This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2811377, IEEE Transactions on Industrial Informatics

Replicating a Trading Strategy by means of LSTM for Financial Industry Applications

Luigi Troiano, *Member, IEEE,* Elena Mejuto Villa, *Student Member, IEEE,* Vincenzo Loia, *Senior Member, IEEE*

***Abstract*—This paper investigates the possibility of learning a trading rule looking at the relationship between market indicators and decisions undertaken regarding entering or quitting a position. As means to achieve this objective we employ a Long-Short Term Memory (LSTM) machine, due its capability to relate past and recent events. Our solution is a first step in the direction of building a model-free robot, based on Deep Learning, able to identify the logic that links the market mood given by technical indicators to the undertaken investment decisions. Although preliminary, experimental results show that the proposed solution is viable and promising.**

I. Introduction

DEEP LEARNING (DL) is gaining a wide popularity for industrial applications due its capability to train complex non-linear models in very large parameter spaces over massive datasets. Architectures and models have been investigated in the past two decades, but availability of GPU computing enabled DL industrial applications such as automotive [1], control systems [2], fault detection and recovery [3], [4] and biomedicine [5], [6], just to mention some.

Financial Industry is one the sectors that can benefit more of DL in automating complex decision making, due to the wider range of information today made real-time available by multiple sources and because of DL capability to explore non-linear relations within and/or between different sources of information. Most of the decisions regard the issue of buy/sell orders according to the market mood captured by technical indicators, that are metrics whose value is derived from price and volume time series in a stock or asset. A trading strategy is the set of rules followed in taking such decisions by human traders or algorithms, being this second option preferred to the first to trade in fast paced financial markets.

Our research hypothesis is that a robot trained by DL is able to replicate the logic underlying a strategy only by looking at undertaken trading decisions. As said, those decisions are taken by traders according to technical indicators. So, they are the only source of information given to the robot, without assuming any model for the rules followed by the trader. The remainder of this paper is organized as follows: Section 2 provides a very brief overview of related literature, Section 3 describes the DL model we consider for our experimentation, Section 4 reports the experiment and Section 5 outlines conclusions and future directions.

Manuscript received October 6, 2017; revised January 22, 2018.

L. Troiano and E. Mejuto Villa are with the Department of Engineering, University of Sannio, Benevento, 82100 Italy (email: troiano@unisannio.it, mejutovilla@unisannio.it).

V. Loia is with the Department of Innovation Systems, University of Salerno, Fisciano, 84084 Italy (email: loia@unisa.it).

II. Existing applications of ML/DL to Finance

Besides model driven approaches (e.g., see [7], [8]), a vast literature concerns applications of machine learning (ML) to the Financial Industry with respect to time series forecasting, prediction of price movements, portfolio management, risk assessment, identification of trading strategy parameters, and similar. More recently, DL follows the same directions.

Various DL architectures have been investigated for predicting different kinds of financial time series. For instance, the forecasting of stock prices has been studied by Cai, Hu and Lin [9], where they propose a combined approach consisting of a Restricted Boltzman Machine (RBM) to extract discriminative low-dimensional features and a Support Vector Machine (SVM) for regression. A different approach is followed by Chen, Zhou and Dai [10] in order to predict the stock market returns by means of Long-Short Term Memory (LSTM). Persio and Honchar [11] investigate different artificial neural network (ANN) architectures, namely Multilayer Perceptron (MLP), Convolutional Neural Network (CNN) and LSTM, to stock price movement forecasting, where the prediction of future trend movements are based on past returns. Authors also consider a feature extraction based on Wavelet Transform as preliminary to the prediction task which yields better results. A similar problem is faced by Dixon, Klabjan and Hoon Bang [12], where they make use of a Deep Neural Network (DNN) as predictor of price movements over the following 5-minutes for several commodities and forex futures. As input they consider price differences, price moving averages and return pair-wise correlations to build a memory from historical data and capture co-movements between symbols. Portfolio management have been investigated by Heaton, Paulson and Witte [13] where they present an automated portfolio selection procedure based on, first encoding a large dataset of historical returns by means of an Autoencoder (AE) and then decoding it by solving an optimization problem.

Closer to the problem faced in this paper, Arévalo, Niño, Hernández and Sandoval [14] propose a High-Frequency Trading (HFT) strategy based on the output given by a DNN used to forecast the next 1-minute average price. Instead, we aim to replicate an existing strategy by learning hidden rules that link technical indicators to decisions in assuming long/short positions over time. DL is being widely implemented to process temporal information regarding video, audio and text [15], [16]. Our approach differs from other financial studies described above since we aim to exploit this capability for teaching a robot about how to trade financial markets by means of LSTM and using only technical indicators of historical data.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2811377, IEEE Transactions on Industrial Informatics

III. MODEL

In our experimentation, we assume that the strategy is implemented by a rational agent according to some undisclosed algorithm. The agent plays the role of teaching the strategy to a robot, which plays the role of learner. The training procedure is used to identify the relationship that stands between indicators and market positions, coded as +1 for "long" (positions acquired trough a buy first and sell after action sequence), -1 for "short" (positions acquired trough a sell first and buy after action sequence) and 0 for "hold" (no position acquired). Once the training is complete, the LSTM is used to get decisions which are expected to replicate the original strategy. The learning process is outlined by Fig. 1. This approach has the advantage of producing a minimal organizational overhead, since the training can have place online in parallel to the actual trading activities, with the robot taking over the agent when the training is complete.

