Time series forecasting using a hybrid ARIMA and LSTM model

Oussama FATHI,
Velvet Consulting,
64, Rue la Boétie, 75008,
ofathi@velvetconsulting.com

***Abstract*— Inspite of its great importance, there has been no general consensus on how to model the trend and the seasonal component in time-series data. Box and Jenkins auto-regressive integrated moving average (ARIMA) is one of the more popular linear models in time series forecasting of the past four decades. Meanwhile, artificial neural networks (ANN) have gathered some meritorious results among forecasting models. Since time-series data is sequential, we will be focusing more on recurrent neural networks (RNN) because they are the most suited to the problem . Both the trend and the seasonal component are important to model in order to build a robust prediction and it is needless to say how capital this is in many fields such as sales forecast, supply chain optimization and so on. In this paper, a hybrid methodology that combines ARIMA and RNN models is proposed to take advantage of the unique strength of ARIMA in seasonal component modeling and RNN in trend forecasting. Experimental results with real data sets indicate that the hybrid modeling approach can be an effective way to make forecasting accuracy higher than what it would have been by either of the models used separately.**

I. INTRODUCTION

Time series exploration is a very important area of data analysis, extracting knowledge from past observations to identify the evolution of a phenomenon in the present and facilitate its projection into the future. This exploration also highlights certain correlations that may seem unavoidable to the naked eye and identifies certain characteristics necessary to understand the current state of this phenomenon. Much effort has been made by the scientific community to improve the exploration and analysis of time series.

One of the most famous and popular models for this type of analysis is the ARIMA model introduced by Box and Jenkins in 1976. The popularity of this model comes mainly from its flexibility, quality and versatility. It is available for autoregressive processes (AR), moving average processes (MA) and the combination of the two (ARMA). ARIMA operates on processes that are called stationary, but can adapt to processes that are not in its integrated version (The "I" in ARIMA). Last but not least, this model may also have seasonal fluctuations in the SARIMA time series. A major limitation of this model is its high error rate as soon as normality and/or linearity in the data is lost.

Recently, the evolution of information technologies has contributed to generating massive amounts of data on the one hand, and to preparing the necessary IT support for calculations on the other. This evolution has also introduced the implementation of learning models that are more efficient than those proposed by conventional statistics. Neural networks then reach their peak and outperform traditional models, whether it is a question of regression, classification or more advanced problems such as the processing of multimedia data (image, sound) thanks to deep learning. The particularity of neural networks is their tendency to adapt to the features created by the data itself. This data oriented approach facilitates learning despite the loss of certain essential characteristics in classical statistics such as linearity, normality, homosedasticity or observation independence. Like the ARIMA model, neural networks are available in several architectures, each corresponding to a type of problem (RNN, CNN, DQN, GAN, etc.).

In this paper, we try to couple the effect of these two forms of modelling to produce a result that could exceed the individual performance of each model. We present a hybrid modelling approach for time series analysis. The latter combines ARIMA and artificial neural networks. Combining the two would help us to capture certain patterns that would be inaccessible to one of the two models without the support of the other and thus provide satisfactory results in time series prediction. The rest of this paper is organized as follows: The following section puts the two models within their respective theoretical frameworks. Sections III and IV describe the empirical part of the study by providing an overview of the nature of the data involved and the metrics used to measure the performance of our model on these data. The last section summarizes the remarkable results of the study.

II. MODEL DESCRIPTION

This section is devoted to the presentation of the models, which are the subjects of the comparison. We start with the most famous and widely used method, that of Box and Jenkins, still known as the "ARIMA family of models". Then, we explore a little bit the descriptive literature of recurrent neural networks, of which we present a class closely related to our study, the Long Short-Term Memory (LSTM) neural networks. We end by introducing our hybrid ARIMA/LSTM approach, the subject of this paper.

