Effect of Architecture in Recurrent Neural Network Applied on the Prediction of Stock Price

Zahra Berradi, Mohamed Lazaar, Hicham Omara, Oussama Mahboub

Abstract—**The recurrent neural network is generally utilized in an assortment of areas, such as pattern recognition, natural language processing and computational learning. Time series prediction is one of the most challenging topics for many years due to its application in finance and decision making. This article centers chiefly on the methods and techniques of forecasting future prices of two stocks: IAM.PA and ORA.PA. An experimental investigation is grounded on two years of historical information. Furthermore, the statistics of the stocks and the predictions are made for 22 days in advance. The prediction performance compares three approaches of the recurrent neural network: Elman recurrent neural network in the first stage, Long Short-Term Memory recurrent neural network for the next phase, and Gated Recurrent Unit in the third phase. The article aims to accommodate a comparative analysis among these three models based on mean square error, time per step, memory and the number of hidden nodes required for excellent accuracy.**

Index Terms—**recurrent-neural-networks, forecasting-stock-prices, long-short-term-memory, gated-recurrent-units**

I. Introduction

FORECASTING stock price is a technique used to have a clear vision about the future price based on the previous statistics of the stock. The stock market is considered as a non-linear dynamic system. Though, every investor desires a model to guess the future stock values to help him make appropriate decisions.

Numerous analysis methods have been developed to calculate stock prices. Moreover, various statistical models for forecasting stocks are available to decide the right time to sell or hold. Many factors have influenced the forecasting of the stock market. Among all, the most common technical factors are opening value, high value, low value, closing value, and volume.

In the journey to have a suitable prediction of time series data, the researchers concentrate on two major phases namely: the choice of features and the best models. Some of them make efforts to add new features that affect the stock prices such as using twitter data [1], [2], [3], or reduce the number of technical features (if they are too many) like the work of [4] who used 60 financial and monetary features. Later on, further reducing them to 11. They opted for three-dimensionality decrement methods. These included methods of Principal Component Analysis (PCA), Fuzzy Robust Principal Component Analysis (FRPCA), and Kernel-based Principal Component Analysis (KPCA).

The researchers used many linear and non linear models for the prediction of time series data. Some linear models include the Autoregressive (AR) model and the Autoregressive Integrated Moving Average (ARIMA). Other non-linear models such us Autoregressive Conditional Heteroscedasticity (ARCH), Generalized Autoregressive Conditional Heteroscedasticity (GARCH), artificial neural networks, Support Vector Machine (SVM), Random Forest (RF) as well as Support Vector Regression (SVR).

Some researchers have tried to propose a suitable combination of the existing models like [5] who offers an approach which consists of the fusion of many models involving Support Vector Regression (SVR) and Artificial Neural Network (ANN), Random Forest (RF) and SVR resulting into SVR-ANN, SVR-RF and SVR-SVR models. On the other hand, [6] have proposed an integrated approach based on Genetic Fuzzy Systems (GFS) and Artificial Neural Networks (ANN).

Other researchers [7] have proposed a hybrid methodology, which is a fusion of Empirical Mode Decomposition with the exponential smoothing method. [8] has proposed a model based on the chaotic mapping, firefly algorithm, and support vector regression (SVR). Another approach was developed by [9] for the multilayer perceptron; they used the Genetic Algorithm (GA) to obtain the optimal architect of MLP. Another technique was further worked on and improved by [10], called Probabilistic Self-Organizing Map (PRSOM). They proposed an architecture optimization model that is combined integer nonlinear optimization model under linear constraints, resolved by the genetic algorithm. The application of ANN is also extending to medical field like the work of [11], they proposed an artificial neural network to identify the thermal neutrons flux to treat brain tumors. [12] focused on the theoretical part of ANN, they studied the existent and exponential stability of the periodic solution for fuzzy cellular neural networks with time-varying delays. While [13] adopted Hidden Markov Model (HMM) technique to forecast stock price for some airline companies.

The time for the prediction depends on the need of the forecaster: microseconds, seconds, hours, days, weeks, months, or years (in rare cases). There are numerous recurrent neural networks used by the researchers to predict the prices such us Elman recurrent neural network, long short-term memory and gated recurrent unit. The simplest one is Elman recurrent neural network, whereas the most used ones to characterize long-term memory are the long short-term memory neural network (LSTM) and gated recurrent neural network (GRU). Other architectures, such us Deep Belief Network (DBN), are used in the case of short-term forecasting.

