoregressive Integrated Moving Average (ARIMA), Generalized Autoregressive Conditional Heteroskedasticity (GARCH), as well as other stochastic volatility models (see, for example, [1]–[4]). The use of these models often entail making assumptions about the data, its underlying distribution, and the different processes affecting it. Because of these assumptions, these methods often generalize poorly for new, out-of-sample data, even though they fit the current data well and do provide valuable insights into the time series [5]. Recently, developments in machine learning and neural networks have given rise to non-linear time series models that are increasingly being adapted for financial applications. Support vector machines (SVM), restricted Boltzmann machines (RBM), random forests, gradient boosted trees (GBM), and multilayer perceptrons (MLP) are just some examples of the machine learning models that are being used [6]–[10]. Amongst these models, one particular type of machine learning architecture, a recurrent neural network (RNN), has been shown, compared to others, to be better suited for sequential data such as time series. The suitability is due to the feedback loops in RNNs that allow them to use information not just from the current input, but also from past inputs. This is unlike other neural networks that, in general, process inputs as separate, independent data points. There is however, one major problem with RNNs - their inability to learn long-term dependencies due to the infamous vanishing gradient problem [11]–[13]. To address this, the long short-term memory (LSTM) was introduced.

In this paper, an LSTM model is used. A type of RNN, an LSTM also has feedback loops, but moreover, it can also regulate its memory by using a gating mechanism that learns which information to keep, to pass on, and to forget. It is widely used and has been shown to have excellent predictive capabilities in natural language processing, handwriting recognition, image recognition, and image captioning. See, for example, [14]–[18]. In finance, LSTMs have been increasingly used for time series analysis. For example, applications for price predictions on major stock market indices all over the world such as the S&P500, Shanghai's SSE Index, India's NIFTY 50, and Brazil's Ibovespa are studied in [11], [19]–[22]. In addition, Tsantekidis et al. [23] used LSTMs on Finnish companies to predict price movements through high frequency trading data on a limit order book. Apart from predicting prices, Yeung et al. [24] employed LSTMs to detect jumps in the values of different stock market indices, and Xiong et al. [25] applied LSTMs on the S&P500 and Google domestic trends data to forecast price volatility. These are just some examples of LSTM implementations on financial time series showing the neural network to produce promising results. Comparisons with other methods have also been made. Siami-Namini et al. [26], for example, compared LSTM with ARIMA for time series forecasting. They not only used data from major exchanges such as the Dow Jones Industrial Average (DJIA) and Nasdaq Composite, but also other economic time series such as the M1 money supply, currency exchange indices, and transportation data. Results from their study show that LSTM forecasts have significantly less root mean square

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 01,2025 at 18:01:22 UTC from IEEE Xplore. Restrictions apply.

error (RMSE) than those from ARIMA. Fischer and Krauss [27] applied LSTM to S&P500 data for price prediction and compared the results with random forest, a standard deep neural network (DNN), and logistic regression. Their findings indicate that LSTM does indeed have higher accuracy than the other approaches, and that LSTM-based portfolios offer higher returns and lower volatilities. Di Persio and Honchar [10], on the other hand, compared LSTM and MLP with their own method, which is an ensemble of wavelets and a convolutional neural network (CNN). Although they reported that their method appears to be superior, the results are very close to those from LSTM [21].

Most of the literature on the application of LSTM on financial time series have been made on major market indices such as the DJIA and S&P500. In this paper, an LSTM is applied to Stockholm's OMX30 to explore what advantages an LSTM-based approach can provide for a smaller, less perfect market. The method used is inspired by both Fischer and Krauss [27] and Barra et al. [28]. LSTMs are applied to sequences of daily returns; however, as opposed to applying the network to the index itself, the model is applied to the individual stock constituents. Rather than the usual regression problem that is commonly seen in literature, a binary classification problem is used based on the daily median across the different stocks. The target is therefore whether or not the stock's next day return will be above or below the median. Furthermore, instead of just one LSTM, an ensemble of independent and parallel LSTMs is used, where the ensemble's prediction depends on the majority of the individual results. This is in line with [28] who argue that such ensembles can eliminate much of the randomness in the model and increase the reliability of outcomes. The combination of a median-based binary classification and an LSTM ensemble that this paper implements on the relatively less efficient Swedish market is a unique approach. Results show that, when compared to a randomly chosen portfolio and a portfolio containing all the stocks considered, the LSTM-based method gives rise to portfolios that yield higher returns, lower volatility, and higher risk-return ratios.

The rest of the paper is organized as follows: Section 2 presents LSTMs and their mechanisms in detail. Section 3 describes the method, where the data is introduced and the neural network architecture, ensemble, and trading strategy are explained. Section 4 is the presentation and discussion of results. Finally, section 6 provides a summary and conclusions.

II. Long Short-Term Memory

As mentioned above, RNNs are especially suited for sequential data due to their feedback loops that enable them to use both current and past inputs, therefore allowing information to persist. This feature of RNNs means that they are able to learn and take into account trends and context when training and making predictions. There is, however, one major limitation - RNNs lose memory in the long run because of the vanishing gradient problem. To tackle this, Hochreiter and Schmidhuber [29] introduced the LSTM network in 1997. Since then, it has been modified and improved upon over the years by, for example, [15], [16], [30], [31].

