Applied Intelligence (2021) 51:7177–7195
https://doi.org/10.1007/s10489-021-02249-x

Two robust long short-term memory frameworks for trading stocks

Dušan Fister[1] · **Matjaž Perc**[2,3,4] · **Timotej Jagrič**[1]

Accepted: 28 January 2021 / Published online: 27 February 2021

Abstract
This paper aims to find a superior strategy for the daily trading on a portfolio of stocks for which traditional trading strategies perform poorly due to the low frequency of new information. The experimental work is divided into a set of traditional trading strategies and a set of long short-term memory networks. The networks incorporate general and specific trading patterns, where the former takes into account the universal decision factors for trading across many stocks, while the latter takes into account stock-specific decision factors. Our research shows that both long short-term memory networks, regardless of whether they are based on universal or stock-specific decision factors, significantly outperform traditional trading strategies. Interestingly, however, on average neither has the edge compared to the other, thus remaining ambivalent as to whether universality or specificality is to be preferred when it comes to designing long short-term memory networks for optimal trading.

Keywords Long short-term memory · Algorithmic trading · Mechanical trading system · Portfolio of stocks

1 Introduction

Stock trading is a market exchange process that involves transactions of company stocks among shareholders. These transactions commonly encompass buying and selling stocks, with some holding period between. The aim of stock trading is to obtain a profit, either dividend payouts or capital profit, where the first is expressed as actual earnings that the company is willing to pay out and the latter as the difference between the buying and selling prices. Stock trading is about making the right decision at the right time, either focused on a single stock or broadened over a portfolio of stocks, i.e., multiple stocks held simultaneously. Importantly, the latter variant diversifies (decreases) the unsystematic risk and is thus commonly employed in practice.

Trading decisions by shareholders are traditionally made by either a top-down or a bottom-up approach. A top-down approach sequentially follows the macroeconomic, industry sector and then the individual company (fundamental) analysis [11], while a bottom-up approach, also called stock picking, does that in reverse. Macroeconomic, sectoral and fundamental analyses require extensive information and understanding; trading decisions derived from such processes are typically very rare, either because of the low frequency of some information, e.g., quarterly in the best case, or because of the lengthy analysis process itself. As an alternate to the top-down or bottom-up approaches, technical analysis relies heavily on only past and current asset prices. Defined as chartists, technical analysts analyze supply and demand functions continuously to identify investor behavior and sentiment and execute immediate actions. Due to its rapidity, technical trading can be defined as a high-frequency trading (HFT) process; these processes are currently gaining in popularity among investors. For

✉ Timotej Jagrič
timotej.jagric@um.si

Dušan Fister
dusan.fister1@um.si

Matjaž Perc
matjaz.perc@um.si

1 Faculty of Economics and Business, University of Maribor, Razlagova 14, 2000 Maribor, Slovenia

2 Faculty of Natural Sciences and Mathematics, University of Maribor, Koroška cesta 160, 2000 Maribor, Slovenia

3 Department of Medical Research, China Medical University Hospital, China Medical University, Taichung, Taiwan

4 Complexity Science Hub Vienna, Josefstädterstraße 39, 1080 Vienna, Austria

example, Zhang [62] reported that HFT in 2010 accounted for more than 70% of transactions; we can assume that this ratio is currently much higher. Interestingly, in the HFT domain, raw financial data are supplied directly from trading platforms to high-performance computers and servers that run automated and computerized decision algorithms. These decode data and synchronously and proactively intervene on the stock market. The HFT process is completely automated, regardless of whether the decision algorithms are a very simple set of rules or extremely complex machine learning algorithms. Both share a common purpose, that is, to identify or extract past and recurrent hidden patterns that might be beneficial for future intervention.

The problem and hypothesis that this article addresses is whether it is possible to earn higher-than-normal (excess) profits on an open and transparent stock market using innovative decision algorithms. Economic theory on efficient markets strongly teaches us that the answer to this question is no – Fama [16] has stated that when the markets are efficient, (1) asset prices fully reflect all available information, and (2) new information is quickly incorporated into the asset price, which means that the only way the investor can increase expected profit is to take higher risk. Since new information comes in an erratic or unpredictable way, asset prices are erratic and unpredictable as well; in other words, asset prices follow random walk movements. According to the theoretical grounds, it is therefore impossible to earn higher profits based on only past (old) information, and whoever attempts to do so cannot surpass even a randomly selected portfolio of stocks of comparable risk (stated by Malkiel [32], cite taken from [41]).

Of course, many market inefficiencies have contradicted these theoretical concepts, such as speculative bubbles, irrational pricing, herd instinct, and market changes. A more comprehensive list can be found in [29]. Many empirical studies have shown that various trading strategies and ideas have indeed achieved excess profits and thus have intrinsically negated the theorem that asset prices strictly follow random walks and are unpredictable [31]. Fernández-Rodríguez et al. [17] reported the applicability of a simple artificial neural networks (ANN) trading rule to stock index trading, where the authors obtained higher profitability than a buy-and-hold (passive) strategy through periods of bear and stable markets and lower profitability for the bull market periods only. Chiang et al. [6] proposed an adaptive trading decision support system with a denoising function, which scored 41.86% of the adjusted return, compared to 12.98% for a buy-and-hold trading strategy. Chong and Ng [8] realized (based on 60-year data for London's FT30 index) that trading with the simple momentum strategies relative strength index (RSI) and moving average convergence divergence (MACD) generated higher returns than the buy-and-hold strategy in most cases. Similarly, Wong et al. [59] showed the benefits of MACD and RSI on the Singapore Stock Exchange. They claimed that by applying simple technical indicators, one can generate significantly positive returns, and interestingly, they also claimed that larger companies have specialized trading teams that apply technical analysis only. Teixeira and De Oliveira [52] proposed a combination of technical analysis with the nearest neighbor classification algorithm, exclusively based on past daily stock closing prices and volumes. The proposed method generated significantly higher profits on 12 out of 15 stocks than the buy-and-hold trading strategy. Ruta [40] showed many options for continuous 24/7 machine learning trading, and Creamer [10] proposed the machine learning algorithm Logitboost, which generated highly significant positive returns. The most recent research on the stock trading field encompasses deep reinforcement learning methods for stock trading [60] and many other machine learning, artificial intelligence and deep learning trading algorithms, which have been reviewed by *cbailes* on GitHub's library *awesome-deep-trading*.[1] Additionally, many relevant articles considering convolutional neural networks, long short-term memory, generative adversarial networks, etc., for trading, are accessible here. Furthermore, fuzzy inference systems may emerge in the near future [13, 33].

The purpose of this article is to implement an online (simultaneously updating) long short-term memory (LSTM) neural network investment trading strategy (ITS) for daily portfolio trading and test whether it is possible to earn higher-than-normal profits compared to benchmark trading strategies over a longer trading period. LSTM's innovative architecture should identify (extract) hidden patterns in financial data, if existing, and sufficiently well intervene in out-of-sample backtesting. A case study on the German stock market during the years 2010 and 2020 was implemented, where the German market was chosen as the example because it is one of Europe's leading, most open, and transparent stock markets. Financial data were fetched from a popular trading platform to construct a dataset and were later split into two parts: offline LSTM training was employed for the first part; while online LSTM training was employed for the second. The quality of the proposed idea was then evaluated.

In this study, we did not rely on either fundamental, sectoral or macroeconomic analyses but instead relied on the current market and stock (financial) quotations only, which were technically expressed for a given moment by movement, volatility, trend in stock prices, and suitable technical indicators. Using the different pattern extraction methods, we struggled to capture any speculative market

[1] https://github.com/cbailes/awesome-deep-trading by Craig Bailes

patterns from these movements and trends. We supposed that some of these patterns would affect multiple stocks in a similar (general) way synchronously and that others would only be relevant for specific stocks. Hence, we constructed the two following frameworks (Sirignano and Cont [46]):

1. A single, universal, general *common-model* that treats all the stocks in the same way, without differentiating between them. The number of LSTM models equaled 1, regardless of how many stocks were included in a portfolio (Filos [18] also called such decision systems 'universal trading agents').
2. A stock-specific *unique-model* that aimed to be used for trading with a single stock only. The number of LSTM models equaled the number of stocks held in a portfolio.

Model universality, as exploited in present paper, was extensively populated by Ruiz-Cruz in [38], who proposed a universal deterministic control law model that was able to simultaneously trade multiple stocks in real-time. Investment profiles were freely managed from conservative to aggressive, and the findings on 4 Mexican stocks showed that the proposed approach can be exploited for universal multiple stock trading. Wilson [58] stated that one chaotic model can be effectively exploited to trade hundreds of stocks over a long time period without changing any system inputs; in his case, he traded with 600 stocks over 10 years using a single model. Sezer and Ozbayoglu [42] employed a universal convolutional neural network (CNN) and performed trading on the 30 Dow stocks, where CNN backtesting gave promising results for the future. In contrast, Chihab et al. [7] explicitly stated that no universal model captured everything that might happen on stock markets due to many external (either objective or subjective) factors. According to Shiller [44], a sample of these includes psychological, sociological and anthropological factors, such as greed, emotional pain, speculation, overconfidence, underreaction, the perceived irrelevance of history, magical thinking, herd instinct and others. These factors may all influence different stocks similarly or completely disparately, similar to more objectively given factors, such as interest rates, GDP growth and other commonalities [1].

Our goal was to find an autonomous, superior and robust ITS for daily trading on a selected set of stocks for which classical fundamental analysis could not be applied due to the too low frequency of data (information). We expected that this study would provide a superior ITS and effectively disclose the benefits of both LSTM frameworks. The architecture of both LSTM neural networks remained equal in both frameworks. We assumed that regardless of whether the hidden (recurrent) patterns across many stocks were present through time, the common-model would output adequate performance. Nevertheless, we expected that the unique-model would adapt to the specifics of each stock and therefore outperform the common-model. This paper was written as an extension of Fister et al. [20] and Fister and Jagrič [19] and comprises the following main novelties:

- extend experiments from a single stock to complete portfolio,
- widen the online LSTM principle for automated daily stock trading,
- validate that LSTM ITS can outperform a passive buy-and-hold ITS on a diversified portfolio,
- implement two different LSTM ITS frameworks (ideas) and compare their robustness,
- focus on European (German) financial markets,
- compare and extract the benefits of LSTM ITSs compared to traditional ITSs.