A. BOX AND JENKINS METHOD

The Box-Jenkins methos, named after the statisticians George Box and Gwilym Jenkins, applies autoregressive integrated moving average (ARIMA) models to find the best fit of a time-series model to past values of a time series.

The original model is based on an iterative three-stage

modeling approach :

- **Model identification and model selection** : making sure that the variables are stationary, identifiying seasonality in the dependent series and extract knowledge from the autocorrealtion, partial autocorrelation and inverse autocorrelation plots to decide which (if any) autoregressive or moving average component should be used in the model (by checking which model minimizes the Akaïke Information Criterion (AIC) and the Bayesian Information Criterion (BIC)).
- **Parameter estimation** using computation algorithms to arrive at coefficients that best fit the selected ARIMA model. The most commom methods use maximum likelihood estimation or non-linear least-squares estimation.
- **Model checking** by testing whether the estimated model conforms to the specifications of a stationary univariate process. In particular, the residuals should be independent of each other and constant in mean and variance over time. If the estimation is inadequate, we have to return to step one and attempt to build a better model.

1) **Stationary processes:** As we have seen, ARIMA models work on stationary processes, this means that the mean and the variance must be constant over time. Additionally, there should be no dependency between residuals. These hypotheses can easily be checked owing to augmented Dickey and Fuller test or Philippe-Perron test. Still, what makes a process non-stationary is the presence of two identifiable components over a time series: the trend and the seasonal component. Therefore, to make a time-series process stationary, we should eliminate its trend and seasonal components.

The seasonal component is not hard to eliminate. The idea is to differentiate the observations over the period of the cycle. Formally, if y_t is a seasonal process and T its period, we can define a new process z_t as :

$$z_t = y_t - y_{t-T}$$

This operation is called differencing and its output, z_t, is a stationary process.

In [1], more attention is paid to the trend component. The latter models the global motion of the time series. As a reminder, we speak of a trend when the series y_t can be written, with the exception of an adjustment error ε_t, as a combination of m time functions, chosen a priori:

$$y_t = \sum_{i=1}^{m} \alpha_i f_i(t)$$

The presence of a trend prevents the process from being stationary. To overcome this problem, two types of stationarity are defined:

- **By eliminating the trend (trend stationary, denoted as TS)**: the procedure consists in estimating the trend (with ordinary least squares for example) and subtracting it from the original series. This technique is used when the trend component is deterministic. For example:

$$y_t^{(TS)} = a + bt + \varepsilon_t$$

- **By differencing the series (differencing stationary, denoted as DS)**: the procedure consists in differentiating the series $\Delta y = y_t - y_{t-1}$ until it reaches a stationary process. This technique is used when the trend component is random. For example:

$$y_t^{(DS)} = y_{t-1}^{(DS)} + b + \varepsilon_t$$

A first difference between the two processes $y_t^{(TS)}$ and $y_t^{(DS)}$ is that the second one involves a previous observation while the first one is completely detached from the past. We will now try to explore the recurring relationship of the DS process. We have:

$$\begin{aligned} y_t^{(DS)} &= y_{t-1}^{(DS)} + b + \varepsilon_t = (y_{t-2}^{(DS)} + b + \varepsilon_{t-1}) + b + \varepsilon_t \\ &= y_{t-2}^{(DS)} + 2b + (\varepsilon_t + \varepsilon_{t-1}) \end{aligned}$$

By pushing the development to the end, we find:

$$y_t^{(DS)} = y_0^{(DS)} + bt + \sum_{i=1}^{t} \varepsilon_i$$

Once this recurrence formula has been established, two new differences appear: The TS process is only noisy by ε_t at the moment t, the effect of noise is said to be temporary. The DS process is additionally noisy due to all past errors, it is said that the noise is permanent. The constant part of the TS process (denoted as a) is deterministic, while its equivalent in the DS process (noted $y_0^{(DS)}$) is random.