Fig. 1 shows an example of an LSTM network with one input feature x, one hidden layer with several units, and one output y. An LSTM unit, also called a *memory cell*, is magnified to show its inner components. A memory cell contains three gates, each controlling how much information should be kept in memory, forgotten, and passed on as cell output. A $sigmoid$ activation function is used for all three gates as its value ranges from 0, corresponding to no information, to 1, corresponding to all information

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{1}$$

The notation in the diagram are as follows:

- $x = (x_1, x_2, \ldots, x_n)$ is the input vector where x_t, $t = 1, \ldots, n$ is the data point at time t in a sequence of length n
- s_t is the cell state, i.e., the memory of the cell at time t
- $\tilde{s}_t$ is the candidate cell state
- h_t is the output of the cell, also called hidden state
- f_t, i_t, and o_t are the values for the forget, input, and output gates, respectively
- W_f, W_i, W_o, and $W_{\tilde{s}}$ are the weight matrices associated with the input x
- U_f, U_i, U_o, and $U_{\tilde{s}}$ are the weight matrices associated with the output h_t
- b_f, b_i, b_o, and $b_{\tilde{s}}$ are the bias vectors

When the model is created, the cell states and outputs are initialized, say to s_0 and h_0, respectively. For a forward pass, an input $x = (x_1, x_2, \ldots, x_n)$ in the form of a sequence is fed into the model, where the memory cells take the data points x_t consecutively to calculate the new cell states and outputs. Fig. 1's magnified memory cell shows its evolution over time as it processes one data point after another: x_{t-1}, x_t, x_{t+1}, ...

The values for the gates at time t are calculated based on the input x_t and the previous output h_{t-1} as follows:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{4}$$

Similarly, a candidate cell state $\tilde{s}_t$ is computed, still based on the input x_t and the previous output h_{t-1}, but with a $tanh$ activation function. This represents the new information that the memory cell has received.

$$\tilde{s}_t = tanh(W_{\tilde{s}} x_t + U_{\tilde{s}} h_{t-1} + b_{\tilde{s}}) \tag{5}$$

The actual cell state s_t is then calculated based on the previous cell state s_{t-1} and candidate cell state $\tilde{s}_t$ from (5), where the values of the forget and input gates, f_t and i_t, determine how much should be forgotten from s_{t-1} and retained from $\tilde{s}_t$:

$$s_t = f_t s_{t-1} + i_t \tilde{s}_t \tag{6}$$

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 01,2025 at 18:01:22 UTC from IEEE Xplore. Restrictions apply.

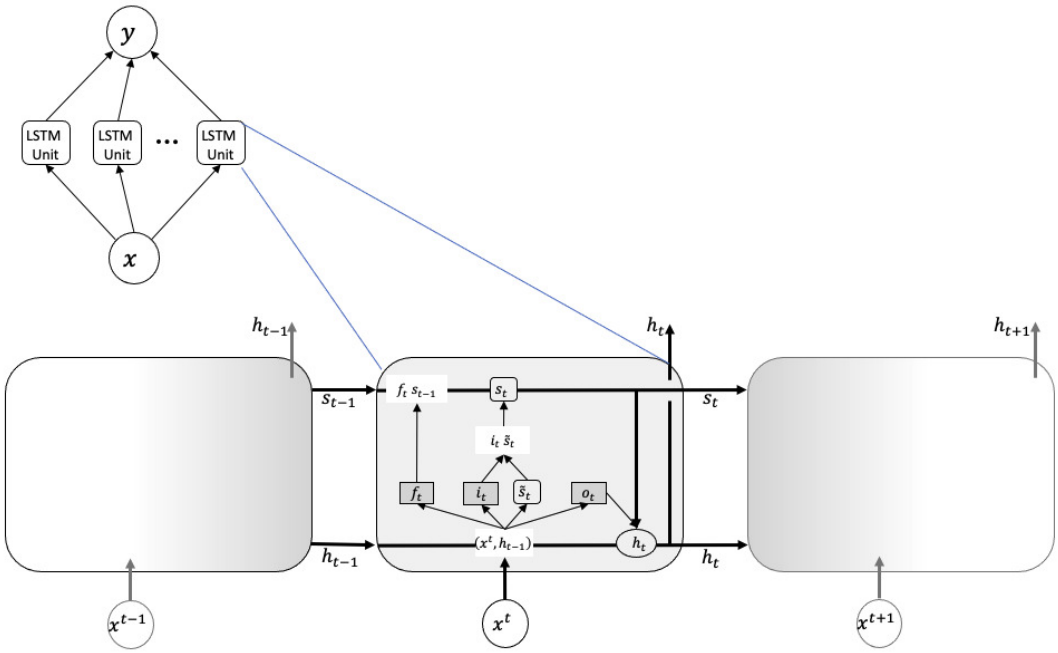


Fig. 1: LSTM network with a magnified LSTM unit (memory cell)

Finally, the output h_t is computed based on the value of the output gate o_t and current cell state s_t calculated above:

$$h_t = o_t tanh(s_t) \tag{7}$$

Note that the outputs h_t are not just looped into the same memory cell, but are also passed to the other memory cells in the network as shown in Fig. 2.

