Deep learning stacking for financial time series forecasting: an analysis with synthetic and real-world time series

Eder F. Urbinate[1], **Leonardo K. Felizardo**[1], **Emilio Del-Moral-Hernandez**[1]

[1]Polytechnic School, University of São Paulo, São Paulo, Brazil

{eder.urbinate,leonardo.felizardo,emilio.delmoral}@usp.br

Abstract. *The forecasting problem is one of the main applications arising from the synergy between finance and artificial intelligence. With the advancement in the field of deep learning, some ANN achieved very satisfactory results and gained more attention. One approach to increase the time series forecasting model's performance is ensemble models, combining each model's prediction (stacking). However, there are some difficulties in combining and evaluating these models for a good performance in financial time series. We use synthetic and real-world time series to evaluate the model stacking, trying to understand the main financial time series components. Using this ensemble method, we reduced the prediction error for both scenarios.*

1. Introduction

Time series forecasting is a research subject that has been around for a long time with various application topics such as life sciences, medicine, business decision-making, finance, climate modeling and physics, to name a few. Time series models are constantly developing and, more recently, have incorporated machine learning as a crucial solution tool in many situations. The first approaches used mathematical and statistical models such as regression, auto-regressive, exponential smoothing, and even structural time series models. All these models depend on the underlying function to explain the relationship between the responses and explanatory variables. At the same time, modern machine learning methods are generally entirely data-driven.

Artificial neural networks (ANN) arise from biology, and from the first proposed models, a great diversity of derivations of it emerged. For example, long short-term memory (LSTM) is a type of ANN that can capture long and short-time dependencies, and it was introduced in 1997 by [Hochreiter and Schmidhuber 1997] with many applications that employ this architecture, including stock price prediction [Selvin et al. 2017]. The research community improved and created new ANN architectures based on these architectures and ingenious techniques. Architectures based on LSTM and convolutional neural networks (CNN) are now being employed on multivariate and univariate time series forecasting [Sezer et al. 2020]. Although CNN approaches are usually associated with image processing, recent works have shown that the CNN architectures also have competitive results for the time series forecasting and time series classification domain [Ismail Fawaz et al. 2019].

We can extract better performance by employing different architectures of artificial neural networks. Some training techniques combine different architectures based on the performance of the training data set. When performing these combinations, we need to consider what was obtained through the training data and what will be reflected in

the test data. As the finance series has a high noise level, their analysis is quite uncertain and usually requires testing on several different data sets. To better understand the factors affecting different architectures' performance and verify the generalization capacity of neural networks in different situations, we used synthetic series generated by different mathematical processes. We show that some synthetic series generation processes better reflect the behavior obtained in real-world series. Furthermore, combining different networks outperforms individually trained architectures for most real-world activities. We summarize the contributions of this work in the:

1. We propose an ensemble of state-of-the-art deep learning approaches to predict financial time series outperforming all the individual methods.
2. We compare the results obtained in the real-world time series with the synthetically generated ones, highlighting possible characteristics or components of the real-world data that the synthetic time series can simulate.

2. Brief review of the time series modeling using DL

With the appearance of the LSTM method [Hochreiter and Schmidhuber 1997] research community shifted its attention to testing it. Using a combination of the decoupled extended Kalman filter learning and LSTM, [Pérez-Ortiz et al. 2002] presents impressive results considering LSTM limitations on learning over a long sequence of symbols. [Cheng et al. 2006] presents favorable positive results for the vanilla recurrent neural network (RNN), which outperformed the multilayer perceptron (MLP) on a multi-step-ahead prediction problem using multiple data sets for comparison. Throughout the years, LSTM and other RNN architectures remained a baseline for applications in time series predictions. For instance, we can observe in [Sezer et al. 2020] how RNN dominated almost all of the period of 2005 to 2019 for time series forecasting in financial markets. More recently, we have observed convolutional methods in the time series forecasting for finance.

CNN emerges as an alternative solution in the deep learning (DL) domain for time series forecasting/classification employed on various applications [Ismail Fawaz et al. 2019], [Sezer et al. 2020]. One of the first applications of CNN in the time series classification was in the work of [Zheng et al. 2014], outperforming the MLP. Other applications merged time series modeling and CNN, which was the case of [Yang et al. 2015] that applied the CNN architecture to recognize human activity from body-worn inertial sensors. [Di Persio and Honchar 2016], compared the CNN method against the MLP and RNN methods and surpassed them when analyzing the root mean square error (RMSE) accuracy metric. CNN was used to predict future stock prices trend (up or down), but using the order book as input. When analyzing the precision and recall metrics, CNN architecture overcomes the compared methods, MLP, and support vector machine (SVM). More recently, [Wang et al. 2019] employed CNN for the multivariate problem compared with a set of usual techniques such as auto-regressive, MLP, LSTM, and other machine learning methods was presented a different approach for multivariate time series forecasting applying CNN surpassing the contender algorithms. As we perceive in [Ismail Fawaz et al. 2019] work, we see more techniques using CNN for time series models, such as residual network (RESNET) and fully convolutional networks approaches.