Both processes can, however, be expressed in a linear form, with a slope, an intercept and the effect of an additive noise. To check whether the process is stationary or not (yet), we have a multitude of tests in the literature such as the augmented Dickey-Fuller (ADF) test or the Phillips-Perron (PP) test. However, it is very difficult to determine whether a process is TS or DS and the tests cited can be mistaken about the nature of the processes.

2) **ARIMA model:** ARIMA was first introduced by Box and Jenkins in [2] in 1976 in a book that received tremendous attention from the scientific community, working on research works oriented towards prediction at that time. This method is therefore applied in a wide variety of fields and remains one of the most robust models in data processing and operational prediction [3]. ARIMA characterizes time series by going from three fundamental aspects:

- Autoregressive terms (AR) that model past process information.
- Integrated terms (I) that model the differences needed to make the process stationary.
- The moving average (MA) that controls the past information of noise around the process.

Specifically, the terms AR give a representation of the series based on its p past observations:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} = \sum_{j=1}^{p} \phi_j X_{t-j}$$

where the sequence $(\phi_j)_j$ represents the auto-regressive coefficients and p is the number of previous observations necessary to predict the value of the series at the present time. The MA component, instead of focusing on past

observations, presents a rolling average of previous error terms in a regression model (innovation processes):

$$X_t = \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} = \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

where the sequence $(\theta_j)_j$ represents the MA coefficients, $(\varepsilon_{t-j})_j$ the previous innovation processes at the moment t and q is the number of moving average jumps deployed to predict the current value of the series. Finally, in addition to all this, there is the integrated part (it is the I in ARIMA), characterized by a parameter d which represents the order of differentiation :

$$X_t^* = X_t - X_{t-1} - \cdots - X_{t-d}$$

Generally, taking $d = 1$ is more than enough [4]. A remarkable class of processes is the one for which $d = 0$ called ARMA.

By introducing the "Backward" operator $BX_t = X_{t-1}$, an ARIMA model can formally be written:

$$(I - \sum_{j=1}^{p} \phi_j B^j)(I - B)^d X_t = (I + \sum_{j=1}^{q} \theta_j B^j)\varepsilon_t$$

Box and Jenkins also generalized the ARIMA model to recognize seasonal fluctuations in a time series. We therefore speak of the SARIMA model (the S which is added means "Seasonal") and we write [5]:

$$X_t = SARIMA(p, d, q)(P, D, Q)_m$$

The parameters P, D and Q present the number of seasonal terms that are auto-regressive, integrated and mobile respectively. The periodicity of the seasonal phenomenon (daily, weekly,...) is set by m.

3) **Limitations of the model:** Despite its widespread and high performance, the ARIMA model still has some limitations that hinder predictions and lead to a decrease in accuracy.

We saw in the section dedicated to explaining stationary processes that there were two types of stationarity and despite their dissimilar construction, it is, however, often difficult whether a given series is TS or DS. Examples can be easily constructed to illustrate the arbitrary closeness of TS and DS models [6]. Although statistical tests sych as the Dickey-Fuller test and the Phillips-Perron test, among others, have been developed in the time-series econometrics litterature, these tests suffer from very low power in distinguish between a unit-root and a near-unit-root process [7].

On another hand, as known in statistical inference, there is always a need to put the proposed estimate for a parameter within an interval with a certain level of confidence. The reason behind this process is simple, if the probability of belonging to this confidence interval is low, the estimator could very well be biased or have a non-optimal variance. To demonstrate a second limitation of the ARIMA models, we will focus on the MA part and build a confidence interval for the coefficients of this part of the model. We have:

$$(\forall h < q), \sqrt{n}(\hat{\rho}(h) - \rho(h)) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \hat{\sigma}^2)$$

Therefore:

$$(\forall h < q), \sqrt{n}\frac{\hat{\rho}(h) - \rho(h)}{\hat{\sigma}} \xrightarrow{\mathcal{L}} \mathcal{T}(n-1)$$