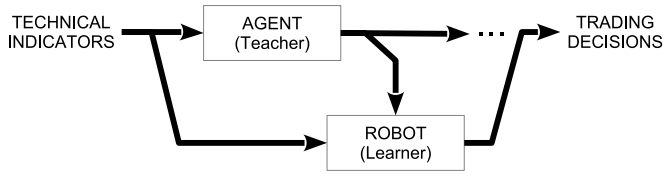

Fig. 1. Reference model

RNNs can be employed as Universal Turing Machine learners, thus suitable to learn a trading strategy. Among them, LSTM [17], [18] gained interest due their capability in taking into account dependencies from both longer and shorter term events. There several variants of LSTM, e.g., see [19]–[21]. In common they share the overall recurrent architecture outlined in Fig. 2 (top). They are a class of RNN able to capture the state, given enough units in a high dimensional space. The weighting matrix is trained in order to control the evolution of states. LSTM, is capable of replicating the output of any computable function [22]. This makes LSTM suitable for processing tasks entailing sequence processing, such as in case of speech recognition [23], [24], natural language processing [25], [26], -omics sciences [27], [28], automatic control [29], [30], and others. In addition, LSTM is proving to be effective in time series modeling and prediction [31], [32].

Each LSTM unit consists of a structure called *memory cell* which is able to store information over long periods of time, by updating the internal state. A memory cell is composed of four main elements: *input gate*, *forget gate*, *output gate* and *cell state*. There are several variations of the original model proposed by Hochreiter and Schmidhuber [17]. In this paper we adopt the model presented in [33] which is implemented given the set of equations below and schematically described by Fig. 2 (bottom).

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \quad (1)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f), \quad (2)$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \quad (3)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o), \quad (4)$$
$$h_t = o_t \tanh(c_t). \quad (5)$$

where x is the input vector to the LSTM unit, i, f, o are *activation vectors* used respectively to control the forget gate, the input gate and the output gate; c represents the cell input activation vector, and h is the cell output vector. The function σ is the logistic sigmoid function and W and b represent weight matrices and bias terms associated to the different activation vectors for each equation.

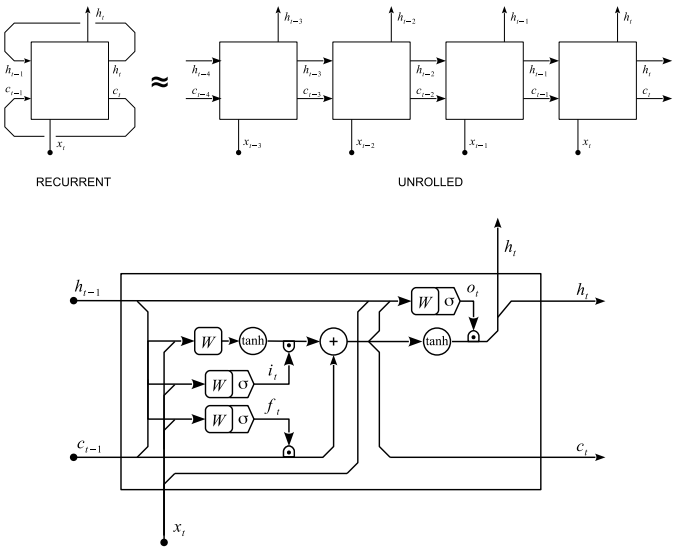

Fig. 2. LSTM unrolled structure (top) and internal block structure (bottom)

The key to success of LSTM is the cell input activation vector, c_t, which keeps memory about the cell state at each time t. The state is refreshed according to Eq.(3), regulated by the forget f_t and the input gates i_t. The role of f_t is to allow the cell to remember or forget its previous state c_{t-1}, (Eq.(2)). The forget gate is counterbalanced by the input gate that, making use of the same information (Eq.(1)), has instead the role of allowing or blocking the incoming signal to update the cell state (Eq.(3)). Once the cell state is recomputed at time t, the LSTM emits the output h_t as a filtered version of the cell state. First, the output gate o_t imposes what parts of the cell state will be transmitted to the output and also to the next memory cell (Eq.(4)). Then, the output h_t is computed by multiplying a normalized version of the cell state (through *tanh* function) by the value of the output gate (Eq.(4)). All gates modulate the respective signals according to their activation level by means of a logistic function σ. Instead for the the inputs and outputs is preferred to use a tanh dues its capability to provide both negative and positive values. They are necessary in the loop-back in order to compensate current information within the cell activation potential. Therefore, the internal structure of LSTM cell is made of multiple perceptrons and back propagation algorithm is generally the most common choice for training.

With respect to other popular RNNs, e.g. those based on Elman's and Jordan's reference models, the main advantage of this architecture is in using the cell state c_t that is updated under the control of the mentioned gates. This allows to modulate the interactions between the memory cell and its environment, by limiting the gradient to the last stage (also known as constant error carousels [34], [35]. This helps to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2811377, IEEE Transactions on Industrial Informatics

prevent the LSTM from vanishing the gradient too quickly [36], [37].

As part of current developments, we mention the joint application of CNN and LSTM for time series and sequence processing, e.g. see [38]. In general, the purpose is to map raw data to a feature space by CNN as first stage, eventually embedding partial correlations, and to use values in this space as input to LSTM at second stage.

IV. Experimentation setting and results

A. Data

For the experiments, we use the historical data concerning the adjusted close price series of the 30 components of the Dow Jones Industrial Average index, that are listed alphabetically in Table I. All of them are used along the process of model tuning performed during the experimentation in order to make our conclusions more robust. The date range is from 1 January 2012 to 31 December 2016. The plots of some of these price series are shown in Fig. 3.

TABLE I
THE 30 DOW JONES STOCKS.

Symbol	Company	Symbol	Company
AAPL	Apple Inc.	KO	Coca-Cola Co.
AXP	American Express	MCD	McDonal's
BA	Boeing Co.	MMM	3M
CAT	Caterpillar Inc.	MRK	Merck
CSCO	Cisco Systems	MSFT	Microsoft
CVX	Chevron Corp.	NKE	Nike Inc.
DD	DuPont	PFE	Pfizer
DIS	The Walt Disney Company	PG	Procter & Gamble
GE	General Electric	TRV	Travelers Co.
GS	Goldman Sachs	UNH	UnitedHealth Group
HD	Home Depot	UTX	United Technologies
IBM	Int. Business Machines Corp.	V	Visa
INTC	Intel Corp	VZ	Verizon
JNJ	Johnson & Johnson	WMT	Wal-Mart Stores
JPM	JP Morgan Chase & Co.	XOM	Exxon Mobil