The idea of an ensemble model combining ANN is not so recent, as can be seen in [Dietterich 2000] and [Jin et al. 2004]. These models combine different architectures of neural networks and treat them as large neural networks. We can see some applications related to time series in epilepsy detection [Akyol 2020] and applications in forecasting [Maqsood et al. 2004]. Here we propose an ensemble based on best time series forecasting deep neural network architectures with RESNET, CNN, and LSTM. The next chapter will give more details about how this ensemble was assembled.

3. Methods

We divide this section into three topics: A summary of the models employed for time series forecasting, a description of the synthetic time series generation techniques, and present metrics for evaluation.

3.1. Predictor models

3.1.1. LSTM, CNN and RESNET

Long Short-Term Memory (LSTM) architecture is a variant of the recurrent networks that can capture the series's long and short-term time dependencies. LSTM has an internal gate mechanism regulating the information flow carrying the relevant learning through the sequence processing. This architecture is commonly used in time series forecasting, obtaining state-of-the-art results in many cases [Hu et al. 2021]. Some application examples are stock price prediction [Selvin et al. 2017], acoustic modeling, speech recognition, and weather forecasting.

Convolutional Neural Networks (CNN) have convolutional kernels acting like each neuron's receptive field. CNN is inspired by the optical nervous system and was initially designed for two-dimensional shape recognition. The architecture CNN gained greater visibility in 2012 due to competitions such as ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Krizhevsky et al. 2012], considerably surpassing previous years' results. Recent works have extended the use of CNN for time series, natural language processing, and financial time series [Hu et al. 2021] [Lim and Zohren 2021]. Here we use CNN adapted for 1D time series applications.

The Residual Network (RESNET) [He et al. 2016] is an ANN architecture proposed to solve image classification problems. One of the main benefits is that the RESNET can deal better with the vanishing or exploding gradients that very deep nets may face. The mechanism of the RESNET relies on the approximation of the residual maps $\mathcal{F}(x)$, employing a block of stacked convolutions layers and using a short skip connection to feed the block output. More recently, works have been employing the RESNET architecture with slight modifications to solve time series problems such as prediction and classification [Ismail Fawaz et al. 2019].

3.1.2. Ensemble modeling

Ensemble modeling is a way of stacking neural networks into a package to work as a single predictor. Here we have a combination that uses the values generated by each predictor in the set. Besides this adopted method, there are also other ensemble

approaches. Figure 1 shows how we ensemble the three different neural networks. The stacking models give the predictor instance the loss value from the training and errors for the testing series. Once we select the model by the smallest loss in the training set, we return the error for the model picked. The input dataset is divided into training, validation, and testing, and then it is inserted into the ensemble with RESNET, CNN, and LSTM, where the learning and prediction of each model are processed. Then, the predictor selection is made based on the neural network that generated the smallest training loss, this network is selected, and its error values are selected as the error of the ensemble model. The tests performed in this work present how prediction capabilities can be improved by stacking networks.

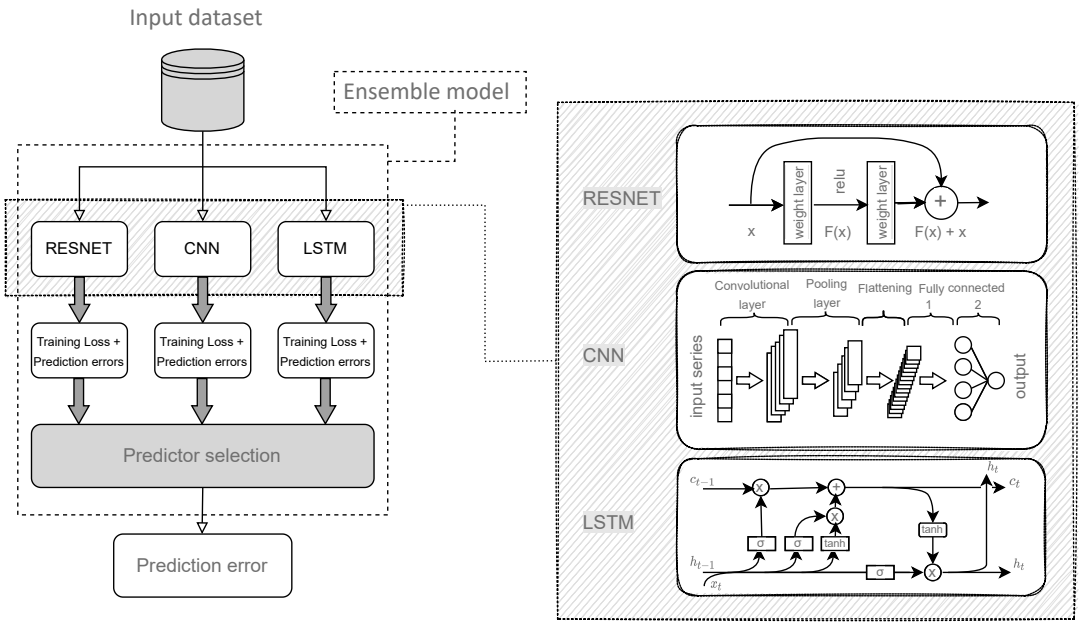

Figure 1. Left: Ensemble model scheme. Right: RESNET, CNN, and LSTM network architectures ensemble components.