With m being the number of hidden units and n the number of input neurons, the total number of trainable parameters in the neural network is

$$4mn + 4m^2 + 4m \tag{8}$$

where $4mn$ is the number of weights associated with the input, $4m^2$ is the number of weights associated with the outputs h_t, and $4m$ is the number of biases.

III. Method

A. Data

Data for the empirical investigation were taken from the constituents of Stockhom's OMX30 (http://www.nasdaqomxnordic.com). Table I lists the top ten securities in the index by weight, while Table II presents the different industries. Both are from December 2019 as that was the latest available information at the time the data were extracted.

Daily closing prices for the constituent stocks[1] from May 2002 to January 2020 were downloaded. In order to avoid keeping track of the changes in constituents over time, the stocks used were kept to be those comprising the index as of February 2020. With the closing prices p_t, the daily returns R_t were calculated as follows:

$$R_t = \frac{p_t}{p_{t-1}} - 1 \tag{9}$$

[1]Essity B excluded due to lack of data for the dates of interest.

TABLE I: OMX30 Top 10 Securities by Weight (as of December 2019)

Ticker	Security	Weight (%)
ATCO A	ATLAST COPCO	7.40
HM B	HENNES & MAURITZ	6.56
VOLV B	VOLVO	6.11
ERIC B	ERICSSON	5.91
INVE B	INVESTOR	5.49
ASSA B	ASSA ABLOY	5.45
SAND	SANDVIK	5.40
SHB A	SVENSKA HANDELSBANKEN	4.56
ESSITY B	ESSITY	4.54
SEB A	SKAND. ENSKILDA BANKEN	4.51

TABLE II: OMX30 Industry Breakdown (as of December 2019)

Industry	Weight (%)	No. of Securities
Oil & Gas	0.00	0
Basic Materials	3.27	3
Industrials	37.73	10
Consumer Goods	9.21	4
Health Care	4.07	2
Consumer Services	6.56	1
Telecommunications	6.13	2
Utilities	0.00	0
Financials	22.82	6
Technology	10.21	2

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 01,2025 at 18:01:22 UTC from IEEE Xplore. Restrictions apply.

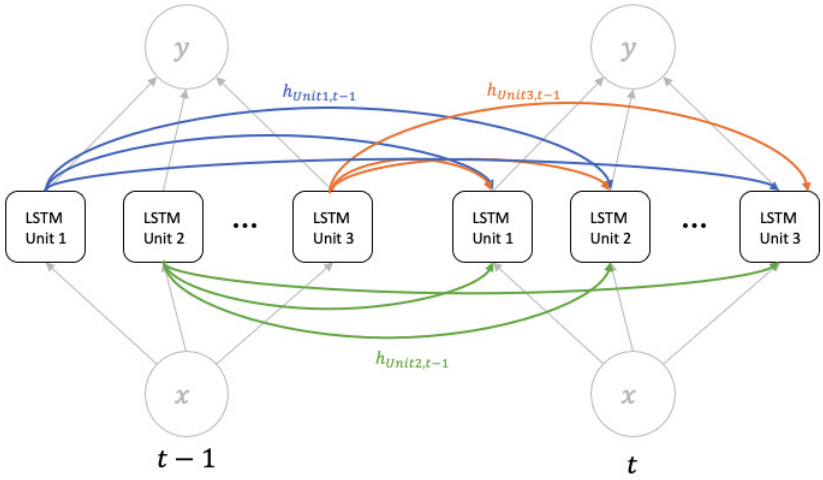


Fig. 2: Hidden states h_t from an LSTM unit do not only get looped within the unit, but are also passed on to other LSTM units

In line with [27] and [9], the daily medians of the returns of the stocks were calculated. Each stock was then classified as either 0 if its daily return is below the daily median, or 1 if its daily return is above. To create the inputs to the LSTM, sequences of returns were created for each stock, where the target for each sequence is the one day ahead prediction of whether the return will be above or below the median. Fig. 3 illustrates this.

B. Network Architecture

The LSTM model consists of one input neuron, one hidden layer, and one output neuron. A sigmoid activation function is used for the output, which can be interpreted as a measure of confidence. Closer to 1 means that the model is more confident that the return will be above the median, while closer to 0 means it is more confident that it will be below. Adam optimizer was used together with a learning rate of 0.0075, which was chosen with the help of Bayesian optimization [32]. The same Bayesian optimization algorithm was also used to determine the values of the other hyperparameters to be:

- number of neurons in hidden layer = 3
- dropout = 0.06
- recurrent dropout = 0.14
- batch size = 6800

1) Training and Testing: For training and testing, the data for each stock were divided into blocks of length 750 days for training (approximately three years of trading), 270 days for valuation (more than a year of trading), and 270 days for testing, as illustrated in Fig. 4. Sequences of length 240 were created for each of these data sets[2]. A rolling window of 30 days was used, which results in 30 non-overlapping days of prediction for each block. In practice, it also means that the model is retrained approximately every six weeks. The 30-day rolling window was chosen based on trial and error, where a shorter rolling window resulted in overfitting, while a longer rolling window resulted in lower accuracy for predictions towards the end of testing as the data becomes further from the training dates.