The structure of the paper is as follows: Section 2 addresses the fundamentals of financial markets, particularly the German financial market, the dataset used, traditional investment trading strategies and the fundamentals of LSTM. Section 3 describes the two proposed robust online LSTM ITSs. Section 4 addresses the experiments and results, and Section 5 concludes the paper and outlines directions for the future.

2 German financial market, stock companies and investment trading strategies

This case study focused on one of the most important and prominent financial markets in Europe – the German financial market, which is located in Frankfurt (Germ. Börse Frankfurt). In Frankfurt, many various financial instruments can be traded, such as stocks, bonds, ETFs/ETPs, funds, and commodities. Every day, the Frankfurt stock exchange calculates the average performance of the stocks quoted there and publishes the so-called stock indices accordingly. The most popular stock index in Germany is the DAX30 (Germ. Deutsche Aktien Index 30, German stock index 30), which comprises 30 blue-chip (very prominent) companies and is calculated using a free-float (free-adjusted) capitalization method. Table 1 lists the DAX30[2] stock companies, their tickers and market capitalizations.[3]

2.1 Dataset

In the table of German blue-chip representatives, companies from textile, automotive, aviation, civil engineering, healthcare, insurance, banking, and other industry sectors can be found. The largest market capitalization on the given date was held by SAP SE company, which is a representative

[2] https://www.finanzen.net/index/dax/30-werte

[3] https://finance.yahoo.com/

Table 1 List of stocks obtained from Finanzen.net in May 2020. Additionally, minimum, maximum and mean close prices, the JB test of normality result and the ADF test of stationarity result are attached

Company	Ticker	Beta	MC	min	max	mean	JB	ADF
Adidas AG	ADS.DE	0.88	51.834	35.01	316.05	116.79	1015.43	0
Allianz SE	ALV.DE	1.10	80.483	57.47	232.00	138.11	413.81	0
BASF SE	BAS.DE	1.26	52.748	39.03	97.67	69.35	644.01	0
Bayer AG	BAYN.DE	1.29	64.987	36.23	143.88	81.38	424.32	0
Beiersdorf AG	BEI.DE	0.17	22.473	39.35	116.35	73.39	395.75	0
Bayerische Motoren Werke AG	BMW.DE	1.37	38.426	28.65	122.60	73.29	1142.10	0
Continental AG	CON.DE	1.66	19.858	32.13	251.30	135.18	286.37	0
Covestro AG*	1COV.DE	1.41	6.876	n/a	n/a	n/a	n/a	n/a
Daimler AG	DAI.DE	1.68	42.605	21.84	95.79	55.47	658.73	0
Deutsche Börse AG	DB1.DE	0.36	28.569	36.13	157.20	75.13	1039.76	0
Deutsche Bank AG	DBK.DE	1.63	17.981	4.87	46.90	21.59	418.91	0
Deutsche Post AG	DPW.DE	1.29	39.154	9.13	40.99	23.70	417.79	0
Deutsche Telekom AG	DTE.DE	0.59	72.140	7.71	18.05	12.69	259.05	0
E.ON SE	EOAN.DE	0.87	26.204	6.03	26.61	12.34	1794.42	1
Fresenius Medical Care AG & CO. KGaA	FME.DE	1.12	22.940	36.10	93	63.17	321.07	0
Fresenius SE & CO. KGaA	FRE.DE	0.93	29.821	13.93	79.65	44.31	310.49	0
HeidelbergCement AG	HEI.DE	1.39	10.357	24.57	95.50	60.22	456.54	0
Henkel AG & Co. KGaA	HEN3.DE	0.58	35.031	35.21	128.90	81.85	396.51	0
Infineon Technologies AG	IFX.DE	1.42	26.953	3.77	25.49	11.92	513.43	0
Deutsche Lufthansa AG	LHA.DE	1.22	4.498	7.18	31.12	15.19	2443.49	0
Linde plc	LIN.DE	0.73	116.213	75.96	208.60	96.35	3009.16	0
MERCK KGaA	MRK.DE	0.69	45.782	28.41	125.60	70.90	318.83	0
MTU AERO engines AG	MTX.F	n/a	9.292	35.25	286.40	100.51	2218.86	0
Münchener Rückversicherungs-Gesellschaft AG	MUV2.DE	0.76	33.946	79.55	282.60	158.76	966.62	0
RWE AG	RWE.DE	1.11	18.687	9.20	68.73	27.48	2667.97	1
SAP SE	SAP.DE	1.07	**144.599**	31.11	129.44	69.30	615.22	0
Siemens AG	SIE.DE	1.23	88.218	59.76	133.20	92.72	577.96	0
Volkswagen AG	VOW3.DE	**1.72**	77.899	55.50	255.20	145.20	1129.20	0
Vonovia SE*	VNA.DE	0.41	29.727	n/a	n/a	n/a	n/a	n/a
Wirecard AG*	WDI.DE	0.47	11.848	n/a	n/a	n/a	n/a	n/a

The examined data are from 1 Jan 2010 to 12 May 2020

"MC" stands for market capitalization and is given in billion euros. Market capitalization data and beta (5-year monthly) were taken on 6th June from Yahoo Finance. Three stock companies, designated by "*", e.g., 1COV.DE, VNA.DE and WDI.DE, were not suitable for this case study since no financial data were available for full coverage from 2010 on. Bold values denote the highest values. The "min", "max" and "mean" prices are stated in euros. High JB values show that none of the close prices were practically normally distributed. The ADF test of stationarity shows that only the close prices of two companies, e.g., EOAN.DE and RWE.DE, are time stationary. We concluded that the observed non-normality, non-stationarity and high level of multicollinearity of financial data make the classical regression approach infeasible

Source: Finanzen.net[2] and Yahoo Finance[3]

of a multinational enterprise focused on computer software applications. By contrast, the lowest market capitalization was held by Deutsche Lufthansa AG, i.e., the German national airline company. The presented "betas" express the empirically calculated risk of each company stock value relative to the overall DAX30 stock index. The riskiest among those on the list was the automotive Volkswagen AG company, and the least risky was the personal-care company Beiersdorf AG. Stock companies, declared with "*", e.g., 1COV.DE, VNA.DE, WDI.DE, were excluded from the case study due to missing (insufficient) data, while for the remaining companies, basic underlying financial data features could be fetched from the platform Yahoo Finance, consisting of open price, close price, adjusted close, highest daily price, lowest daily price and volume.

Open price was defined as the price of the security (in the morning) at the opening of the trading day. Similarly, the close price was defined as the current security price in the evening when the market closed. The adjusted close price was defined as the close price corrected for dividend payouts, stock splits, right offerings and others. A high price was derived as the highest price on the given day during the market operating hours, while a low price was derived as the lowest price. The volume determined the number of transactions on the given day as it was summed throughout the trading day. These 6 basic financial data features were adopted to derive several technical indicators and thus (1) expanded the dataset and (2) included additional information in the dataset. The minimum, maximum and mean values of each underlying company across the complete time period observed can be found in Table 1. Additionally, the results of the Jarque-Bera (JB) test of normality [26] and augmented Dickey-Fuller (ADF) test of stationarity [12] are attached to the table to validate that none of the close prices in the observed time period followed a normal distribution and that only two of the time series were practically stationary. Additionally, the fetched prices exhibited high levels of multicollinearity. All mentioned findings severely impair the assumptions of classical regression analysis and thus raise the question of whether the classical approach is feasible. Table 2 further summarizes the list of original market, stock and date data and the derived technical indicators. Such an 'expanded' dataset (hereinafter dataset) was generated for each stock and then used during the case study.

Table 2 List of explanatory variables

Explanatory variables	No. of feat.
1. Market (DAX30) data: open, close, high, low, adjusted close, volume	6
2. Stock data: open, close, high, low, adjusted close, volume	6
3. Date data: month, day, day of week, days to next trading day, days from previous trading day	5
4. Technical indicators:	
RET: $n = \{1, 2, 3, ..., 10\}$-day period	10
DIFF: $n = \{1, 2\}$-day period	2
RSI: 14-day period RSI, standardized	1
MACD: 12-day short, 26-day long and 9-day signal period	1
INCL: $n = \{5, 10, 15, 20\}$-day period	4
CHG_RET: $n = \{1, 2\}$-day period	2
DIFF_INCL	4
COEFs	2
Sum	43

Market and stock data each comprised 6-element financial data. Date data were composed of 5-element periodic data. Technical indicators consisted of n-day returns, n-day differences, RSI and MACD indicators, inclinations over $n = \{5, 10, 15, 20\}$-day periods, changes (differences) in returns for $n = \{1, 2\}$, differences in inclination values between (1) 5-day inclination to a 5-day inclination shifted for 5-days, (2) 5-day inclination to a 10-day inclination, (3) 5-day inclination to a 15-day inclination and (4) 5-day inclination to a 20-day inclination. Finally, two coefficients were calculated, e.g., the relative quotient between the close to open prices and the relative quotient between high and low prices, both decreased by 1

Source: Fister et al. [20].

2.2 Traditional investment trading strategies

An ITS is a decision algorithm that scrutinizes financial data and, according to the internal state, generates three fundamental trading signals, i.e., "Buy", "Hold" and "Sell". Fister et al. [20] distinguished among (1) traditional trading strategies, such as passive, RSI and MACD; (2) machine learning classifiers (MLC), such as decision trees, random forests, SVM, k-NN; and (3) ANNs, e.g., LSTM. Traditional trading strategies rely on rule-based guidelines only, which enable them to conduct instantaneous actions based on signals without any effort to build surrogate models. On the other hand, the surrogate model, which has been described by [34], based ITS performs in two subsequent stages: (1) training and (2) validation (prediction), usually by splitting the dataset into two samples, e.g., training (in-sample) and validation (out-of-sample). The in-sample is used for training only and the out-of-sample for unbiased prediction only. The quality of predictions is evaluated using a mechanical trading system (MTS), which is a tool that exploits the idea of ex post backtesting [21, 36]. Practically, this means that the MTS and ITS are both put into history, without knowing any "future" trends, movements or information. The MTS is given an initial amount of cash

that can be used to buy stocks. Then, simulation trading is performed day by day for each stock in the portfolio. The MTS reacts to the ITS trading signal accordingly: it buys, holds and sells stocks on demand. Backtesting is especially useful for testing ITSs since it incorporates a zero initial investment concept (trading is performed with virtual money only).

Traditional (classical) trading strategies are a kind of rule-based ITS [61], where instead of the trading signal being derived from some intelligent decision algorithm, the decision is made according to the internal state moving between two values (maximum and minimum). When the internal state hits one of the preset limits, the relevant trading signals are triggered; when the internal state holds within these two margins, it is treated as a hold signal.