3.1.3. Auto-regressive integrated moving average

In addition to ANN, the research community applies classical methods, such as auto-regressive integrated moving average (ARIMA), as a baseline. ARIMA relies on the fundamental principle that the future values of a time series came from a linear function of past observations and white noise terms. The ARIMA model pioneered by Box and Jenkins is the most popular and effective statistical model for time series forecasting.

3.2. Synthetic time series generation and real-world data

3.2.1. ARMA, harmonic and Gaussian noise

Although financial time series generally do not present simple auto-correlations, we employed an auto-regressive moving average (ARMA) as a less complicated problem.

We assume that if the model can not predict rightly in this simple scenario, the model will also be unsatisfactory in the real-world scenario. It is possible to generate synthetic time series from simulated stochastic processes using many functions. Our first choice to simulate a stochastic process is employing auto-correlated moving average series with second-order auto-correlation.

We combined harmonic functions (sine) with the ARMA to simulate a long-term dependent function. With this combination, we can evaluate the ability of predictive models to certain seasonal behaviors of the series, such as the seasonal behaviors we observe in real-world actions. Finally, we included Gaussian noise to simulate normally distributed interference on the time series. We chose an online library with some implemented functions to generate univariate time series (https://github.com/TimeSynth).

3.2.2. NARMA synthetic series

Another synthetic series we tested was a non-linear auto-regressive moving average (NARMA) [Waheeb et al. 2019]. We use this synthetic time series based on the assumption that financial time series may have a non-linear component in the auto-correlation. This non-linear component would explain the recent success of non-linear models in predicting financial time series. The following equation briefly formulates how we generated the NARMA series:

$$x(t) = 0.5x(t-1)e(t-1) + e(t) \tag{1}$$

where $e(t)$ is the white noise in the uniform distribution $N(0, 1)$.

3.2.3. FBM as synthetic financial series

In finance, it is common to find approximations of some time series by models that use Brownian motion, and this can be observed, for example, for stock price [Osborne 1959]. It is possible to find in the literature authors who use fractional Brownian motion (FBM) [Dieker and Mandjes 2003] model to simulate financial series either for commodity or stocks [Ibrahim et al. 2021], [Imperial and Segura 2018]. As initially proposed by [Mandelbrot and Van Ness 1968], FBM has parameter H called Hurst parameter, $0 < H < 1$, where $H = 1/2$ is a two-sided Brownian motion, $H = 1$ are straight lines with a normally distributed slope, $H \in (1/2, 1]$ the correlation of two increments over non-interlapping time-intervals is positive, and for $H \in (0, 1/2)$ it is negative:

$$B_H(t) := \frac{1}{\Gamma + 1/2}\left(\int_{-\infty}^{0} [(t-s)^{(H-1/2)} - (-s)^{(H-1/2)}]\, dB(s) + \int_{0}^{t} (t-s)^{(H-1/2)}\, dB(s) \right) \tag{2}$$

Then we moved on to a further exploration experiment with different series constructed by varying values of H and compared with what was obtained in real series. To implement FBM, we use the library in `https://pypi.org/project/fbm/`.

3.2.4. Real-world financial series

We use the logarithmic return for real series instead of the value itself. This transformation prevents the predictive model from overfitting the value at $t - 1$ of the series, which can quickly occur in applications like this. We also used different real financial time series from different types and sectors to execute our experimentation with the best models selected from the synthetic time series tests. We ran the models for 30 financial products of different types and sectors, such as Stocks (Information Technology, Financials, Consumer Discretionary, Communication Services, Energy, Industrials, Health Care), Funds, Crypto Currency and Index.

3.3. Performance metrics and evaluation

To evaluate the performance among the models, we choose two distinct metrics. These methods' accuracy measures are largely used in the literature. The root mean square error (RMSE) is the first chosen and most used. Moreover, we suggest using the mean absolute scaled error (MASE) to better compare results with different datasets and not be affected by the data scale.

4. Experiments

4.1. Experiment parameterization and data

We employed a computer with an Intel Core i7-7500U CPU and NVIDIA GeForce 940MX GPU. We implemented the models using Keras and Tensorflow as the main libraries of ANN to execute the tests. Additional simulations were also executed using the Google Colaboratory (`https://colab.research.google.com/`).

4.1.1. Architectures setup

We employed different model configurations inspired by the literature and our empirical tests in our experiments. We use the Auto-ARIMA, a module to automatically search the hyper-parameters (p, q, d) for the ARIMA model. The LSTM parameterization employed was extracted from [Sagheer and Kotb 2019] that has extensively tested recurrent architectures for the time series forecasting problem of petroleum production. The employed LSTM architecture has two layers, with 128 for the entrance and 64 cells for the last LSTM layer, followed by a dense output layer with size M. For the CNN architecture, we applied the hyper-parameters of the [Mehtab et al. 2020] that also tested different CNN configurations for the stock prices prediction, and we chose the winner configuration. Finally, we employed RESNET, using an architecture inspired by the [Ismail Fawaz et al. 2019], using one-dimensional convolutions and an LSTM layer at the end.

