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

Luigi Troiano, *Senior Member, IEEE*, Elena Mejuto Villa, *Student Member, IEEE*, and 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 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.**

***Index Terms*—Recurrent neural networks, robot Learning, stock markets.**

I. Introduction

DEEP LEARNING (DL) is gaining a wide popularity for industrial applications due its capability to train complex nonlinear 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 of 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 nonlinear 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 II provides a very brief overview of the related literature; Section III describes the DL model we consider for our experimentation; Section IV reports the experiment; and Section V outlines conclusions and future directions.

Manuscript received October 6, 2017; revised January 22, 2018; accepted February 7, 2018. Date of publication March 1, 2018; date of current version July 2, 2018. Paper no. TII-17-2357. *(Corresponding author: Vincenzo Loia.)*

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

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

Digital Object Identifier 10.1109/TII.2018.2811377

II. Existing Applications of Machine Learning (ML)/DL to Finance

Besides model-driven approaches (e.g., see [7] and [8]), a vast literature concerns applications of 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 *et al.* [9], where they propose a combined approach consisting of a restricted Boltzman machine to extract discriminative low-dimensional features and a support vector machine for regression. A different approach is followed by Chen *et al.* [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 architectures, namely multilayer perceptron, 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 that yields better results. A similar problem is faced by Dixon *et al.* [12], where they make use of a deep neural network (DNN) as a predictor of price movements over the following 5 min. for several commodities and Forex futures. As an input, they consider price differences, price moving averages (MAs), and return pairwise correlations to build a memory from historical data and capture comovements between symbols. Portfolio management has been investigated by Heaton *et al.* [13], where they present an automated portfolio selection procedure based on: first encoding a large dataset of historical returns by means of an autoencoder, and then decoding it by solving an optimization problem.

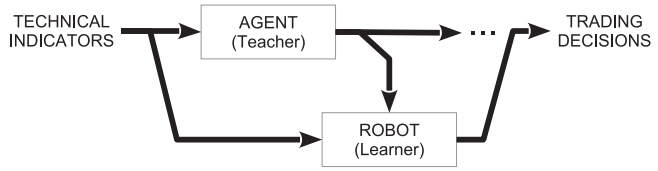

Fig. 1. Reference model.

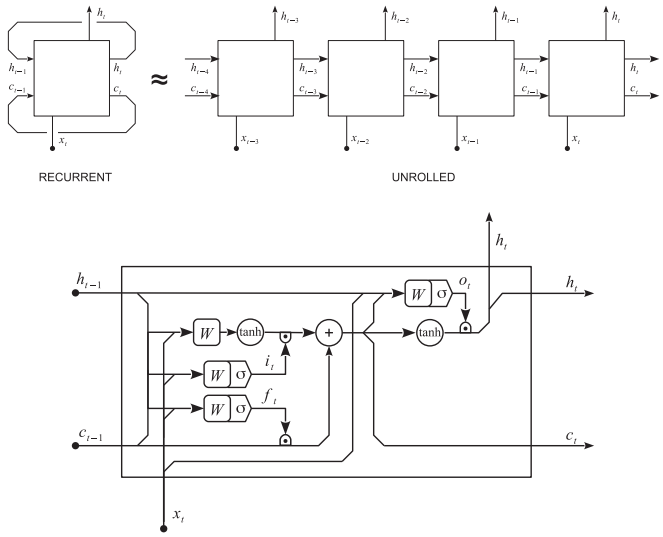

Fig. 2. LSTM unrolled structure (top) and internal block structure (bottom).

Closer to the problem faced in this paper, Arévalo *et al.* [14] propose a high-frequency trading strategy based on the output given by a DNN used to forecast the next 1-min 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.

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 a learner. The training procedure is used to identify the relationship that stands between indicators and market positions, coded as +1 for "long" (positions acquired through a buy first and sell after action sequence), −1 for "short" (positions acquired through 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 that 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 place online in parallel to the actual trading activities, with the robot taking over the agent when the training is complete.

Recurrent neural networks (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 are 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* that 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, $i, f,$ and 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.