Manuscript received July 28, 2019; revised February 9, 2020.

Z. Berradi is with the National School of Applied Sciences, Abdelmalek Esaadi University, Tetouan, Morocco. e-mail: (berradi.zahra@gmail.com).

M. Lazaar is with the National School of Computer Science and Systems Analysis, Mohammed V University, Rabat, Morocco. e-mail: (mohamed.lazaar@um5.ac.ma).

H. Omara is with the Faculty of Sciences of Tetouan, Abdelmalek Essaadi University, Tetouan, Morocco. e-mail: (hichamomara@gmail.com).

O. Mahboub is with the National School of Applied Sciences, Abdelmalek Esaadi University, Tetouan, Morocco. e-mail: (mahboubousama@gmail.com).

This work aims to compare the most used recurrent neural networks for time series data problem and to select the best ones for forecasting the stock prices. The comparison will be made based on the accuracy of models using the same dataset of IAM.PA stock and with the same number of epochs. The primary purpose is to achieve as much efficiency as possible by choosing the optimal parameters.

The paper is organized into different sections. The first section is dedicated to introduction, while the second section is about related works. In the third section, the models ERNN, LSTM and GRU are explained with their architectures. In the fourth section, the three methods of RNN are compared for forecasting stock prices of IAM.PA then the same optimal parameters are used to forecast the stocks IAM.PA and ORA.PA for 22 days. In the last section, the conclusion is drawn.

II. Related work

Predicting the stock price has been one of the most overwhelming tasks that keep the research in continuous improvement. Many efforts have been made to find the best existing methods in the literature such as Fuzzy system, recurrent neural network, ARIMA, ARMA, Genetic Algorithm, etc. Currently, Neural network is the most used one because it has the ability to adapt to a dynamic system that change continuously such us the stock market.

A considerable amount of literature has been published on the predicting of the stock prices. [14] compared Probabilistic Neural Networks (PNN) and Support Vector Machines (SVM) to predict the stock market using economic and technical Information. The results show that PNN gives the best accuracy with technical indicators while SVM gives the best results using economic features. [15] explored the power of using SVM, Multi-Layer Perceptron (MLP), and regression to predict the price of S&P 500. They developed SVM model with RBF kernel model yielded a good prediction with respect to the regression and ANN models. [16] selected 25 papers satisfying certain criteria. His work assumed that the use of ANN combined with other machine learning techniques yields better results. [17] proved that ARIMA model yields good results for forecasting Gold prices. On the other hand, the prediction of the Silver price using ANN gave better results. [18] compared two different models of neural network; LSTM and DNN. The work proves that LSTM and DNN both give good accuracy whereas LSTM outperform DNN in term of weekly forecasting. [19] surveyed more than 100 related published articles that focus on neural and neuro-fuzzy techniques applied to forecast stock market, they concluded that ANN yields better results with higher accuracy. For short time forecasting, [20] compared five types of recurrent neural networks; their works show that ERNN and Echo State Network (ESN) are more suitable for short time forecasting than LSTM and GRU. [21] suggested LSTM model and emotional analysis model to predict the stock price of Shanghai.

In this paper, we suggest techniques to choose the best parameters used in the Architect of the recurrent neural network. Then, we provide a comparison of three types of recurrent neural network namely ERNN, LSTM, and GRU based on MSE. Finally, we give the prediction of IAM.PA and ORA.PA to forecast the closing price for 22 days.

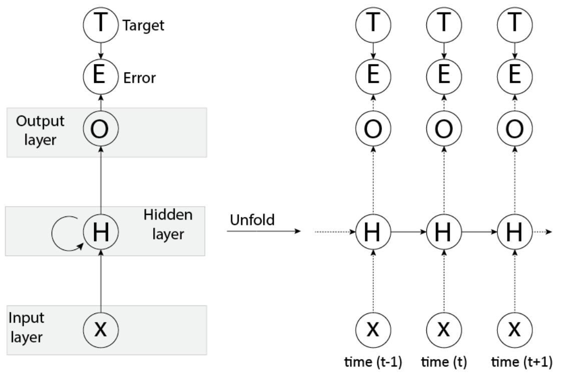

Fig. 1. A recurrent neural network.