4.1.2. Data preparation

We simulated a time series with ten thousand time steps for the tests with the synthetic dataset. This size is compatible with financial time series data when dealing with daily data. We employed 30 different series of different sizes for the real-world time

series, ranging from 2862 to 17112, with most being intermediate values. These series were taken from Kaggle databases (`https://www.kaggle.com/`), specifically: Cryptocurrency Historical Prices, Huge Stock Market Datasets, and Financial time series datasets. Each time series was separated into three data sets in our experiments, 70% for training, 20% for testing, and 10% for validation. After this separation, each training round was performed with an observation window of N (32, 64, 128, and 256). Next, we perform an empirical analysis employing the value of N = 256 for the test series, as it presented the best results. Windows larger than 256 would bring limitations to real series that could be chosen due to the structure of the experiments and the division between training, testing, and validation. Only more extensive series were selected for our experiments. The prediction step M means that a single value M steps ahead of the observation window is evaluated. Thus, our experiments displayed N and M values: the observation window and the desired future time step.

4.2. Empirical results for real-world financial time series

First, we compare the most employed deep learning techniques in financial time series forecasting, LSTM and CNN. As the M increases, we observe that the error of the predictions equalizes. As we can observe in the table 1, especially for lower values of M, CNN had lower RMSE and MASE values than the LSTM approach. Interestingly, CNN was better regarding the error for most assets, being better than LSTM for 57% of the assets. On the other hand, LSTM had a lower overall average. We choose the prediction window of size $M = 10$ to extend our analysis by comparing four techniques: Auto-ARIMA, RESNET, LSTM, CNN, and the ensemble method of the three ANN architectures. We employ the ensemble technique described in Section 3 to improve the results baseline. The table 2 presents all the RMSE and MASE errors for each asset and each model, including the ensemble. From this result, we already observe that for most of the assets, the ensemble model error was lower than LSTM and CNN. In the results, compared between CNN and LSTM in the financial series, it is possible to see that LSTM has a smaller average error when considering all assets. CNN manages to do better most of the time, so this behavior can be, in a way, captured by the ensemble that makes the proposed method perform very close or better in the average values and present the minor error for individual assets more times than the other models. Considering the RMSE, we calculate the average error among the CNN, LSTM, and ensemble assets with similar performance. The Auto-ARIMA had the second-best performance regarding the number of assets, with the lowest error for MASE and RMSE metrics. However, considering the MASE metric, the ensemble performed close to CNN, which had the best result. To better understand the characteristics causing this behavior, we extend our analysis using the ensemble technique in synthetically generated time series.

4.3. Empirical results for synthetic time series

We evaluated each model by training and testing on synthetic time series constructed using auto-correlation, Gaussian noise, and harmonic functions composition (GN+H), the non-linear time series NARMA and FBM. Table 3 presents the result for all the mentioned predictive models for both the RMSE and the MASE. In this table, it is possible to notice the best results obtained by the ensemble model. When analyzing the NARMA series, the ensemble outperforms the other techniques for both metrics, RMSE and MASE. Considering linear auto-correlated time series with Gaussian noise and

Table 1. Results for 30 financial series. Column LSTM and CNN presents MASE error for each model. Column dif presents the difference between models, i.e $dif = MASE_{CNN} - MASE_{LSTM}$, negative values mean that CNN has a minor error. The Asset column represents the asset ticker in the U.S. market. CNN is better in 57% of cases, LSTM in 37% and tied in 6%. Times-best shows how many times the model has the minor error for each series.