[2]Note that a 270-day testing data with sequences of length 240 will result in 30 days of predictions.

C. Ensemble and Threshold

Using ensembles of neural network models has been argued to reduce variation and improve generalization. Therefore, as with [28], instead of just one LSTM network, several LSTMs were used, all independent of each other and trained in parallel. A diagram of the ensemble is shown in Fig. 5, where each LSTM has the same architecture described in Section III-B, but with a different weight initialization.
The different initializations used were those readily available in Keras [33] and the details of which are provided in Table III. In total, 11 LSTMs were used.

As the LSTMs are independent, so are their outputs. The final ensemble prediction as to whether the next day return will be above or below the median is decided based upon a minimum number, referred to as a threshold, of LSTMs with that result. In other words, the threshold is the minimum number of LSTMs that must agree on a prediction. Since there are 11 LSTMs, any threshold equal to or above six is considered majority.

D. Trading Strategy

For the trading strategy, stocks predicted by the LSTM ensemble to perform better than the median are bought and added to an equally weighted portfolio. A stock is held until the model no longer predicts it to be above the median, in which case, the position is closed. The resulting portfolio is rather dynamic and is adjusted daily based on the LSTM forecasts. There is no fixed number of stocks in the portfolio; it contains however many stocks the model predicts to perform well.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 01,2025 at 18:01:22 UTC from IEEE Xplore. Restrictions apply.

	x_stock1	x_stock2	...	x_stock29	Median	y_s1	y_s2	...
Sequence 2 / Sequence 1	0,00000	-0,00510	...	-0,01023	-0,01000	1	1	...
	-0,04145	-0,03077	...	-0,01807	-0,01759	0	0	...
	0,00000	0,00529	...	0,01052	0,00527	0	1	...
	0,00542	0,00000	...	0,01821	0,00000	1	1	...
	0,00537	-0,01579	...	0,00257	0,00000	1	0	...
	⋮	⋮	...	⋮	⋮	⋮	⋮	⋮
	0,02211	0,03067	...	-0,00611	0,01059	1	1	...
	0,00723	-0,01786	...	-0,01536	-0,01240	Seq 1 target 1	0	...
	-0,05736	-0,01212	...	-0,02813	-0,01893	Seq 2 target 0	1	...

Fig. 3: Input sequences and targets

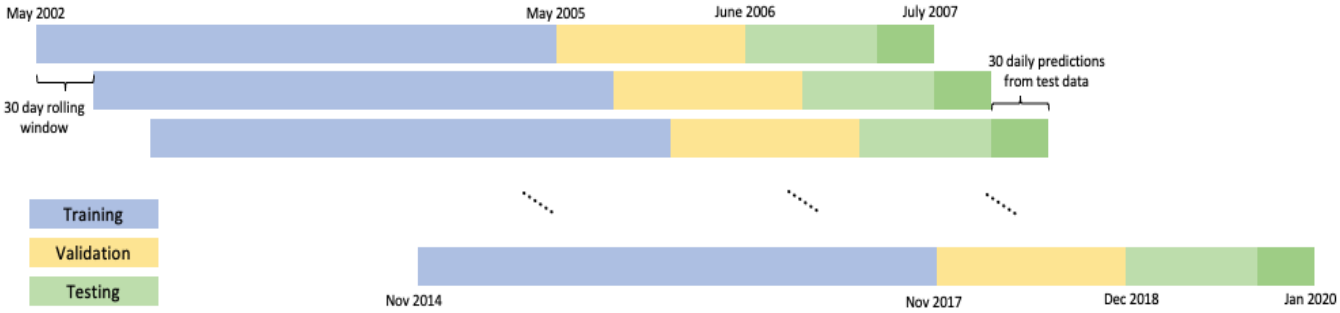


Fig. 4: Blocks of testing, validation, and training sets

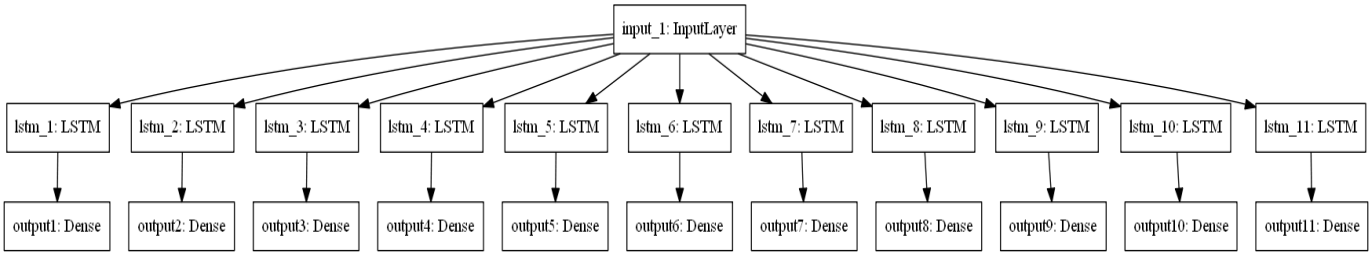


Fig. 5: LSTM ensemble. The LSTMs are independent of each other and are trained in parallel.