Many of the traditional rule-based trading strategies exist. In this paper, two well-known technical indicators from Table 2, the RSI and MACD, were used (more technical indicators can be viewed in [37]). For the RSI indicator, which finds extremes between 0 and 100, this means that when (1) the internal state over the preferred time series, e.g., a stock's close price, hits the upper limit (in the professional trading community set at 70), the stock is considered to be overbought, and a sell signal is triggered immediately. Conversely, when the (2) internal state hits the lower limit (usually 30), this is treated as oversold, and a buy signal is triggered. A MACD indicator is slightly more sophisticated since it derives actual signals by subtracting two exponential moving averages (EMAs), e.g., between the 12-period EMA and 26-period EMA, where the period is typically given in days. The result of the subtraction is the so-called MACD line, which is next compared to the 9-period EMA signal line. The subtraction between the two ultimately provides the MACD histogram, which is used for trading directly. Both the RSI and MACD ITSs incorporate initial blackout periods due to their momentums.

By contrast, a passive trading strategy, i.e., buy at the first trading day and hold the capitalization until the last day, is also a typical representative of traditional ITSs. Due to its simplicity, it does not incorporate any blackout periods. The three traditional trading strategies mentioned, i.e., passive, RSI, and MACD, are referred to in this article as backtesting benchmarks [53].

2.3 LSTM investment trading strategy

In this article, MLC-ITSs are not discussed. Rather, the focus of this article is on the design of the ANN trading strategy or, more precisely, the use of LSTM, i.e., a kind of recurrent neural network (RNN). RNNs were proposed by Rumelhart et al. [39] specifically to cope with time series problems. Their benefit compared to a more common multilayer perceptron network lies in the integration of a feedback loop, which connects the previous output to the current input. While the RNNs in general mechanically connect the output from the previous step to the input in the current step directly, an innovative variant of RNNs called LSTM, proposed by Hochreiter and Schmidhuber [23], do this more delicately by selectively filtrating the unnecessary information from the previous step and selectively inputting new data in the current step.

LSTMs consist of three gates, forget, input, and output gates, and a central memory cell. In effect, they are much more complex than usual RNNs. The complexity of LSTMs allows them to prevent the occurrence of exploding and vanishing gradients and simultaneously enables them to capture longer-term dependencies in data without forgetting much. Figure 1 shows the basic outline of a single LSTM cell [3, 22, 24].

Let us designate the input that is about to be propagated through the LSTM neural network as $\mathbf{x}_t$, and the trainable LSTM weights, i.e., the input, recurrent and bias weights, as $\mathbf{W}$, $\mathbf{R}$ and $\mathbf{b}$, respectively. Whether the underlying weights comprise the input, forget, output and cell candidate components as follows:

$$\mathbf{W} = \begin{bmatrix} W_i \\ W_f \\ W_o \\ W_g \end{bmatrix}, \mathbf{R} = \begin{bmatrix} R_i \\ R_f \\ R_o \\ R_g \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_i \\ b_f \\ b_o \\ b_g \end{bmatrix}, \tag{1}$$

then, the propagation of information through the LSTM can be denoted by the following transition equations [24, 50]:

$$\begin{aligned} \mathbf{i}_t &= \sigma\left(W_i \cdot \mathbf{x}_t + R_i \cdot \mathbf{h}_{t-1} + b_i\right), \\ \mathbf{f}_t &= \sigma\left(W_f \cdot \mathbf{x}_t + R_f \cdot \mathbf{h}_{t-1} + b_f\right), \\ \mathbf{o}_t &= \sigma\left(W_o \cdot \mathbf{x}_t + R_o \cdot \mathbf{h}_{t-1} + b_o\right), \\ \tilde{\mathbf{c}}_t &= \tanh\left(W_g \cdot \mathbf{x}_t + R_g \cdot \mathbf{h}_{t-1} + b_g\right), \\ \mathbf{c}_t &= \mathbf{i}_t \odot \tilde{\mathbf{c}}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1}, \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right), \end{aligned} \tag{2}$$

where σ represents a sigmoid function, and $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\tilde{\mathbf{c}}_t$ represent input, forget, and output gate components and the cell candidate component, respectively. Vectors $\mathbf{c}_t$ and $\mathbf{h}_t$ represent memory cell and hidden states, respectively, and $\mathbf{c}_{t-1}$ represents the previous memory cell state from the constant error carousel. Operator $\odot$ stands for elementwise multiplication.

Similar to LSTM propagation stipulating the forward pass of the LSTM network, LSTM training stipulates the backward pass. Training the LSTM is a very complicated and computationally intensive task; therefore, we only mention it briefly. The LSTM training (in effect RNN training) is concluded using a special variant of backpropagation, i.e.,

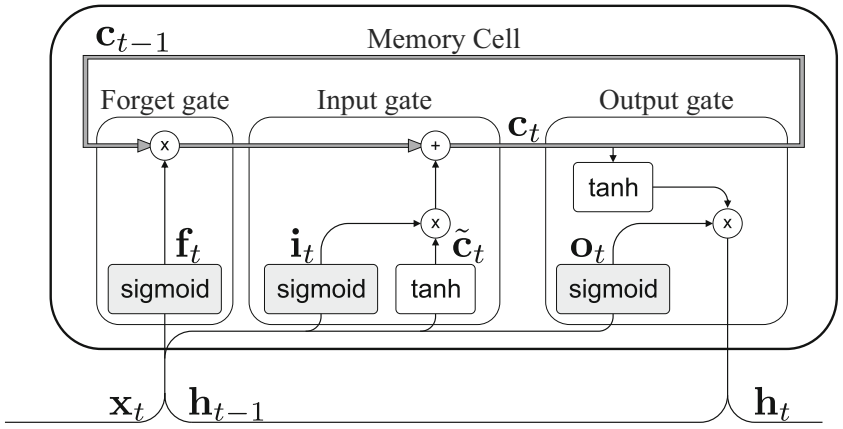

Fig. 1 Outline of the LSTM cell. LSTMs consist of forget, input and output gates and a central memory cell. The latter makes the LSTM a representative of recurrent neural networks. Source: Buduma and Locascio [5]

backpropagation through time (BPTT) [57]. Typical actions of BPTT encompass (1) presenting the input pattern to LSTM and forward-propagating it to obtain predicted output $\hat{y}$, (2) comparing the predicted output $\hat{y}$ to actual output y and calculating the classification error $e = \hat{y} - y$ (in this case $\hat{y}$ and y are classification labels), (3) calculating the LSTM network weight derivatives, and, finally (4) exploiting the calculated derivatives to modify (adjust) the network weights to minimize the classification error e [4]. The original mathematical outline for calculating derivatives was denoted in detail by Hochreiter and Schmidhuber [23], while Sutskever [49] presented novel state-of-the-art RNN training methods.

2.3.1 Online LSTM investment trading strategy

As already mentioned, when dealing with surrogate modeling, the dataset is split into training and testing samples (also called in-sample and out-of-sample). Usually, the model is trained on the basis of the in-sample, and the out-of-sample is used for evaluation of the trained model only. However, such strict splitting may be irrational when considering the real-world problem of portfolio trading. Here, new information (new instances of data) are disclosed every day, and these can be rationally exploited to update the model. Behavior on stock markets changes daily; therefore, it is essential for the user to keep the model continuously updated with the most recent information. Practically, this means that the learning process continues out-of-sample rather than mechanically stops at the end of the in-sample. Considering that only a single instance (one day) of progress is made in each step, this practically simulates instance-by-instance (day-by-day) trading. This concept only complies with a strict dataset split initially, i.e., the first split to design the initial network model, whereas later, the out-of-sample instances become available to retrain (update) the model one-by-one. According to Fister and Jagrič [19], who called such LSTM "online", the concept is entirely physically viable and substantially beneficial to use in practice.

Algorithm 1 An online LSTM training algorithm.

```
1:  procedure ONLINE LSTM TRAINING
2:      LOAD dataset X;
3:      SPLIT dataset into in-sample and out-of-sample instances;
4:      DEFINE learning rate α value and decay;
5:      for no. of offline epochs do ▷ number of offline epochs
6:          TRAIN model with in-sample; ▷ offline LSTM-training
7:      end for
8:      INITIALIZE empty array backtest_vector;
9:      DEFINE fixed learning rate α value;
10:     for t in out-of-sample instances do
11:         EXTRACT new x_t;
12:         ŷ_t = PREDICT (x_t); ▷ LSTM-simulation
13:         backtest_vector = APPEND(ŷ_t);
14:         CALCULATE actual y;
15:         for no. of online epochs do ▷ number of online epochs
16:             RETRAIN model with latest x_t, and actual y_t; ▷ online LSTM-training
17:         end for
18:     end for
19:     SIMULATE_MTS(backtest_vector); ▷ simulate actual real-time trading
20:     REPORT results;
21: end procedure
```

Algorithm 1 presents the details of the online LSTM procedure [19] as was used during the experiments. First, the complete dataset $\mathbf{X}$ was loaded into the programming

environment and split into two samples: in-sample and out-of-sample. The algorithm parameters, such as the learning rate α, decay and other mandatory parameters, were initialized. After initializing, the prospective LSTM model was trained on the in-sample. The purpose of in-sample training was to capture the general behavior of financial markets. In-sample represented offline training and was run for a number of offline epochs. Due to the mass of in-sample instances, it was a time-consuming process. After the offline training, the so-called *backtest_vector* and fixed learning rate α were initialized. The first was used to store the out-of-sample trading signals for later evaluation, while the second was used for online training. As the process turned into the online LSTM training part, the new out-of-sample instances $\mathbf{x}_t$ were extracted one at a time. This sequential day-by-day process followed real-world circumstances. Therefore, the model only progressed by a single out-of-sample instance at each time, first by making forthcoming trading signal predictions and appending the results into the *backtest_vector* and second by retraining the LSTM model with actual y. After the online training concluded, i.e., all out-of-sample instances went through, the MTS simulation (SIMULATE_MTS in Alg. 1) was performed using the generated array *backtest_vector*. Finally, the results for the completed out-of-sample period were visualized and reported to the user.