		M = 1			M = 2			M = 5			M = 10			M = 15			M = 20		
Sector	**Asset**	**LSTM**	**CNN**	**dif**	**LSTM**	**CNN**	**dif**	**LSTM**	**CNN**	**dif**	**LSTM**	**CNN**	**dif**	**LSTM**	**CNN**	**dif**	**LSTM**	**CNN**	**dif**
Inf. Tech.	AAPL	**0.622**	0.700	0.078	**0.628**	0.917	0.289	**0.627**	0.750	0.123	**0.624**	0.770	0.146	**0.622**	0.803	0.181	0.616	**0.613**	-0.003
Financials	AIG	0.674	**0.646**	-0.028	0.656	**0.650**	-0.006	0.662	**0.659**	-0.003	0.654	**0.645**	-0.009	0.652	**0.645**	-0.007	0.654	**0.645**	-0.009
Financials	ALL	0.492	**0.461**	-0.031	0.492	**0.463**	-0.029	0.478	**0.462**	-0.016	0.464	**0.455**	-0.009	**0.457**	0.469	0.012	0.480	**0.451**	-0.029
Cons. Disc.	AMZN	**0.420**	0.426	0.006	0.419	**0.418**	-0.001	**0.407**	0.410	0.003	0.413	**0.408**	-0.005	0.438	**0.403**	-0.035	0.409	**0.397**	-0.012
Financials	AXP	**0.634**	**0.634**	0.000	0.635	**0.633**	-0.002	**0.632**	**0.632**	0.000	0.638	**0.632**	-0.006	0.644	**0.637**	-0.007	**0.640**	0.643	0.003
Cripto Curr.	Bitcoin	**1.088**	1.093	0.005	**1.083**	1.099	0.016	1.012	**1.001**	-0.011	**0.916**	0.926	0.010	1.098	**1.086**	-0.012	1.024	**1.017**	-0.007
Funds	HO1	**0.732**	0.739	0.007	**0.733**	0.743	0.010	**0.749**	0.760	0.011	**0.777**	0.782	0.005	**0.773**	0.781	0.008	**0.767**	0.772	0.005
Funds	NG1	0.618	**0.613**	-0.005	0.622	**0.621**	-0.001	**0.612**	0.613	0.001	**0.602**	0.603	0.001	**0.591**	0.591	0.000	**0.597**	0.598	0.001
Com. Serv.	DIS	**0.626**	0.627	0.001	0.629	**0.627**	-0.002	0.631	**0.625**	-0.006	0.627	**0.625**	-0.002	0.630	**0.624**	-0.006	0.630	**0.625**	-0.005
Inf. Tech.	GOOGL	**0.802**	0.805	0.003	0.810	**0.807**	-0.003	**0.818**	0.820	0.002	**0.849**	0.852	0.003	0.820	**0.811**	-0.009	**0.906**	0.910	0.004
Energy	HES	0.611	**0.607**	-0.004	0.605	**0.586**	-0.019	0.592	**0.565**	-0.027	0.576	**0.560**	-0.016	0.492	**0.474**	-0.018	0.494	**0.474**	-0.020
Industrials	MMM	0.635	**0.620**	-0.015	0.622	**0.620**	-0.002	0.622	**0.621**	-0.001	0.622	**0.620**	-0.002	**0.613**	0.620	0.007	0.613	**0.610**	-0.003
Inf. Tech.	MSFT	**0.535**	0.538	0.003	0.536	**0.531**	-0.005	0.536	**0.534**	-0.002	0.534	**0.533**	-0.001	0.531	**0.529**	-0.002	0.537	**0.532**	-0.005
Inf. Tech.	MSI	**0.903**	0.908	0.004	0.910	**0.900**	-0.010	**0.854**	0.862	0.008	**0.801**	0.814	0.012	**0.851**	0.852	0.001	**0.953**	0.957	0.003
Com. Serv.	OMC	0.623	**0.614**	-0.009	**0.612**	0.616	0.004	**0.610**	**0.610**	0.000	**0.604**	0.610	0.006	0.603	**0.596**	-0.008	0.591	**0.588**	-0.002
Energy	OXY	0.641	**0.635**	-0.006	**0.636**	**0.636**	0.000	0.637	**0.645**	0.007	**0.635**	0.637	0.002	**0.634**	0.638	0.004	**0.639**	0.640	0.001
Health Care	PFE	**0.663**	0.665	0.002	**0.662**	0.668	0.006	**0.666**	0.667	0.001	0.675	**0.668**	-0.007	**0.663**	0.668	0.005	0.677	**0.672**	-0.005
Cons. Disc.	POOL	0.813	**0.805**	-0.007	0.777	**0.771**	-0.006	**0.819**	0.839	0.020	**0.882**	0.903	0.021	**0.972**	0.986	0.015	**0.888**	0.892	0.004
Cons. Disc.	RL	**0.699**	0.700	0.001	0.681	**0.670**	-0.012	0.628	**0.605**	-0.023	0.617	**0.578**	-0.039	**0.580**	0.588	0.008	0.517	**0.514**	-0.003
Industrials	RSG	0.613	**0.603**	-0.009	**0.600**	0.602	0.002	0.671	**0.646**	-0.025	0.712	**0.649**	-0.064	0.605	**0.599**	-0.006	**0.586**	0.590	0.004
Cons. Disc.	SBUX	0.521	**0.516**	-0.005	**0.520**	0.526	0.005	0.522	**0.518**	-0.003	**0.511**	0.518	0.007	0.510	**0.508**	-0.002	0.504	**0.503**	-0.001
Energy	SLB	0.623	**0.613**	-0.010	0.640	**0.613**	-0.026	0.617	**0.608**	-0.009	**0.607**	0.607	0.000	0.630	**0.612**	-0.019	**0.605**	0.604	0.000
Index	SP500	0.563	**0.554**	-0.009	0.563	**0.554**	-0.009	0.560	**0.553**	-0.007	0.562	**0.555**	-0.007	0.557	**0.553**	-0.004	**0.555**	0.559	0.004
Inf. Tech.	STX	0.641	**0.630**	-0.011	**0.643**	0.664	0.021	0.621	**0.620**	-0.002	0.644	**0.638**	-0.006	**0.551**	0.575	0.023	0.607	**0.606**	-0.001
Health Care	TMO	0.582	**0.575**	-0.006	0.578	**0.575**	-0.003	0.579	**0.577**	-0.002	**0.582**	**0.582**	0.000	0.574	**0.572**	-0.001	0.584	**0.572**	-0.012
Health Care	VRTX	**0.423**	0.430	0.007	**0.415**	0.416	0.001	0.419	**0.416**	-0.002	**0.380**	0.383	0.003	**0.337**	0.340	0.004	**0.363**	0.369	0.006
Com. Serv.	VZ	0.729	**0.718**	-0.011	0.722	**0.718**	-0.005	0.721	**0.713**	-0.008	0.720	**0.717**	-0.003	0.732	**0.719**	-0.014	0.727	**0.718**	-0.009
Inf. Tech.	WDC	0.579	**0.575**	-0.004	**0.573**	0.580	0.007	**0.559**	0.561	0.002	**0.560**	0.564	0.004	**0.550**	**0.550**	0.000	0.587	**0.586**	-0.001
Inf. Tech.	XLNX	0.593	**0.590**	-0.002	0.602	**0.594**	-0.008	0.585	**0.585**	-0.001	**0.584**	**0.584**	0.000	0.585	**0.581**	-0.004	**0.583**	0.584	0.002
Cons. Disc.	YUM	0.387	**0.383**	-0.004	0.390	**0.383**	-0.008	0.380	**0.379**	-0.001	**0.367**	**0.367**	0.000	**0.366**	0.367	0.001	0.364	**0.360**	-0.004
average		0.6362	0.6341		0.6331	0.6400		0.6275	0.6285		0.6246	0.6262		0.6220	0.6259		0.6232	0.6200	
times-best		12	19		11	20		11	21		16	17		14	17		12	18	