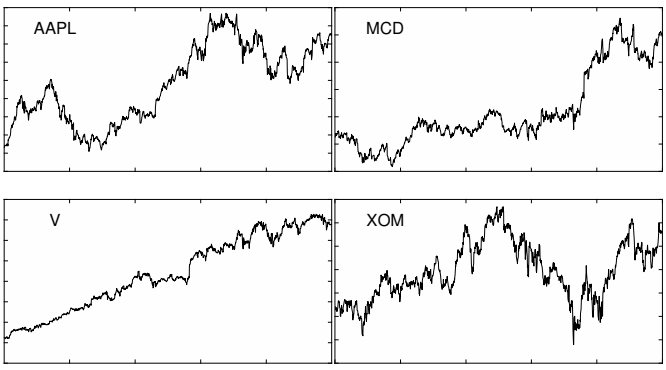

Fig. 3. Adjusted close price series of four stocks of the Dow Jones index.

B. Methodology

Three strategies are used in order to test replication capabilities in different scenarios: (i) Random Choice (RC), (ii) Moving-Average (MA) Crossover and (iii) Moving Average Convergence/Divergence (MACD).

Algorithm 1 Routine used for simulating the agent which follows Random Choice strategy.

1: **procedure** RANDOMCHOICE(n)
Input: n, number of days to simulate
Output: y, the sequence of decisions to take

2: $y[0] \leftarrow 0$, decision on initial day
3: $t_0 \leftarrow 0$

4: **for** $t = 1...n-1$ **do**
5: $p \leftarrow e^{t-t_0}$
6: $d \leftarrow trial(\{Keep, Change\}, p)$
7: **if** $d = Change$ **then**
8: $y[t] \leftarrow trial(\{-1,0,1\} \setminus y[t-1], \frac{1}{2})$
9: $t_0 \leftarrow t$
10: **end if**
11: **end for**
12: **return** y

13: **end procedure**

1) Random Choice: This strategy offers a borderline case in which the agent takes decisions completely at random without respect to underlaying technical indicators. In particular, the agent follows the sequence of decisions as obtained by Algorithm 1. At the beginning, the agent takes a neutral position (line 2), while t_0 marks the beginning of the position (line 3). The simulation is iterated over the subsequent $n-1$ days (line 4). Each the probability p (line 5) of keeping a position (line 6) decays exponentially over the time. If the decision of changing position is taken (line 7), the agent chooses one at random among the the others (line 8) and time t_0 moves ahead (line 9). A the end of the process, the sequence of decisions is returned (line 12).

2) Crossover Strategy: This is a very basic strategy based on moving averages (MAs). A price crossover has place when a short-term, i.e., faster MA crosses a long-term, i.e., slower MA. Crossover is used to identify shifts in the momentum, which dives to long/short and short/long position switches in the market. In particular, as depicted by Fig. 4, when the faster MA crosses over the slower MA, this is used as a signal of uptrend and a long position is entered. Conversely, when the faster MA crosses below the slower MA, this is used as signal of downtrend, so that the long position is released and a short position is entered.

In our experiment, we simulated an agent that makes use of a 20-period short-term MA and a 40-period long-term averages MA. Fig. 4 plots both the faster and the slower MA, together with the position assumed at the corresponding crossover points over a limited period of time.

3) Moving Average Convergence/Divergence (MACD) Histogram Strategy: In this case the agent implements a strategy using the signals originated by the MACD Histogram, that is given as the difference between the MACD Line and the Signal Line. The Signal Line is the 9-period exponential moving average (EMA) of the MACD Line that is, in turn, the difference between a slower 26-period EMA and a faster 12-period EMA of the market price. The resulting MACD Histogram is therefore an oscillator that moves above and below the zero line that can be assumed as momentum indicator in order to generate entry/exit signals as outlined below:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2811377, IEEE Transactions on Industrial Informatics

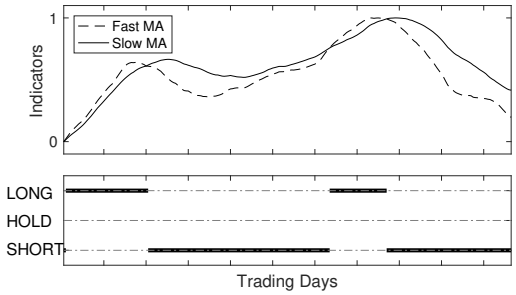

Fig. 4. Crossover strategy for AAPL stock in a limited period.

- Buy Entry: Hist. value is below -0.4.
- Buy Exit: Hist. value gets bigger than -0.1.
- Sell Entry: Hist. overcomes the value 0.4.
- Sell Exit: Hist. crosses below 0.1.

We notice that the value assumed by the MACD Histogram depends on the range of price variations. In order to define a strategy that is independent from stock-related thresholds, the Histogram's positive and negative values have been normalized within the range $[-1,+1]$. Fig. 5 offers an extract of trading decisions assumed by the MACD strategy over a limited period.

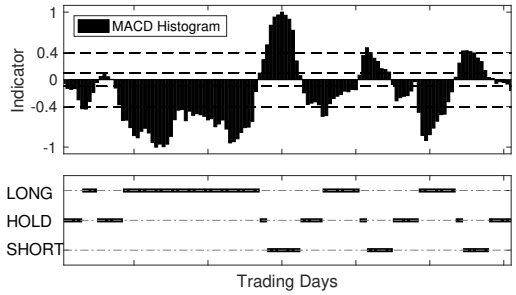

Fig. 5. MACD strategy for AAPL stock in a limited period.

The capability of replicating each strategy is measured as the matching between the position predicted by the robot against to the position assumed by the agent, with the robot unaware of the strategy used by the agent. Therefore, the quality of results does not reflect the strategy performance in terms of profits and losses.