We then find, at the risk of α:

$$\mathrm{P}\left(\left| \sqrt{n}\frac{\hat{\rho}(h)}{\hat{\sigma}} \right| \leq t_{1-\frac{\alpha}{2}} \right) = 1 - \alpha$$

where $t_{1-\frac{\alpha}{2}}$ is $1 - \frac{\alpha}{2}$ the Student's law with $n-1$ degrees of freedom percentile. For $\alpha = 5\%$, $t_{1-\frac{\alpha}{2}} \approx 2$, we get, with a 95% confidence level:

$$\hat{\rho}(h) \in \left[-\frac{2\hat{\sigma}}{\sqrt{n}}, \frac{2\hat{\sigma}}{\sqrt{n}} \right]$$

The two bounds of the interval depend on h and therefore, the further away from the past, the wider the confidence interval becomes. In addition, the (S)ARIMA model is very expensive in terms of calculation time. The steps of the modeling are numerous:

- Estimate the trend and seasonality and isolate them ;
- Perform the noise whiteness test ;
- Perform unit root tests to examine stationarity ;
- Calculate the model parameters ;
- Estimate the model coefficients.

It is clear that, although efficient, this model remains quite heavy and a mistake in one step could lead to a snowball effect.

B. NEURAL NETWORKS

Neural networks are not a new model. Indeed, they were introduced in the 1950s (the perceptron was introduced in 1958)[8]. Nevertheless, they became obsolete in the 1970s because they were too expensive in terms of calculation and there was not enough data at the time to train them. From the 1990s to today, neural networks have enjoyed their heyday and have aroused the curiosity of data enthusiasts and leading researchers alike thanks to digital transformation cradled by abundant data flows, thanks to the computing power offered by GPUs, but above all, thanks to deep architecture (deep learning). To familiarize oneself with the basics necessary for understanding neural networks, the literature is plethoric and there are many publications on this subject [9]. The RNNs, which are more complicated to understand, will be clarified in this section.

1) **RNN and vanishing gradient:** RNNs are used to process sequential data, a sequence of inputs $x_1, \cdots, x_m$,indeed, at the time t , they calculate their output according to the input x_t but also the state of the hidden layer at the previous time. Thus, they develop an internal state that acts as a short-term memory and makes it possible to take into account the temporal dependencies that the inputs manifest.The simplest RNNs are described as follows. Let: $n, k, p \in N, x_0, \cdots, x_m$ where $x_t \in R^n, W \in M_{kn}(R), W_h \in M_{kk}(R), O \in M_{pk}(R)$ and $h_{-1} \in R^k$.

Then, the dynamics of the associated RNN is described by:

- $h_t = \sigma(Wx_t + W_h h_{t-1})$
- $y_t = O.h_t$

with $h_t \in R^k$ the state of the hidden layer and $y_t \in R^p$ the state of the output layer. In practice, we often take $h_{-1} = 0$.

These RNNs can also be visualized in the form of a graph. Conventional recurrent neural networks (such as Vanilla) are exposed to the problem of vanishing gradient that prevents them from changing their weight according to past events. During training, the network tries to minimize an error function that depends in particular on the output $R(x_t)$. For our example, according to the chain derivation rule, the contribution to the gradient $\frac{\partial R(x_t)}{\partial U_{mn}}$ of the entry x_{t-k} is:

$$\frac{\partial R(x_t)}{\partial U_{mn}} = \cdots + \frac{\partial R(x_t)}{\partial x_t}\frac{\partial x_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\cdots\frac{\partial h_{t-k+1}}{\partial h_{t-k}}\frac{\partial h_{t-k}}{\partial U_{mn}} + \cdots \tag{1}$$

$$= \cdots + \frac{\partial R(x_t)}{\partial x_t}\frac{\partial x_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\cdots\frac{\partial h_{t-k+1}}{\partial h_{t-k}} \times \sigma'(Wx_{t-k} + W_h h_{t-k-1})E_{mn}x_{t-k} + \cdots \tag{2}$$