MTS served as a backtesting routine that, according to the ITS, bought, sold or held the stocks in the portfolio. It is based on real market and financial data that were downloaded from web portal. In general, the MTS can be used in two ways: sequentially (day by day) or once only (when the trading is fully completed). Due to its ease, we used the second approach in our case, but this introduced the need to implement the *backtest_vector*. The following assumptions were considered for the MTS: (1) the MTS does not deal with short selling, (2) the close price is obtained exactly a moment before the market closes so that we can execute orders, (3) the market is highly liquid so that the MTS can execute any orders requested, (4) the market is large enough that any buy or sell orders executed do not affect prices or expected market behavior, (5) the maximum number of stocks is always bought when the buy trading signal is triggered and all of the stocks are always sold when the sell trading signal is triggered – hence, some of the free cash assets usually remain in the portfolio after buying (only integer values of stocks can be traded), (6) once the instrument is bought or sold, no further transactions are allowed in the n-days time horizon.

2.4 Response variable

The training signal during the training stage was expressed as a percentage change in the close price in given time t compared to the previous close price $t - n$. Formally, the response variable equation was derived from [20] as follows:

$$\Delta_n = \frac{x_t^{(\text{close})}}{x_{t-n}^{(\text{close})}} - 1, \tag{3}$$

where $x_t^{(close)}$ was the close price in given period (day) t, and $x_{t-n}^{(close)}$ was the close price n-days ago. Here, n was called the time horizon and was defined prior to use; Fister et al. [20] varied n from 1 to 5. The Δ_n represented the percentage change in the close price for a given time horizon and naturally cannot be used for the classification problem. An additional transformation step was thus employed before actual use comparing percentage changes to a predefined variable $threshold$. Here, three scenarios were applied: (1) when the percentage change Δ_n was higher than the defined $threshold$, the training signal was a "Buy" signal; (2) when the percentage change Δ_n was lower than the defined (negative) $-threshold$, the training signal was a "Sell" signal; (3) if the percentage change Δ_n remained between the positive and negative thresholds ($\pm threshold$), the training signal was "Hold":

$$y_t^{(n)} \begin{cases} \text{Buy}; & \Delta_n > threshold \\ \text{Hold}; & -threshold < \Delta_n < threshold \\ \text{Sell}; & \Delta_n < -threshold \end{cases}. \tag{4}$$

The authors in [20] fixed the threshold at $threshold = 0.05$. In this article, we agreed that some further experimentation on this value may be useful/convenient, mainly due to the fact that each transaction incorporates trading or transaction costs/fees/provisions (more frequent transactions imply higher trading costs).

3 Proposed solution

As already stated in the introduction, the purpose of this paper was to implement two robust/superior LSTM neural network ITSs and check whether they can perform efficiently on the financial stock market. The two following frameworks for automated online LSTM ITS were presented in detail:

1. a *common-model* for all stocks held in the portfolio that traded according to the general (average) behavior of the portfolio,
2. a *unique-model* for each specific stock in the portfolio that traded according to the behavior of each stock uniquely (specifically).

Although the two mentioned frameworks were different, they shared a common ground of propagating sequence (financial) data. The propagation of sequence (daily) data

was performed as follows: first, the length of the time steps T_s was defined, and for a given day t, the previous T_s time steps were extracted from the dataset. These were next supplied into the LSTM neural network one-by-one. The time step $\mathbf{x}_t$ was propagated through the LSTM last. Whether the length of the time step T_s was sufficiently long, LSTM established stable internal memory cell ($\mathbf{c}_{t=T_s}$) and robust weight settings. This was important, since internal memory cell was next employed to output the predicted signals. These were next compared to the actual response variable in a given time t, e.g., y_t, to drive the LSTM training procedure. Finally, t became $t+1$ and the process was repeated. Each instance participated multiple times in LSTM propagation. Figure 2 graphically shows the propagation of time steps through the LSTM.

3.1 Common-model for all stocks

Common modeling was the first proposed framework for automated online LSTM ITS for portfolio trading. Here, a single LSTM model was used for training and predicting for all stocks. The common-model aimed to capture the general or average behavior of stocks held in a portfolio. The benefit of this idea lies in its decreased complexity, i.e., a single-model could be especially appropriate for (minimalistic) dedicated hardware over a wide range of stocks. The negative aspects of the single-model are (1) that a considerable set of underlying financial data must be available for a complete update and (2) two or more contradictory stock behaviors may cancel each other out and thus contribute little or nothing to the model. It is thus desired that the stocks in this model are similar to each other (homogeneous). The following paragraph describes the single-model working principle.

For the in-sample, every day, k instances or k training samples, where k equals the number of stocks held in a portfolio, were used $N_{offline}$ times for offline LSTM-training. The offline LSTM-training ran sequentially from the oldest data towards the newest data, always considering all the k instances of all stocks in a given time t. After the LSTM-training concluded for all k instances in a given time t, the t variable was incremented by 1 to obtain $t+1$, and the procedure was repeated. During the out-of-sample, for each of the stocks, the past sequence data of length T_s was supplied to the common-model. Next, the LSTM-simulation or prediction was run, which was in effect the trading signal prediction $\hat{y}$ for the next trading day (tomorrow). At the end of the (tomorrow's) trading day, the actual trading signal y was extracted, and the difference between the actual y and predicted $\hat{y}$ trading signals was supplied to the online LSTM-training procedure. Figure 3a shows the outline of a common-model.

3.2 Unique-model for each stock

A unique model for each stock was tailored to the separate specifics of each stock. Practically, this means that for each stock company, exactly one model was initialized, trained and simulated. Hence, the unique-model allowed diverse financial instruments, such as bonds, cryptocurrencies or commodities, to be held in portfolio simultaneously (although only stocks were considered). The benefit of the unique-model compared to the common-model lies in the much less time consuming updating and more tolerable execution pace for different financial instruments distributed over several computers. However, the main drawback of such design was a much higher complexity and thus the need for better hardware. Figure 3b shows the outline of the unique-model for each stock.

4 Experiments and results

To test the underlying hypothesis, whether it is possible to earn higher-than-normal profits using the online LSTM neural network, comprehensive experimental work was

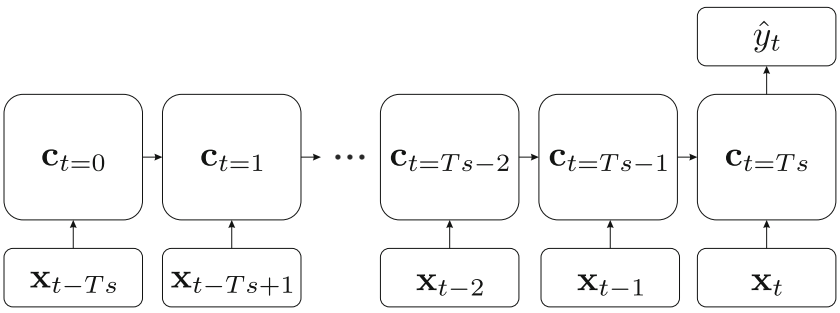

Fig. 2 The model shown is a kind of many-to-one sequence model. The past market and financial data are supplied and propagated via LSTM cell. After the predefined time steps T_s, the LSTM establishes stable memory cell state and can output the predicted trading signal $\hat{y}_t$, e.g., during the learning procedure after comparing to actual output y_t

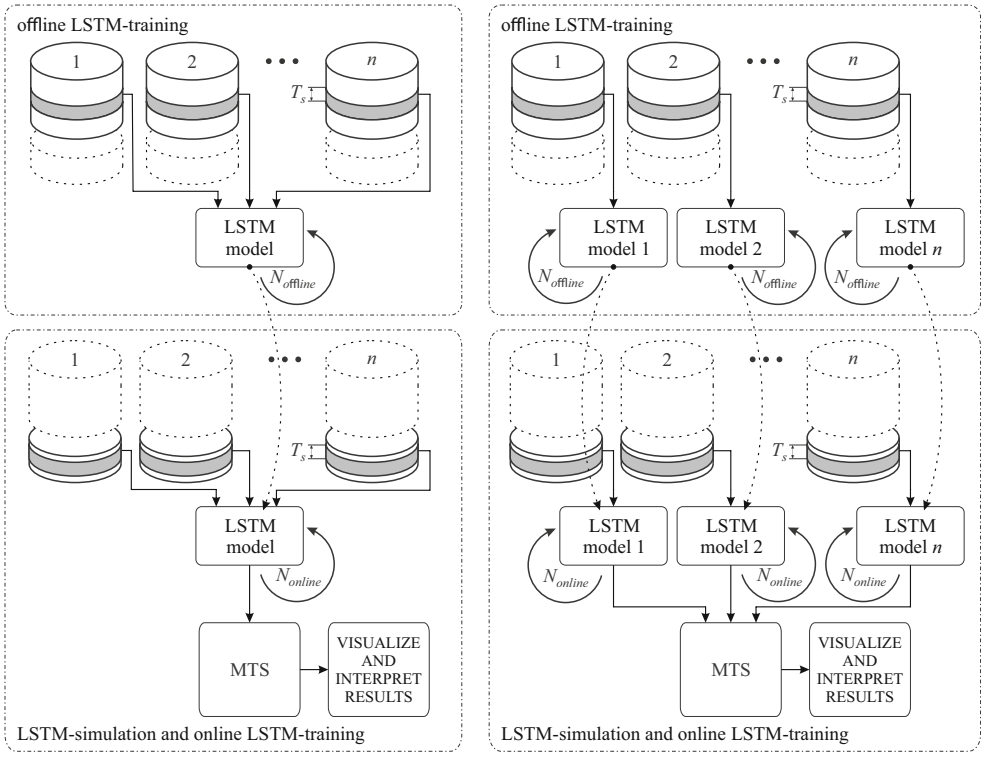

Fig. 3 A solid black line indicates the data-sample in use, i.e., in case of an offline LSTM-training, the in-sample (the upper) and in case of an online LSTM-training with LSTM-simulation, the out-of-sample (the lower). Unbiasedness was preserved entirely using this approach. $N_{offline}$ represents the number of offline epochs and N_{online} the number of online epochs **a** Common-model. **b** Unique-model.

performed. The experimental work was conducted in the following experimental stages:

- fetching financial data for the majority of German DAX30 blue-chip stocks and constructing a dataset,
- expanding and transforming the obtained dataset into a suitable form,
- defining the trading costs treatment,
- implementing and employing offline and online LSTM-training procedures,
- implementing and employing the LSTM-simulation procedure, and
- visualizing the results.