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 (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} [see (2)]. The forget gate is counterbalanced by the input gate that, making use of the same information [see (1)], has instead the role of allowing or blocking the incoming signal to update the cell state [see (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 [see (4)]. Then, the output h_t is computed by multiplying a normalized version

TABLE I
30 DOW JONES STOCKS

Symbol	Company	Symbol	Company
AAPL	Apple, Inc.	KO	Coca-Cola Co.
AXP	American Express	MCD	McDonald'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	International Business Machines Corp.	V	Visa
INTC	Intel Corp.	VZ	Verizon
JNJ	Johnson & Johnson	WMT	Walmart Stores
JPM	JP Morgan Chase & Co.	XOM	Exxon Mobil

of the cell state (through tanh function) by the value of the output gate [see (4)]. All gates modulate the respective signals according to their activation level by means of a logistic function σ. Instead for the inputs and outputs, it is preferred to use a $\tanh$ function due to its capability to provide both negative and positive values. They are necessary in the loopback in order to compensate current information within the cell activation potential. Therefore, the internal structure of LSTM cell is made of multiple perceptrons and the backpropagation 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 prevent the LSTM from vanishing the gradient too quickly [36], [37].

As a 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 the first stage, eventually embedding partial correlations, and to use values in this space as the 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, which 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 January 1, 2012 to December 31, 2016. The plots of some of these price series are shown in Fig. 3.

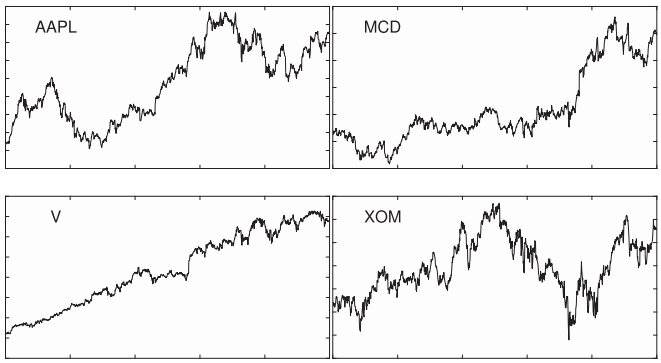

Fig. 3. Adjusted close price series of four stocks of the Dow Jones index.

B. Methodology

Three strategies are used in order to test replication capabilities in the following different scenarios:

1) random choice (RC);
2) MA crossover; and
3) MA convergence/divergence (MACD).

1) Random Choice: This strategy offers a borderline case in which the agent takes decisions completely at random without respect to underlying 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), whereas 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 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).

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: $y[t] \leftarrow trial\left(\{-1, 0, 1\} \setminus y[t-1], \frac{1}{2}\right)$
9: $t_0 \leftarrow t$
10: **end if**
11: **end for**
12: **return** y
13: **end procedure**

2) Crossover Strategy: This is a very basic strategy based on MAs. A price crossover has placed when a short term, i.e., faster MA crosses a long term, i.e., slower MA. Crossover is used

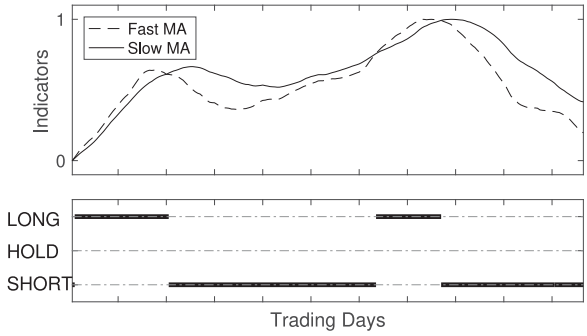

Fig. 4. Crossover strategy for AAPL stock in a limited period.

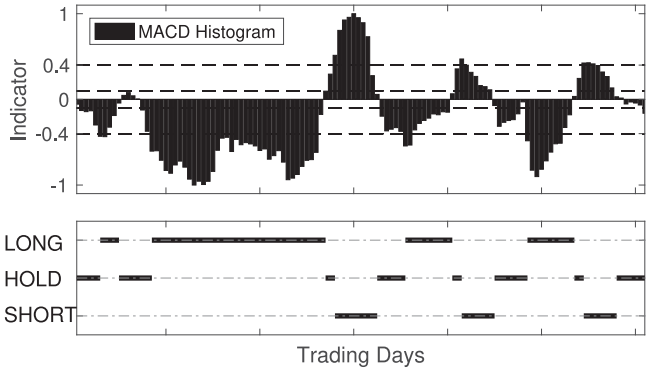

Fig. 5. MACD strategy for AAPL stock in a limited period.

to identify shifts in the momentum, which dives to long/short and short/long position switches in the market. In particular, as depicted in 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 a 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) MA 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 nine-period exponential MA (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 a momentum indicator in order to generate entry/exit signals, which are outlined as follows:

1) Buy Entry: histogram value is below -0.4.
2) Buy Exit: histogram value gets bigger than -0.1.
3) Sell Entry: histogram overcomes the value 0.4.
4) Sell Exit: histogram 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.