where E_{mn} represents the matrix such that the only non-zero term is the term (m, n) equal to 1. For standard values of the activation function and network weights, the product $\frac{\partial h_t}{\partial h_{t-1}}\cdots\frac{\partial h_{t-k+1}}{\partial h_{t-k}}$ decreases exponentially according to k. Therefore, the correction of the error if it involves a distant event will decrease exponentially with the time interval between that event and the present. The network will be unable to take into account past entries. Actually ,to update its weights, a neural network backpropagates the gradient where it calculates the derivative of its activation function evaluated in the activation term. A classic RNN, not choosing which terms to keep, is forced to keep them all, which causes the error to decrease exponentially. The network stops learning wrongly because it believes that the error is minimal (since it is zero).

2) **LSTM:** The architecture of an LSTM model is more complex to present. In this document, we will simply present an overview diagram as well as the equations that govern the set and a quick description. The Long Short-Term Memory (LSTM) network is the most widely used architecture in practice to address the problem of gradient disappearance. This network structure was proposed by Sepp Hochreiter and Jrgen Schmidhuber in 1997 [10]. The idea associated with the LSTM is that each computational unit is linked not only to a hidden state h but also to a state c of the cell that plays the role of memory. The change from c_{t-1} to c_t is done by constant gain transfer equal to 1, so that errors are propagated at previous steps without any gradient disappearance phenomenon. The status of the cell can be modified through a door that allows or blocks the update (input gate). Similarly, a door controls whether the state of the cell is communicated at the output gate of the LSTM unit. The most common version of LSTMs also uses a gate to reset the cell state (forget gate). The dynamic equations of this model are as follows:

- $F_t = \sigma(W_F x_t + U_F h_{t-1} + b_F)$ (forget gate)
- $I_t = \sigma(W_I x_t + U_I h_{t-1} + b_I)$ (input gate)
- $O_t = \sigma(W_O x_t + U_O h_{t-1} + b_O)$ (output gate)
- $c_t = F_t \circ c_{t-1} + I_t \circ tanh(W_C x_t + U_C h_{t-1} + b_C)$
- $h_t = O_t \circ (tanh(c_t)$
- $o_t = f(W_o h_t + b_o)$

The terms b_F, b_I, b_O, b_c and b_o represent the different biases. Operator $\circ$ represents Hadamard's product (element-wise product). Initially, we take $c_0 = h_0 = 0$.
With the advent of deep architectures, LSTM is widely used today in sequence learning and has demonstrated great experimental power [11]. However, this architecture is subject to many changes, it has almost as many variations as the papers that use it. In 2014, a simplified version of LSTM called GRU (Gated Recurrent Unit) was introduced [12]. It has the advantage of being less computationally cumbersome because it has fewer parameters and equations. Nevertheless, it remains as efficient as its ancestor. Here are the equations that govern the dynamism of the model:

- $Z_t = \sigma(W_Z x_t + U_Z h_{t-1} + b_Z)$ (update gate)
- $R_t = \sigma(W_R x_t + U_R h_{t-1} + b_R)$ (reset gate)
- $h_t = Z_t \circ h_{t-1} + (1 - Z_t) \circ tanh(W_h x_t + U_h(R_t \circ h_{t-1}) + b_h)$

In the following, we will confuse the two architectures but the calculations were done with LSTM.