III. Neural Network Models

In this section, three architectures of deep learning namely Elman recurrent neural network (ERNN), long short-term memory recurrent neural network (LSTM) and gated recurrent unit (GRU) are described.

A. *Elman recurrent neural network*

The simple recurrent neural network also knows as Vanilla recurrent neural network or Elman recurrent neural network, [22], considered to be the most basic type of the recurrent neural network. It contains an input layer, hidden layer, and output layer. In each layer, there is at least one node, and each node is connected to the previous nodes with edges called weights. Training recurrent neural network consists of two parts: going forward and going backward.

Going forward: on time t; the network processes the input vector, updates its hidden state via an activation function and uses it to predict its output. On time $t + 1$, the hidden layer receives the inputs vector and the outputs of the hidden layer from the last time step, as shown in figure 1.

More formally, given a sequences x_t of N dimension and $t = (1, ..., T)$. The RNN updates its recurrent hidden state by the equation

$$h_t = f(Wx_t + Uh_{t-1} + b) \tag{1}$$

Where f is a nonlinear function such as logistic sigmoid or a hyperbolic tangent. b is bias.W and U are the matrices which represent input and hidden weights respectively. The output O_t is calculated using this equation

$$O_t = g(Vh_t + c) \tag{2}$$

Where g is usually linear transformation, c is bias, and V is the matrix of output weight. In the process of going forward, the model initializes the weights to an identity matrix. After that, it calculates the value of the hidden unit and then calculates the error function (or the loss function) that depends on the error between the estimated output and the target. The loss function that we used in this work is Mean Square Error (MSE) and Mean Absolute Error(MAE):

$$E_{MSE} = \frac{1}{T}\Sigma_{i=1}^{T}|O_t - T_t|^2 \tag{3}$$

$$E_{MAE} = \frac{1}{T}\Sigma_{i=1}^{T}|O_t - T_t| \tag{4}$$

Where O_t and T_t are the estimated output and the actual output respectively.
Going backward: to compute the loss function over the entire training set to perform a single update of the network parameters is very expensive regarding time and memory. A famous approach to solve this problem is by computing the gradient over mini-batches of the training data. This method called Stochastic Gradient Descent (SGD). The update equation is:

$$W_{k+1} = W_k + \eta \nabla L(W_k) \tag{5}$$

Where η is the learning rate. And W_k is the update weights. The convergence of SGD is slow. A good solution for this issue has been addressed by several contributions proposed in the literature for updating the network parameters. In the following, three effective strategies are described in the literature to accelerate the convergence of SGD.
Nestrov momentum: the update equation is

$$W_{k+1} = W_k + \mu W_{k-1} - \eta \nabla (W_k + \mu W_{k-1}) \tag{6}$$

where W_k is the previous update of W_{k+1}, μ is a hyperparameter, a common choice is to set$\mu = 0.9$.
RMSprop: proposed by [23]. The update equation is:

$$W^{(i)}_{k+1} = W^{(i)}_k - \eta V^{(i)}_k$$

Such us

$$V^{(i)}_k = \begin{cases} (1-\delta) W^{(i)}_{k-1} + \delta \nabla L(W^{(i)}_k)^2 & if \quad \nabla L(W^{(i)}_k) > 0 \\ (1-\delta) W^{(i)}_{k-1} & outherwise \end{cases}$$

The value of δ is in between $[0, 1]$ and$\eta = 0.01$.
Adam: This algorithm, for efficient stochastic optimization, was proposed by [24]. The update equation of Adam is:

$$W_{k+1} = W_k + \frac{\eta}{\sqrt{\widehat{v}_k + \varepsilon}} \widehat{m}_k$$

Such us:

$$\widehat{m}_k = \frac{m_k}{1-\beta_1^k} \quad \widehat{v}_k = \frac{v_k}{1-\beta_2^k}$$

Where m_k update biased first-moment estimate such as: $m_k = \beta_1 m_{k-1} + (1-\beta_1)\nabla L(W^{(i)}_k)$ and v_k update biased second raw moment estimate such as:$v_k = \beta_2 v_{k-1} + (1-\beta_2)\nabla L(W^{(i)}_k)^2$ Where the default values of the hyper parameters are $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. The work of [25] mentioned that while training RNNs, the model forgetting the information if the model trained for a long time. To solve this problem, [26] propose an advanced hidden unit that can preserve the memory for a long time.

