UNIVERSITY OF WARSAW

Faculty of Economic Sciences

WORKING PAPERS
No. 29/2022 (405)

THE EFFICIENCY OF VARIOUS TYPES OF INPUT LAYERS OF LSTM MODEL IN INVESTMENT STRATEGIES ON S&P500 INDEX

THI THU GIANG NGUYEN
ROBERT ŚLEPACZUK

WARSAW 2022

WORKING PAPERS 29/2022 (405)

The efficiency of various types of input layers of LSTM model in investment strategies on S&P500 index

Thi Thu Giang Nguyen[a], *Robert Ślepaczuk*[b]*

[a] *Faculty of Economic Sciences, University of Warsaw; Quantitative Finance Research Group*
[b] *Faculty of Economic Sciences, Department of Quantitative Finance, University of Warsaw; Quantitative Finance Research Group*

Corresponding author: rslepaczuk@wne.uw.edu.pl

Abstract: The study compares the use of various Long Short-Term Memory (LSTM) variants to conventional technical indicators for trading the S&P 500 index between 2011 and 2022. Two methods were used to test each strategy: a fixed training data set from 2001–2010 and a rolling train–test window. Due to the input sensitivity of LSTM models, we concentrated on data processing and hyperparameter tuning to find the best model. Instead of using the traditional MSE function, we used the Mean Absolute Directional Loss (MADL) function based on recent research to enhance model performance. The models were assessed using the Information Ratio and the Modified Information Ratio, which considers the maximum drawdown and the sign of the annualized return compounded (ARC). LSTM models' performance was compared to benchmark strategies using the SMA, MACD, RSI, and Buy&Hold strategies. We rejected the hypothesis that algorithmic investment strategy using signals from LSTM model consisting only from daily returns in its input layer is more efficient. However, we could not reject the hypothesis that signals generated by LSTM model combining daily returns and technical indicators in its input layer are more efficient. The LSTM Extended model that combined daily returns with MACD and RSI in the input layer generated a better result than Buy&Hold and other strategies using a single technical indicator. The results of the sensitivity analysis show how sensitive this model is to inputs like sequence length, batch size, technical indicators, and the length of the rolling train - test window.

Keywords: algorithmic investment strategies, machine learning, testing architecture, deep learning, recurrent neural networks, LSTM, technical indicators, forecasting financial-time series, technical indicators, hyperparameter tuning S&P 500 Index,

JEL codes: C15, C45, C52, C53, C58, C61, G14, G17

Acknowledgements: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Introduction

Predicting financial time series is a challenging task due to the nature of the data. Among the challenges are the high volatility and constant changes over time, which means that we are not able to parameterize the time series and find any sustainable patterns to predict the prices efficiently. Despite the obstacles, the effort to predict the market has never stopped. For institutional investors such as investment funds and insurance companies, an efficient model can help anticipate market's volatility and optimize portfolio allocation. Individual investor can also apply algorithmic trading to find investment opportunities that maximize returns and minimize risks. Because of all these benefits, we have seen numerous research in various directions, such as using macroeconomics indicators, analyzing market sentiment, or using technical analysis.

In recent years, the financial industry has joined with other industries to take advantage of modern technological advancements, especially Deep Learning. Since computational resources have become more accessible, there has been increasing attention to exploring the application of complicated models such as neural networks in predicting financial time series. However, the majority of the literature we found focused on solving the forecasting problem, while only a few works analyzed the application of these models in investment strategies.

This research aimed to investigate the application of Long Short-Term Memory (LSTM), a Recurrent Neural Network (RNN), in predicting and trading the S&P500 index. We compared the LSTM models' performance with other traditional trading strategies that have been used in the market for a long time, including the Simple Moving Average (SMA), Moving average convergence divergence (MACD), and Relative Strength Index (RSI). We conducted the research using daily S&P500 closed prices from 2001.01.01 to 2022.04.30, in which the out-of-sample period was from 2011.01.01 to 2022.04.30.

Since LSTM models have been proved superior in handling sequential data, we presumed there was great potential in developing an effective trading strategy based on LSTM models compared to other conventional approaches. We searched for results in order to verify the following hypotheses:

(RH1) *The signals from algorithmic investment strategy using predictions from the LSTM model consisting only from daily returns in its input layer are more efficient than using technical analysis indicators.*

(RH2) *The signals from algorithmic investment strategy using predictions from the LSTM model combining daily returns and technical analysis indicators in its input layer are more efficient than using only technical indicators.*

The research was developed using Python 3.7 and leveraged GPUs and TPUs from Google Colab. The neural networks were built on the framework from TensorFlow and Keras packages. Due to the complexity of the models, one full training and hyperparameter tuning took approximately 12 hours to complete.

The research is structured into five sections. First section reviews recent research on approaches to predict and trade financial assets. Second section describes the data, methodology, and details of the strategies responsible for trading signals generation process. The third section summarizes

the results of the tested strategies. The fourth section conducts a sensitivity analysis of how the performance of the selected model changes with respect to the change of the main hyperparameters or input data. The last section provides the conclusion about the hypothesis based on the results of the experiments we conducted.

1 Literature Review

Traders have been using indicators such as SMA, MACD, and RSI to find short-term trends and make trading decisions. These trading strategies have the advantages of being simple to implement and have been tested in academics and in the market for a long time (Sang & Di Pierro, 2019).

The moving average is a smoothing technique to verify the emergence of new trends. Ellis and Parbery (2005) conducted research comparing the performance of SMA strategies using fixed and adaptive window lengths. The research was performed on three stock market indices: the Australian All Ordinaries (AOI), the Dow Jones Industrial Average (DJIA), and the S&P500. The authors concluded that the returns generated by adaptive moving average were not significantly higher than the fixed SMA_{200} strategy if we consider transaction cost. The authors pointed out that the adaptive strategy might be superior in reflecting market trends and generating more trading signals; however, the returns from such a strategy could not compensate for the transaction cost, which resulted in lower net returns.

In order to analyze the momentum of the market, the MACD indicator derived from the 12-day exponential moving average, and 26-day EMAs is commonly used. There are also variations of the analysis, such as instead of using the fixed weight to calculate the EMA, we can use a changing weight based on the historical volatility, which shows 55.55% higher accuracy in terms of trend recognition compared to the conventional MACD (Wang & Kim, 2018).

Pradipbhai (2013) studied whether using a simple moving average instead of an exponential moving average had an impact on the performance of MACD strategy on the CNX Nifty during the period from 2011-01-04 to 2012-03-31. The study confirmed that using an exponential moving average was better in terms of generated returns.

In another recent study, the author tested various MACD strategies in different markets' indices, including DAX, NIKKEI 225, RTS, SSE, and HOSE. In all MACD strategies, the author used the popular MACD parameters (12,26,9), which was also the default parameter used in almost all technical analysis software. The results show that the strategies could generate positive returns, and the best strategy was to buy when the MACD line crossed over the signal line and both lines were above the zero line. Alternatively, we should sell when the MACD line crossed under the signal lines and both lines were below the zero line. (Hoang Hung, 2016). The same MACD strategy was also tested on London Stock Exchange's FT30 index during the period of 60 years, from 1935 to 1994, and the strategy was proved to outperform the buy and hold (Chong & Ng, 2008).

The profitability of strategy using the RSI oscillator was investigated for trading CHFUSD from January 1998 to May 2009 (Anderson & Li, 2015). The author computed RSI based on the

past 14-day period, which was a typical period length used in practice, and constructed trading strategies using different thresholds: (40,60), (30,70), (20,80), (15,85), and (10,90). The strategy that used (40,60) thresholds generated the most significant profit, and the authors concluded that investors should customize the strategy instead of using the conventional parameters. The same conclusion was confirmed by Bartkus (2018), who tested RSI in trading the EURUSD pair from January 2016 to July 2017 and used hourly data. Instead of testing different thresholds, the author tested the strategy using different periods and suggested that an RSI modification that used a daily or weekly period generated less number of trades and could potentially bring a better result. Nor and Wickremasinghe (2014) investigated RSI on the Australian stock market, utilizing daily data of the Australian All Ordinaries Index from 1996 to 2014. Using the parameter values from Wilder's original research about RSI (Wilder, 1978), the authors found that this strategy showed the possibility of making profits. The research results indicated that the strategy worked well in some periods but performed poorly in others. Therefore, the authors proposed the idea of constantly revisiting and optimizing the parameters to improve its performance. NSE stock market prediction using deep-lear The research papers that we reviewed about applying technical indicators such as SMA, MACD, and RSI confirmed that these strategies could predict trends. However, we were unable to compare strategies because their performance was mostly summarized by the rate of returns instead of using any risk-adjusted return metrics, for example, information ratio. All of the research emphasized tuning of each indicator's parameters in order to find out the best configurations.

As technology advances, attention is turning to Machine Learning and Deep Learning to understand the market's behavior and discover the most efficient indicator for developing better-automated trading strategies.

Heaton et al. (2017) applied deep learning models to perform smart indexing for the biotechnology IBB index from January 2012 to April 2016 and concluded that deep learning had better predictive performance in comparison to conventional approach. Grudniewicz and Ślepaczuk (2021) used machine learning models such as Neural Networks, KNN, Regression Tree, Random Forest, Bayesian Generalized Linear and Support Vector Machines to trade different market indices from 2002 to 2020. The results showed that Bayesian Generalized Linear was the best performing model, outperformed the Buy & Hold strategy. Kanwal et al. (2022) proposed a hybrid approach that combined Bidirectional Cuda Deep Neural Network Long Short-Term Memory (BiCuDNNLSTM) and 1-D Convolutional Neural Network (CNN) to predict five different financial time-series. The outcome showed that this hybrid model had the highest prediction accuracy, compared to other models such as LSTM-DNN, LSTM-CNN, and LSTM.

Recurrent Neural Network (RNN) is a class of artificial neural networks specializing in modeling sequential data. RNNs were firstly applied in domains such as Natural Language Processing or predicting DNA sequences. Technical traders have soon recognized the similarity of these prediction problems with the stock price prediction problem. For example, a series of sentences or words could be viewed as a time series with some degree of auto-correlation. Long Short-Term Memory (LSTM)

is a variant of RNN, firstly introduced in 1997, and since then, researchers have been trying to use these models to predict the market.