TABLE III: Different weight initializations with chosen parameters for the individual LSTMs

Initialization	Description	Parameters
RandomNormal	Normal Distribution	$\mu = 0.0$, $\sigma = 0.05$
RandomUniform	Uniform Distribution	$min = -0.05$, $max = 0.05$
TruncatedNormal	Normal Distribution, values $\sigma \geq 2$ are redrawn	$\mu = 0.0$, $\sigma = 0.05$
Zeros	Initialized to 0	
Ones	Initialized to 1	
GlorotNormal	Normal Distributed	$\mu = 0.0$, $\sigma = \sqrt{\frac{2}{fan\_in+fan\_out}}$
GlorotUniform	Uniform Distribution$[-limit, limit]$	$limit = \sqrt{\frac{6}{fan\_in+fan\_out}}$
Identity	Identity matrix	
Orthogonal	Orthogonal matrix from QR decomposition of random matrix drawn from normal distribution	
Constant	Initialized to a constant	$constant = 0.05$
VarianceScaling	TruncatedNormal	$\sigma = \sqrt{\frac{1}{fan\_in}}$

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 01,2025 at 18:01:22 UTC from IEEE Xplore. Restrictions apply.

IV. Results

A. Accuracy

Fig. 6 below displays the test accuracy for three of the LSTMs, with Random Normal, Random Uniform, and Glorot Uniform[3] weight initializations. Other LSTMs showed similar results so these were chosen as examples. The accuracies are shown to be just near 50%, which is consistent with findings in literature (see, for example, [10], [21], [28]). Because of the complexity of financial time series, accuracy achieved with machine learning methods is often around 50%, unlike the much higher accuracies achieved in other areas such as image recognition.

B. Threshold and Minimum Accuracy

To examine the effects of different thresholds and minimum required accuracy on the average daily returns, these variables of the LSTM ensemble were varied. Fig. 7a shows the result for changing the threshold. Excluding three, which appears to be an anomaly, one can see that the daily average portfolio return grows as the threshold increases, until it reaches eight, at which point the return starts to decrease. Having too low of a threshold means that even stocks predicted by many of the LSTMs to perform worse than the median may be added to the portfolio. On the other hand, after the optimal number of eight, requiring more LSTMs to have the same prediction becomes too strong of a condition and results in less or even no stocks included in the portfolio.
Similarly, Fig. 7b shows the result for varying the minimum required accuracy, where it is shown that the average daily return decreases as the minimum required accuracy increases. An increase in minimum accuracy means requiring the model to be more confident of its prediction. However, as shown above, the predictive accuracy stays very close to 50%, which means that, as with the threshold, having a higher accuracy requirement may be too strict of a condition, leading to fewer or no stocks being included in the portfolio.

C. Portfolio Returns

Apart from looking at the effects of the threshold and required accuracy, the daily returns from the LSTM-based portfolio were compared with a portfolio containing all 29 stocks, to be referred to as an all-stock portfolio[4], and a randomly chosen portfolio. The random portfolio contains randomly chosen stocks from the 29 that were studied and the number of stocks included in the portfolio daily is also random. Table IV displays the overall results. As seen in the table, both the LSTM and the all-stock portfolios have higher average daily returns than the random portfolio, with the LSTM portfolio having the highest. The LSTM portfolio also has a lower standard deviation than the all-stock portfolio, indicating that the portfolio chosen by the LSTM method may incur less risk. It is also worth noting that although the all-stock portfolio appears to have a slightly higher maximum daily return, the minimum return for the LSTM is less negative, which may suggest that the LSTM portfolio has a much less downside that of the all-stock portfolio.

[3]Glorot Uniform is the default Keras initialization.

[4]A comparison with the OMX30 index cannot be made directly for two reasons. First, ESSITY B was excluded due to lack of data for the period of interest. Second, the stocks examined here are fixed to be those making up the index as of February 2020, while the index constituents in practice are regularly adjusted.

TABLE IV: Comparison of daily returns (%) for the LSTM, all-stock, and randomly chosen portfolios

	LSTM	All-Stock Portfolio	Random Portfolio
Mean	0,0397	0,0336	0,0159
StDev	1,2274	1,4291	0,7371
Min	-6,1097	-8,3910	-4,7493
Max	9,7507	9,9950	4,9016

For more details, Table V compares the average daily returns by year, where the best results are highlighted in bold. During the 2007 - 2008 financial crisis, the random portfolio provided better returns than both the LSTM and all-stock portfolios. Afterwards, however, as in the overall results in Table IV, the random portfolio is outperformed by the LSTM and all-stock portfolios. The results of these two portfolios are very close to each other, where in some years, the LSTM portfolio appears to have a higher average daily return, while in others, the all-stock portfolio does. Similar to Table IV, however, one can see that for periods of negative returns, such as for the years 2007 - 2008, 2011, and 2018, the LSTM appears to do better than the all-stock portfolio by having less negative results. Again, this implies that the LSTM portfolio is certainly a more attractive choice when losses are to be expected.