seasonal fluctuations, LSTM had the best performance for the MASE metric, followed by the ensemble technique. When considering the RMSE, the ensemble matched the LSTM with equally good performance or a little better. When we look at the FBM series, CNN was the best technique outperforming the others on both RMSE and MASE metrics. When we observe the average and how many times the model has a minor error, we can see that the proposed method has better results.

Table 4 shows that FBM series behavior depends mainly on the H parameter. As we approach the $H = 1$, the positive auto-correlated factor becomes more relevant, and as it approaches $H = 0.5$, the series becomes more similar to the Brownian motion or the Winner process. Values below 0.5 are negative auto-correlations that we decided were not worth exploring. As we approach $H = 1$, the results show that the ensemble starts to outperform the LSTM and the CNN. We believe that for series that are more similar to a Brownian motion, ensemble and LSTM fail to generalize the behavior. However, LSTM and the ensemble method perform better as the auto-correlation in the FBM becomes relevant. This increase in performance may be caused by the fact that CNN can better deal with noise data without overfitting, while LSTM and the ensemble can better capture the auto-correlation implicitly present in the series. Here we can see that for the RMSE, the ensemble has the second best result on average, behind the CNN, while for the MASE, the ensemble has the best result but also has the minor error more times than the other models.

We highlight some exciting insights from the results obtained in the real-world and synthetically generated time series: from the observations, CNN and LSTM had better performance for series with more noise and long-term dependent components such as

Table 2. Predictors RMSE and MASE comparison for Auto-ARIMA, CNN, LSTM, and ENSEMBLE in 30 real-world financial series. N = 256 and M = 10. Times-best shows how many times the model has the minor error for each series.