For the sake of experimental tests, 10 separate simulations of both the common- and unique-models were completed first. The use of 10 separate simulations was inspired by a common validation tool, the k-fold validation, where k is usually set to $k = 10$. We do not provide cross-validation in this paper but monitor the bias-variance error using a number of experiments. LSTMs perform differently with each run; thus, it is important to extract the best, worst and mean performances (performance is represented by the final portfolio value). Ten different models were built for the purpose of testing the common-model, and 270 models were built for the unique-model experiments, i.e., each of the 27 stocks in the portfolio had 10 different unique-models. Upon completing 10 simulations, the following experimental tests were conducted:

1. proof-of-concept of the common- and unique-models compared to traditional ITSs,
2. robustness tests between common- and unique-models, and
3. detailed evaluation tests of common- and unique-model performance compared to that of traditional ITSs.

Experiments were performed in the Python scripting language with the following libraries: financial data were fetched using Python's pandas-datareader[4] via the Yahoo Finance web portal; dataset transformation was performed using the pandas,[5] numpy[6] and talib[7] libraries. Here, the 6-element basic market and stock data (open, close, high, low, adjusted close prices and volume), were transformed into the dataset of 43-element dataset **X** (the list of dataset **X** variables shown in Table 2). The trading costs were implemented manually and were set at 1% according to suggestions of [45, 47, 55, 56]. The procedures for LSTM-training and LSTM-simulation were implemented using the

[4] https://pandas-datareader.readthedocs.io/en/latest/

[5] https://pandas.pydata.org/

[6] https://numpy.org/

[7] https://ta-lib.org/

Table 3 Algorithm setup

Variable	Variable name	Value
No. of LSTM units		100
Learning algorithm		Adam
Threshold	$threshold$	0.035
No. of epochs for offline training	$N_{offline}$	200
No. of epochs for online training	N_{online}	10
Offline initial (variable) learning rate	ϵ	0.002
Online (fixed) learning rate		0.001
Time horizon	n	1
Time steps	T_s	32
Dataset split		66.66%-33.33%
In-sample size		1717
Out-of-sample size		843
Transaction costs		1%
Initial cash per stock		10,000.00 euros
No. of stocks in portfolio	k	27
No. of independent runs	N	10

The decay in the offline (variable) learning rate was defined as $\epsilon \cdot 0.1 \cdot \exp(i/N_{offline})$, where i represented the current epoch. The equation guaranteed an exponentially decreasing learning rate. The initial offline learning rate was set as double the keras recommendation, while the online learning rate was held at the default. The time horizon was set at $n = 1$, which means that one-day returns were used to calculate the response variable. The Adam learning algorithm [30] was used as a robust training algorithm due to its efficacious characteristic of combining AdaGrad and RMSProp. The number of offline training sessions was determined experimentally. The number of offline epochs were set very low (compared to [19]) to ensure that only general relations between data were captured (to avoid overfitting problems, since the batch size was fixed at 1). The time steps $T = 32$ were held as a compromise between the actualization of the LSTM network on the one hand and the stability of the internal memory cell on the other. The initial cash per stock was set for a moderate investor (due to the relative evaluation of all ITSs, the initial cash is not among the most relevant setup variables)

keras deep learning API[8] and the results were visualized with the help of the matplotlib library.[9]

Experiments were conducted using the algorithm setup as designated in Table 3. Both common- and unique-models were implemented as a combination of the subsequent (1) LSTM layer and (2) dense layer, where the dense layer consisted of the 3 output neurons (buy, hold, sell) with a softmax activation function. The MTS was given initial cash (free assets) per stock. The offline LSTM-training incorporated the variable (decaying) learning rate ϵ and the online LSTM-training fixed learning rate. Although the dataset was fetched and generated from 1 June 2009 to 12 May 2020, only the data from 4 Jan 2010 to 12 May 2020 were used for simulations. The second half of 2009 was used to avoid a blackout period (missing data) due to momentum indicators. Setting the threshold was a purely experimental task for two reasons: (1) a high threshold guaranteed many hold instances and just a sample of buys and sells; a low threshold classified many instances into buy or sell groups but diminished the number of holds proportionally. The proportion of buys and sells turned out to be crucial, as these instances were directly presented to LSTM: when the frequency of buys and sells was too low, the LSTM turned nonreactive; when their frequency was too high, LSTM traded for nothing. (2) When setting the threshold, focus was also kept on the balance between the buys and sells, i.e., the greater the divergence between the two, the less symmetric the operation of LSTM (otherwise special remedies should be taken, such as [48]). It was experimentally determined, for the given sample of data, that $threshold = 0.035$ gives a reasonable compromise between the frequency and skewness of trading signals. The dataset was split 2/3 (67% was used for the training stage, and the remaining 33% was used for validation) as a compromise between the extended out-of-sample period presented in [20] and the usual data mining practice for 5- or 10-fold validation, e.g., 80% or 90%. Prior to conducting experiments, the variables were normalized between range 0-1 to make them suitable for the LSTM neural network. The next paragraph gives a short comment on financial movements during the observed trading period.

During the observed period, markets performed well. Due to very low interest rates, stock markets acted as substitutes for bank deposits and bonds and therefore attracted many new investors. The performance of markets was so positive in the last time period that market analysts reported potential markets may burst to correct for inflated prices. A large correction occurred in March 2020, when the Coronavirus crisis struck. Although it did not hurt financial institutions as badly as the last financial crisis, it caused significant losses and delays in real sectors. Although monetary policy drivers reacted rapidly in most countries, so that stock markets recovered quickly, many stock prices plummeted momentarily, and stock companies had to address serious financial problems. The period examined in this paper thus addresses both stock market occurrences, bullish and bearish trends, which makes it especially important for ITSs to identify trends and act accordingly.

4.1 Proof-of-concept

The proof-of-concept was conducted by (1) selecting the best performances with regard to final portfolio value (on the last trading day) among the 10 runs and (2) adding them together into a "best" portfolio. For the common-model, this means that the model with the highest final portfolio value

[8] https://keras.io/

[9] https://matplotlib.org/

among the 10 was chosen as the best. For the unique-model, the best portfolio performance was extracted for each stock independently among the 10 runs, and the 27 individual best performances were then summed into a single portfolio. Further steps in the proof-of-concept incorporated providing visuals of both performances on the chart and comparing these to traditional ITSs. Figure 4 shows the obtained results.

In the figure, "LSTM1" represented the best unique-model performance and "LSTM2" represented the best common-model LSTM ITS performance. According to the final portfolio value, the former was the most profitable among all the ITSs. The common-model was not as prominent as its predecessor but still scored higher profit compared to the traditional ITS. Although the passive ITS was the least sophisticated among all, it performed the best during the bullish trend, but on the other hand, it lost much of its potential profit during the Coronavirus crisis. Even if the portfolio value of the passive ITS recovered quickly, at the end of the trading period, it suffered an overall loss. The RSI ITS delivered solid performance for the majority of examined period, except for the last 150 trading days, when it missed the potential profit just before the last bullish trend. MACD ITS performed worst of all ITSs on a given set of stocks and did not give any indication of profitability.

The experiments were run on the HP ProDesk 400 G4 MT Business PC using an Intel Core i5-7500 CPU and 8 GB RAM. On average, it took 23 hours 3 min and 40 sec to complete common-model LSTM-training and 29 min and 2 sec to complete online LSTM-training (for the complete portfolio). The offline unique-model LSTM-training lasted an average of 50 min 35 sec per stock, while the online LSTM-training procedure lasted an average of 1 min and 5 sec per stock.

4.2 Robustness test between the common- and unique-models

The robustness test was conducted by analyzing all 10 independent runs that were initially carried out. For each of the stocks separately, the final (overall) portfolio value was derived, and the basic statistical momentums, such as minimum, maximum and mean values, were calculated. The purpose of the robustness test was to determine and compare the mean values and min-max spreads between the two proposed LSTM ITSs. Namely, the larger the min-max spreads were, the worse the ITS robustness.

Table 4 shows the results of the robustness tests, where maximum (denoted as "max"), minimum ("min") and mean ("mean") values were extracted for each of the stocks in the portfolio, for each of the LSTM ITSs and the differences Δ between mean values of both ITSs were calculated. The results show that some greater disparities in a few stocks arose between the common- and unique-models. For example, the unique-model scored a much higher performance difference, i.e., more than 2,000.00 euros of difference between the common- and unique-models for four stocks: BMW.DE, CON.DE, FRE.DE and HEN3.DE. While the first two belonged to a group of riskier assets, the former two were considered less risky. On the other hand, the common-model scored a much higher mean final portfolio value twice for ADS.DE and DB1.DE assets. We speculated that the large difference between the common- and unique-models was due to the riskiness of the assets, but this was not the case. As a matter of fact, we realized that (1) the variance (and thus the standard deviations) of the common-model was significantly lower than that for the unique-model; (2) the common-model in effect scored a higher mean value 16 out of 27 times, and we concluded that the generalization abilities of common-model were shown

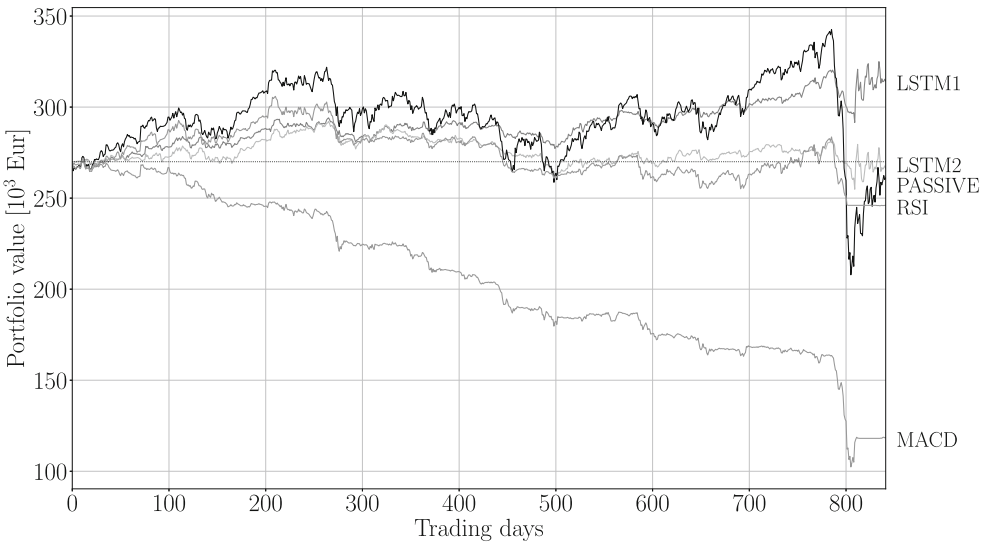

Fig. 4 "LSTM1" represents the unique-model, and "LSTM2" represents the common-model. The former was most profitable among all ITSs. The passive ITS was the most prominent during the bullish trend, while RSI missed the potential profit in the last stage of trading. MACD did not show any indication of profitability. The dotted line shows the initial free cash assets