B. LSTM recurrent neural network

LSTM can be interpreted as a variation or as an extension of ERNN. The LSTM was invented to solve the problem of overfitting of the simple RNN. An LSTM architecture composed of the input layer, the hidden recurrent layer, and the output layer. Each unit in hidden layer have three input (previous hidden value H_{t-1}, previous cell memory value C_{t-1} and actual input variable x_t). Inside the hidden unit there is four gate (input gate I_t, forget gate F_t, new memory gate M_t and output gate O_t). Their role is to control the flow of the information, keep the important information and forget the unnecessary information as shown in figure 2.

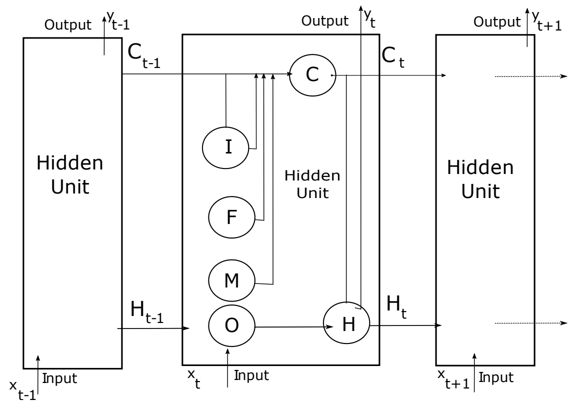

Fig. 2. Long short term memory recurrent neural network.

Unlike the recurrent unit which simply computes a weighted sum of the input signal and applies a nonlinear function, each LSTM hidden unit calculates her hidden value by using all the gates with a specific combination

$$H_t = O_t tanh(C_t) \tag{7}$$

Where O_t is an output gate that modulates the amount of memory content on time t, and C_t is cell memory. The output gate is computed by

$$O_t = \sigma(W_o x_t + U_o H_{t-1}) \tag{8}$$

Where σ is a logistic sigmoid function that define by $\sigma(x) = \frac{1}{1+\exp(-x)}$. W_o and U_o is the weight matrix. The memory cell C_t is updated by partially forgetting the existing memory and adding a new memory content M_t

$$C_t = F_t C_{t-1} + I_t M_t \tag{9}$$

Where the new memory content is

$$M_t = tanh(W_m x_t + U_m H_{t-1}) \tag{10}$$

The extent to which the existing memory is forgotten is modulated by a forget gateF_t.

$$F_t = \sigma(W_f x_t + U_f H_{t-1}) \tag{11}$$

The degree to which the new memory content is added to the memory cell is modulated by an input gate I_t:

$$I_t = \sigma(W_i x_t + U_i H_{t-1}) \tag{12}$$

C. Gated recurrent neural network

A gated recurrent unit (GRU) was proposed by [27] and there is another variation of GRU proposed by [28];[29];[30]. The simple RNN calculates the next hidden layer directly. Similarly to the LSTM unit, the GRU has gating units that modulates the flow of information inside the unit as shown in figure 3.

In the first step, each unit calculates the update gate z_t based on the current state and the previous hidden states:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{13}$$

Where σ is a sigmoid function. In the second step, it calculates the reset gate r_t but with different weight

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{14}$$

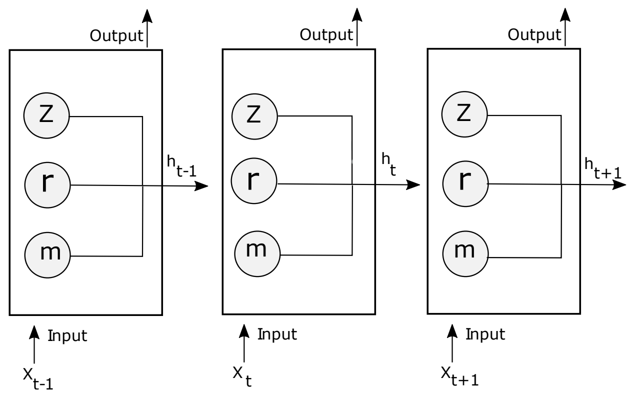

Fig. 3. Gated recurrent unit.