	Auto-ARIMA		RESNET		CNN		LSTM		ENSEMBLE	
Asset	RMSE	MASE	RMSE	MASE	RMSE	MASE	RMSE	MASE	RMSE	MASE
AAPL	0.0160	0.7226	0.0148	0.6204	0.0150	0.6200	0.0150	0.6240	**0.0145**	**0.6198**
AIG	0.0131	0.6873	0.0132	**0.6438**	**0.0130**	0.6450	0.0132	0.6545	0.0131	0.6438
ALL	0.0120	0.6461	0.0085	0.4550	**0.0080**	**0.4550**	0.0082	0.4635	0.0081	0.4553
AMZN	0.0219	0.7334	0.0124	0.4193	0.0120	0.4080	0.0120	0.4130	**0.0118**	**0.4039**
AXP	0.0135	0.7048	0.0130	0.6334	0.0130	**0.6320**	0.0126	0.6377	**0.0126**	0.6340
Bitcoin	**0.0165**	**0.3755**	0.0533	0.9310	0.0530	0.9260	0.0530	0.9160	0.0531	0.9220
HO1	**0.0248**	**0.6905**	0.0280	0.7787	0.0280	0.7820	0.0280	0.7770	0.0279	0.7791
NG1	0.0466	0.6995	0.0386	0.6086	**0.0380**	0.6030	**0.0380**	**0.6020**	0.0383	0.6051
DIS	0.0122	0.6842	0.0117	0.7298	**0.0110**	**0.6250**	**0.0110**	0.6270	0.0113	0.6257
GOOGL	**0.0083**	**0.7419**	0.0100	0.8790	0.0100	0.8520	0.0100	0.8493	0.0099	0.8547
HES	0.0240	0.8135	**0.0170**	0.5919	**0.0170**	**0.5600**	0.0171	0.5762	0.0171	0.5955
MMM	0.0100	0.6566	0.0098	0.6280	0.0100	**0.6200**	**0.0097**	0.6223	**0.0097**	0.6246
MSFT	0.0165	0.6910	0.0130	0.5348	0.0130	0.5330	0.0130	0.5340	**0.0128**	**0.5322**
MSI	**0.0079**	**0.7426**	0.0108	0.8635	0.0106	0.8139	0.0107	0.8014	0.0106	0.8135
OMC	0.0116	0.7030	0.0111	0.6044	0.0108	0.6098	0.0109	**0.6039**	**0.0108**	0.6057
OXY	0.0160	0.6810	0.0154	0.6746	**0.0152**	0.6370	0.0152	0.6350	**0.0152**	**0.6326**
PFE	**0.0110**	0.7047	0.0112	0.7426	0.0110	0.6680	0.0110	0.6750	**0.0110**	**0.6640**
POOL	**0.0116**	**0.8143**	0.0135	0.8955	0.0133	0.9034	0.0134	0.8823	0.0133	0.8936
RL	0.0132	0.7050	0.0130	0.5904	0.0129	0.5778	0.0130	0.6166	**0.0129**	**0.5772**
RSG	**0.0072**	**0.5913**	0.0094	0.6609	0.0094	0.6486	0.0095	0.7122	0.0094	0.6488
SBUX	0.0138	0.6975	0.0108	0.5309	0.0107	0.5177	**0.0106**	**0.5110**	0.0107	0.5143
SLB	0.0165	0.7023	0.0147	0.6168	**0.0143**	0.6069	0.0144	**0.6068**	**0.0143**	0.6074
SP500	0.0097	0.6812	0.0078	0.6059	0.0080	0.5550	0.0080	0.5620	**0.0077**	**0.5531**
STX	**0.0216**	0.6442	0.0226	0.6400	0.0225	**0.6377**	0.0223	0.6436	0.0227	0.6568
TMO	0.0133	0.6775	0.0125	0.6439	**0.0121**	0.5820	0.0121	0.5821	**0.0121**	**0.5802**
VRTX	0.0249	0.7007	0.0151	0.4090	0.0148	0.3834	**0.0146**	**0.3805**	0.0148	0.3833
VZ	**0.0098**	**0.6922**	0.0107	0.7516	0.0104	0.7168	0.0105	0.7202	0.0104	0.7162
WDC	0.0255	0.6435	0.0174	0.5640	0.0174	0.5638	0.0176	0.5603	**0.0173**	**0.5587**
XLNX	0.0154	0.6877	0.0137	**0.5812**	0.0136	0.5840	0.0136	0.5844	**0.0136**	0.5845
YUM	0.0137	0.6614	0.0085	0.4063	0.0081	0.3665	0.0082	0.3669	**0.0081**	**0.3633**
average	0.0159	0.6859	0.0154	0.6412	0.0152	0.6211	0.0152	0.6247	0.0152	0.6216
times-best	9	7	1	2	8	6	5	5	15	10

seasonality. Also, we observe that for some of the assets CNN, LSTM, and Auto-ARIMA had a better performance. When considering non-linear auto-correlation, the ensemble was the best technique. Therefore, we could infer that non-linear auto-correlations are an important component of most financial time series. In this segment, another inference is that the models have to deal with a higher level of noise and long-term dependent factors. The real-world financial time series noise probably changes as we change the asset, requiring different treatments to approximate the underlying function.

5. Conclusion

This work proposes stacking CNN, LSTM, and RESNET as an ensemble model to predict financial time series returns. From the results in real-world time series, we can observe that our proposed ensemble could outperform the architectures individually and

Table 3. Predictors RMSE and MASE comparison for Auto-ARIMA, CNN, LSTM, and ENSEMBLE in synthetic series Gaussian noise, NARMA, and FBM. Times-best shows how many times the model has the minor error for each series.

	Auto-ARIMA		RESNET		CNN		LSTM		ENSEMBLE	
series	RMSE	MASE	RMSE	MASE	RMSE	MASE	RMSE	MASE	RMSE	MASE
GN+H	0.645	1.169	0.525	0.915	0.439	0.774	**0.426**	**0.737**	**0.426**	0.742
NARMA	0.091	1.012	0.106	0.974	0.091	0.899	**0.088**	0.759	**0.088**	**0.758**
FBM	0.015	9.138	0.008	4.778	**0.005**	**1.714**	0.006	2.190	**0.005**	1.749
average	0.2502	3.7730	0.2130	2.2223	0.1783	1.1290	0.1733	1.2287	0.1731	1.0832
times-best	-	-	-	-	1	1	2	1	3	1

Table 4. Predictors RMSE and MASE comparison for Auto-ARIMA, CNN, LSTM, and ENSEMBLE in FBM series for H from 0.6 to 0.9. N = 256 and M = 10. Times-best shows how many times the model has the minor error for each series.