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, 3.4 GHz x8, 31.3 GB RAM, and GPU GeForce GTX 980 with 4 GB 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 pretrained network, in order to make experimental results independent from that kind of choice.

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 the 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 the 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 one 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

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

TABLE II
ACCURACY IN EXPERIMENT A

1—NO EARLY STOPPING

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

2—WITH EARLY STOPPING

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

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 3000 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 the 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, which 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, which 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, the accuracy is high in the case of the crossover since the network has to learn that position depends only on the sign of the input data that is, as we said, the difference between two MAs. Finally, the 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 *et al.* [40]. This is designed to stop the training when the model 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 set the *patience* limit, i.e., the maximum period standing with no improvement, to 2500 epochs, as this gave us the best results among other values we tested. Table II-2 shows the results when early stopping is employed. The accuracy is not affected, but reached within fewer epochs. In particular, crossover is able to further improve 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.

TABLE III
ACCURACY AND TRAINING EPOCHS IN EXPERIMENT B

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

2) Experiment B–Multiple Units: As the 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 the input to the dense layer, which is aimed at providing the decision as the output, using tanh as the 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 100% of the 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 the accuracy along the training period considerably increased). Indeed, by adding a dense layer, the 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 five as this value seems to offer an appropriate tradeoff between memory and risk of overfitting.

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 the following:

1) what is the output range expressed by the activation function used for the output layer;
2) the coding; and
3) the cost function used to train the network.

Table IV-1 reports experimental results when the sigmoid is used as the activation function. The output can be coded (C) as

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

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). The accuracy for crossover is practically comparable to previous results. However, we notice a substantial reduction of the training epochs. For the 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 the 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 that 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 the above-mentioned experiments, 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 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 are 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 five 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
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

Table VI-1 outlines that the 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 Tables VI-2 and 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 the 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.

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 that 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 a 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.

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% ±0.36%	98.95% ±0.91%	5798 ±1267	97.24% ±1.00%	79.48% ±12.33%	4884 ±1953
U	MSE	98.77% ±0.56%	99.07% ±0.79%	5769 ±1133	99.31% ±0.36%	89.07% ±5.52%	5998 ±1776
U	CE	98.96% ±0.37%	98.89% ±1.00%	5254 ±1046	99.66% ±0.22%	92.64% ±4.25%	6563 ±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% ±0.41%	95.96% ±2.14%	2984 ±354	98.35% ±0.66%	78.10% ±8.68%	3507 ±707
U	MSE	99.25% ±0.44%	95.57% ±1.99%	3107 ±531	99.17% ±0.41%	76.79% ±11.00%	2862 ±222
U	CE	99.17% ±0.53%	96.09% ±1.98%	2973 ±357	99.66% ±0.27%	77.44% ±10.52%	2924 ±305

3—CORRELATED INFORMATION WITH PREVIOUS DECISION

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

4—CORRELATED INFORMATION AND UNRELATED INDICATORS WITH PREVIOUS DECISION

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

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 an input, a coded output, and the 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 have been placed 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 toward 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 in 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 in 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 period. The learning process improves when direct information comes together with unrelated information, as this avoids 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

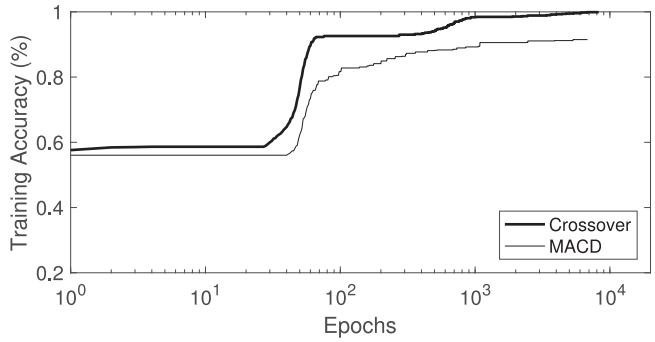

Fig. 6. Accuracy over training epochs for crossover and MACD strategies in Experiment C. The output is coded. MSE is the loss function.