Kim and Kang (2019) tested various deep learning models, including MLP, 1D CNN, LSTM, and attention network, to predict the trends of Korean's KOSPI200 index. The authors found that LSTM and weighted attention networks worked well with sequential data. Fischer and Krauss (2018) noticed that LSTM model outperformed random forest and logistic classifier when used to predict the movements of S&P500 from 1992 to 2015. Before that, M et al. (2018) concluded in his work that LSTM and CNN models outperformed linear model such as ARIMA. The research results was obtained from highly traded stocks of India's NSE index and NYSE. However, the authors used different trading periods for NSE and NYSE stocks, which might make the conclusion biased.

Lv et al. (2021) used a general LSTM model and a LightGBM-optimized LSTM model to predict the daily closing prices of Shanghai and Shenzhen 300 indices. Their experiments showed that the LightGBM-LSTM version had the best results in predicting the stock indices' trend, measured by accuracy and F1 scores. LSTM model was also used to predict the next-day closing price of S&P500 index (Bhandari et al., 2022). In this research, the authors used predictors representing the macroeconomic data, technical indicators, and the time series itself as input to generate predictions. The technical indicators used in this research were MACD, Average True Range (ATR), and RSI. The model performance was measured using RMSE, MAPE, and R^2 - which measured the precision of predicted prices versus the actual price. The findings showed that the LSTM model with a single layer and 150 hidden neurons provided better results than stacked LSTM models.

Sang and Di Pierro (2019) combined the LSTM model and traditional technical indicators: SMA, MACD, and RSI, to use the advantages of both approaches. The technical indicators could provide LSTM with information about the market behavior, and LSTM had the ability to process sequential data. The experiments were conducted on the top five stocks in each of the nine sectors constituting the S&P500 and the ETFs representing these nine sectors. Daily data for 2014 was used as train data set, while the out-of-sample period lasted from 2015 to 2018. From these findings, the authors concluded that applying LSTM had the potential to improve the performance of technical trading algorithms.

The mentioned research above focused on the predicting ability of LSTM models. However, they did not investigate the profitability of the strategies based on LSTM models' predictions. Michańków et al. (2022) compared the performance of different trading strategies using LSTM prediction and on different asset classes: the S&P500 index and Bitcoin price using data from 2013 to the end of 2020 with the 15-minute, hourly, and daily frequency data. The authors also used a customized Mean Absolute Directional Loss (MADL) loss function to train the LSTM model to improve the usefulness of LSTM prediction with regard to algorithmic investment strategies. The authors concluded that the LSTM models were sensitive to the hyperparameters and the configuration of the LSTM models, especially the loss function played a crucial role. Overall, during the tested period from 2017 to 2020, the LSTM strategies could not outperform Buy& Hold strategies, regardless of the asset class or the data frequency used. Kijewski and Ślepaczuk (2020) applied LSTM model to predict and

trade S&P 500 index in the period from 2001 to 2020 and came to similar conclusion. Although the authors found that LSTM model was able to outperformed Buy&Hold strategy this period, the results were not robust to changes in most of the hyperparameters. The strategy that combined signals from LSTM models and other classical methods outperformed market significantly.

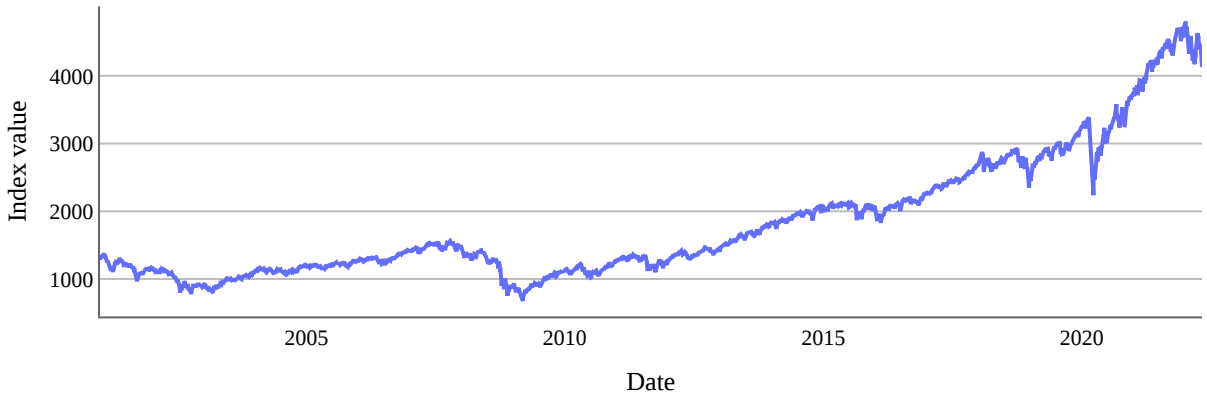
Previous authors' work enabled us to develop our approach for this research. We utilized the MADL as the loss function to train the LSTM network, as this function took into account the magnitude of the loss resulting from the wrong predictions. We investigated three of the most used technical indicators - SMA, MACD, and RSI - separately and in combination with the LSTM models and were able to compare the performance among the strategies by using a standardized set of performance metrics.

2 Methodology and Data

2.1 Data

The research used the daily data of S&P500 index in the period from 2001.01.01 to 2022.04.30, and the trading period started from 2011.01.01. The index was imported from Yahoo Finance via Python's package *yfinance*. Figure 1 shows the index's closed price of the whole period, and the descriptive statistic of the daily returns are summarized in Table 1.

Figure 1: S&P 500 Closed Price from 2001-01-01 to 2022-04-30



Note: The index was imported from Yahoo Finance via Python's package *yfinance* and covered the period from 2001-01-01 to 2022-04-30.

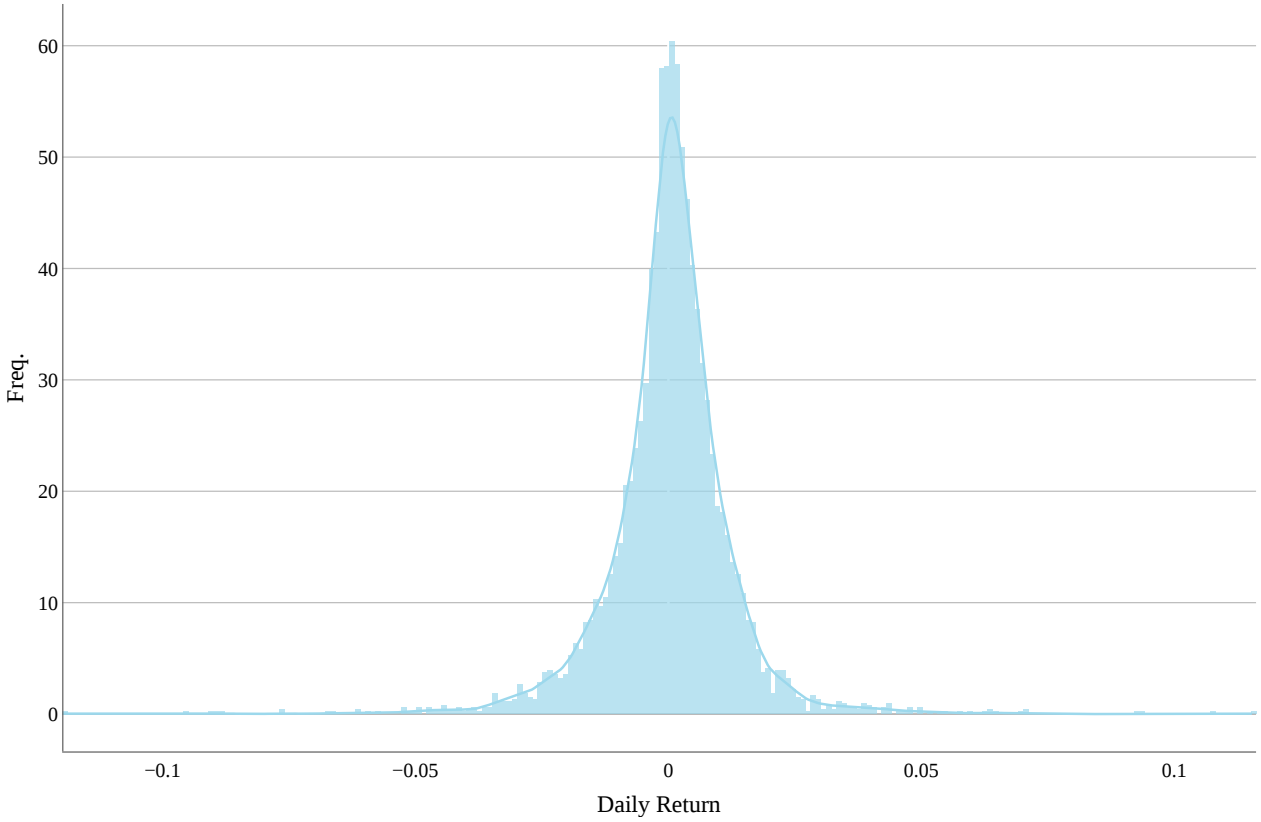
Table 1: S&P500 daily return descriptive statistic

Count	5365	**Mean**	0.000294
Std	0.012311	**Min**	-0.119841
25% percentile	-0.004544	**50% percentile**	0.000673
75% percentile	0.005772	**Max**	0.115800
Kurtosis	11.28	**Skewness**	-0.174437
KS Test stat.	0.4779	**KS Test pvalue**	0.0

Note: Descriptive statistics calculated of S&P500's daily returns from 2001-01-01 to 2022-04-30. Kolmogorov-Smirnov test for normality had test statistic = 0.48 and p-value = 0.0. Therefore, we rejected the null hypothesis that the data were normally distributed.

S&P500's daily simple returns are leptokurtic, with the Kurtosis statistic of 11.28, significantly higher than the normal distribution. It means that there are high probabilities of extraordinarily high or low returns. The Skewness of -0.17 indicates a relatively symmetrical distribution. The density plot of daily returns in Figure 2 showed a symmetrical distribution of daily returns around zero, which meant that the number of gain days and loss days in the research period are more or less balanced.

Figure 2: S&P 500 Daily return density plot from 2001.01.01 to 2022.04.30



Note: Daily returns from 2001-01-01 to 2022-04-30 are distributed symmetrically but are leptokurtic (Kurtosis statistic of 11.28).

LSTM model learns by using functions that map a sequence of past observations to an output observation. Therefore, we split the time series into sub-sequences of input and output. The splitting process uses the sequence length hyperparameter, which dictates how many time steps, in this case - previous trading days, are used as input to predict the next value in the sequence. We scaled the S&P500 data by transforming the daily index values to the index daily simple return in order to reduce the training time of the LSTM's models.