TABLE V: Comparison of average daily returns (%) by year. Best results are highlighted in bold.

Year	LSTM	All-Stock Portfolio	Random Portfolio
2007	-0,0531	-0,0755	**-0,0407**
2008	-0,0872	-0,1743	**-0,0641**
2009	0,1911	**0,2322**	0,1184
2010	**0,1052**	0,1046	0,0621
2011	**-0,0085**	-0,0462	-0,0246
2012	**0,0805**	0,0667	0,0404
2013	0,0639	**0,0699**	0,0275
2014	0,0453	**0,0519**	0,0253
2015	0,0181	**0,0204**	0,0126
2016	0,0142	**0,0441**	0,0141
2017	**0,0344**	0,0344	0,0131
2018	**-0,0403**	-0,0497	-0,0447
2019	**0,1095**	0,1088	0,0434

To see the development of profits over time, Fig. 8 graphs the cumulative returns for all three portfolios from 2007 - 2020. It can be seen that the LSTM portfolio clearly has the highest accumulated returns for the whole period. Once again, during intervals of decline, as is evident in 2008 - 2009 for example, the graph shows that the LSTM portfolio returns does not decrease as much as the all-stock portfolio.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 01,2025 at 18:01:22 UTC from IEEE Xplore. Restrictions apply.

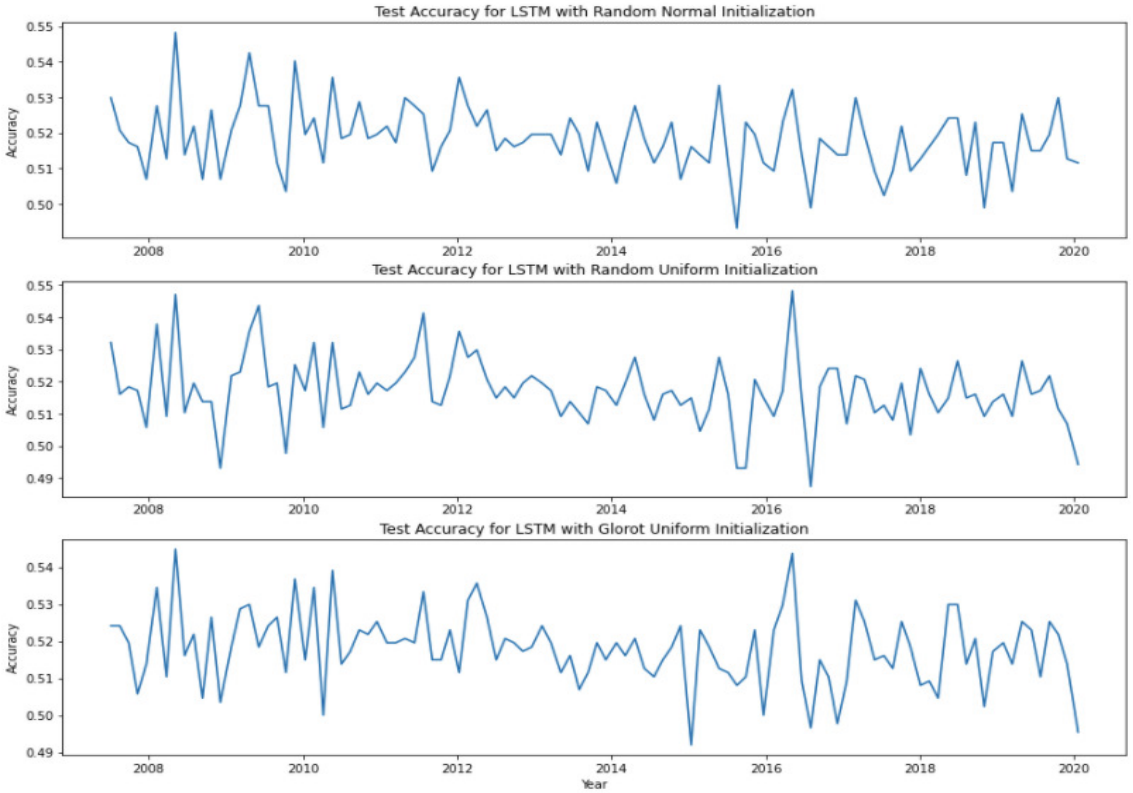


Fig. 6: Samples of test accuracy for the LSTMS with different weight initializations