C. Experimental Setting

All the experiments were carried out on a workstation equipped with an Intel® Core™ i7-6700 Processor, 3.4GHz x8, 31.3GB RAM and GPU GeForce GTX 980 with 4GB RAM on board. The software environment used was Anaconda Python 3.5.[1] LSTM was implemented by means of Keras,[2] a high-level neural networks API, using TensorFlow[3] as backend engine. Technical indicators were computed by means of a Python wrapper of the software library TA-Lib.[4]. We did not make use of any pre-trained network, in order to make experimental results independent from that kind of choice.

[1]https://www.continuum.io/
[2]https://keras.io/
[3]https://www.tensorflow.org/
[4]http://ta-lib.org/

D. Findings

1) Experiment A - Single unit: In the first experiment we try to replicate each of the strategies giving as input only the information strictly required to make decisions. That is, for the Crossover strategy we assume as input the difference between the faster MA and the slower MA; for MACD strategy, the input given is the Histogram; instead, since RC does not depend on any technical indicator, we used at random one of them as input.

LSTM, as other neural networks, is sensitive to the range of input data because of different activation functions that are employed along the model, in our case the *sigmoid* or *tanh* functions. This requires to normalize the data. The output and target are series of the same length of input series. They provide the decision vector given respectively by the robot and the agent. We assumed standard values $\{-1,0,1\}$ to represent short, hold and long positions respectively. Note that in the case of Crossover strategy, no neutral position is assumed, so the output can only take values $\{-1,+1\}$.

Since we are working with time series, the sequence of values is important, therefore the split of the dataset was given assuming for each stock a first period of 80% for training and the remaining 20% for testing. No validation set was considered in this first experiment.

In this experiment we consider a simple architecture, consisting of one single layer made of LSTM with 1 unit for the state, so that the output is direct function of that unit. The batch size is computed to be as longer as possible (it depends on the number of samples, since it must be a common divisor between the training and testing sets). The state is reset after each training batch. The network was trained along 3,000 epochs in the case of RC, and 15,000 epochs for the Crossover and MACD. The algorithm used for training is a stochastic gradient descent optimization as implemented by *RMSProp*[39], an adaptive learning rate method proposed by Hinton. The loss function is the conventional *mean square error* (MSE).

The training of LSTM can adopt samples made of multiple time steps as input. So, we repeated the experiment by considering different sample lengths, in order to check how performance is affected by an increasing lookback period.

Results are reported in Table II-1, that gives the accuracy reached along both the training and the testing periods. Since 30 different stocks have been used, accuracy is summarized in terms of mean and variance. As expected, in the case of the RC, accuracy is approximately one third, that is the probability of getting the correct prediction at random. Because of this, RC will not be further considered in the remainder of experimentation. Instead accuracy is high in the case of the Crossover, since the network have to learn that position depends only on the sign of the input data that is, as we said, the difference between two MAs. Finally, accuracy for the MACD is good but lower. This is because of the coexistence of multiple thresholds to learn in making decisions.

In order to avoid the model to overfit the data and to make shorter the training period, we consider an *early-stopping* regularization as proposed by Yao, Rosasco and Caponnetto [40]. This is designed to stop the training when the model

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2811377, IEEE Transactions on Industrial Informatics

TABLE II
ACCURACY IN EXPERIMENT A.

1 - NO EARLY STOPPING

	Random		Crossover		MACD	
lookback	train	test	train	test	train	test
1	32.63%	31.76%	85.78%	90.24%	85.08%	80.03%
	±1.63%	±5.55%	±3.06%	±4.14%	±3.20%	±6.15%
5	32.87%	33.71%	87.26%	91.14%	88.68%	84.21%
	±1.79%	±5.19%	±2.34%	±3.38%	±2.66%	±6.00%
10	32.84%	33.71%	87.06%	90.92%	89.92%	85.11%
	±1.85%	±5.21%	±2.96%	±3.97%	±2.85%	±6.51%
15	32.94%	33.55%	87.04%	90.99%	90.66%	85.49%
	±1.87%	±5.22%	±4.18%	±4.65%	±2.64%	±5.61%
20	32.99%	33.84%	86.46%	90.21%	91.00%	85.42%
	±1.82%	±5.43%	±5.88%	±6.42%	±2.47%	±5.31%

2 - WITH EARLY STOPPING

	Crossover			MACD		
lookback	train	test	epochs	train	test	epochs
1	85.89%	91.17%	22237	85.40%	76.53%	6142
	±4.26%	±3.68%	±5289	±3.93%	±8.45%	±2311
5	85.08%	89.51%	5632	90.30%	85.17%	5004
	±2.82%	±3.47%	±2981	±3.63%	±4.64%	±2254
10	86.68%	91.00%	17046	90.75%	83.82%	5845
	±3.29%	±3.82%	±4264	±3.18%	±6.39%	±2632
15	86.85%	91.12%	15789	91.61%	83.88%	6864
	±3.55%	±3.48%	±3929	±2.96%	±6.44%	±2364
20	86.29%	90.85%	15566	91.59%	83.48%	5337
	±4.83%	±4.92%	±4051	±3.38%	±6.12%	±1479

performance does not improve anymore. We apply early-stopping by monitoring the accuracy on a validation period made of a trailing 20% of the training period and we set the *patience* limit, i.e., the maximum period standing with no improvement, to $2,500$ epochs, as this gave us best results among other values we tested. Table II-2 shows the results when early stopping is employed. Accuracy is not affected, but reached within fewer epochs. In particular, Crossover is able to improve further the accuracy and this leads to delay the early stopping. The better fit is also confirmed by lower variance for the accuracy over the 30 stocks. Instead, MACD capability to learn the decision rule is in general lower and this leads to get an early stopping signal earlier.

2) Experiment B - Multiple units: As next step, we consider a different architecture made of a fully-connected dense layer over an LSTM layer made of 15 units. This solution increases the memory capability of the LSTM and shorten the training. The output of the 15 LSTM units is used as input to the dense layer, that is aimed at providing the decision as output, using *tanh* as activation function. We employ a 0.25 *dropout rate* in order to curb the possible overfitting [41]. Table III reports the results of this experiment. The improvement for Crossover is evident, reaching even the 100% of accuracy for some stocks. MACD replication does not show any improvement, being even worse for longer lookback periods, as it seems to be more affected by overfitting (as accuracy along the training period considerably increased). Indeed, by adding a dense layer, accuracy on the training improves as the model is able to better fit the training period, but not necessarily improving over the testing period. In the remainder, we will assume a standard lookback period of 5 as this value seems to offer an appropriate trade-off between memory and risk of overfitting.