Table 4 Robustness test results, where the common- and unique-model columns outline the minimum, maximum and average portfolio values for each of the stocks

	Common-model				Unique-model				Δ
	max	min	mean	max. profit	max	min	mean	max. profit	
ADS.DE	15,845.5	8,372.44	12,450.76	5,845.5	11,385.29	8,135.82	9,326.09	1,385.29	-3,124.67
ALV.DE	12,271.80	9,903.88	11,282.14	2,271.80	11,765.34	10,087.04	10,681.49	1,765.34	-600.65
BAS.DE	7,104.99	3,053.42	5,081.94	-2,895.01	6,963.76	4,202.84	5,075.23	-3,036.24	-6.71
BAYN.DE	7,808.00	4,850.7	5,921.02	-2,192.00	8,962.23	5,479.92	7,122.02	-1,037.77	**1,201.00**
BEI.DE	10,821.01	7,824.75	9,482.37	821.01	9,516.12	8,167.38	8,969.41	-483.88	-512.96
BMW.DE	5,826.42	4,438.61	5,363.15	-4,173.58	11,862.33	5,112.01	8,482.25	1,862.33	**3,119.10**
CON.DE	6,785.01	3,981.95	4,826.42	3,214.99	9,471.72	4,586.51	7,170.49	-528.28	**2,344.07**
DAI.DE	8,932.15	4,467.33	6,216.41	-1,067.850	9,106.97	4,160.38	6,697.23	-893.03	**480.82**
DB1.DE	20,847.38	14,574.18	16703.23	10,847.38	16,962.52	10,174.6	13,591.40	6,962.52	-3.111,83
DBK.DE	9,542.53	3,096.58	4,663.74	-457.47	5,622.59	2,645.7	3,892.91	-4,377.41	-770.83
DPW.DE	10,595.00	7,781.89	8,607.79	595.00	11,284.49	5,145.3	8,574.62	1,284.49	-33.17
DTE.DE	11,694.17	5,796.68	9,093.95	1,694.17	11,026.45	7,844.71	9,298.95	1,026.45	**205.00**
EOAN.DE	14,490.76	9,733.88	12,650.07	4,490.76	12,470.04	9,910.51	11,143.65	2,470.04	-1,506.42
FME.DE	11,869.51	6,910.99	9,079.01	1,869.51	14,035.77	8,654.79	10,986.33	4,035.77	**1,907.32**
FRE.DE	10,330.59	5,929.14	7,364.41	330.59	12,579.45	8,647.24	11,322.85	2,579.45	**3,958.44**
HEI.DE	10,672.38	4,748.70	6,403.63	672.38	9,446.03	4,542.3	6,193.55	-553.97	-210.08
HEN3.DE	8,899.58	6,360.35	7,018.19	-1,100.42	10,166.29	6,398.33	9,023.49	166.29	**2,005.30**
IFX.DE	10,263.59	7,278.52	9,089.54	263.59	11,884.12	6,693.77	8,969.47	1,884.12	-120.07
LHA.DE	14,300.22	6,242.53	8,003.62	4,300.22	12,858.99	5,223.11	7,807.86	2,858.99	-195.76
LIN.DE	11,794.11	7,444.30	10,614.41	1,794.11	11,539.76	8,537.72	9,900.82	1,539.76	-713.59
MRK.DE	13,024.75	7,009.77	10,173.94	3,024.75	10,859.94	8,409.54	9,761.69	859.94	-412.25
MTX.F	13,254.14	8,330.39	10,659.51	3,254.14	13,753.86	8,422.72	11,191.40	3,753.86	**1,012.71**
MUV2.DE	10,743.51	9,835.41	10,314.48	743.51	10,560.48	8,508.62	9,623.23	560.48	-691.25
RWE.DE	21,311.35	13,306.65	17,427.40	11,311.35	26,878.27	10,000.0	17,151.77	16,878.27	-275.63
SAP.DE	13,041.15	9,215.15	10,341.75	3,041.15	11,533.6	8,946.38	10,366.35	1,533.6	**24.60**
SIE.DE	9,112.07	5,838.69	7,343.57	-887.93	10,244.38	4,954.37	7,610.84	244.38	**267.27**
VOW3.DE	12,338.66	6,636.57	9,124.14	2,338.66	11,540.49	4,160.31	7,712.78	1,540.49	-1,411.36

The number of samples amounts to $N = 10$

Differences Δ were calculated between the mean values of the unique-model compared to common-model, i.e., $\Delta = mean(\text{unique-model}) - mean(\text{common-model})$. Bold values denote positive differences (differences where the unique-models scored higher mean values than the common-models). Maximal profit shows the amount of profit per stock, e.g., if initial free cash assets amounted to 10,000.00 euros per stock, and the maximal final value of the individual stock amounted to 15,845.5 euros (for ADS.DE), then the maximal profit was 5,845.5 euros. For stocks that were traded with a negative maximal profit for both proposed LSTM frameworks, we can assume that they suffered a downtrend during the out-of-sample

to be positive; (3) the min-max spread of 16 out of 27 stocks varied minimally in terms of mean values, i.e., less than 1,000.00 euros between the common- and unique-models; (4) the common-model proved to be more robust than the unique-model, but due to its higher generalization capabilities, it was also more rigid (lower profitability on certain stocks may be implied); (5) the difference between the sum of mean values for the common- (245,300.59 euros) and unique-models (247,648.17 euros) was also minimal and below that of the passive ITS. This demonstrated that both ITSs traded similarly to each other and that both LSTM ITSs were superior to the passive ITS in the best cases but less superior in the average cases. We conclude that a robust and solid LSTM portfolio performance occasionally beats the market and that extra experiments with adjusting control parameters, architectures and time periods would further increase performance.

4.3 Detailed evaluation tests

The purpose of this test was to objectively evaluate the quality of each ITS. The test was carried out by deriving

three groups of well-known financial indicators: (1) final profit/loss and beta riskiness, (2) volume (number) of ITS transactions, and (3) the Sharpe ratio, Jensen's alpha and Treynor ratio. The first two groups were extracted directly from the trading results, while the third group employed an additional manipulation step.

The financial beta (β) [14] acted as a standardized financial indicator that showed the ratio of covariance of asset (portfolio) and index prices relative to the variance in index prices. It represents the movement of asset prices relative to the index and thus acts as a substitute for the risk of an asset. Beta was calculated as follows:

$$\beta = \frac{\mathrm{Cov}(r_i, r_m)}{\mathrm{Cov}(r_m)} \tag{5}$$

where r_i and r_m represent the investment (portfolio) and market (benchmark) daily returns, respectively. Here, the asset returns r_i were substituted by an ITS portfolio value, and benchmark returns r_m were substituted by "DAX30" returns during the observed period. Next, the Sharpe ratio [43], or the return-to-risk financial indicator, was introduced. It measured the proportion of expected return per unit of risk. It was calculated as follows:

$$S_a = \sqrt{N} \cdot \frac{\mathbb{E}(r_i - r_m)}{\sqrt{\mathrm{Var}(r_i - r_m)}}, \tag{6}$$

where S_a represents the annualized asset Sharpe ratio, and N represents the number of days within a year, e.g., set at $N = 252$ in our case, since one year consists of approximately 252 trading days. The Sharpe ratio was calculated on the basis of raw daily data, and negative values indicate that the expected profit of the underlying investment was below the market return and positive values indicate that the expected profit was higher than the market return.

The following indicators measured value on the basis of final values rather than on daily data. Jensen's alpha α_J [27] was provided to measure the excess return over the theoretical expected return. It considered the relative risk of underlying assets (portfolio) compared to the market $beta_{iM}$ and the capital asset pricing model as follows:

$$\alpha_J = R_i - \left(R_f + \beta_{iM} \cdot \left(R_M - R_f\right)\right) \tag{7}$$

where α_J represents Jensen's alpha, and R_i stands for realized returns on assets (portfolio). R_f and R_M represent the risk-free rate and overall market return, respectively. A positive α_J means that the investment is performing better than expected; while a negative α_J means that it is performing worse. Finally, the Treynor ratio T [54] measures the amount for the excess reward of underlying investment compared to the risk-free rate per given unit of volatility. It was calculated as the difference between realized return and the risk-free rate, compared to the asset (portfolio) risk as follows:

$$T = \frac{R_i - R_f}{\beta_{iM}}. \tag{8}$$

Here, the greater the difference is between the realized return and the risk-free rate, the higher the Treynor ratio, or the greater the beta, the lower the Treynor ratio. A positive T determines that the investment is more suitable than the risk-free investment, while a negative T determines that the investment is less suitable. The obtained results are shown tabularly in Table 5 on the basis of best portfolios only (not on all 10 independent runs).

The first group of extracted results considered final portfolio value, overall profit/loss and beta riskiness. Given an amount of cash at the beginning of a trading period (10,000.00 euros per stock) meaned that 270,000.00 euros were initially given to complete portfolio. By trading stocks, this amount varied according to the assets' close prices. Table 5 shows that only a single ITS - unique-model LSTM ITS exceeded the initial amount of cash at the end of the trading period and thus consolidated an overall profit (16.40%). The common-model LSTM ITS followed the unique-model, with a -1.38% of overall loss, but surpassed the passive ITS, which also consolidated very little loss (-3.29%). The RSI ITS accounted for -9.27% of the overall loss, while MACD ITS accounted for an extreme loss of -56.12%. According to the beta indicator, the passive ITS was the riskiest ITS among all, although this riskiness was still lower than the riskiness of the benchmark (complete DAX30 portfolio) due to the three companies not included in the analysis. On the other hand, the common-model and, especially, the unique-model showed very little riskiness compared to the benchmark, and they even scored negative betas. This intrinsically implied that whatever movement happened with the index prices, it did not relate to the LSTM ITSs. We agreed that such implication came as a consequence of prudent, vigilant and reasonable day-to-day trading.