D. Portfolio Risk

For a more rigorous assessment of the risk, the annualized volatility, Sharpe ratio, and Sortino ratio were examined. The Sharpe ratio is a commonly used measurement of return per unit of risk and is given by the following formula:

$$Sharpe\ ratio = \frac{R_p - r_f}{\sigma_p} \tag{10}$$

where the R_p is the portfolio return, r_f the risk-free return, and σ_p the standard deviation of the portfolio returns. On the other hand, the Sortino ratio is a variant of the Sharpe ratio that only takes into account the downside standard deviation. It is given by:

$$Sortino\ ratio = \frac{R_p - r_f}{\sigma_d} \tag{11}$$

where σ_d is the standard deviation of the negative portfolio returns. By only considering the negative returns, the Sortino ratio distinguishes between good and bad volatility. Regardless of which is used, higher values are preferred for both ratios. In the calculations, the risk-free rate was taken to be the Swedish 1-month Treasury bill (accessed from https://www.riksbank.se/en-gb/statistics/search-interest--exchange-rates/). Results are presented in Table VI.

Looking at the annualized volatility columns, it is evident that the LSTM has lower volatility throughout, denoting a less risky portfolio. Comparing the Sharpe and Sortino ratios, one can see that apart from 2013 - 2016, the LSTM portfolio appears to have a higher return per unit of risk than the all-stock portfolio. Negative Sharpe and Sortino ratios in 2007 - 2008, 2011, and 2018 indicate the portfolios' returns during these periods are less than the risk-free rate (see Table V). Care must always be taken when comparing negative ratios as they can be misleading. For example, for the same return, a larger standard deviation, i.e., higher risk, will result in a less negative ratio, which may lead some to conclude better performance. In this case, however, the less negative ratios of the LSTM portfolio do indeed come from the combination of higher returns and lower volatility.

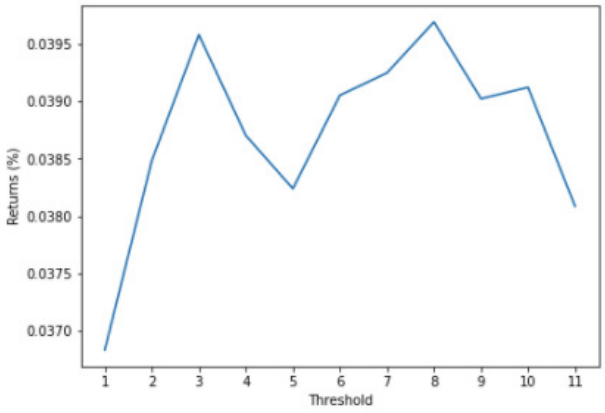
V. Summary and Conclusions

In this paper, we present an approach for the prediction of stock price movement by using an ensemble of independent and parallel LSTM neural networks. A binary classification problem based on the median of returns is used and the ensemble's forecast depends on how many of the LSTMs agree on the same output. The model is applied to the constituents of the relatively smaller and less efficient OMX30 index as opposed to the commonly used major stock market indices such as the S&P500 and DJIA. A straightforward trading strategy is then implemented based on the LSTM forecasts. Compared to a randomly chosen portfolio and a portfolio containing all the stocks in the index, the LSTM-based portfolio appears to have better average daily returns and a higher cumulative

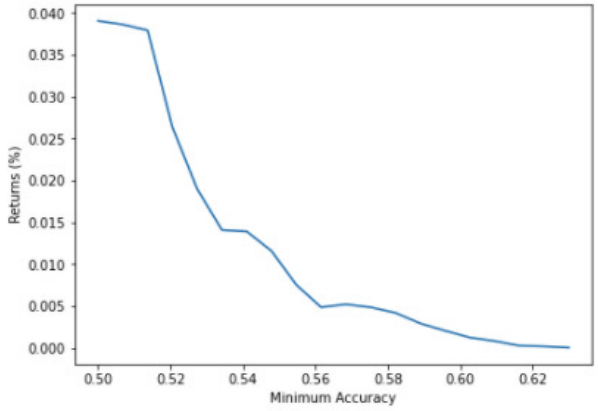
Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 01,2025 at 18:01:22 UTC from IEEE Xplore. Restrictions apply.

TABLE VI: Comparisons of risk measures and risk-reward ratios between the LSTM and all-stock portfolios. Better results are highlighted in bold.

	Annualized Volatility		Annualized Sharpe Ratio		Annualized Sortino Ratio	
Year	LSTM	All-Stock Portfolio	LSTM	All-Stock Portfolio	LSTM	All-Stock Portfolio
2007	**20,5268**	22,8983	**-0,7673**	-0,9333	**-1,1907**	-1,4488
2008	**35,0996**	41,8415	**-0,7025**	-1,1130	**-1,2706**	-1,9361
2009	**23,3151**	32,2950	**2,0473**	1,7982	**3,5995**	3,3155
2010	**18,6689**	19,8941	**1,3992**	1,3045	**2,4743**	2,2925
2011	**26,0842**	28,1767	**-0,1284**	-0,4556	**-0,2094**	-0,7234
2012	**18,3997**	19,2223	**1,0462**	0,8206	**1,7852**	1,3671
2013	**12,4070**	12,6676	1,2403	**1,3343**	2,1114	**2,2801**
2014	**13,1716**	13,7736	0,8410	**0,9246**	1,4495	**1,5971**
2015	**18,5197**	20,5986	0,2574	**0,2590**	0,4248	**0,4295**
2016	**14,8670**	21,0785	0,2701	**0,5470**	0,4108	**0,8534**
2017	**9,9064**	10,7538	**0,9232**	0,8501	**1,5297**	1,4159
2018	**13,6962**	15,5125	**-0,7053**	-0,7752	**-1,1330**	-1,2500
2019	**13,8528**	14,5898	**2,0087**	1,8948	**3,3842**	3,1913