In the third step, it calculate the new memory content m_t

$$m_t = tanh(Wx_t + r_t \circ Uh_{t-1})$$

And finally, it calculate the next hidden state h_t

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ m_t \tag{15}$$

Where $\circ$ is an element wise multiplication.

IV. Results and discussion

The dataset was collected over two years, from 20^{th} February 2017 to 20^{th} February 2019, in order to predict the closing price of the following month: 21^{st} February 2019 to 21^{st} March 2019. The data used in this article are taken from the website ”www.finance.yahoo.com”. The different values as a result of the different features can reduce the effectiveness of training procedures, however, this problem was resolved by data normalization. The data contained several missing values that required preprocessing; subsequently, these missing values are replaced by the average value. After the preprocessing, the dataset of stock, IAM.PA, was used as a tool to choose the best optimizer and the number of hidden nodes. Then, these parameters were employed in the architect of the neural network. After that, a comparison was made among ERNN, LSTM and GRU, based on the MSE to forecast the closing price of IAM.PA and ORA.PA for 22 days. Table I demonstrates that the number of features used in this experiment are 6, based on the work of [31]. 20% of IAM.PA dataset was used as the testing sample and 80% was used as the training sample, as shown in figure 4.

TABLE I
THE 6 FEATURES USED IN THE FORECASTING

Features	**The source**
Opening price	from the data
lowest price	from the data
high price	from the data
volume	from the data
SMA20$_t$	$SMA20 = \frac{p_{t-1}+\dots+p_{t-20}}{20}$ where p_t is the closing price
EMA20$_t$	$(p_{t-1} \times 0.3) + EMA20_{t-1} \times 0.7$

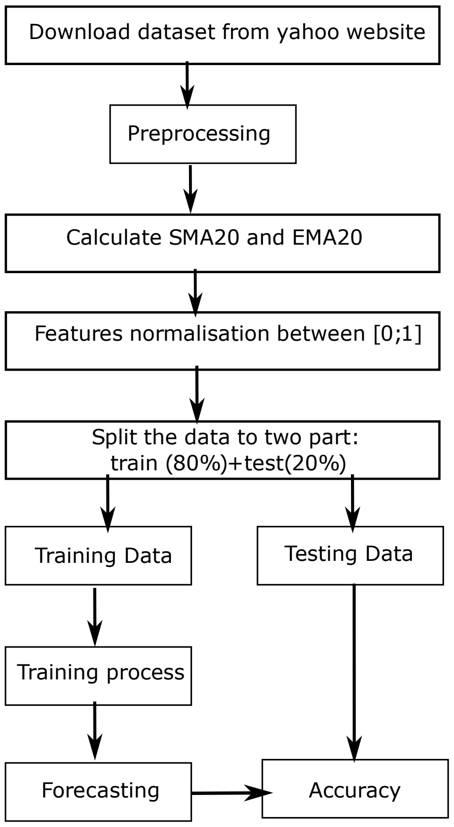
Download dataset from yahoo website
Preprocessing
Calculate SMA20 and EMA20
Features normalisation between [0;1]
Split the data to two part: train (80%)+test(20%)
Training Data
Testing Data
Training process
Forecasting
Accuracy

Fig. 4. Flowchart used in this work

The architect of all Recurrent Neural Network was formed by six input nodes in the input layer, 18 hidden nodes in the hidden layer, and one output node in the output layer. The model ERNN uses the activation function *tanh*. The number of epochs employed were 100. The output layer is a fully connected layer with one node and one output via a linear activation function. The models LSTM and GRU use *tanh* as the activation function, and *sigmoid* as the hidden activation function. To find the best parameters that can be used in the previous architects, the training dataset of IAM.PA is employed to select the most fitting optimizer and the optimal number of nodes. The optimizers used in this simulation are Adam, RMSprop, and Nestrov momentum.

TABLE II
MEAN SQUARE ERROR USING THREE DIFFERENT OPTIMIZERS FOR THE MODELS ERNN ,LSTM AND GRU.