TABLE III
ACCURACY AND TRAINING EPOCHS IN EXPERIMENT B.

	Crossover			MACD		
lookback	train	test	epochs	train	test	epochs
1	98.88%	99.34%	6531	86.24%	79.36%	3300
	±0.49%	±0.70%	±1403	±4.43%	±6.53%	±690
5	97.93%	98.46%	7280	92.91%	89.21%	6931
	±0.71%	±0.93%	±1782	±3.83%	±3.32%	±3007
10	98.65%	97.93%	7693	96.42%	87.40%	5951
	±0.60%	±0.96%	±1672	±2.00%	±4.19%	±2184
15	98.81%	97.25%	6195	97.13%	83.41%	4724
	±0.70%	±1.72%	±1630	±2.30%	±5.63%	±1361
20	99.07%	95.85%	6379	97.42%	81.11%	3792
	±0.76%	±2.14%	±184	±1.56%	±6.06%	±943

3) Experiment C - Output representations: Up to now we did not consider the issue related to how the output space is represented. In this experiment we consider several options regarding (i) what is the output range expressed by the activation function used for the output layer, (ii) the coding, and (iii) the cost function used to train the network.

Table IV-1 reports experimental results when the sigmoid is used as activation function. The output can be coded (C) as an index, each representing one of the possible trading decisions (market positions). Alternatively we can get an output that is "uncoded" (U), so that each decision/position is given as an individual binary value. In this case, besides the conventional loss function based on MSE, we can use the *cross-entropy* (CE). Accuracy for Crossover is practically comparable to previous results. However, we notice a substantial reduction of the training epochs. For MACD strategy, the integer coding does not stand due to saturation of sigmoid for values over 1. We achieve a better accuracy when a binary coding is used together with CE loss function, preventing a too early stopping. The larger number of epochs for training supports this conclusion.

Table IV-2 outlines results when the *softplus* is used in place of the sigmoid. Crossover performance is not affected when MSE is used as loss function. Instead, the CE loss function used for binary coding does not allow the network to learn the strategy. This is because accuracy does not improve quick enough to avoid early stopping. This problem does not stand for MACD, although we observe a general worsening.

In addition, we also tested the application of *softmax* and *ReLU* functions. However we did not obtained any improvement. For the sake of readability we avoid to report experimental tables for both cases. In reason of results outlined above, in the remainder of the experimentation we will make use of the *sigmoid* activation which yields better results, except when using integer labels for MACD. In the latter case we will consider the *softplus* activation function.

4) Experiment D - Input space: In all experiments above, we used as input solely the indicator that provides the trading signals. This simplifies the learning task as decisions can be directly linked to the input, for example just considering the sign for Crossover or thresholds for MACD.

In this experiment we complicate the task by requiring the network to combine multiple series in order to get trading signals. In particular, we assume the two MAs in place of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2811377, IEEE Transactions on Industrial Informatics

TABLE IV
ACCURACY AND TRAINING EPOCHS IN EXPERIMENT C.

1 - SIGMOID ACTIVATION

		Crossover			MACD		
output	loss	train	test	epochs	train	test	epochs
C	MSE	98.85% ±0.35%	98.87% ±0.71%	5356 ±1817	–	–	–
U	MSE	98.79% ±0.38%	98.96% ±0.78%	4759 ±1600	92.17% ±3.76%	86.17 ±5.65%	6649 ±3719
U	CE	98.74% ±0.43%	98.78% ±0.81%	4678 ±1323	95.35% ±2.51%	92.97% ±2.34%	8971 ±3347

2 - SOFTPLUS ACTIVATION

		Crossover			MACD		
output	loss	train	test	epochs	train	test	epochs
C	MSE	98.53% ±0.55%	98.53% ±1.05%	5931 ±1373	92.87% ±3.38%	88.01% ±4.04%	7138 ±3324
U	MSE	98.65% ±0.47%	98.66% ±0.96%	6254 ±1571	92.53% ±3.86%	88.22% ±3.85%	6868 ±2861
U	CE	39.93% ±8.13%	42.30% ±9.59%	3076 ±467	83.01% ±5.85%	79.08% ±6.53%	3937 ±1586

their difference for the Crossover and the two price EMAs together with the MACD Line and the Signal Line in place of the Histogram for the MACD. Therefore, in both cases, the network is demanded to learn how to combine different inputs, although all correlated to the decision. In addition, in order to complicate further the learning task and to consider more realistic situations, we mix the relevant inputs (direct or correlated information to decisions) with unrelated data. For this experiment we use 5 additional indicators that are not related to trading signals, as summarized in Table V.

TABLE V
LIST OF UNRELATED INDICATORS

Indicator Description	Type of Indicator
Midpoint over period	Overlap studies
Hilbert Transform - Dominant Cycle Period	Cycle Indicator
Highest value over a specified period (MAX)	Math Operator
Vector Trigonometric Sine (SIN)	Math Transform
Absolute Price Oscillator (APO)	Momentum

Table VI-1 outlines that model accuracy gets worse when using correlated data. The Crossover strategy deteriorates even along the training period, thus more epochs may be required to improve. Instead, MACD shows up a slight improvement along the training, although accuracy decreases significantly along the testing period, suggesting a lack of generalization.

In Table VI-2 and Table VI-3, we report performance statistics when, respectively, direct and correlated inputs are mixed to the five unrelated indicators. Table VI-2 outlines a worser performance, since additional indicators represent a relevant source of noise to discover trading signals. So that, accuracy slightly improves along the training period, but it is lower on the testing period, possibly because of overfitting along the training. Results are different when unrelated indicators are mixed with correlated input series. In some cases the performance improves, possibly because the noise introduced by additional indicators is able to limit the overfitting and to speed up the training process.