(a) Average Daily Returns (%) vs. Threshold



(b) Average Daily Returns (%) vs. Minimum Accuracy

Fig. 7: Average daily returns as threshold and minimum accuracy are varied

return over time. Even more remarkably, the LSTM portfolio exhibits less volatility throughout the whole period than the portfolio containing all the stocks. The combination of this lower volatility with the higher returns results in higher risk-return ratios.

With such encouraging outcomes, we identify several ways to improve the approach even further. For example, using a learning rate decay might increase speed of convergence and accuracy. Having a more systematic way of deciding when the model should be retrained may also lead to better results and prevent overfitting. For example, statistics from the data may be taken regularly, where retraining is done whenever a significant shift in distribution or parameters is observed. Several input neurons corresponding to the number of stocks instead of a single one for the whole neural network may force the model to process the stocks simultaneously and learn the correlations between them. Instead of being equal, having a trading strategy that weighs the stocks depending on the model's confidence may also lead to higher returns. Finally, incorporating other types of architecture in the ensemble, such as Transformer-based methods which have also been shown to work well with time series data, may raise efficiency and accuracy even more. These are just some examples on how to increase the performance of a method which already gives promising results.

REFERENCES

[1] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[2] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[3] R. H. Shumway and D. S. Stoffer, *ARIMA Models*. Cham: Springer International Publishing, 2017, pp. 75–163.

[4] R. S. Tsay, *Analysis of Financial Time Series*. John Wiley & Sons, 2005.

[5] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 5, pp. 1407–1418, 2018.

[6] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 24–38, 2009.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 01,2025 at 18:01:22 UTC from IEEE Xplore. Restrictions apply.

Fig. 8: Comparison of cumulative returns (%)

[7] C. Krauss, X. A. Do, and N. Huck, "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500," *European Journal of Operational Research*, vol. 259, no. 2, pp. 689–702, 2017.

[8] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques," *Expert Systems with Applications*, vol. 42, no. 1, pp. 259–268, 2015.

[9] L. Takeuchi and Y.-Y. A. Lee, "Applying deep learning to enhance momentum trading strategies in stocks," in *Technical Report*. Stanford University, 2013.

[10] L. Di Persio and O. Honchar, "Artificial neural networks architectures for stock price prediction: Comparisons and applications," *International Journal of Circuits, Systems and Signal Processing*, vol. 10, no. 2016, pp. 403–413, 2016.

[11] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PloS One*, vol. 12, no. 7, p. e0180944, 2017.

[12] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[14] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki, "Scene labeling with LSTM recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3547–3555.

[15] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.

[16] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[17] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[18] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3156–3164.

[19] J. Cao, Z. Li, and J. Li, "Financial time series forecasting model based on CEEMDAN and LSTM," *Physica A: Statistical Mechanics and its Applications*, vol. 519, pp. 127–139, 2019.

[20] J. B. Heaton, N. G. Polson, and J. H. Witte, "Deep learning in finance," *CoRR*, vol. abs/1602.06561, 2016.

[21] D. M. Nelson, A. C. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with LSTM neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1419–1426.

[22] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1643–1647.

[23] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Using deep learning to detect price change indications in financial markets," in *2017 25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 2511–2515.

[24] J. F. A. Yeung, Z.-k. Wei, K. Y. Chan, H. Y. Lau, and K.-F. C. Yiu, "Jump detection in financial time series using machine learning algorithms," *Soft Computing*, vol. 24, no. 3, pp. 1789–1801, 2020.

[25] R. Xiong, E. P. Nichols, and Y. Shen, "Deep learning stock volatility with google domestic trends," *arXiv preprint arXiv:1512.04916*, 2015.

[26] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, "A comparison of ARIMA and LSTM in forecasting time series," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 1394–1401.

[27] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[28] S. Barra, S. M. Carta, A. Corriga, A. S. Podda, and D. R. Recupero, "Deep learning and time series-to-image encoding for financial forecasting," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 683–692, 2020.

[29] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997.

[30] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 10 2000.

[31] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005, IJCNN 2005.

[32] F. Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python," 2014. [Online]. Available: https://github.com/fmfn/BayesianOptimization

[33] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io

[34] Nasdaq, "OMX Stockholm 30 Index, Fact Sheet," 2020. [Online]. Available: https://indexes.nasdaq.com/docs/FS_OMXS30.pdf

[35] C. Olah, "Understanding LSTM networks," 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs

[36] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 01,2025 at 18:01:22 UTC from IEEE Xplore. Restrictions apply.