	ERNN	LSTM	GRU
Nestrov momentum(Train data)	0.015699	0.036332	0.026128
Nestrov momentum(Test data)	0.009602	0.028055	0.035105
RMSprop (train data)	0.019797	0.020871	0.019162
RMSprop (Test data)	0.014243	0.016683	0.014584
Adam (train data)	**0.013381**	**0.016036**	**0.014999**
Adam (test data)	**0.008851**	**0.015142**	**0.011339**

Table II shows the Mean square Error (MSE) obtained

TABLE III
TIME/STEP USING THREE DIFFERENT OPTIMIZERS FOR THE MODELS ERNN, LSTM, AND GRU.

	ERNN	LSTM	GRU
Nestrov momentum(Train)	236us/step	**274**us/step	256us/step
Nestrov momentum(Test)	291us/step	**331**us/step	425us/step
RMSprop (train data)	238us/step	297us/step	**242**us/step
RMSprop (Test data)	307us/step	286us/step	**279**us/step
Adam (train data)	**208**us/step	299us/step	243us/step
Adam (test data)	**270**us/step	290us/step	384us/step

by the three optimizers for the models ERNN, LSTM, and GRU. There is a significant difference between the MSE obtained by the training data using Adam optimizer (0.013381 for ERNN, 0.016036 for LSTM and 0.014999 for GRU) and the MSE obtained by the other optimizers (RMSprop and Nestrov momentum). Based on this result, the chosen optimizer for RNN architect is Adam optimizer.

Table III shows the execution time for each step using the three models. The execution time using the training data of ERNN with the optimizer Adam is much better (208 us/step) compared to RMSprop (238 us/step) and Nestrov momentum (236 us/step). For the LSTM, the execution time is better using the Nestrov momentum optimizer (274 us/step) compared to RMSprop (297 us/step) and Adam(299 us/step). Finally, for the GRU, the optimizers RMSprop and Adam almost have the same time execution (242 us/step versus 243 us/step), while for Nestrov momentum, it's (256 us/step). The major finding from table II and table III was that the model ERNN with Adam optimizer has less MSE with a suitable execution time compared to other models.

After selecting the best optimizer, it was necessary to know the optimal number of hidden nodes in the hidden layer. The number of nodes in the hidden layer was set in the previous experiment to 18 nodes. So, to find the ideal number of hidden nodes, the MSE error was calculated from using one node to 100 nodes in the model ERNN with Adam optimizer.

As shown in table IV, the memory is increased by increasing the number of hidden nodes, whereas the MSE of the data shows fluctuations. These findings suggest that, generally there are no specific number of nodes convenient for all the models; only the experiment can help to choose the right number of hidden nodes. As we can see from table IV, the optimal number of hidden nodes are 20 because it have the smallest MSE (0.00885), an average execution time (270 us/step) with average memory. Returning to the targets posed at the beginning of this paper, we are able to say that the optimal architect of the RNN will be one input layer with six nodes, one hidden layer with 20 nodes, one output layer with one node, and with the Adam optimizer. These parameters will be used in RNN architect to predict the price IAM.PA and ORA.PA.

As shown in table V, the MSE of the stock ORA.PA is quite the same for all the models. Whereas the MSE of IAM.PA is the same for ERNN and LSTM but less effecient using GRU campared to other models. From data in figure 5 and figure 6, it is apparent that the prediction is significantly good for ORA.PA and IAM.PA for 22 days using the RNN models.

TABLE IV
MEAN SQUARE ERROR OF THE TEST DATA WITH DIFFERENT NUMBER OF HIDDEN NODES FOR ERNN MODEL.

	MSE	Time/step	Memory
1	0.009803	266 us/step	39
2	0.00811	262us/step	40
3	0.014136	260us/step	40
4	0.014118	257us/step	40
5	0.008434	332us/step	40
6	0.011303	267us/step	41
7	0.008969	291us/step	41
8	0.011881	272us/step	41
9	0.012049	321us/step	42
10	0.0118	291us/step	42
20	**0.008851**	**270us/step**	**42**
30	0.009449	375us/step	43
40	0.008294	618us/step	43
50	0.009227	305us/step	43
60	0.009543	298us/step	44
70	0.012283	340us/step	44
80	0.008369	359us/step	44
90	0.006329	334us/step	44
100	0.00933	404us/step	45