We used the daily S&P500 closed price for the other benchmark strategies without any transformation.

2.2 Investment Strategies Descriptions

We used the below general assumptions for all the strategies in this research:

- We assumed that we could trade a fraction of the index.
- When entering the position on day t, we assumed that we bought the index at day t's closing price.
- When exiting the position on day t, we assumed that we sold at day t's closing price.

For each indicator, we used a specific strategy to identify trading signals:

- **LSTM predictions** At the end of day $t-1$, we generated a prediction of day t's return. If the predicted returns in positive, we placed an order to buy the index at day t. Otherwise, we exited the existing long position or did not enter the long position.

$$pos_t = \begin{cases} 1 & \text{if } \hat{r_t} > 0 \\ 0 & \text{else} \end{cases} \tag{1}$$

  where pos_t is position for day t, $\hat{r_t}$ is predicted return of day t, 1 means a long position, and 0 means no position.

- **Simple Moving Average:** We entered a long position at t if the index's closed price crossed over the SMA line at day $t-1$. We exited the long position at t the index's closed price crossed under the SMA line at day $t-1$. Otherwise, we kept the previous position. The SMA was calculated using the index level.

$$pos_t = \begin{cases} 1 & \text{if } SP500_{t-1} > SMA_{t-1} \text{ and } pos_{t-1} = 0 \\ 0 & \text{if } SP500_{t-1} < SMA_{t-1} \text{ and } pos_{t-1} = 1 \\ pos_{t-1} & \text{else} \end{cases} \tag{2}$$

  where pos_t is position of day t, $SP500_{t-1}$ is SP500 index level at end of day $t-1$, and $SMA_{t-1} = \frac{1}{n}\sum_{i=1}^{n} SP500_{t-i}$ is the simple average of price of period $[t-n, t-1]$.

- **Relative Strength Index:** We entered a long position at day t if the RSI line crossed over the oversold threshold at day $t-1$ and sold at t when the RSI crossed below the overbought threshold at day $t-1$. Otherwise, we kept the previous position.

$$pos_t = \begin{cases} 1 & \text{if } RSI_{t-1} > oversold \text{ and } pos_{t-1} = 0 \\ 0 & \text{if } RSI_{t-1} > overbought \text{ and } pos_{t-1} = 1 \\ pos_{t-1} & \text{else} \end{cases} \tag{3}$$

where pos_t is position of day t, RSI_{t-1} is RSI value at end of day $t-1$.

- **Moving Average Convergence Divergence:** When the MACD line indicating the difference between the two EMA lines crossed over the signal line at day $t-1$, we opened a long position on day t. We exited the long position when the MACD line crossed under the signal line. Otherwise, we kept the previous position.

$$pos_t = \begin{cases} 1 & \text{if } MACD_{t-1} > signal_{t-1} \text{ and } pos_{t-1} = 0 \\ 0 & \text{if } MACD_{t-1} > signal_{t-1} \text{ and } pos_{t-1} = 1 \\ pos_{t-1} & \text{else} \end{cases} \tag{4}$$

where pos_t is position of day t, $signal_{t-1}$ is the difference between two EMAs, and $MACD_{t-1}$ is MACD value at end of day $t-1$.

- **Buy and Hold:** We bought the index at the beginning of the test period and held it until the end.

2.3 The Evaluation of Model's Performance

Adopting the approach from Michańków et al. (2022), we calculated to following metrics to evaluate the performance of the tested strategies:

- **Portfolio value:** We calculated the daily portfolio value using this formula:

$$E_t = E_{t-1}(1 + r_t \times pos_{t-1}) \tag{5}$$

where

pos_{t-1} : trading position we had at the end of day $t-1$, which was generated by the model on day $t-2$. The position was one if we had a long position and 0 if we did not have a position.

r_t: daily simple returns of the index of day t.

E_{t-1}: portfolio value at the end of day $t-1$

The position of each trading day was generated a day before because we used the closed price up to the previous trading day as input to our models. For example, the position of day t was

generated at the end of day $t-1$. If models generated a buy signal and we already had a long position on the day $t-1$, we continued holding the position, and the portfolio value at day t grew at the same rate as the return of day t. In the other case, when we did not have any position on the day $t-1$, the portfolio value at the end of day t was the same as day $t-1$ since we used all the available funds from day $t-1$ to enter the long position at day t's closed price.

In case there is a transaction fee applied:

$$E_t = E_{t-1}(1 + r_t \times pos_{t-1} - |pos_t - pos_{t-1}| \times fee) \tag{6}$$

where

fee: the transaction fee, and was a percentage of the trade's value.

- **Annualised Return Compounded (ARC):** The metric shows annualised rate of return for the given strategy over the period from $(0, ..., T)$. We assumed there were 250 trading days in a year.

$$ARC = \left(\frac{E_T}{E_0}\right)^{\frac{250}{T}} - 1 \tag{7}$$

where

T: the number of trading days in the investment period.
E_T: portfolio's value at the end of the investment period.
E_0: portfolio's value at the beginning of the investment period.

- **Annualised Standard Deviation (ASD):** This is a risk measure showing the annualised deviation of returns from their long-term average.

$$ASD = \sqrt{\frac{S}{T}\sum_{t=0}^{T}(r_t - \overline{R})^2} \tag{8}$$

where

T: the number of trading days in the investment period.
S: the number of trading days in a year. We assumed $S = 250$.
r_t: the daily return of investments.
$\overline{R}$: the average daily return of the whole period.

- **Information Ratio (IR*):** This metric measures the amount of return for a given unit of risk.

$$IR^* = \frac{ARC}{ASD} \tag{9}$$

- **Maximum Drawdown(MD):** The maximum percentage drawdown during the investment period. We calculate the change from the highest point to the lowest point before a new

peak is established and take the greatest of such movements. A low MD is preferred as this means that losses from the investment are small. According Ryś and Ślepaczuk (2018), we can calculate MD using below formula:

$$MD = sup_{(x,y)\in\{[0,T]^2:x\leq y\}} \frac{E_x - E_y}{E_x} \tag{10}$$

where

$[0, T]$: investment period from 0 to T

x, y : instances in the investment period.

E_x, E_y: portfolio's value at the moments of x and y respectively.

- **Modified Information Ratio (IR**):** The metric adjusts the IR^* by considering maximum drawdown and the sign of ARC.

$$IR^{**} = IR^* \times ARC \times \frac{sign(ARC)}{MD} \tag{11}$$

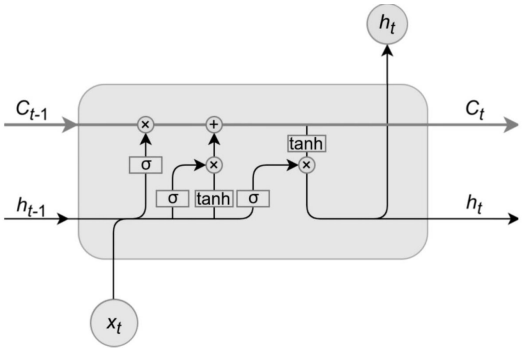
2.4 Methodology

2.4.1 LSTM Model and Hyperparameters

LSTM is an RNN network with long-term memory cells that can keep track of long-term trends. Because of this characteristic, LSTM is used in forecasting time series, especially when the time series show some trends.

An LSTM network consists of layers similar to a feed-forward neural network: the input layer, optional hidden layer(s), and output layer. LSTMs also have Recurrent Units to learn from sequential data. There are two primary operations in each standard recurrent unit: (i) combine the previous hidden state with the new input, and (ii) pass the combined info to the activation function and the next state. LSTMs' recurrent unit is more complex. It uses three gates (input gate, output gate, and forget gate) to decide which information to keep or discard. It also has a cell state to keep the long-term memory of the network. The diagram in Figure 3 shows how a LSTM's recurrent unit processes information:

Figure 3: Architecture of LSTM cell



Note: Referenced from Neural Networks LSTM (Matsumoto, 2019).

- Hidden state of the previous timestep h_{t-1} and the input of the current timestep x_t are combined and then passed to the three gates: forget gate, input gate, and output gate.
- At forget gate, the activation function is defined as:

$$f_t = \sigma(U_f.x_t + V_f.h_{t-1} + b_f) \tag{12}$$

where f_t is the activation function of the forget gate, b, U, V denote biases, input weights, and recurrent weights of the network cells. The function ranges from 0 to 1, and it sets to which extent the values in the previous cell states C_{t-1} will be discarded.

$$C_t^{'} = f_t.C_{t-1} \tag{13}$$

- The input gate identifies which information from the previous hidden state and current input will be added to the current cell state C_t:

$$i_t = \sigma(U_i.x_t + V_i.h_{t-1} + b_i) \tag{14}$$

$$C_t^{+} = tanh(U_c.x_t + V_c.h_{t-1} + b_c) \tag{15}$$

$$C_t = C_t^{'} + i_t.C_t^{+} \tag{16}$$

- The output gate is also a sigmoid activation function and uses inputs from the hidden state of the previous timestep h_{t-1} and the input of the current timestep x_t.

$$o_t = \sigma(U_o.x_t + V_o.h_{t-1} + b_o) \tag{17}$$

The output gate decides how the hidden state of the current timestep will be updated.

$$h_t = o_t.tanh(C_t) \tag{18}$$

In this research, we used KerasTuner (Keras, 2022) framework to identify the optimal specifications for our LSTM models. We identified three main groups of hyperparameters to tune:

- **Input data hyperparameters**

  Sequence length: The number of previous trading days to use as input to predict the return for the next trading day.

- **Model configuration hyperparameters**

  Number of LSTM layers: How many LSTM layers to include in the model. If we include too many layers, we can have the issue of overfitting and increasing training time.

  Number of neurons per layer: The number of nodes included in each LSTM layer.

  Activation function: We checked two options, either *tanh* or *relu* for activation function.

  Input dropout rate: Input dropout is used as a regularization method to reduce overfitting and improve the model performance. This is the rate where inputs are probabilistically excluded from updating weights during model training. We can also place a dropout layer after each LSTM layer, however, due to the computational constraint we did not incorporate this layer in the scope of this research. Future study can consider including the drop out layer in tuning process and examine if it can improve the model's performance.

  Learning rate: Defines how much the model's weights will change in response to the loss function's gradient change after each epoch. The learning rate is considered one of the most critical hyperparameters since it controls the training process. A low learning rate can help us make sure we do not miss the local minimum of the loss function; however, it will increase the training time. Setting a higher learning rate helps us train the model faster but might not reach the optimal weights. In the hyperparameter tuning process, we set the range of learning rates from low to high, and the optimal learning rate was chosen from trials with the lowest loss value.