TABLE VI
ACCURACY AND TRAINING EPOCHS IN EXPERIMENT D.

1 - CORRELATED INFORMATION

		Crossover			MACD		
output	loss	train	test	epochs	train	test	epochs
C	MSE	94.25% ±2.58%	90.51% ±6.30%	6143 ±3327	93.96% ±2.96%	67.55% ±16.64%	4462 ±1515
U	MSE	93.33% ±2.49%	87.99% ±8.52%	6080 ±3789	93.81% ±3.68%	72.60% ±12.33%	5087 ±2738
U	CE	95.63% ±1.84%	92.25% ±6.07%	6654 ±4505	96.54% ±2.31%	81.09% ±8.39%	5818 ±3024

2 - DIRECT USEFUL INFORMATION AND UNRELATED INDICATORS

		Crossover			MACD		
output	loss	train	test	epochs	train	test	epochs
C	MSE	98.78% ±0.70%	94.40% ±3.46%	3063 ±823	97.58% ±1.82%	69.80% ±14.19%	3431 ±1010
U	MSE	98.81% ±0.56%	94.19% ±2.99%	2832 ±300	96.99% ±1.76%	72.98% ±11.49%	3742 ±1355
U	CE	98.94% ±0.63%	94.71% ±2.92%	2972 ±487	98.91% ±1.23%	81.06% ±7.48%	4346 ±1657

3 - CORRELATED INFORMATION AND UNRELATED INDICATORS

		Crossover			MACD		
output	loss	train	test	epochs	train	test	epochs
C	MSE	96.89% ±1.39%	89.89% ±4.91%	4595 ±1437	95.96% ±2.42%	70.65% ±11.43%	3584 ±795
U	MSE	97.05% ±1.19%	89.23% ±6.17%	4548 ±1443	96.18% ±1.86%	66.10% ±13.77%	4634 ±1503
U	CE	97.06% ±1.18%	91.08% ±5.24%	4460 ±1270	98.14% ±1.34%	78.11% ±8.14%	4486 ±1476

5) Experiment E: Keeping memory of decisions: Generally, trading decisions on future positions are influenced by the current position. This makes possible to filter out spurious signals that may lead to unstable decisions. This suggests to include an additional input which consists on the position assumed the day before. The experiment processes the training and the testing period differently. Along the training, we are able to use the agent position (target) as this is available in advance. Instead, at testing time, we inject the position assumed by the robot (output) the day before.

Table VII-1 shows the performance generally improved in terms of slightly higher accuracy along the training, especially for MACD, but no significant improvement is recorded along the testing period. Note that the learning process is bit faster. Tables VII-2, VII-3 and VII-4 summarize the model performance when adding the decision of the day before to the cases outlined in Section IV-D4. We conclude that including a memory does not always lead to better accuracy. Indeed, keeping the wrong position in memory may propagate this error several days ahead.

E. Convergence and Qualitative Assessment

One aspect to consider is how accuracy evolves along the training period. As a note of methodology, as we are interested on qualitative aspects concerning how the model accuracy behaves at each run, we will focus only on AAPL stock as example of a more general behavior. We will consider the best accuracy obtained so far at each epoch.

In Fig. 6, we compare Crossover and MACD strategies when we make use of direct information as input, coded

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2811377, IEEE Transactions on Industrial Informatics

TABLE VII
ACCURACY AND TRAINING EPOCHS IN EXPERIMENT E.

1 - DIRECT USEFUL INFORMATION WITH PREVIOUS DECISION

		Crossover			MACD		
output	loss	train	test	epochs	train	test	epochs
C	MSE	98.92%	98.95%	5798	97.24%	79.48%	4884
		±0.36%	±0.91%	±1267	±1.00%	±12.33%	±1953
U	MSE	98.77%	99.07%	5769	99.31%	89.07%	5998
		±0.56%	±0.79%	±1133	±0.36%	±5.52%	±1776
U	CE	98.96%	98.89%	5254	99.66%	92.64%	6563
		±0.37%	±1.00%	±1046	±0.22%	±4.25%	±1804

2 - DIRECT USEFUL INFORMATION AND UNRELATED INDICATORS WITH PREVIOUS DECISION

		Crossover			MACD		
output	loss	train	test	epochs	train	test	epochs
C	MSE	99.22%	95.96%	2984	98.35%	78.10%	3507
		±0.41%	±2.14%	±354	±0.66%	±8.68%	±707
U	MSE	99.25%	95.57%	3107	99.17%	76.79%	2862
		±0.44%	±1.99%	±531	±0.41%	±11.00%	±222
U	CE	99.17%	96.09%	2973	99.66%	77.44%	2924
		±0.53%	±1.98%	±357	±0.27%	±10.52%	±305

3 - CORRELATED INFORMATION WITH PREVIOUS DECISION

		Crossover			MACD		
output	loss	train	test	epochs	train	test	epochs
C	MSE	97.38%	68.17%	3194	96.75%	66.64%	3972
		±0.35%	±14.27%	±933	±1.09%	±15.21%	±1051
U	MSE	97.39%	69.39%	3781	97.47%	69.41%	3588
		±0.38%	±13.36%	±1296	±0.95%	±11.66%	±1430
U	CE	97.41%	79.65%	4184	98.40%	73.10%	3322
		±0.38%	±12.06%	±1401	±0.59%	±12.53%	±839

4 - CORRELATED INFORMATION AND UNRELATED INDICATORS WITH PREVIOUS DECISION

		Crossover			MACD		
output	loss	train	test	epochs	train	test	epochs
C	MSE	98.99%	89.67%	2774	97.51%	61.99%	3405
		±0.42%	±5.34%	±126	±0.89%	±17.19%	±709
U	MSE	98.96%	90.61%	2959	98.28%	63.94%	2933
		±0.31%	±3.65%	±424	±0.79%	±11.48%	±408
U	CE	99.11%	88.77%	3173	98.99%	60.32%	2934
		±0.49%	±7.33%	±737	±0.58%	±16.27%	±295

output and MSE as loss function. As expected, the learning for Crossover provides better accuracy along all the epochs. This is because Crossover is simpler to learn with respect to MACD. The convergences has place by means of steps within specific ranges of epochs, earlier for Crossover. This is expected, because the training algorithm finds a sequence of local optima and gets attracted to move towards them. The early stopping conditions triggers first for MACD.