C. HYBRID MODEL

We have seen that in order to successfully identify the behaviour of a time series, it is essential to control its two main components: the trend and the cyclical component. The first describes the overall movement while the second points to periodic fluctuations. The ARIMA model operates on series that do not contain these elements: it estimates them, isolates them and then eliminates them. The parameters resulting from this operation are the model parameters. ARIMA then estimates the model coefficients by maximum likelihood. It was noted that for a long-term prediction, the estimation is less and less accurate. However, it is also noted that ARIMA is effective in estimating the seasonal component. Cyclic, this sequence keeps the same values for a given period. On the other hand, it is this seasonal component (in addition to noise to a lesser extent) that makes the trajectory of the time series erratic and volatile. Thus, if the series lacks this component, the function obtained is smoother. It is known that if one seeks to learn/approach a function through a neural network, the results would be more fruitful if this function is smooth. The idea is therefore to transform the original time series y_t into a smooth function by isolating it from its seasonal component. To do this, we proceed with the first step of the Box and Jenkins method, which is the $y_t = m_t + s_t + \varepsilon_t$ decomposition. Three functions are obtained: the trend m_t, the seasonal component s_t and the noise ε_t. s_t and ε_t are retained to reproduce seasonality and put the predicted values into confidence intervals using ε_t. From now on, m_t is smooth because it is free of any fluctuation. We can therefore approach it through a network of neurons to know its dynamics and thus predict its future. As this is a time series, the observations are sequential and therefore we use LSTM-type RNNs.
This approach not only avoids the cumbersome calculations required by the ARIMA model, but also avoids the confusion

created by the TS and DS processes. In addition, we take the best performance of both ARIMA and LSTM models: the first is optimal for the seasonal component and the second is effective for the trend. Once this is done, we can cross-check the predicted trend $\hat{m}_t$ and the reproduced cyclical component s_t to construct a forecast of the original series, noted $\hat{y}_t$.

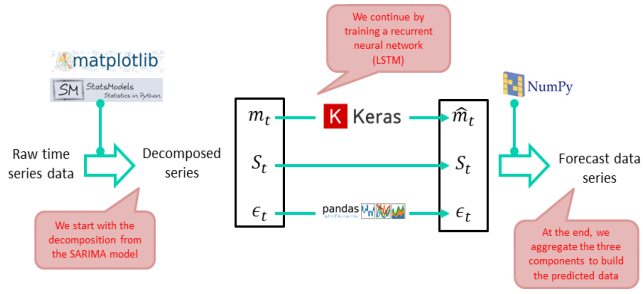

Fig. 1: Data pipelines for the hybrid model

III. EXPERIMENTS

A. FORECAST PERFORMANCE METRICS

To be able to validate the accuracy of forecasting, several performance measures have been proposed in the litterature [13]. It is a common practice to consult various metrics to analyze time series. We have selected three metrics that possess different properties, which is important to efficiently understand the forecasting capabilities of the models seen from different points of view. Each of the metrics summarized here is a function of the actual time series (denoted as a_i) and the forecast results (denoted as f_i). n refers to the number of observations.

1) **Root Mean Squared Logarithmic Error (RMSLE):** The RMSLE is defined as :

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(f_i+1) - log(a_i+1))^2}$$

Because it uses a logarithmic scale, the RMSLE is less sensitive to outliers than the standart root mean square error (RMSE). Moreover, the RMSE reacts to over underestimation and overestimation the same way, whilst the RMLSE penalizes more the underestimated predictions.

2) **Mean Absolute Scaled Error (MASE):** The MASE was chosen owing to its scale independence, i.e. it can be used to compare the relative accuracy of many models applied to different datasets. To calculate the MASE, each of the errors is scaled with the in-sample mean absolute error (MAE). This metric is estimated by :

$$MASE = \frac{1}{n}\sum_{i=1}^{n}\frac{|f_i - a_i|}{\sum_{j=2}^{n}|a_j - a_{j-1}|}$$

3) **Mean Absolute Percentage Error (MAPE):** The MAPE can be defined as :

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{f_i - a_i}{a_i}\right|$$

The MAPE is also scale-independent, since it is a percentage error measure of the average absolute error. We still should be careful when using this metric, because if the time series have zero values, the MAPE yields undefined results due to division by zero. One more warning, the MAPE is biased towards underestimated values and does not penalize large errors.