TABLE V
MEAN SQUARE ERROR OF THE THREE MODELS FOR THE IAM.PA AND ORA.PA STOCKS

	ERNN	LSTM	GRU
IAM.PA (train)	0.011474	0.014561	0.017307
IAM.PA (test)	0.015737	0.017323	0.022942
ORA.PA (train)	0.015919	0.016838	0.012966
ORA.PA (test)	0.008535	0.008632	0.006723

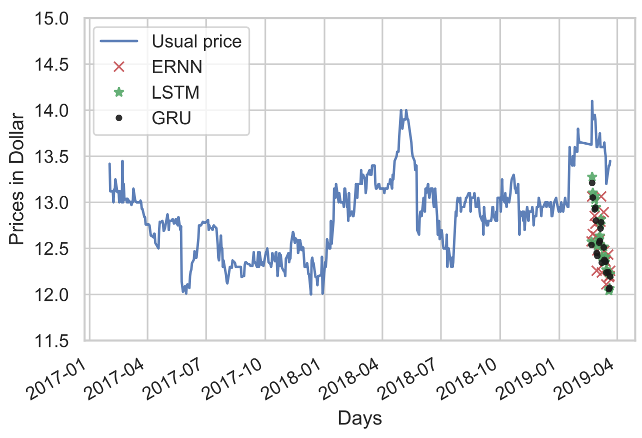

Fig. 5. The prediction of IAM.PA using the models ERNN, LSTM, and GRU for 22 days.

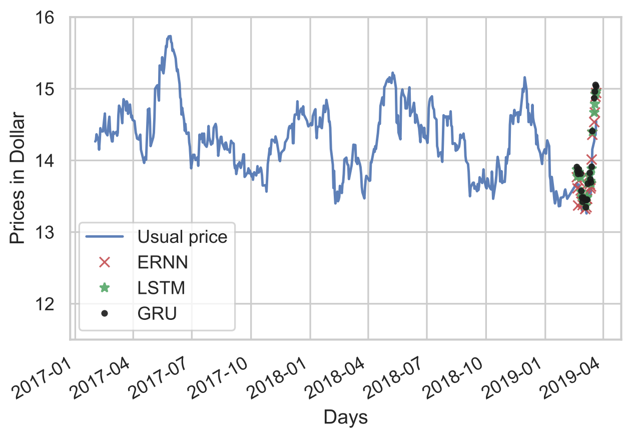

Fig. 6. The prediction of ORA.PA using the models ERNN, LSTM, and GRU for 22 days.

V. Conclusion

The aim of the paper is to predict the stock prices of IAM.PA and ORA.PA from Paris stock exchange (Euronext). The stock prices of IAM.PA were used to determine the optimal parameters to select the best optimizer and optimal number of hidden nodes.

Three models were compared: ERNN, LSTM and GRU. The predictions of IAM.PA and ORA.PA are quite impressive because the MSE is too small for all the models, especially ERNN. The RNN method is one of the most practical ways to forecast the stock price due to its high accuracy.

References

[1] J. Bollen, H. Mao, and X. Zeng, “Twitter mood predicts the stock market,” *Journal of computational science*, vol. 2, no. 1, pp. 1–8, 2011.

[2] N. Oliveira, P. Cortez, and N. Areal, “The impact of microblogging data for stock market prediction: using twitter to predict returns, volatility, trading volume and survey sentiment indices,” *Expert Systems with Applications*, vol. 73, pp. 125–144, 2017.

[3] M. Skuza and A. Romanowski, “Sentiment analysis of twitter data within big data distributed environment for stock prediction,” pp. 1349–1354, 2015.

[4] X. Zhong and D. Enke, “Forecasting daily stock market return using dimensionality reduction,” *Expert Systems with Applications*, vol. 67, pp. 126–139, 2017.

[5] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, “Predicting stock market index using fusion of machine learning techniques,” *Expert Systems with Applications*, vol. 42, no. 4, pp. 2162–2172, 2015.

[6] E. Hadavandi, H. Shavandi, and A. Ghanbari, “Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting,” *Knowledge-Based Systems*, vol. 23, no. 8, pp. 800–808, 2010.

[7] M. Ismail and A. M. Awajan, “A new hybrid approach emd-exp for short-term forecasting of daily stock market time series data,” *Electronic Journal of Applied Statistical Analysis*, vol. 10, no. 2, pp. 307–327, 2017.