Next, the number of transactions for each ITS is shown, which was derived as follows: for each of the LSTM ITSs, the best portfolio was selected (traditional ITSs were run once only). Then, the number of transactions (no. of times the MTS bought or sold stocks) during the whole period was summed. The maximum, minimum, mean values and standard deviations were calculated from the portfolios, e.g., the unique-model maximally performed 36 trades for one of the stocks in the portfolio, and minimally performed only 1 trade for a stock in the portfolio. Across the complete portfolio, the unique-model performed 9.15 trades per stock on average, with a standard deviation of 7.96. The common-model, on the other hand, traded slightly more frequently, with a mean value of 12.85 per complete

Table 5 The results in this table were obtained by analyzing the best results (among the 10 model runs)

		passive	RSI	MACD	common-	unique-
Value	euros	261,136.06	244,964.33	118,463.91	266,268.29	**314,281.28**
Profit	R_i (%)	-3.29	-9.27	-56.12	-1.38	**16.40**
Beta	β	**0.7746**	0.1812	0.5153	-0.0290	-0.0091
No. of transactions	max	1	11	**88**	30	36
	min	1	2	**34**	2	4
	mean	1	7.37	**68.04**	12.85	9.15
	stdev	0	1.95	**9.88**	6.57	7.96
Sharpe ratio	S_a	0.0765	-0.1336	-1.7529	0.0032	**0.2384**
Jensen's alpha	α_J(%)	1.38	-9.30	-53.69	-3.63	**14.32**
Treynor ratio	T	-0.07	-0.60	-1.13	**1.16**	-15.77

For the calculation of Jensen's α_J and Treynor's T, we assumed a risk-free rate $R_f = 2\%$

The highest values are in bold, "common-" stands for the common-model and "unique-" stands for unique-model. The most profitable ITS was the unique-model LSTM. The riskiest among the ITSs was the passive ITS. The latter also demanded the highest expected return. Beta (β) was calculated compared to the benchmark "DAX30"

portfolio. The passive ITS executed only a single buy order per stock and thus scored equal maximum, minimum and mean (average) numbers of transactions. The RSI ITS scored 7.37 transactions per stock on average, and the MACD ITS executed 68.04 transactions on average. Very frequent trading was one of the reasons for poor MACD ITS performance – each transaction incorporated trading costs, and the more trades made, the higher the transaction costs. These costs substantially lowered the portfolio value. Additionally, note that transaction costs imposed significant volatility as well. For instance, if transaction costs were set at 1%, a drastic 1% loss was always imposed when buying or selling stocks instantly; among others, this negatively affected the calculation of beta and the rest of the indicators.

Given the investment performances, RSI and MACD ITS scored negative Sharpe ratios, while the passive and robust LSTM ITSs scored positive ones. The highest Sharpe ratio among the last three was scored by the unique-model LSTM ITS and was thus denoted as bold in the table. According to Jensen's alpha α_J, the passive and unique-model ITSs performed better than (theoretically) expected. Due to two of the negative betas, we could not perform objective and unbiased Treynor ratio analysis, as the results would not be meaningful.

5 Concluding remarks

We have implemented two promising variants of online (constantly-updating) LSTM neural networks for the purpose of stock trading and compared their trading performance to that of traditional ITSs. We have extended the experiments from a single asset to a complete portfolio. The German financial market represented by DAX30 blue-chip stocks was exploited for this study. We have experimentally shown that LSTM neural networks can be exploited as robust ITSs that score solid portfolio performance in average cases and beat the market occasionally in the best cases. Both proposed LSTM ITSs performed outstandingly well in the bearish market when selling assets, and limiting losses came into play. We determined that additional steps towards improving the efficiency of LSTM ITS during bullish markets should be taken to increase the mean final overall profit of multiple independent runs. The robustness tests that were carried out in the paper showed that LSTM can indeed be a very reliable and robust tool. Therefore, additional experiments with control parameters and architecture optimization are necessary to guarantee that the mean overall portfolio value outperforms that of passive ITS. Considering the given outlines, we rejected any doubts of the underlying hypothesis and thus concluded that it was indeed possible to earn higher-than-normal profits on an open and transparent stock market during both bullish and bearish market conditions using innovative LSTM neural networks. Further, we attached important conclusions of the paper point-wise:

- The limiting loss capability importantly contributed to the negativity of the calculated beta, which is in accordance with Kay [28].
- Universal models are to be stipulated when the complexity of the trading system needs to be low. For well-differentiated financial instruments, some form of preprocessing, such as clustering [51], might be beneficial. Nevertheless, a crucial step for combining clustering with later building a trading system is the selection

of features (variables). One must remember that all fundamental, sectoral or macroeconomic information was discarded in this paper, as all information originated from market quotations only.
– Common- and unique-models incorporated entirely different approaches to modeling, but both showed exceptionally similar performance on average. On the basis of 10 independent runs, the mean final portfolio values between them differed by less than 1%.
– The single common-model for intervening on stock market was exploited to verify the existence of subjective factors mentioned earlier that drive asset prices. As the common-model performed well on a broad set of stocks, we concluded that similar factors existed on market that affected the asset prices of many stocks in the same (similar) way. Implicitly, this meant that investor (psychological) behavior was similar across many stocks and dependent on fundamental factors to some degree. Of course, we could not capture investor behavior directly, but we exploited the common-model as a proxy.

Kay defended the concept that a negative beta came as a consequence of a decreasing algorithmic performance relative to the general index, and vice versa, which in effect caused the trading algorithm to underperform in most (bullish) situations and outperform the general index during the index's negative (bearish) returns. This is exactly what we observed in the results. Chourmouziadis and Chatzoglou [9] performed a study very similar to ours; by emphasizing the effect of economic, political and psychological factors on financial markets, they exploited a fuzzy stock trading system and realized that their proposed ITS produced lower returns than the passive ITS on the bull market but effectively avoided large losses on the bear market. Berutich et al. [2] showed that a genetic programming ITSs handled avoiding large losses that were caused in the 07-09 subprime crisis well. Using a portfolio of 21 Spanish stocks, they managed to earn minimal positive returns during the market collapse. These two findings were in accordance with our results. Hu et al. [25] have in literature survey explicitly stated that many algorithmic trading systems commonly suffered from poor trading performance during uptrends but adequate limiting losses in downtrends. The conclusion on similarities of common- and unique-models (mean portfolio values differing for less than 1%) contributed to the field of automatic trading as follows: strong psychological (deterministic) factors exist on financial markets that similarly affect numerous financial instruments. These factors could be divided into two subgroups: (1) universal factors and (2) financial instrument-specific factors (also called universal price formation mechanisms [46] or stylized facts [15]). On average, models exploiting any of these provide similar portfolio returns.

For future work, external data information that are consequence of rapid information technology outbreak should be included. An excellent example of such data seem to be support and confidence models that daily search the internet for comments and news regarding the stocks' or other financial instruments' portfolio and subjectively asses these information as positive or negative. External information at high frequency would drastically increase the resilience and reliability of current trading models (comprehensive examples of analyzing sentiment from information sources, such as blogs and social media, are shown in [35, 63]). Finally, performed case study could be extended by increasing the number of unique- and common-model runs from $N = 10$ to $N = 100$ or 1000. We expected that the trading performance of the unique- and common-models would become even better through this extension. Further, experiments could be extended by incorporating other foreign stock markets, such as the US and Asian and incorporating other financial instruments, such as bonds, cryptocurrencies, commodities, forex, etc. Finally, the LSTM trading strategy could be exploited to actually intervene in stock markets, either as a recommendation system or combined with a trading bot and dedicated hardware, as a real-time web trading system. Here, the trading period could be lowered significantly to enable intraday buying and selling orders. Finally, the effect of poor performance during uptrends should be carefully studied, and appropriate remediation should be implemented.

Acknowledgements Dušan Fister and Timotej Jagrič acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P5-0027). Matjaž Perc was supported by the Slovenian Research Agency (Grant Nos. P1-0403, J1-2457, J4-9302, and J1-9112).

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Barrot LD, Servvn L (2018) Gross capital flows, common factors, and the global financial cycle the world bank. https://doi.org/10.2139/ssrn.3116778
2. Berutich JM, López F, Luna F, Quintana D (2016) Robust technical trading strategies using GP for algorithmic portfolio selection. Expert Systems with Applications 46:307–315. https://doi.org/10.1016/j.eswa.2015.10.040, https://www.sciencedirect.com/science/article/pii/S0957417415007447
3. Brownlee J (2016) Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras - Machine Learning Mastery. Machine Learning Mastery pp. 1–135. http://machinel

earningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/
4. Brownlee J (2017) Long Short-Term memory networks with python develop sequence prediction models with deep learning. Machine Learning Mastery 46(313):192–202. https://doi.org/10.19083/ridu.11.498
5. Buduma N, Locascio N (2017) Fundamentals of deep learning: designing Next-Generation machine intelligence algorithms. O'Reilly Media Inc, Newton
6. Chiang WC, Enke D, Wu T, Wang R (2016) An adaptive stock index trading decision support system. Expert Syst Appl 59:195–207. https://doi.org/10.1016/j.eswa.2016.04.025
7. Chihab Y, Bousbaa Z, Chihab M, Bencharef O, Ziti S (2019) Algo-Trading Strategy for intraweek foreign exchange speculation based on random forest and probit regression. Applied Computational Intelligence and Soft Computing, p 2019. https://doi.org/10.1155/2019/8342461
8. Chong TTL, Ng WK (2008) Technical analysis and the London stock exchange: Testing the MACD and RSI rules using the FT30. Applied Economics Letters 15(14):1111–1114. https://doi.org/10.1080/13504850600993598. http://www.tandfonline.com/doi/abs/10.1080/13504850600993598
9. Chourmouziadis K, Chatzoglou PD (2016) An intelligent short term stock trading fuzzy system for assisting investors in portfolio management. Expert Syst Appl 43:298–311. https://doi.org/10.1016/j.eswa.2015.07.063
10. Creamer G (2012) Model calibration and automated trading agent for Euro futures. Quantitative Finance 12(4):531–545. https://doi.org/10.1080/14697688.2012.664921
11. Damodaran A (2016) Damodaran on valuation: security analysis for investment and corporate finance, vol 324. John Wiley & Sons, Hoboken
12. Dickey DA, Fuller WA (1979) Distribution of the estimators for autoregressive time series with a unit root. J Am Stat Assoc 74(366):427. https://doi.org/10.2307/2286348
13. Egrioglu E, Yolcu U, Bas E (2019) Intuitionistic high-order fuzzy time series forecasting method based on pi-sigma artificial neural networks trained by artificial bee colony. Granular Computing 4(4):639–654
14. Elton EJ, Gruber MJ (1997) Modern portfolio theory, 1950 to date. Journal of Banking and Finance 21(11-12):1743–1759. https://doi.org/10.1016/S0378-4266(97)00048-4
15. Engle RF, Patton AJ (2007) What good is a volatility model? In: Forecasting volatility in the financial markets, pp. 47–63. Elsevier
16. Fama EF (1970) Efficient Capital Markets: A Review of Theory and Empirical Work: Discussion. The Journal of Finance 25(2):421. https://doi.org/10.2307/2325488. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1970.tb00518.x
17. Fernández-Rodríguez F, González-Martel C, Sosvilla-rivero S (2000) On the profitability of technical trading rules based on artificial neural networks:: Evidence from the Madrid stock market. Economics Letters 69(1):89–94. https://doi.org/10.1016/s0165-1765(00)00270-6
18. Filos A (2019) Reinforcement learning for portfolio management. arXiv:1909.09571
19. Fister D, Jagrič T (2019) Online long short-term memory network for stock trading. StucoSRec Proceedings of the 2019 6th Student Computer Science Research Conference, pp 5–8. https://doi.org/10.26493/978-961-7055-82-5.5-8
20. Fister D, Mun JC, Jagrič V, Jagrič T (2019) Deep learning for stock market trading: a superior trading strategy? Neural Network World 29(3):151–171. https://doi.org/10.14311/NNW.2019.29.011
21. Harvey CR, Liu Y (2015) Backtesting. The Journal of Portfolio Management 42(1):13–28
22. Hochreiter S, Bengio Y, Frasconi P, Schmidhuber J (2001) Gradient flow in recurrent nets: the difficulty of learning long-term dependencies
23. Hochreiter S, Schmidhuber J (1997) Long Short-Term Memory. Neural Computation 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735. http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735
24. Hochreiter S, Schmidhuber J (1997) LSTM Can solve hard long time lag problems. In: Advances in neural information processing systems, pp. 473–479
25. Hu Y, Liu K, Zhang X, Su L, Ngai EW, Liu M (2015) Application of evolutionary computation for rule discovery in stock algorithmic trading: a literature review. Applied Soft Computing Journal 36:534–551. https://doi.org/10.1016/j.asoc.2015.07.008
26. Jarque CM, Bera AK (1980) Efficient tests for normality, homoscedasticity and serial independence of regression residuals. Economics Letters 6(3):255–259. https://doi.org/10.1016/0165-1765(80)90024-5. https://www.sciencedirect.com/science/article/pii/0165176580900245
27. Jensen MC (1968) The performance of mutual funds in the period 1945-1964. The Journal of Finance 23(2):389. https://doi.org/10.2307/2325404
28. Kay J (2019) Sunshine on a Cloudy Day: Evidence in Support of a Moving Sunshine on a Cloudy Day: Evidence in Support of a Moving Average Strategy Across Down Markets Using ETFS Average Strategy Across Down Markets Using ETFS. https://digitalcommons.library.umaine.edu/honors
29. Kindleberger CP, Aliber RZ (2011) Manias, panics and crashes: a history of financial crises Palgrave Macmillan
30. Kingma DP, Ba J (2014) Adam:, A method for stochastic optimization. arXiv:1412.6980
31. Lo AW, MacKinlay AC (2014) Stock market prices do not follow random walks: evidence from a simple specification test. A Non-Random Walk Down Wall Street 1(1):17–46. https://doi.org/10.1515/9781400829095.17
32. Malkiel BG (2003) The efficient market hypothesis and its critics. J Econ Perspect 17(1):59–82. https://doi.org/10.1257/089533003321164958
33. Melin P, Sánchez D. (2019) Optimization of type-1, interval type-2 and general type-2 fuzzy inference systems using a hierarchical genetic algorithm for modular granular neural networks. Granular Computing 4(2):211–236
34. Meyer CF (1972) Surrogate modeling. Water Resour Res 8(1):212–216. https://doi.org/10.1029/WR008i001p00212
35. Nguyen TH, Shirai K, Velcin J (2015) Sentiment analysis on social media for stock movement prediction. Expert Syst Appl 42(24):9603–9611
36. Ni J, Zhang C (2005) An Efficient Implementation of the Backtesting of Trading Strategies. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol. 3758 LNCS, pp. 126–131. https://doi.org/10.1007/11576235_17
37. Pojarliev M (2005) Performance of currency trading strategies in developed and emerging markets: some striking differences. Fin Mkts Portfolio Mgmt 19(3):297–311. https://doi.org/10.1007/s11408-005-4692-2
38. Ruiz-Cruz R (2018) Portfolio modeling for an algorithmic trading based on control theory. IFAC-PapersOnLine 51(13):390–395. https://doi.org/10.1016/j.ifacol.2018.07.310
39. Rumelhart DE, Hinton GE, Mcclelland JL (1986) A General framework for Parallel Distributed Processing

40. Ruta D (2014) Automated trading with machine learning on big data. In: Proceedings of the 2014 IEEE international congress on big data, bigdata congress 2014, pp. 824–830. https://doi.org/10.1109/BigData.Congress.2014.143. https://ieeexplore.ieee.org/abstract/document/6906878/
41. Samuelson PA, Nordhaus WD (2009) Economics 19th International Edition
42. Sezer OB, Ozbayoglu AM (2018) Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. Applied Soft Computing Journal 70:525–538. https://doi.org/10.1016/j.asoc.2018.04.024
43. Sharpe WF (1966) Mutual fund performance. J Bus 39(1):119–138
44. Shiller RJ (1999) Human behavior and the efficiency of the financial system. Handb Macroecon 1(PART C):1305–1340. https://doi.org/10.1016/S1574-0048(99)10033-8
45. Shin HW, Sohn SY (2004) Segmentation of stock trading customers according to potential value. Expert Syst Appl 27(1):27–33. https://doi.org/10.1016/j.eswa.2003.12.002
46. Sirignano J, Cont R (2019) Universal features of price formation in financial markets: perspectives from deep learning. Quantitative Finance 19(9):1449–1459
47. Šonje V, Alajbeg D, Bubaš Z (2011) Efficient market hypothesis: is the Croatian stock market as (in) efficient as the US market. Financial theory and practice 35(3):301–326
48. Sun J, Li H, Fujita H, Fu B, Ai W (2020) Class-imbalanced dynamic financial distress prediction based on adaboost-svm ensemble combined with smote and time weighting. Information Fusion 54:128–144
49. Sutskever I (2013) Training recurrent neural networks. University of Toronto, Toronto, Canada
50. Tai KS, Socher R, Manning CD (2015) Improved semantic representations from tree-structured long short-term memory networks. ACL-IJCNLP 2015 - 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing of the asian federation of natural language Processing. Proceedings of the Conference 1:1556–1566
51. Tay FEH, Cao LJ (2001) Improved financial time series forecasting by combining support vector machines with self-organizing feature map. Intelligent Data Analysis 5(4):339–354
52. Teixeira LA, De Oliveira ALI (2010) A method for automatic stock trading combining technical analysis and nearest neighbor classification. Expert Syst Appl 37(10):6885–6890. https://doi.org/10.1016/j.eswa.2010.03.033
53. Theofilatos K, Likothanassis S, Karathanasopoulos A (2012) Modeling and trading the EUR/USD exchange rate using machine learning techniques. Technology & Applied Science Research 2(5):269–272. www.etasr.com
54. Treynor JL (1961) Market value, time, and risk. Time, and Risk (August 8, 1961)
55. Wang JL, Chan SH (2009) Trading rule discovery in the US stock market: an empirical study. Expert Syst Appl 36(3 PART 1):5450–5455. https://doi.org/10.1016/j.eswa.2008.06.119
56. Weber BW (1999) Screen-based trading in futures markets: Recent developments and research propositions. In: Proceedings of the Hawaii International Conference on System Sciences, p. 247. IEEE. https://doi.org/10.1109/hicss.1999.772767
57. Werbos PJ (1990) Backpropagation through time: what it does and how to do it. Proc IEEE 78(10):1550–1560. https://doi.org/10.1109/5.58337
58. Wilson CL (1994) Self-organizing neural network system for trading common stocks. In: IEEE International conference on neural networks - conference proceedings, vol. 6, pp. 3651–3654. https://doi.org/10.1109/icnn.1994.374924
59. Wong WK, Manzur M, Chew BK (2003) How rewarding is technical analysis? Evidence from Singapore stock market. Applied Financial Economics 13(7):543–551. https://doi.org/10.1080/0960310022000020906
60. Wu X, Chen H, Wang J, Troiano L, Loia V, Fujita H (2020) Adaptive stock trading strategies with deep reinforcement learning methods Information Sciences
61. Ye W, Duo W (2019) Autonomous forex trading agents. In: ACM International conference proceeding series, pp. 205–210. https://doi.org/10.1145/3373419.3373436
62. Zhang F (2010) High-frequency trading, stock volatility, and price discovery Available at SSRN 1691679
63. Zhang W, Skiena S et al (2010) Trading strategies to exploit blog and news sentiment. In: Icwsm

Dušan Fister received a BSc and MSc in mechatronics from the University of Maribor, Slovenia and is currently working towards the PhD in economics. He is certified in quantitative risk management (CQRM). His areas of interest include econometrics, data mining, optimization, robotics and controllers. He has published more than 60 research articles in referred journals, conferences and book chapters. He is currently an assistant at Faculty of Economics and Business, University of Maribor.

Matjaž Perc is professor of physics at the University of Maribor. He is a member of Academia Europaea and the European Academy of Sciences and Arts, and among top 1% most cited physicists according to 2020 Clarivate Analytics data. He is also the 2015 recipient of the Young Scientist Award for Socio and Econophysics from the German Physical Society, and the 2017 USERN Laureate. In 2018 he received the Zois Award, which is the highest national research award in Slovenia. In 2019 he became Fellow of the American Physical Society.

Timotej Jagrič, received his first PhD from University of Maribor and his second PhD from University of Primorska. He is certified in quantitative risk management (CQRM), and is full professor of applied economics and econometrics and full professor of finance. He works at University of Maribor and is head of the Institute for Finance and Artificial Intelligence. He gives classes at the Faculty of Economics and Business, Faculty for natural sciences and mathematics and other Universities in and outside Slovenia. He was visiting researcher at Humboldt University of Berlin in the Collaborative Research Center (SFB) 649 "Economic Risk" and is currently visiting professor at Technical University of Graz. He has worked as chairmen of the supervisory board of the largest re-insurance company in Slovenia, as member of the Strategic council of the government of the Republic of Slovenia, and as advisor for many government and other public institutions in CEE region. He is actively working as consultant in the field of risk management and artificial intelligence, is member of the steering committee of the Slovenian risk association - SI.RISK, and acts as an expert for WHO.