Similar behavior can be found for different combination of output representations (Coded/Uncoded) and loss functions (MSE/CE), as outlined by Fig. 7. The coded representation shows an initial advantage in accuracy as this requires a lower dimension output with respect to the uncoded representation. However this difference stands only at the beginning, vanishing at later epochs.

As last case, we consider the MACD accuracy profile at varying the source of input information. As depicted by Fig. 8, the hardest case is when we use correlated (but not direct) information, as the LSTM is forced to find the function relationship between different quantities before taking a decision. This delays the convergence along the training

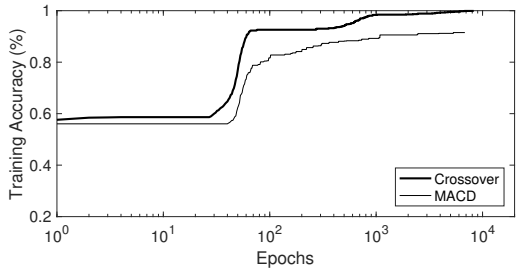

Fig. 6. Accuracy over training epochs for Crossover and MACD strategies in Experiment C. The output is coded. MSE is the loss function.

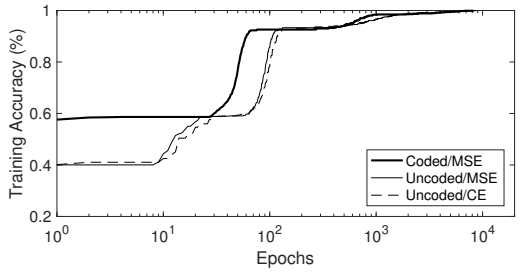

Fig. 7. Accuracy over training epochs for different output representations. Case of Crossover in Experiment C.

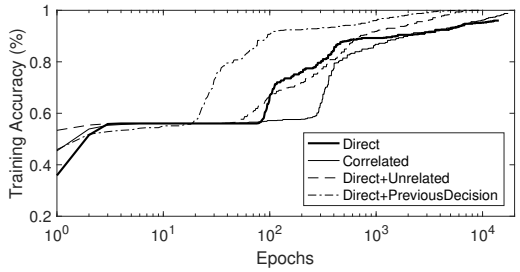

Fig. 8. Accuracy over training epochs for different input spaces. Case of MACD. The output is coded. CE is the loss function.

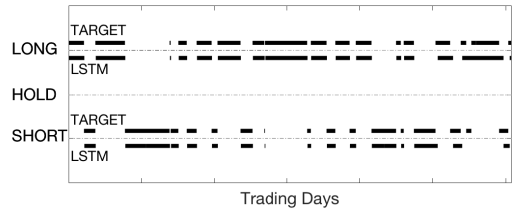

Fig. 9. Strategy decision (TARGET) and LSTM prediction. Case of Crossover in Experiment E-3. The output is coded. MSE is the loss function.

period. The learning process improves when direct information comes together with unrelated information, as this avoid to get trapped in local optima, and even more when we keep memory of the position assumed the day before.

A second aspect we take into account is about the nature of errors that affects accuracy. As shown in Fig. 9, a lower accuracy is mostly due to disalignments in positions, as they can be assumed by LSTM bit earlier or bit later, more than to sudden and episodic position changes. In other terms, errors are due to synchronism of agent and robot decisions, that can differ of few days. Therefore, the two sequences of decision are basically only shifted in time.

V. Conclusions and Future Work

In this paper we provided an investigation on the possibilities offered by LSTM machines to decode trading decisions in financial markets. The result of experimentation pointed out that this approach is viable. For simple behaviors, as those used for experimentation in this paper, an increase in the number of input features makes shorter the training but can easily lead to overfitted models. This effect should be limited by looking at more complex decisions as those attaining human traders. There are several directions that it worths to investigate for the future. Among them, a larger comparison with other and more complex architectures, possibly involving a preliminary convolutional step as some experiences in other fields are suggesting. This should be able to provide a layer of local feature correlations and implicit feature selection, able to better link indicators to decisions and reject unrelated information. Indeed, in general we observed a rapid performance deterioration by increasing the number of technical indicators given as input to the network. Possibly this because of combinatorial effects that multiple information sources produces on the state evolution. However, the use of technical indicators is generally performed by analyzing them individually, and combining them into a trading decision after. This suggests that an architecture based on multiple LSTMs, each devoted to a small subset of indicators and combined by further stacked layers over them, may be beneficial to network performance. Since a strategy is independent from the specific stock, it would be interesting to study how the network performs when trained over a larger or different set of stocks. The ultimate goal is to face the challenge of learning and performing the complex human behavior exhibited by traders in taking financial decisions.

References

[1] A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, and B. Vorster, "Deep learning in the automotive industry: Applications and tools," in *IEEE Int. Conf. on Big Data*, 2016. doi: 10.1109/BigData.2016.7841045 pp. 3759–3768.

[2] K. Cheon, J. Kim, M. Hamadache, and D. Lee, "On replacing pid controller with deep learning controller for dc motor system," *JOACE*, vol. 3, no. 6, pp. 452–456, 2015. doi: 10.12720/joace.3.6.452-456

[3] Z. Liu, Z. Jia, C. M. Vong, S. Bu, J. Han, and X. Tang, "Capturing high-discriminative fault features for electronics-rich analog system via deep learning," *IEEE Trans. on Industrial Informatics*, vol. 13, no. 3, pp. 1213–1226, 2017. doi: 10.1109/TII.2017.2690940

[4] K. B. Lee, S. Cheon, and C. O. Kim, "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes," *IEEE Trans. on Semiconductor Manufacturing*, vol. 30, no. 2, pp. 135–142, 2017. doi: 10.1109/TSM.2017.2676245

[5] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G. Z. Yang, "Deep learning for health informatics," *IEEE J. of Biomedical and Health Informatics*, vol. 21, no. 1, pp. 4–21, 2017. doi: 10.1109/JBHI.2016.2636665

[6] S. Lee and J. H. Chang, "Oscillometric blood pressure estimation based on deep learning," *IEEE Trans. on Industrial Informatics*, vol. 13, no. 2, pp. 461–472, 2017. doi: 10.1109/TII.2016.2612640

[7] L. Troiano and P. Kriplani, "A mean-reverting strategy based on fuzzy transform residuals," in *IEEE Conference CIFEr*, 2012. doi: 10.1109/CIFEr.2012.6327766 pp. 1–7.