B. DATASETS

In this study, two popular datasets are used to demonstrate the effectiveness of the hybrid model : the Wolf's sunspot data and the US dollar/British pound exchange rate data. These time series have different statistical characteristics and have been widely studied in the statistical and the neural network litterature owing to their mixed linear and non-linear properties.

1) **The Wolf's sunspot data:** The sunspot data we consider contains the annual number of sunspots from 1700 to 1988, giving a total of 289 observations. The data series is regarded as nonlinear and non-Gaussian and is well suited for training non-linear models. The plot of this time series (see Figure 1) also suggests that there is a cyclical pattern with a mean cycle of about ten years. In [14], a different kind of hybrid approach has been applied on these data, using ARIMA and simple MLP.

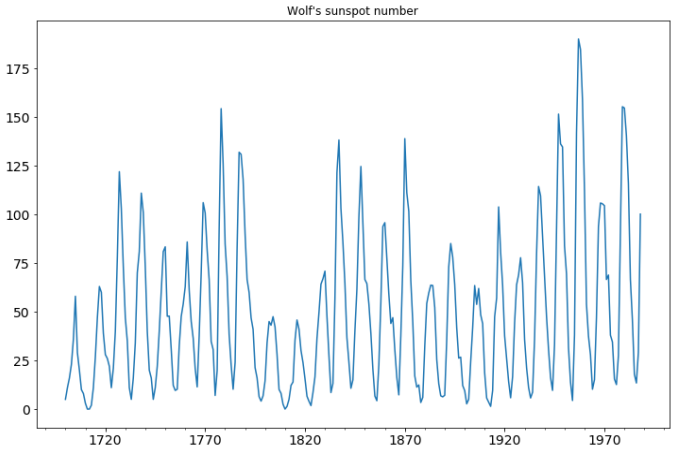

Fig. 2: The Wolf's sunspot data trajectory

2) **The US dollar/British pound exchange rate data:** The second data set is the exchange rate between British pound and US dollar. The volatility in this type of data gives rise to several problems of stationarity and non-linearity, which makes it an interesting subject for our study. Various linear and nonlinear theoretical models have been developed but few are more successful in out-of-sample forecasting than a simple random walk model [14]. The data used in this paper contain the daily observations from 1971 to 2018, giving 11939 data points in the time series. The time series plot is given in Fig. 2, which shows the numerous changing turning points in the series.

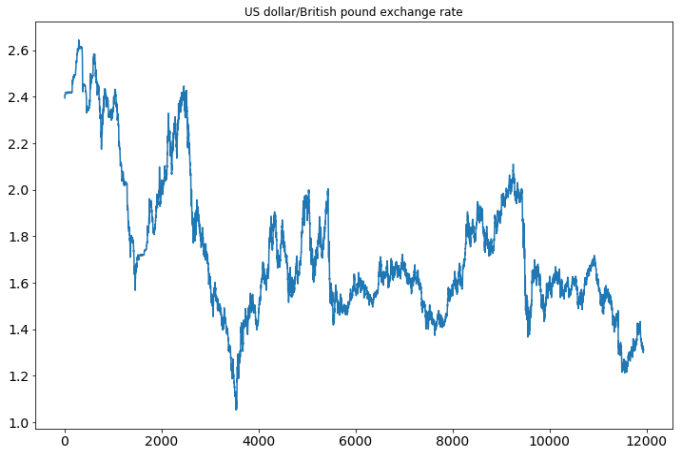

Fig. 3: Daily BP/USD exchange rate data trajectory

IV. METHOD

A. DATA PREPARATION

To assess the forecasting performance of different models, each data set is divided into two samples of training and testing. The training data (approximately 70% of the dataset) is used exclusively for model development and then, the test sample is used to evaluate the established model. The data compositions for the two datasets are given in Table 1. A small specification to mention for the USD/BP exchange rate dataset, in view of the apparent volatility and nature of the trend in the data, a natural logarithmic transformation was required for the study [15].