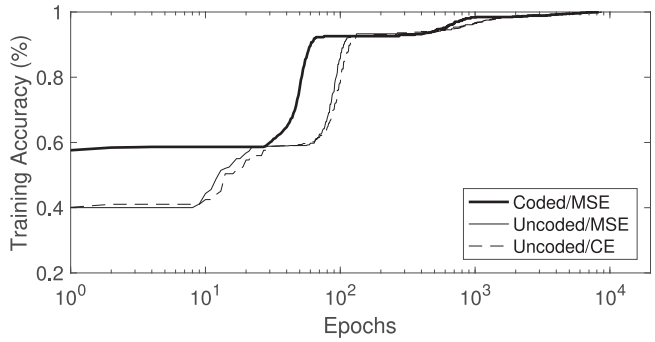

Fig. 7. Accuracy over training epochs for different output representations. Case of crossover in Experiment C.

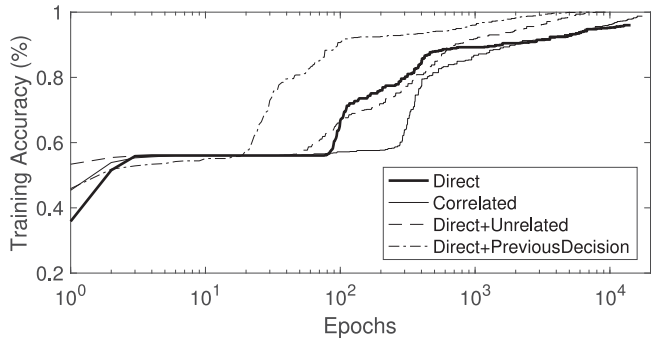

Fig. 8. Accuracy over training epochs for different input spaces. Case of MACD. The output is coded. CE is the loss function.

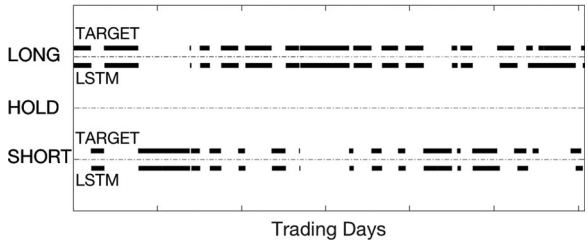

Fig. 9. Strategy decision (TARGET) and LSTM prediction. Case of crossover in Experiment E-3. The output is coded. MSE is the loss function.

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 synchronization of agent and robot decisions, which can differ in few days. Therefore, the two sequences of decision are basically only shifted in time.

V. Conclusion 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 the training shorter, 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 are worth 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 is suggested. 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 an input to the network. Possibly this is 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 of 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 *Proc. IEEE Int. Conf. Big Data*, 2016, 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," *J. Autom. Control Eng.*, vol. 3, no. 6, pp. 452–456, 2015.

[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. Ind. Informat.*, vol. 13, no. 3, pp. 1213–1226, Jun. 2017.

[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. Semicond. Manuf.*, vol. 30, no. 2, pp. 135–142, May 2017.

[5] D. Ravì *et al.*, "Deep learning for health informatics," *IEEE J. Biomed. Health Informat.*, vol. 21, no. 1, pp. 4–21, Jan. 2017.

[6] S. Lee and J. H. Chang, "Oscillometric blood pressure estimation based on deep learning," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 461–472, Apr. 2017.

[7] L. Troiano and P. Kriplani, "A mean-reverting strategy based on fuzzy transform residuals," in *Proc. IEEE Conf. Comput. Intell. Financial Eng. Econ.*, 2012, pp. 1–7.