  Loss function: We used custom loss function MADL (Michańków et al., 2022) to train the model. Unlike MSE, MADL does not focus on the accuracy of the point forecast but on the correctness of the sign and the value of possible loss after using the forecasted return.

$$MADL = \frac{1}{N}\sum_{t=1}^{N}(-1) \times sign(\hat{r_t} \times r_t) \times abs(r_t) \tag{19}$$

  where r_t is the actual return on day t, $\hat{r_t}$ is the predicted return. If the predicted return has the same sign as the actual return, the loss function is reduced by an amount of the actual

return. On the other hand, when the predicted return has the opposite sign with the actual return, the loss function increases by an amount of the actual return.

- **Model training hyperparameters**

  epoch: Number of times the whole training data set is fed through the LSTM networks. This hyperparameter is passed to the *max_epoch* argument of the Hyperband tuner.

  batch_size: Number of samples the LSTM network processes simultaneously.

We identified a search space for each hyperparameter and used Hyperband Tuner to search for the optimal value. We defined the early-stopping condition when the loss function did not show any improvement after 30 consecutive epochs. The Hyperband Tuner trains a large number of models for a few epochs and carries forward the top-performing half of models to the next rounds. The nearest integer to $1+log_{factor}(max\_epochs)$ determines the number of models to train in each round. By default, $factor$ is set to 3. This approach allows Hyperband Tuner to perform 5 to 30 times faster than Bayesian optimization methods in various deep-learning problems (Li et al., 2018).

Table 2: Hyperparameter search space of LSTM network

Hyperparameter	Initial Value Range	Extended Value Range
n_layers	From 1 to 5	From 1 to 5
n_units	[25, 50, 100, 128, 256]	[10, 25, 50, 100, 128, 256]
sequence_length	[5, 10, 20, 40, 60]	[5, 10, 20, 40, 60]
input_dropout	[0.01, 0.02, 0.1, 0.2]	[0.01, 0.02, 0.1, 0.2]
activation	[relu, tanh]	[relu, tanh]
learning_rate	[0.0002, 0.002, 0.02]	[0.0001, 0.0002, 0.002, 0.02]
loss_function	MADL	MADL
batch_size	[10, 32, 64]	[10, 32, 64]
epoch	[10, 20, 40, 80, 100]	[10, 20, 40, 80, 100]

Note: We started the tuning process with the initial value range. If the tuner returned the best value in either the upper or lower bound, we extended the range and checked whether the values outside of the range generated a better loss value.

Table 2 summarizes the search space of hyperparameters of the LSTM model. When the tuner selected either the upper or lower bound of the range as the optimal value, we extended the search range and reran the tuner with the new value range to ensure that we covered all scenarios. If the new model selected by the tuner had a lower loss than the previous model, we used the new model's configurations. Otherwise, if the new model had a greater loss, we kept the initial model's configurations selected by the tuner. In this research, we needed to extend the search range for the number of units and the learning rate after the first tuning. However, the new models selected by the tuner from the new value range did not improve the loss, and we kept the first model returned by the tuner.

2.4.2 Benchmark Strategies' Parameters

- **Simple Moving Average**

  Simple Moving Average is the most used of all technical indicators (Ellis & Parbery, 2005), and is presented by the following formula:

$$SMA_n = \frac{1}{n}\sum_{i=1}^{n} P_{t-i} \tag{20}$$

  where P_t is the index price of day t, and n is the length of the window to calculate the simple moving average.

  If the window is short, the SMA reflects the changes in underlying asset prices closely. Alternatively, a long average is less sensitive to the short-term fluctuations of the underlying asset price and highlights only the major trends. Whether using a long or short window in the SMA strategy will impact the number of trades and the profitability of the strategy.

  In order to choose a reasonable window length for the average, we ran 200 trials on the training data set, using window lengths starting from 5 and incrementing by 5 after each trial. In the end, we selected the window length that generates the best IR^{**} on the training data set. Table 3 summarizes the parameters that we trained and the value range we used to run the trials.

Table 3: Parameter tuning for SMA strategy

Parameter	Value Range
n	5 -¿ 1005
step	5
max_trials	200

Note: n - the window length to calculate the SMA. We ran 200 trials, starting from 5, and incremented the window length by a $step = 5$ after every trial up to 1005.

- **MACD**

  MACD is a technical indicator proposed by Gerald Appel more than 40 years ago and is still widely used by technical analysts today. The standard MACD is computed by subtracting the 26-day EMA from the 12-day EMA. The signal line is then derived by taking the 9-day EMA of the MACD line. When the MACD line crosses over the signal line, it indicates an uptrend and is a buy signal. Similarly, when the MACD line crosses under the signal line, it is a downtrend indicator, and we should exit the position (Wang & Kim, 2018).

  In the scope of this thesis, instead of using the 26-day EMA, 12-day EMA, and 9-day EMA, we fine-tuned all three hyperparameters to find the optimal values for the MACD strategy:

  - The window of slow EMA($slow$)

  - The window of fast EMA($fast$)

- The window of the signal line(m)

$$MACD_t = EMA_t^{fast} - EMA_t^{slow} \tag{21}$$

$$signal_t = EMA^m(MACD_t) \tag{22}$$

The ranges that we used to search for the optimal value of each parameters are summarized in Table 4.

Table 4: Parameters tuning for MACD strategy

Parameter	Value Range
$[EMA_{slow}, EMA_{fast}]$	[26,12], [52,24], [60,15], [90,10], [90,5], [120,5], [130,5]
EMA^m	5 -¿ 1005, each step = 5
max_trials	1400

Note: In each trial, we combined a pair of $[EMA_{slow}, EMA_{fast}]$ and a EMA^m, where m is the window length to calculate the EMA of the MACD line. In total, we ran 1400 trials for all the possible combinations.

We performed a grid search on different combinations of EMA_fast, EMA_slow and EMA^m, and selected the parameters that generated the best IR^{**} in the training data set.

- **RSI**

The RSI was developed in 1978 by J.Welles Wilder Jr. and is a popular momentum oscillator used to evaluate if the asset price changes indicate an overbought or oversold area. The RSI is determined by:

$$RSI_t = 100 - \left(\frac{100}{1 + \frac{Avg_{[t-n,t-1]}(gain)}{Avg_{[t-n,t-1]}(loss)}} \right) \tag{23}$$

where $Avg_{[t-n,t-1]}(gain)$ and $Avg_{[t-n,t-1]}(loss)$ is the average of gain and loss in the period n days prior to the day t^{th}. There are several variations of how to calculate the average. In his original research, Wilder (1978) used a smoothing factor which was a form of exponential average with the decay multiplier α defined as:

$$\alpha = \frac{1}{n} \tag{24}$$

Besides the original RSI, technical analysis softwares also offer option to calculate Cutler's RSI, which uses a simple moving average, and another option to use exponential average. In this research, we chose to use the exponential weighted average, which put more weight on the latest movement in comparison to the other two approaches. The calculation was achieved by using built-in function $ewn(span = n, adjust = False)$ from *pandas* Python's package. The

decay α multiplier is defined as below(Pandas, n.d.)

$$\alpha = \frac{2}{n+1} \tag{25}$$

To identify the trading signals, the RSI oscillator is compared with two thresholds: oversold and overbought. It is a common practice to apply the 30 and 70 for oversold and overbought thresholds. If the RSI crosses over the oversold threshold, it indicates a bullish sign, and when it crosses below the overbought threshold, it is a bearish sign. The thresholds are used as a reference to determine the strength of the trend.

We focused on fine-tuning the three parameters: oversold, overbought, and window length to calculate the average gain and loss. The ranges we used to tune these parameters are summarized in Table 5. The objective is to maximize IR^{**} during the training period.

Table 5: Parameter tuning for RSI strategy

Hyperparameter	**Value Range**
$[oversold, overbought]$	[10,90], [20,80], [30,70], [40,60], [15,85], [25,75], [35,65], [45,55]
n	5 -¿ 1005, each step = 5
max_trial	1600

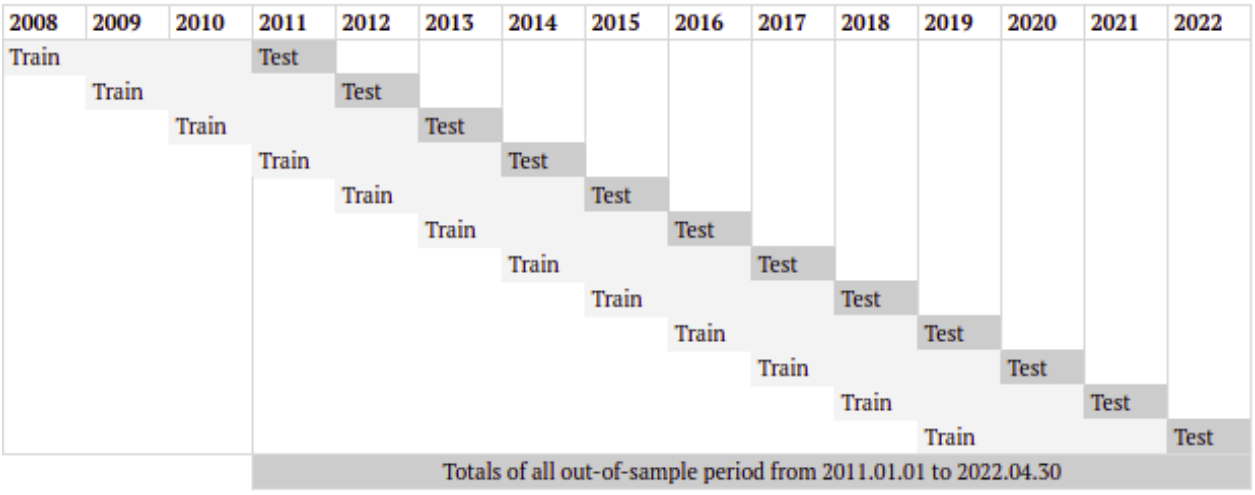
Note: In each trial, we combined a pair of $[oversold, overbought]$ and n - the window length to calculate average loss and gain. In total, we ran 1600 trials for all the possible combinations.