[8] L. Troiano, "Fuzzy co-transform and its application to time series," in *2010 International Conference of Soft Computing and Pattern Recognition*, Dec 2010. doi: 10.1109/SOCPAR.2010.5686735 pp. 379–384.

[9] X. Cai, S. Hu, and X. Lin, "Feature extraction using restricted boltzmann machine for stock price prediction," in *IEEE Int. Conf. ICCSAE*, vol. 3, 2012. doi: 10.1109/CSAE.2012.6272913 pp. 80–83.

[10] K. Chen, Y. Zhou, and F. Dai, "A lstm-based method for stock returns prediction: A case study of china stock market," in *IEEE Int. Conf. on Big Data*, 2015. doi: 10.1109/BigData.2015.7364089 pp. 2823–2824.

[11] L. D. Persio and O. Honchar, "Artificial neural networks approach to the forecast of stock market price movements," *Int. J. of Economics and Management Systems*, vol. 1, pp. 158–162, 2016.

[12] M. Dixon, D. Klabjan, and J. Hoon Bang, "Classification-based financial markets prediction using deep neural networks," *Algorithmic Finance*, 2016. doi: 10.2139/ssrn.2756331

[13] J. B. Heaton, N. G. Polson, and J. H. Witte, "Deep learning for finance: deep portfolios," *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, pp. 3–12, 2017. doi: 10.1002/asmb.2209

[14] A. Arévalo, J. Niño, G. Hernández, and J. Sandoval, "High-frequency trading strategy based on deep neural networks," in *Proc. of 12th ICIC*, 2016. doi: 10.1007/978-3-319-42297-8_40 pp. 424–436.

[15] X. Chang, Y. L. Yu, Y. Yang, and E. P. Xing, "Semantic pooling for complex event analysis in untrimmed videos," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 39, no. 8, pp. 1617–1632, 2017. doi: 10.1109/TPAMI.2016.2608901

[16] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, vol. 24, no. 4, pp. 694–707, 2016. doi: 10.1109/TASLP.2016.2520371

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000. doi: 10.1162/089976600300015015

[19] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE TNNLS*, vol. 28, no. 10, pp. 2222–2232, 2017. doi: 10.1109/TNNLS.2016.2582924

[20] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased lstm: Accelerating recurrent network training for long or event-based sequences," in *30th Conf. NIPS*, 2016, pp. 3882–3890.

[21] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: Modeling user behaviors by time-lstm," in *Proc. of the 26th IJCAI*, 2016. doi: 10.24963/ijcai.2017/504 pp. 3882–3890.

[22] H. Siegelmann and E. Sontag, "On the computational power of neural nets," *J. of Computer and System Sciences*, vol. 50, no. 1, pp. 132 – 150, 1995. doi: 10.1006/jcss.1995.1013

[23] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *IEEE ASRU*, 2013, pp. 273–278.

[24] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang, H. Yang, and W. B. J. Dally, "Ese: Efficient speech recognition engine with sparse lstm on fpga," in *Proc. of the FPGA 2017*, 2017.

[25] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *IEEE SLT Workshop*, 2014. doi: 10.1109/SLT.2014.7078572 pp. 189–194.

[26] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent lstm neural networks for language modeling," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 23, no. 3, pp. 517–529, 2015. doi: 10.1109/TASLP.2015.2400218

[27] G. Leifert, T. Strauß, T. Grüning, W. Wustlich, and R. Labahn, "Cells in multidimensional recurrent neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 3313–3349, 2016.

[28] B. Lee, J. Baek, S. Park, and S. Yoon, "deeptarget: End-to-end learning framework for microrna target prediction using deep recurrent neural networks," in *Proc. of the 7th ACM Int. Conf. on Bioinformatics, Computational Biology, and Health Informatics*, 2016. doi: 10.1145/2975167.2975212 pp. 434–442.

[29] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *J. Mach. Learn. Res.*, vol. 3, pp. 115–143, 2003. doi: 10.1162/153244303768966139

[30] N. Hirose and R. Tajima, "Modeling of rolling friction by recurrent neural network using lstm," in *IEEE ICRA*, 2017. doi: 10.1109/ICRA.2017.7989764 pp. 6471–6478.

[31] M. A. Zaytar and C. El Amrani, "Sequence to sequence weather forecasting with long short-term memory recurrent neural networks," *Int. J. of Computer Applications*, vol. 143, no. 11, 2016.

[32] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, "An overview and comparative analysis of recurrent neural networks for short term load forecasting," *arXiv preprint arXiv:1705.04378*, 2017.

[33] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[34] F. A. Gers, J. A. Schmidhuber, and F. A. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 2000. doi: 10.1162/089976600300015015

[35] F. A. Gers and J. Schmidhuber, "Long short-term memory learns context free and context sensitive languages," in *Artificial Neural Nets and Genetic Algorithms*, 2001, pp. 134–137.

[36] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proc. of the 32nd ICML*, 2015, pp. 2342–2350.

[37] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *ICML*, 2013, pp. 1310–1318.

[38] J. B. Wolfgang Groß, Sascha Lange and M. Blum, "Predicting time series with space-time convolutional and recurrent neural networks," in *Proc. of the 25th ESANN*, 2017, pp. 71–76.

[39] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," 2012.

[40] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007. doi: 10.1007/s00365-006-0663-2

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.