[8] L. Troiano, "Fuzzy co-transform and its application to time series," in *Proc. 2010 Int. Conf. Soft Comput. Pattern Recognit.*, Dec. 2010, pp. 379–384.
[9] X. Cai, S. Hu, and X. Lin, "Feature extraction using restricted Boltzmann machine for stock price prediction," in *Proc. IEEE Int. Conf. Comput. Sci. Automat. Eng.*, 2012, vol. 3, 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 *Proc. IEEE Int. Conf. Big Data*, 2015, pp. 2823–2824.
[11] L. D. Persio and O. Honchar, "Artificial neural networks approach to the forecast of stock market price movements," *Int. J. Econ. Manage. Sys.*, 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*, vol. 6, no. 3–4, pp. 67–77, Dec, 2017, doi: 10.3233/AF-170176.
[13] J. B. Heaton, N. G. Polson, and J. H. Witte, "Deep learning for finance: Deep portfolios," *Appl. Stochastic Models Bus. Ind.*, vol. 33, no. 1, pp. 3–12, 2017.
[14] A. Arévalo, J. Niño, G. Hernández, and J. Sandoval, "High-frequency trading strategy based on deep neural networks," in *Proc. 12th Int. Conf. Intell. Comput.*, 2016, 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. Pattern Anal. Mach. Intell.*, vol. 39, no. 8, pp. 1617–1632, Aug. 2017.
[16] H. Palangi *et al.*, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 4, pp. 694–707, Apr. 2016.
[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, 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 Comput.*, vol. 12, no. 10, pp. 2451–2471, 2000.
[19] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search Space Odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.
[20] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased LSTM: Accelerating recurrent network training for long or event-based sequences," in *Proc. 30th Conf. Neural Inf. Process. Syst.*, 2016, pp. 3882–3890.
[21] Y. Zhu *et al.*, "What to do next: Modeling user behaviors by time-LSTM," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2016, pp. 3882–3890.
[22] H. Siegelmann and E. Sontag, "On the computational power of neural nets," *J. Comput. Syst. Sci.*, vol. 50, no. 1, pp. 132–150, 1995.
[23] A. Graves, N. Jaitly, and A.-R. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proc. IEEE Workshop Automat. Speech Recognit. Understanding*, 2013, pp. 273–278.
[24] S. Han *et al.*, "ESE: Efficient speech recognition engine with sparse LSTM on FPGA," in *Proc. 2017 ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, 2017, pp. 75–84.
[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 *Proc. IEEE Spoken Lang. Technol. Workshop*, 2014, 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, Lang. Process.*, vol. 23, no. 3, pp. 517–529, Mar. 2015.
[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. 7th ACM Int. Conf. BioInformat., Comput. Biol., Health Informat.*, 2016, 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.
[30] N. Hirose and R. Tajima, "Modeling of rolling friction by recurrent neural network using LSTM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, 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. Comput. Appl.*, vol. 143, no. 11, pp. 7–11, 2016.
[32] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, *An Overview and Comparative Analysis: Recurrent Neural Networks for Short Term Load Forecasting.* New York, NY, USA: Springer, 2017.
[33] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/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.
[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*. New York, NY, USA: Springer, 2001, pp. 134–137.
[36] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 2342–2350.
[37] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 1310–1318.
[38] J. B. Wolfgang Groß, S. Lange, J. Boedecker, and M. Blum, "Predicting time series with space-time convolutional and recurrent neural networks," in *Proc. 25th Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn.*, 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," *COURSERA: Neural Netw. Mach. Learn.*, vol. 4, no. 2, 2012, pp. 26–31.
[40] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approx.*, vol. 26, no. 2, pp. 289–315, 2007.
[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.

Luigi Troiano received the M.Eng. degree in IT engineering from the University of Naples, Naples, Italy, in 2000 and the Ph.D. degree in information technology from the University of Sannio, Benevento, Italy, in 2004.

He is currently Assistant Professor with the University of Sannio, where he lectures on data science, artificial intelligence and machine learning. His research interests focus on application of AI models and algorithms to industrial problems. He is coordinator of Computational and Intelligent Systems Engineering Laboratory (CISELab) with the University of Sannio. He developed an extensive experience in designing, experimenting and validating algorithms, designing, and implementing large software systems in industrial environment.

Elena Mejuto Villa was born in Spain in 1988. She received the M.Eng. degree in telecommunications engineering from the University of Oviedo, Oviedo, Spain, in 2014. She is currently working toward the Ph.D. degree in information technologies for engineering from the University of Sannio, Sannio, Italy She is currently a teaching assistant. Her main areas of research interests include artificial intelligence and signal processing techniques to time-varying data in the fields of finance and gravitational wave detection.

Vincenzo Loia received the B.S. degree in computer science from the University of Salerno, Salerno, Italy, in 1985 and the M.S. and Ph.D. degrees in computer science from the University of Paris VI, Paris, France, in 1987 and 1989, respectively.

Since 1989, he has been a Faculty member with the University of Salerno, where he teaches safe systems, situational awareness, and cognitive cyber defense.

He is currently the Chair and Professor of computer science with the Department of Management and Innovation Systems. He is the Editor-in-Chief of *Ambient Intelligence and Humanized Computing*, and *Evolutionary Intelligence* both from Springer and the Associate Editor for various journals.