2.4.3 Defining Train and Test Data Set

We tested all strategies' performance over the out-of-sample period from 2011.01.01 to 2022.04.30. We used two approaches to split train and test data set:

- Approach (I): Train data set was fixed from 2001.01.01 to 2010.12.31, and the test data set is from 2011.01.01 to 2022.04.30
- Approach (II): We used a rolling window of 3-year for the train data set and 1-year for the test data set. After every year, we moved this window one year forward until the end of the data. In the end, we combined the result of each sub-period to get the result for the whole testing period from 2011.01.01 to 2022.04.30. Figure 4 visualizes this approach.

Figure 4: Rolling train-test data set - Approach (II)



Note: The results of the whole period from 2011.01.01 to 2022.04.30 was consolidated from all sub-test periods.

3 Results

3.1 Fixed Period

We trained and tuned the hyperparameters of all strategies using the data set from 2001.01.01 to 2010.12.31. Table 6 summarized the values chosen by the tuning and training process. These fine-tuned strategies were then used for trading S&P500 from 2011 to 2022.04.30.

The strategies based on traditional technical indicators generated fewer signals than the ones that used LSTM models' predictions. It can be explained by the fact that these indicators reflect relatively long-term trends. The SMA strategy was calculated using a 360-day window, while the MACD was derived from the difference of EMA_{52} and EMA_{24}, which was longer than the typically used value of 26 and 12, and the signal was smoothed over a period of 100 days. Similarly, even though the thresholds used in the RSI strategy are lower than the original thresholds of 30-70, the selected window length was 40 days - longer than the commonly used window of 14 days. Therefore, these strategies skip most of the short-term fluctuations in prices.

On the other hand, the LSTM model that used S&P500 daily returns as input generated more signals. The tuning process selected $sequence\_length = 10$, which meant that the next-day return predictions were determined based on the pattern of the previous ten trading days. As a result, the LSTM model strategies captured more short-term trends in its predictions.

Table 6: Hyperparameters

Hyperparameters	Best value	Parameters	Best value
LSTM		**SMA**	
n_layer	3	window length	360
n_nodes	100/128/25	**MACD**	
sequence	10	ema_fast	24
dropout	0.02	ema_slow	52
activation function	tanh	window length	100
learning rate	0.0002	**RSI**	
loss function	MADL	overbought threshold	40
batch_size	32	oversold threshold	60
epochs	80	window length	40

Note: Best values of the hyperparameters of the LSTM network and of the parameters of the technical indicators. Train period: 2001.01.01 to 2010.12.31. Test period: 2011.01.01 to 2022.04.30.

In the SMA strategy, we had long positions in more than 90% of the time, almost similar to the Buy & Hold strategy. However, the ARC of this strategy is only 5.5% ARC compared to 10.88% of Buy & Hold. This indicates that the strategy did not generate good trading signals. We had long positions in the downtrends and were late to enter the up-trend market.

In the MACD and RSI strategies, we stayed in the market more than 50% of the time. The RSI strategy generated only five trading signals and then held the index over a long period. The strategy did not generate a sell signal when the market plunged in March 2020; therefore, it had the same maximum drawdown as the buy and hold strategy. The MACD strategy was able to detect and avoid this significant downturn and is the strategy that had the lowest volatility measures.

LSTM generated more trading signals and only held long positions for 26.87% of the time. This means that most of the trades were only for short-term periods. Figure 5 shows that the LSTM models missed the overall increasing trend from March 2020 to 2022. The market experienced higher short-term volatility in this period, and it impacted the predictions of the LSTM model that use previous 10-day daily returns as input.

When used alone, all of the strategies were unable to outperform the Buy & Hold strategy. Since each strategy has its own advantages, we combined the strengths of each strategy to find the optimal model. We pointed out earlier that the trading signals generated by the SMA strategy were not efficient. In addition, the MACD indicator also encapsulates similar trending properties of the time series as the SMA. Therefore, we added RSI, MACD, and the MACD signal line together with the previous 10-day daily returns to the input layer of the LSTM network.

To tune this LSTM Extended model, we used the same hyperparameters obtained from the LSTM model tuning process. We trained the parameters of the input technical indicators as specified in Table 7. We trained n - the window length used to calculate the average of gain and loss for deriving the RSI. For the MACD, we trained three parameters: the EMA_{slow}, EMA_{fast} and EMA^m, in which m is the window length to compute the MACD signal line.

Table 7: Parameters of MACD and RSI used in the LSTM Extended model

Parameters	Value range	Best value
MACD		
$[EMA_{slow}, EMA_{fast}]$	[26,12], [52,24], [60,15], [90,10], [90,5], [120,5], [130,5]	[26,12]
m	5 - 1000, each step = 1	964
RSI		
n	5 - 1000, each step = 1	17

Note: Train period: 2001.01.01 - 2010.12.31. Test period: 20112.01.01 - 2022.04.30. In RSI, we trained n, the window length to calculate RSI. Since we did not identify the trading signals from RSI solely, we did not need to use and tune the overbought and oversold thresholds.

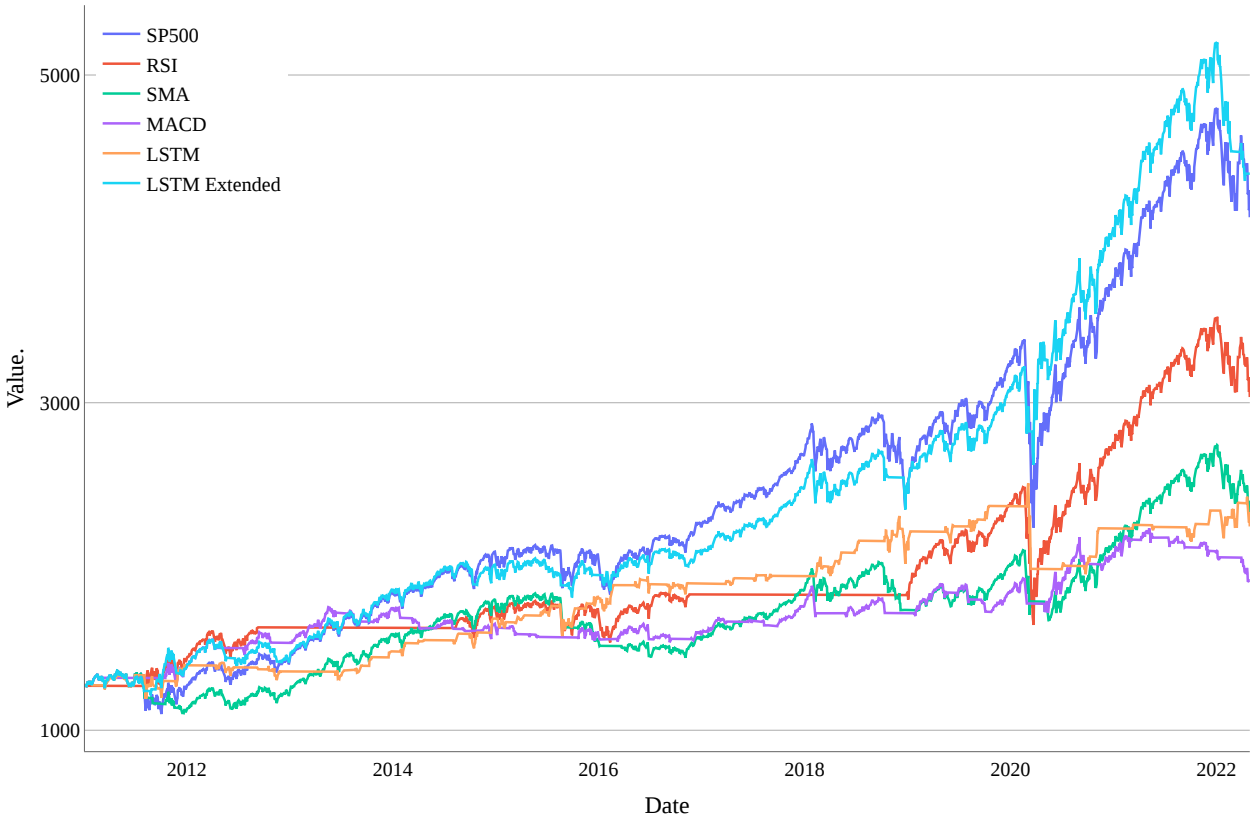
The results in Table 8 showed that this LSTM Extended model improved the performance significantly. The LSTM Extended model was able to capture the significant downtrend in March 2020 and exited the long position on time to avoid further loss. The strategies outperformed Buy & Hold benchmark strategy from March 2020, with a maximum drawdown of 18.65% compared to 33.92% of the Buy & Hold strategy. The strategy also has a slightly higher ARC (11.49%) and better IR^* and IR^{**} ratio. Equity lines for all strategies are compared in Figure 5, where we could see the strategy using LSTM Extended model had the highest equity line.

Table 8: Performance metrics of the strategies.

	ARC	ASD	IR^*	MD	IR^{**}	nTrades	%long pos.
Buy & Hold	10.88%	17.20%	0.63	33.92%	0.20	2	100%
SMA	5.48%	13.27%	0.41	21.42%	0.11	58	90.70%
MACD	3.67%	9.45%	0.39	14.64%	0.1	80	53.65%
RSI	7.92%	15.04%	0.53	33.92%	0.12	6	59.13%
LSTM	5.11%	10.93%	0.47	22.14%	0.11	148	26.87%
LSTM Extended	11.49%	14.00%	0.82	18.65%	0.51	80	91.16%

Note: LSTM Extended is the strategy that combines daily returns, MACD, and RSI in the input layer. The LSMT Extended strategy outperformed all other strategies considered in this research. Train period: 2001.01.01-2010.12.31. Test period: 2011.01.01 - 2022.04.30

Figure 5: Equity lines of strategies on the fixed train and test data set



Note: SP500 represents the Buy & Hold strategy for S&P500 index. LSTM Extended is the strategy that combines daily returns, MACD, and RSI in the input layer. Train period: 2001.01.01 - 2010.12.31. Test period: 2011.01.01 - 2022.04.30.

3.2 Rolling Train and Test Window

We checked the robustness of the models on the rolling train-test window as defined in Figure 4. The hyperparameters of the LSTM models were obtained from the tuning process performed on data from 2001.01.01 to 2010.12.31, and these hyperparameters were fixed over the whole testing period. Table 6 and Table 7 summarizes the values of these hyperparameters. Meanwhile, we fitted LSTM models' parameters every year starting from 2011.01.01 using the last three years of data as training data set. We also used the same approach to update the parameters of the benchmark strategies, and the values of these parameters are summarized in Table 9.