	Auto-ARIMA		RESNET		CNN		LSTM		ENSEMBLE	
H	RMSE	MASE	RMSE	MASE	RMSE	MASE	RMSE	MASE	RMSE	MASE
0.60	0.0434	9.8181	0.5047	183.8652	**0.0138**	**2.0930**	0.0538	18.3445	0.0567	11.6263
0.65	0.0347	**13.0468**	0.4961	380.0690	**0.0333**	19.3465	0.0408	23.9500	0.0519	24.1616
0.70	0.0205	11.6987	0.1099	19.1818	0.0203	9.8568	**0.0110**	**4.8795**	0.0129	15.9911
0.75	0.0071	7.2596	0.0050	2.9711	0.0060	4.6125	0.0040	**1.3440**	**0.0037**	1.4443
0.80	0.0319	41.0465	0.2876	533.2403	0.0123	22.0960	0.0428	55.4170	**0.0034**	**3.0467**
0.85	0.0031	8.2716	0.0020	2.8107	0.0023	2.5195	0.0020	1.7698	**0.0017**	**1.6499**
0.90	0.0327	106.3641	0.2058	797.7468	0.0020	6.7070	0.0210	75.6623	**0.0019**	**6.4832**
average	0.0248	28.2151	0.2302	274.2693	0.0128	9.6045	0.0250	25.9096	0.0189	9.2004
times-best	-	1	-	-	2	1	1	2	4	3

the Auto-ARIMA. We also investigate why individual methods were better for some of the assets in some cases. We tested the methods using synthetically generated time series to identify possible components that generate the financial series. The identification of these components can help in choosing the best predictors. Thus, we hope to have provided subsidies for future investigations using different processes for generating synthetic time series. To better understand which factor could affect the performance of the models, we use three different synthetically generated time series to evaluate all the models with different underlying functions that generate the time series. One of the main inferences is that non-linear auto-correlation may play an essential role in the asset price returns series generation. In future works, long-term dependencies and the noise level need to be better explored, which is heavily related to deep learning overfitting. Long-term dependencies also provide more chances to capture a hidden underlying function, which can be an essential component for some assets. Understanding which processes generate specific types of time series returns can help to select models and assess which ones were better for each technique employed while controlling the improvements of the models.

Acknowledgment

This research was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES - Coordination for the Improvement of Higher Education Personnel, Finance Code 001, grant 88882.333380/2019-01), Brazil

References

Akyol, K. (2020). Stacking ensemble based deep neural networks modeling for effective epileptic seizure detection. *Expert Systems with Applications*, 148:113239.

Cheng, H., Tan, P.-N., Gao, J., and Scripps, J. (2006). Multistep-ahead time series prediction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 765–774. Springer.

Di Persio, L. and Honchar, O. (2016). Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International Journal of Circuits, Systems and Signal Processing*, 10(2016):403–413.

Dieker, A. and Mandjes, M. (2003). On spectral simulation of fractional brownian motion. *Probability in the Engineering and Informational Sciences*, 17(3):417–434.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hu, Z., Zhao, Y., and Khushi, M. (2021). A survey of forex and stock price prediction using deep learning. *Applied System Innovation*, 4(1).

Ibrahim, S. N. I., Misiran, M., and Laham, M. F. (2021). Geometric fractional brownian motion model for commodity market simulation. *Alexandria Engineering Journal*, 60(1):955–962.

Imperial, F. and Segura, A. S. (2018). *Modelling Stock Prices and Stock Market Behaviour using the Irrational Fractional Brownian Motion: An Application to the S&P500 in Eight Different Periods*. PhD thesis, PhD thesis, June.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., and Muller, P. A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*.

Jin, Y., Okabe, T., and Sendhoff, B. (2004). Neural network regularization and ensembling using multi-objective evolutionary algorithms. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 1, pages 1–8 Vol.1.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Lim, B. and Zohren, S. (2021). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200209.

Mandelbrot, B. B. and Van Ness, J. W. (1968). Fractional brownian motions, fractional noises and applications. *SIAM Review*, 10(4):422–437.

Maqsood, I., Khan, M. R., and Abraham, A. (2004). An ensemble of neural networks for weather forecasting. *Neural Computing & Applications*, 13(2):112–122.

Mehtab, S., Sen, J., and Dasgupta, S. (2020). Robust analysis of stock price time series using cnn and lstm-based deep learning models. In *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1481–1486.

Osborne, M. F. (1959). Brownian motion in the stock market. *Operations research*, 7(2):145–173.

Pérez-Ortiz, J. A., Schmidhuber, J., Gers, F. A., and Eck, D. (2002). Improving long-term online prediction with decoupled extended kalman filters. In *International Conference on Artificial Neural Networks*, pages 1055–1060. Springer.

Sagheer, A. and Kotb, M. (2019). Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, 323:203–213.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., and Soman, K. P. (2017). Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647.

Sezer, O. B., Gudelek, M. U., and Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing Journal*.

Waheeb, W., Ghazali, R., and Shah, H. (2019). Nonlinear autoregressive moving-average (narma) time series forecasting using neural networks. In *2019 International Conference on Computer and Information Sciences (ICCIS)*, pages 1–5.

Wang, K., Li, K., Zhou, L., Hu, Y., Cheng, Z., Liu, J., and Chen, C. (2019). Multiple convolutional neural networks for multivariate time series prediction. *Neurocomputing*.

Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L., and Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *IJCAI International Joint Conference on Artificial Intelligence*.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., and Zhao, J. L. (2014). Time series classification using multi-channels deep convolutional neural networks. In Li, F., Li, G., Hwang, S.-w., Yao, B., and Zhang, Z., editors, *Web-Age Information Management*, pages 298–310, Cham. Springer International Publishing.