[8] A. Kazem, E. Sharifi, F. K. Hussain, M. Saberi, and O. K. Hussain, “Support vector regression with chaos-based firefly algorithm for stock market price forecasting,” *Applied soft computing*, vol. 13, no. 2, pp. 947–958, 2013.

[9] M. Ettaouil, M. Lazaar, and Y. Ghanou, “Architecture optimization model for the multilayer perceptron and clustering.” *Journal of Theoretical & Applied Information Technology*, vol. 47, no. 1, pp. 64–72, 2013.

[10] Z. En-Naimani, M. Lazaar, and M. Ettaouil, “Architecture optimization model for the probabilistic self-organizing maps and speech compression,” *International Journal of Computational Intelligence and Applications*, vol. 15, no. 02, p. 1650007, 2016.

[11] J. E. Rivero, R. M. Valdovinos, E. Herrera, H. A. Montes-Venegas, and R. Alejo, “Thermal neutron classification in the hohlraum using artificial neural networks,” *Engineering Letters*, vol. 23, no. 2, pp. 87–91, 2015.

[12] J. Liu, Q. Zhang, and Z. Luo, “Dynamical analysis of fuzzy cellular neural networks with periodic coefficients and time-varying delays,” *IAENG International Journal of Applied Mathematics*, vol. 46, no. 3, pp. 298–304, 2016.

[13] M. R. Hassan and B. Nath, “Stock market forecasting using hidden markov model: a new approach,” pp. 192–196, 2005.

[14] S. Lahmiri, “A comparison of pnn and svm for stock market trend prediction using economic and technical information,” *International Journal of Computer Applications*, vol. 29, no. 3, pp. 24–30, 2011.

[15] A. F. Sheta, S. E. M. Ahmed, and H. Faris, “A comparison between regression, artificial neural networks and support vector machines for predicting stock market index,” *Soft Computing*, vol. 7, no. 8, pp. 55–63, 2015.

[16] O. Ican, T. B. Celik *et al.*, “Stock market prediction performance of neural networks: A literature review,” *International Journal of Economics and Finance*, vol. 9, no. 11, pp. 100–108, 2017.

[17] S. Kumar, “Prediction of gold and silver prices in an emerging economy: Comparative analysis of linear, nonlinear, hybrid, and ensemble models,” *The Journal of Prediction Markets*, vol. 12, no. 3, pp. 63–78, 2019.

[18] D. Shah, W. Campbell, and F. H. Zulkernine, “A comparative study of lstm and dnn for stock market forecasting,” pp. 4148–4155, 2018.

[19] G. S. Atsalakis and K. P. Valavanis, “Surveying stock market forecasting techniques–part ii: Soft computing methods,” *Expert Systems with Applications*, vol. 36, no. 3, pp. 5932–5941, 2009.

[20] 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.

[21] Q. Zhuge, L. Xu, and G. Zhang, “Lstm neural network with emotional analysis for prediction of stock price.” *Engineering Letters*, vol. 25, no. 2, pp. 167–175, 2017.

[22] J. L. Elman, “Language as a dynamical system,” *Mind as motion: Explorations in the dynamics of cognition*, pp. 195–225, 1995.

[23] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[24] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[25] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

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

[27] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[28] G. Chrupała, A. Kádár, and A. Alishahi, “Learning language through pictures,” *arXiv preprint arXiv:1506.03694*, 2015.

[29] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

[30] W. Jozefowicz, R. Zaremba and I. Sutskever, “An empirical exploration of recurrent network architectures,” *International Conference on Machine Learning*, pp. 2342–2350, 2015.

[31] Z. Berradi and M. Lazaar, “Integration of principal component analysis and recurrent neural network to forecast the stock price of casablanca stock exchange,” *Procedia computer science*, vol. 148, pp. 55–61, 2019.

Z. Berradi was born in Ouazzane, Morocco, in 1985. She received Master degree in Applied mathematics and computer science from Faculty of sciences and techniques in Tangier, Morocco, in 2013. Now she is a PHD student at the National school of applied sciences, Tetouan, Morocco. Her current research interest is modelisation, recurrent neural network, deep learnig and stock market movement.