Table 9: Parameters of benchmark strategies in rolling train - test window

Parameter	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022*
SMA												
window length	735	45	495	470	220	690	715	465	175	695	715	460
MACD												
ema_fast	12	12	5	15	24	10	5	12	12	5	5	24
ema_slow	26	26	130	60	52	90	120	26	26	130	130	52
window length	40	25	715	30	845	235	240	15	25	285	300	145
RSI												
oversold	40	30	45	30	45	40	40	40	45	40	45	25
oversold	60	70	55	70	55	60	60	60	55	60	55	75
window length	530	10	10	15	10	10	10	10	110	10	250	385

* Until 2022.04.30

Note: LSTM models used the same hyperparameters as in Table 6 and Table 7 across all periods. The parameters of LSTM networks were updated when we fitted the network with rolling train data set.

Table 10 showed the performance of all strategies when the parameters were updated every year. Compared to using a fixed train and test data set, the benchmark strategies have similar IR^{**}, apart from the strategy using RSI. In this scenario, the RSI strategy generated 66 trading signals, compared to 5 signals previously. However, the percentage of long positions is only 21.96%, which means that most of the trades were short-term. Additionally, the maximum drawdown of the RSI strategy increased from 33.92% to 37.13%, while ARC decreased from 7.92% to 5.2%. In the fixed train-test approach, RSI was computed over a window of 40 days, while in 2021 and 2022, this window was selected as 250 and 385, respectively. This caused the RSI strategy to exit the market too early during the uptrend from 2021 to early 2022, which we observed from the equity line in Figure 6.

Table 10: Performance metrics of the strategies.

	ARC	**ASD**	IR^*	**MD**	IR^{**}	**nTrades**	**%long pos.**
Buy & Hold	10.88%	17.20%	0.63	33.92%	0.20	2	100%
SMA	6.28%	14.63%	0.43	26.69%	0.10	56	91.58%
MACD	4.27%	9.88%	0.43	20.15%	0.09	120	61.75%
RSI	5.20%	10.48%	0.50	37.13%	0.07	66	21.96%
LSTM	5.08%	10.90%	0.47	22.14%	0.11	148	26.23%
LSTM Extended	12.23%	10.60%	1.15	12.10%	1.17	160	46.20%

Note: LSTM Extended is the strategy that combines daily returns, MACD, and RSI in the input layer. The LSTM Extended strategy outperformed all other strategies considered in this research. We used Approach (II) to split train - test data set, as visualized in Figure 4.

The SMA strategy used a long window length in most periods, similar to the fixed train-test approach. As a result, this strategy had very similar results in all terms IR^*, IR^{**}, number of trades, and percentage of long positions.

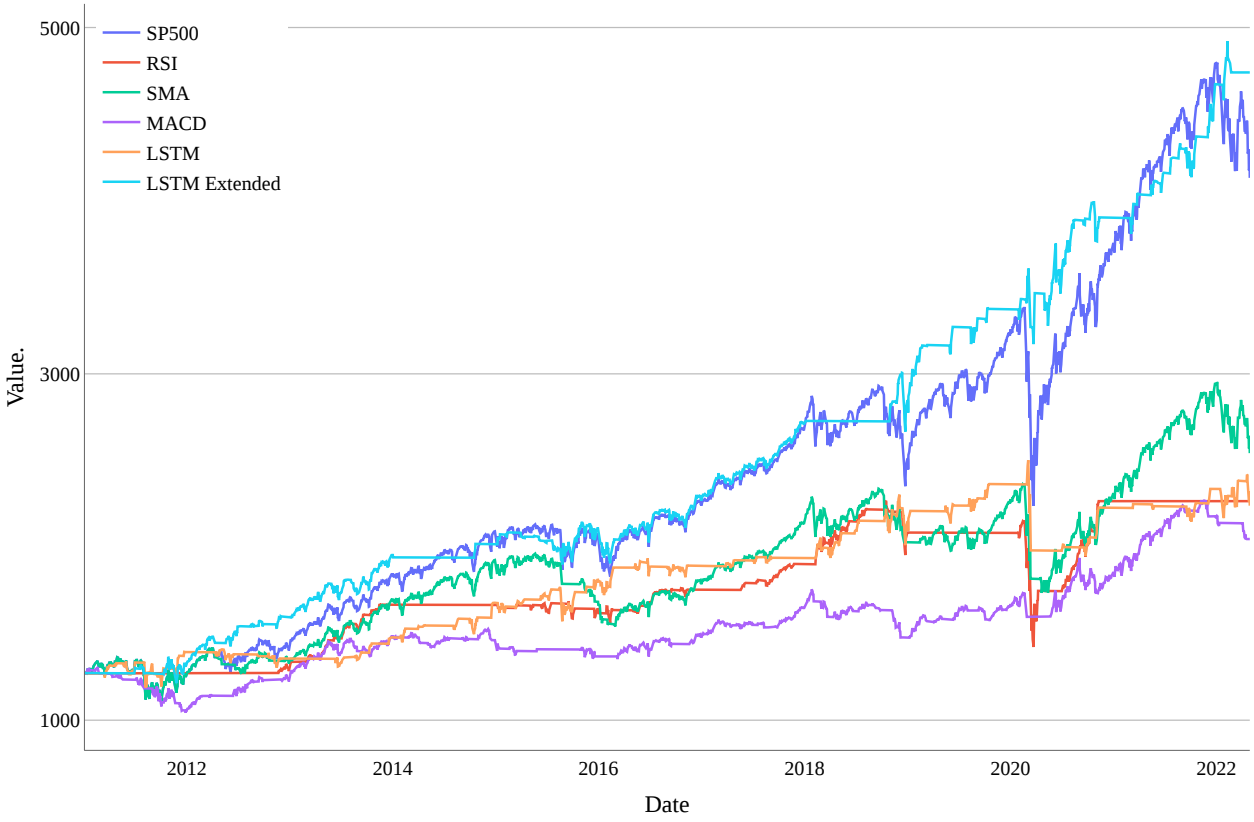
When fitted dynamically, the MACD strategy generated more trade signals (119 trades) than

the fixed approach (79 signals). The percentage of long positions increased from 53.65% to 61.75%. However, the increase in ARC was offset by the increase in MD, which resulted in similar IR^{**}.

The performance of the LSTM model that used 10-day daily returns as input was also indifferent across the two approaches. However, the LSTM Extended network that used 10-day daily returns and technical indicators as input improved significantly.

As seen in Figure 6, the LSTM Extended strategy outperformed Buy & Hold strategy in multiple periods: 2012 to mid-2014, 2019 to 2021, and 2022. The strategy also had the least MD (12.1%) because it was able to anticipate and minimize the impact of the 2020's downturn. The models generated more trades (160) compared to the fixed approach (79) and had long positions 46.2% of the time, lower than in the fixed approach (91.16%). This means that when we fed more updated data to the network, it was able to trade on the short-term fluctuations more efficiently.

Figure 6: Equity lines of strategies in rolling train and test window



Note: S&P500 represents the Buy & Hold strategy for S&P500 index. LSTM Extended is the strategy that combines daily returns, MACD, and RSI in the input layer. The LSMT Extended strategy outperformed Buy & Hold strategy in multiple periods: 2011.07.27 to 2014.06.05, 2015.08.24 to 2018.01.04, 2018.10.28 to 2021.02.08, and 2022.01.14 to 2022.04.30.

The results of the LSTM extended in the rolling train - test window confirm the robustness of this model to the type and length of the training window. The model was able to outperform the Buy & Hold and other benchmark strategies in both circumstances: when we did not refit the

network's parameters and when we fitted its parameters dynamically with the new data.

4 Sensitivity Analysis

We evaluated the sensitivity of the LSTM Extended model performance with regard to factors that played important roles in the model tuning process. The impact of the following factors on the model's performance was investigated: loss function, sequence length, batch size, the technical indicators, and the length of the rolling train - test window. For each scenario, we only changed the selected hyperparameter while keeping everything else the same, and the below scenarios were considered:

- Use MSE as loss function instead of MADL
- Sequence length
    - Decrease sequence length to 5 days
    - Increase sequence length to 20 days
- Batch size
    - Decrease batch size to 16 days
    - Increase batch size to 64 days
- Technical Indicators
    - Increase EMA_{fast} and EMS_{slow} to [24, 52]
    - Decrease window length to calculate MACD signal line to 360
    - Increase window length to calculate RSI to 30
- Rolling train window
    - Increase window of train data set to previous 5 years
    - Decrease window of train data set to previous 2 years
- Rolling test window
    - Increase window of test data set to next 2 years
    - Decrease window of train data set to next 6 months

We generated the predictions for day t on day $t-1$ and had a general assumption that we made the trade for day t at day t's closed price. However, in reality, there are scenarios where we can still enter a position for the day t at day $t-1$'s closed price, for example, during day $t-1$'s after-hours trading sessions. Therefore, we also evaluated the performance of the LSTM Extended model under this circumstance:

- Enter position at day t-1's closed price

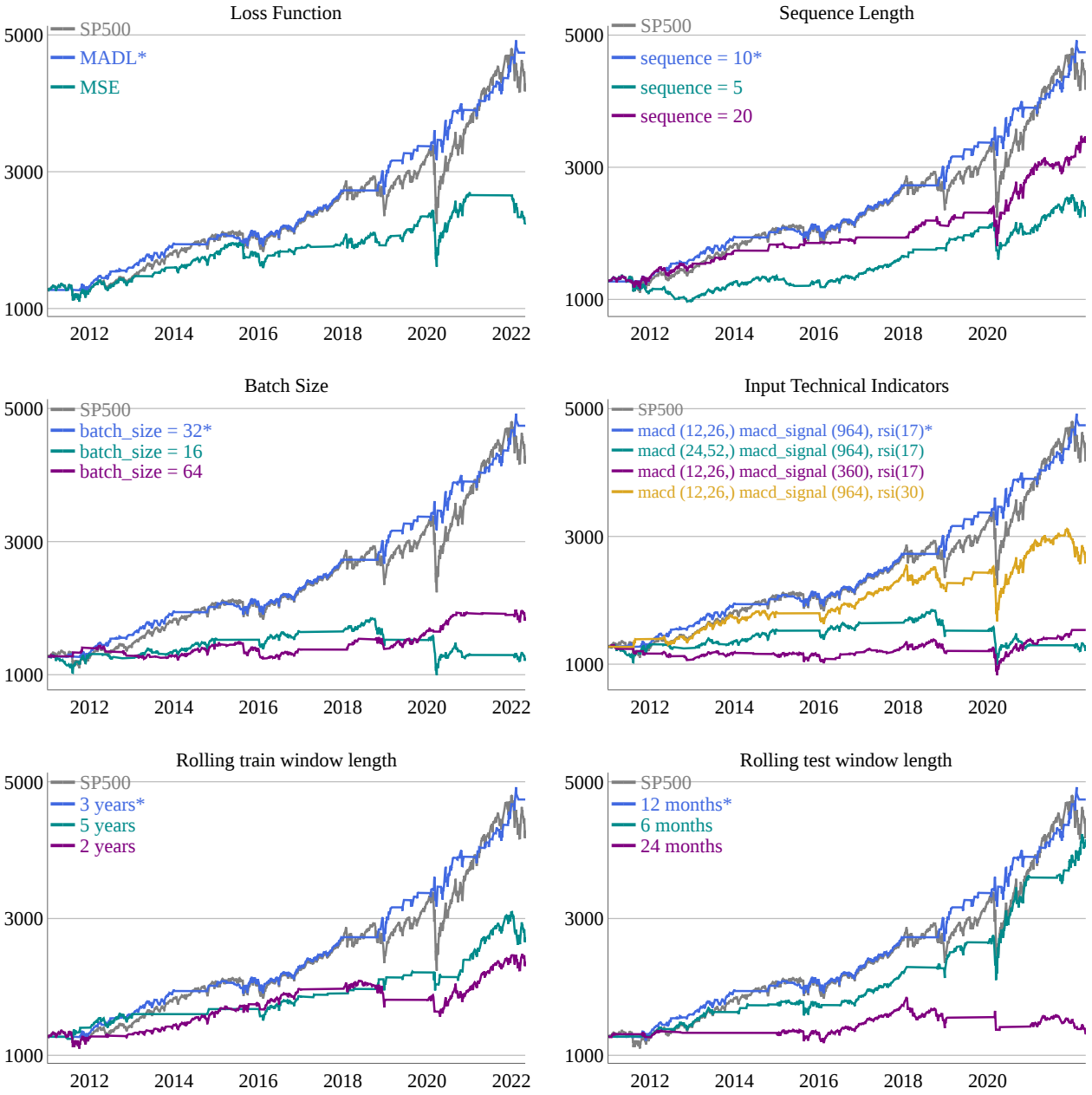
Table 11 and Figure 7 summarize the result of the sensitivity analysis of all the scenarios. Overall, the hyperparameters selected by the tuning process appear to be the optimal values, but at the same time, the strategy is not robust to any changes in the selected hyperparameters.

Table 11: Sensitivity Analysis for the best LSTM Extended model

	ARC	ASD	IR^*	MD	IR^{**}	nTrades	%long pos.
Panel A - Loss function							
Buy & Hold	10.9%	17.2%	0.63	33.9%	0.2	2	100%
Loss function = MADL*	12.23%	10.6%	1.15	12.1%	1.17	160	46.2%
Loss function = MSE	5.11%	15.33%	0.33	33.92%	0.05	98	63.56%
Panel B - Sequence length							
Buy & Hold	10.9%	17.2%	0.63	33.9%	0.2	2	100%
Sequence = 5	5.04%	13.22%	0.38	29.21%	0.07	140	67.00%
sequence = 10*	12.23%	10.6%	1.15	12.1%	1.17	160	46.2%
Sequence = 20	9.16%	12.77%	0.72	24.83%	0.26	150	42.90%
Panel C - Batch size							
Buy & Hold	10.9%	17.2%	0.63	33.9%	0.2	2	100%
batch_size = 16	5.14%	12.99%	0.40	31.45%	0.06	148	60.96%
batch_size = 32*	12.23%	10.6%	1.15	12.1%	1.17	160	46.2%
batch_size = 64	3.05%	9.65%	0.32	18.37%	0.05	144	45.67%
Panel D - Input technical indicator							
Buy & Hold	10.9%	17.2%	0.63	33.9%	0.2	2	100%
macd (12,26,)							
macd_signal (964), rsi(17)*	12.23%	10.6%	1.15	12.1%	1.17	160	46.2%
macd (24,52,)							
macd_signal (964), rsi(17)	3.14%	12.89%	0.24	45.58%	0.02	152	62.78%
macd (12,26,)							
macd_signal (360), rsi(17)	1.66%	13.64%	0.12	40.17%	0.01	162	50.16%
macd (12,26,)							
macd_signal (964), rsi(30)	6.3%	14.17%	0.44	34.76%	0.08	154	65.8%
Panel E - Length of rolling training window							
Buy & Hold	10.9%	17.2%	0.63	33.9%	0.2	2	100%
5 years	6.57%	9.65%	0.68	15.73%	0.28	86	32.76%
3 years *	12.23%	10.6%	1.15	12.1%	1.17	160	46.2%
2 years	5.25%	12.91%	0.41	25.56%	0.08	148	58.65%
Panel F - Length of rolling test window							
Buy & Hold	10.9%	17.2%	0.63	33.9%	0.2	2	100%
6 months	10.59%	11.97%	0.88	23.73%	0.39	132	46.72%
12 months*	12.23%	10.6%	1.15	12.1%	1.17	160	46.2%
24 months	0.26%	10.17%	0.03	29.34%	0.00	126	46.23%
Panel G - Open position day							
Buy & Hold	10.9%	17.2%	0.63	33.9%	0.2	2	100%
Open position on t-1	10.7%	10.13%	1.06	20.59%	0.55	160	46.2%
Open position on t*	12.23%	10.6%	1.15	12.1%	1.17	160	46.2%

(*) Base case. The panels describe model's performance with respect to different loss function, sequence length, batch size, input technical indicators, and length of rolling train-test window. For each scenario, we only changed the selected hyperparameter while keeping everything else the same.

Figure 7: Equity Lines of the Scenarios in Sensitivity Analysis



(*) Base case. Note: SP500 stands for the Buy & Hold strategy. The plot shows the equity lines in the period from 2011.01.01 to 2022.04.30

Panel A of Table 11 showed that the IR^{**} was decreased drastically from 1.17 to 0.05 when we used MSE to train the LSTM network. This proves the importance of using the appropriate loss function customized to the trading strategy, which emphasizes the direction of the changes rather than the exact values.

The impact of sequence length on the model's performance is shown in Panel B of Table 11. The sequence length selected by the tuning process was the best value. When we reduced or increased the length, IR^{**} decreased to 0.07 and 0.26, respectively. However, when the sequence length is

20, the strategy had ARC and ASD more or less comparable to the base case. The equity line of the strategy that used sequence length = 20 was almost flat from 2014 to 2018, which means that we stayed out of the market most of the time. However, we were in the market during 2020. This indicates that the trading signals generated by this strategy were inefficient because we missed the uptrend and could not avoid the downtrend.

Batch size also played an important role. In the scenarios where the batch size was changed to 16 and 64, IR^{**} decreased to 0.06 and 0.05. Other performance metrics also got worse than in the base case and the benchmark Buy & Hold strategy. A smaller batch size seemed to generate better results than a larger batch size (Panel C of Table 11), and the value selected by the tuning process was still the best one.

Panel D of Table 11 presents the scenarios where we altered how the technical indicators were computed. Because the training process is time-consuming, we only considered three scenarios, and each time we changed one parameter of the technical indicators. The worst scenario is when we shortened the window length to calculate the MACD signal line from 964 to 360. The IR^{**} of this strategy is only at 0.01, and MD is 40%, larger than the MD when we buy and hold the index. When the EMA_{fast} and EMA_{slow} was increased from [12, 26] to [24,52] the strategy also performed poorly. The ASD of the strategy was close to the base case; however, it had a lower ARC and greater MD, resulting in an IR^{**} of 0.02. Of all the considered scenarios, extending the window length in RSI from 17 to 30 produced reasonable results. The strategy had IR^{*} of 0.44, close to the Buy & Hold strategy (0.63). However, it had larger MD, therefore, lower IR^{**}.

In Panel E of Table 11, we analyzed how the strategy behaved if we used a different length of rolling training window. When we extended the training data set from the previous three years to five years, the strategy slightly outperformed the benchmark Buy & Hold strategy. It performed better than the benchmark in all risk measures, such as ASD (9.65% compared to 17.2%) and MD (15.73% compared to 33.9%). However, the strategy generated only 85 trades and had long positions for only 32.76% of the time, resulting in lower ARC (6.57% compared to 10.9%). Nonetheless, the strategy was still worse than in the base case. Shortening the rolling train window from 3 to 2 years had a notable negative impact on the performance.

Similarly, we experimented the strategy with a shorter and longer rolling test window. The results are summarized in Panel F of Table 11. If we refitted the model every two years instead of 1 year as in the base case, the strategy hardly generated any profit. The equity line of this scenario in Figure 6 showed that apart from the period from 2015 to 2019, we mostly stayed out of the market. In the remaining scenario, we refitted the model more frequently every six months. This strategy was able to outperform the benchmark Buy & Hold strategy in terms of IR^{*} and IR^{**}. The equity line of this strategy was less fluctuating compared to the Buy & Hold equity line, and it did not have such a big drop in March 2020 as the Buy & Hold. However, the base case strategy was still the dominating strategy.

In the last scenario, we recalculated the strategy's performance metrics if we could make after-hours transactions to buy and sell the index at the day $t-1$'s closed price. If models generated a

buy signal and we already had a long position on the day $t-1$, we continued keeping the position, and the portfolio value at day t grew at the same rate as the return of day t. Otherwise, if we did not have any position on the day $t-1$, the portfolio value at the end of the day t also grew by r_t because we entered the long position at day $t-1$'s closed price. The equity line, in this case, was calculated as follows:

$$E_t = E_{t-1}(1 + r_t \times pos_t) \tag{26}$$

where:

pos_t : trading position we actually had on day t, which was generated by the model at the end of day $t-1$. The position was 1 if we had a long position and 0 if we did not have a position.

r_t: daily simple return of the index of day t.

E_{t-1}: portfolio value at the end of day $t-1$

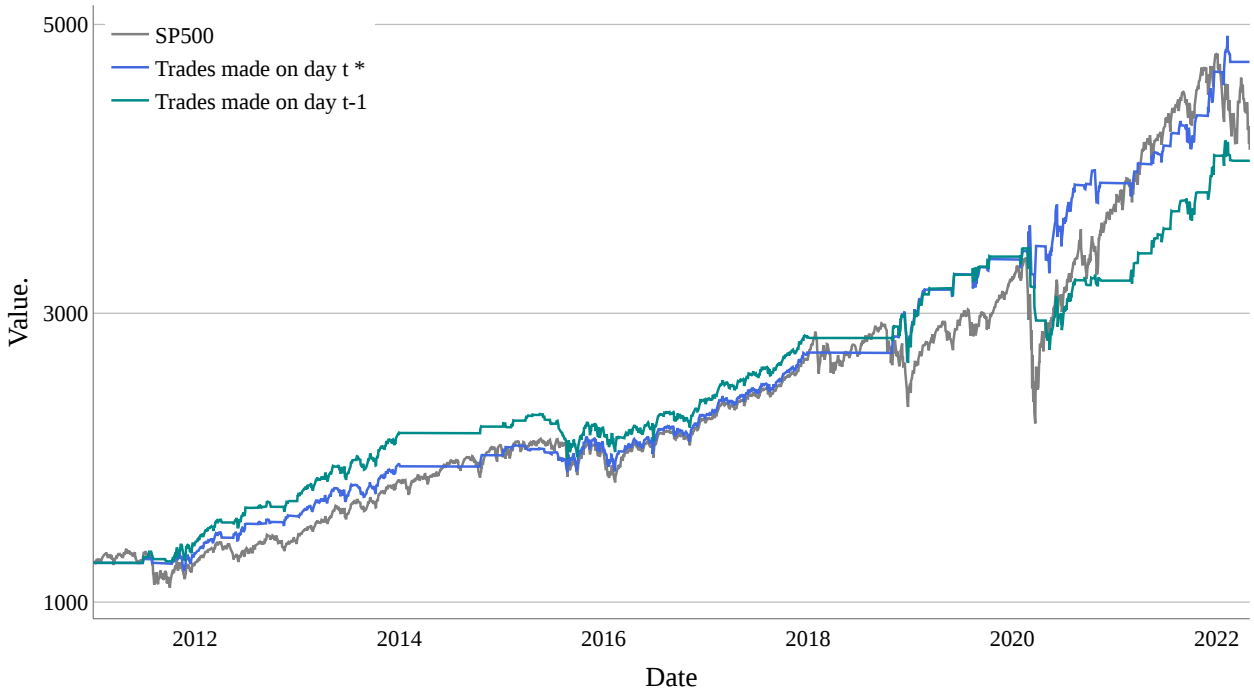
We also applied a transaction fee:

$$E_t = E_{t-1}(1 + r_t \times pos_t - abs(pos_t - pos_{t-1}) \times fee) \tag{27}$$

where fee is the transaction fee percentage.

Figure 8 compares the equity line generated by the scenario where we could make after-hours trade with the base case and the benchmark Buy & Hold. From 2011 until the beginning of 2020, the strategy was able to outperform. However, in the major drop in 2020, we experienced a more extensive loss in comparison to the base case. The IR^{**} ratio in this scenario was better compared to the Buy & Hold strategy; however, it was not as good as in the base case.

Figure 8: Equity Lines of after-hours trade



* Base case. Note: SP500 stands for the Buy & Hold strategy. The plot shows the equity lines in the period from 2011.01.01 to 2022.04.30

The sensitivity analysis demonstrated that the LSTM Extended model chosen by the hyperparameter tuning process was not robust to changes. The results confirmed the challenges in applying neural networks that the quality of the model is highly dependent on the hyperparameter configurations. Because of this, defining an appropriate tuning process is very important. And this process is different depending on the underlying assets, as well as which period to test.

Conclusion

The research aimed to investigate the application of LSTM networks in predicting and trading the S&P500 index compared to other technical indicators. We used Keras Hyperband Tuner to find the optimal hyperparameters of the LSTM networks. We combined the previous 10-day daily returns with the MACD and RSI indicators in the inputs to train the network. We specifically focused on hyperparameter tuning in order to select the appropriate configurations for the network. The model was tested in two approaches: using a fixed train-test data set and using a rolling window of a 3-year train and 1-year test data set. The strategy using forecasts from our best LSTM networks was able to outperform the Buy & Hold strategy and other strategies that used technical indicators alone in both approaches. We also performed a sensitivity analysis to test the robustness of the model with regard to changes in the main hyperparameters. The analysis proved that the LSTM performance was not robust and was sensitive to changes of any hyperparameter and input data.

Based on the results of the research, we were able to verify the hypothesis set at the beginning of the research:

(RH1) *The signals from algorithmic investment strategy using predictions from the LSTM model consisting only daily returns in its input layer are more efficient than using technical analysis indicators.*

(RH2) *The signals from algorithmic investment strategy using predictions from the LSTM model combining daily returns and technical analysis indicators in its input layer are more efficient than using only technical indicators.*

We rejected (RH1) because the strategy using forecast from the LSTM model trained with only daily returns in input layer did not outperform Buy & Hold and other strategies based on technical indicators. We were not able to reject (RH2) since the strategy using the LSTM Extended model outperformed all of the benchmark strategies once we incorporated other technical indicators in the input layer. However, the sensitivity analysis showed that the signals from the LSTM Extended model were not robust. Instead, they were highly dependent on the model configurations such as which loss function was used and batch size. The results were also sensitive to the input data used to train the model. When we altered either the sequence length of input, the parameters used to calculate the input technical indicators, or length of rolling train-test window, the results changed significantly.

There are more possibilities to extend the research in order to improve the performance of the LSTM-based strategy. The first is to explore the combination of LSTM with different technical indicators besides RSI and MACD. The hyperparameter tuning process can be improved as well. In this research, we used the same learning rate throughout the training. We can adapt the techniques that update the learning rate during the training, such as learning rate annealing (Nakamura et al., 2021) or cyclical learning rates (Smith, 2015). We learned that the performance was sensitive to the loss function used to train the model. Therefore, the next research direction can be exploring different loss functions customized to the trading strategies.

References

Anderson, B., & Li, S. (2015). An investigation of the relative strength index. *Banks and bank systems*, *10*(1), 92–96.

Bartkus, C. (2018). Using the relative strength index for active investments in the foreign exchange market. *Modern Management Review*, *23*(4).

Bhandari, H. N., Rimal, B., Pokhrel, N. R., Rimal, R., Dahal, K. R., & Khatri, R. K. (2022). Predicting stock market index using lstm. *Machine Learning with Applications*, *9*, 100320.

Chong, T. T.-L., & Ng, W.-K. (2008). Technical analysis and the london stock exchange: Testing the macd and rsi rules using the ft30. *Applied economics letters*, *15*(14), 1111–1114.

Ellis, C. A., & Parbery, S. A. (2005). Is smarter better? a comparison of adaptive, and simple moving average trading strategies. *Research in international business and finance*, *19*(3), 399–411.

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

Grudniewicz, J., & Ślepaczuk, R. (2021). Application of machine learning in quantitative investment strategies on global stock markets. *Working Papers of Faculty of Economic Sciences, University of Warsaw*, *WP 23*(371).

Heaton, J. B., Polson, N. G., & Witte, J. H. (2017). Deep learning for finance: Deep portfolios. *Applied stochastic models in business and industry*, *33*(1), 3–12.

Hoang Hung, N. (2016). Various moving average convergence divergence trading strategies: A comparison. *Investment management financial innovations*, *13*(2), 363–369.

Kanwal, A., Lau, M. F., Ng, S. P., Sim, K. Y., & Chandrasekaran, S. (2022). Bicudnnlstm-1dcnn — a hybrid deep learning-based predictive model for stock price prediction. *Expert systems with applications*, *202*, 117123.

Keras. (2022, June). *Keras tuner documentation.* https://keras.io/keras_tuner/

Kijewski, M., & Ślepaczuk, R. (2020). Predicting prices of sp500 index using classical methods and recurrent neural networks. *Working Papers of Faculty of Economic Sciences, University of Warsaw*, *WP 27*(333).

Kim, S., & Kang, M. (2019). Financial series prediction using attention lstm. https://arxiv.org/pdf/1902.10877.pdf

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, *18*, 1–52.

Lv, J., Wang, C., Gao, W., & Zhao, Q. (2021). An economic forecasting method based on the lightgbm-optimized lstm and time-series model. *Computational intelligence and neuroscience*, *2021*, 1–10.

M, H., E.A., G., Menon, V. K., & K.P., S. (2018). Nse stock market prediction using deep-learning models [International Conference on Computational Intelligence and Data Science]. *Procedia Computer Science*, *132*, 1351–1362. https://doi.org/https://doi.org/10.1016/j.procs.2018.05.050

Matsumoto, F. R. N. (2019). *Neural netwroks lstm.* https://medium.com/turing-talks/turing-talks-27-modelos-de-predi%5C%C3%5C%A7%5C%C3%5C%A3o-lstm-df85d87ad210

Michańków, J., Sakowski, P., & Ślepaczuk, R. (2022). Lstm in algorithmic investment strategies on btc and sp500 index. *Sensors (Basel, Switzerland)*, *22*(3), 917.

Nakamura, K., Derbel, B., Won, K.-J., & Hong, B.-W. (2021). Learning-rate annealing methods for deep neural networks. *Electronics (Basel)*, *10*(16), 2029.

Nor, S. M., & Wickremasinghe, G. (2014). The profitability of macd and rsi trading rules in the australian stock market. *Investment management financial innovations*, *11*(4).

Pandas. (n.d.). *Pandas.pydata.org. 2022. pandas.dataframe.ewm — pandas 1.4.2 documentation.* https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.ewm.html

Pradipbhai, N. P. (2013). Comparison between exponential moving average based macd with simple moving average based macd of technical analysis. *International Journal of Scientific Research*, *2*(12).

Ryś, P., & Ślepaczuk, R. (2018). Machine learning methods in algorithmic trading strategy optimization – design and time efficiency. *Central European Economic Journal*, *5*(52), 206–229. https://doi.org/doi:10.1515/ceej-2018-0021

Sang, C., & Di Pierro, M. (2019). Improving trading technical analysis with tensorflow long short-term memory (lstm) neural network. *The Journal of finance and data science*, *5*(1), 1–11.

Smith, L. N. (2015). Cyclical learning rates for training neural networks.

Wang, J., & Kim, J. (2018). Predicting stock price trend using macd optimized by historical volatility. *Mathematical problems in engineering*, *2018*, 1–12.

Wilder, J. (1978). *New concepts in technical trading systems.* Trend Research. https://books.google.lu/books?id=WesJAQAAMAAJ

University of Warsaw
Faculty of Economic Sciences
44/50 Długa St.
00-241 Warsaw
www.wne.uw.edu.pl