Series	Sample size	Training set (size)	Test set (size)
Sunspot	289	1700-1920 (221)	1921-1988 (68)
USD/BP	11939	1971-2003 (8357)	2003-2018 (3582)

TABLE I: Sample compositions in datasets

It should also be mentioned that for the LSTM part, a change of scale of the data is planned. The goal is to put all values between 0 and 1 to facilitate model training and achieve better performance.

B. MODELS SPECIFICATION AND PARAMETERS

For the Sunspot dataset, we compared a $SARIMA(10,1,0)(10,0,0)_{10}$ model, a $10 \times 4 \times 1$ LSTM and the hybrid model with the same LSTM structure applied only to the trend. Subba Rao and Gabr in [16] and Hipel and McLeod in [17] certify that the dataset is more suitable for an $ARMA(9,0)$, but the performance measures we have selected favour our model. As for the USD/BP exchange rate dataset, we found that the best model was $ARMA(1,0)$ (random walk). We compared this model to an LSTM $5 \times 4 \times 1$ and a hybrid model, on the same principle as the first dataset. Tables 2 and 3 summarize this setting.

LSTM			
Datasets	n	k	p
Sunspot	10	4	1
USD/BP	5	4	1

TABLE II: LSTM parameters

ARIMA							
Datasets	p	d	q	P	D	Q	m
Sunspot	10	1	0	10	0	0	10
USD/BP	1	0	0	0	0	0	0

TABLE III: ARIMA parameters

V. RESULTS

To train the models, we chose a 10-year period to obtain 10-year predictions for the first dataset and a 5-day period for weekly predictions (considering only the working days of the week) for the second. For each new predicted value, the previous m values (where m is the period) were considered and the prediction error was calculated. The overall error is calculated on a pseudo-average basis of the errors obtained.

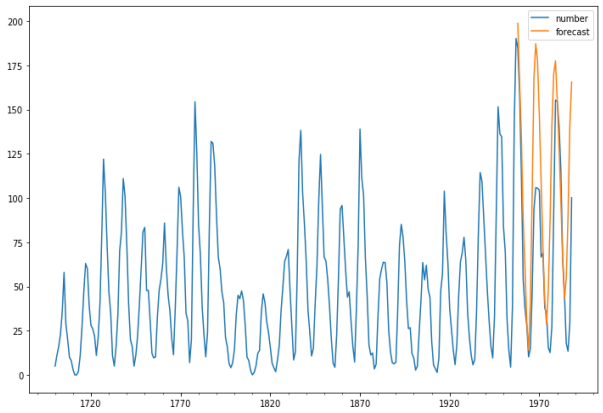

Fig. 4: Forecasted data for the Wolf dataset

Fig. 5: Forecasted data for the USD/BP exchange rate dataset

It can be seen that the predictions from the ARIMA model follow the reality at the peak level and gradually detach themselves from the original curve for the other periods. With regard to the USD/BP exchange rate data, we see that the adjustment made by the ARIMA model is almost perfect.

This challenges LSTM even more. Figure 4 shows a the forecasted data for USD/BP exchange rate data.
We will now quantify the error using our previously introduced performance metrics:

	ARIMA	LSTM	Hybrid
RMSLE	0.967	0.971	0.804
MASE	1.034	1.000	0.741
MAPE	18.7	16.8	3.2

TABLE IV: Errors metrics for the Sunspot data

	ARIMA	LSTM	Hybrid
RMSLE	0.001	0.014	0.008
MASE	0.057	0.931	0.055
MAPE	4.1	4.1	3.9

TABLE V: Errors metrics for the USD/BP exchange rate data

Except for the RMSLE, we see that LSTM is more efficient than ARIMA in the sense that the predictions made are more relevant. Again, the grid search used to estimate the LSTM parameters is not optimal in this study. It should be noted that for the XXX dataset, the ARIMA model chosen is the most powerful. LSTM almost matches its performance and the hybrid model surpasses it, even slightly. LSTM could very well have a better performance if this operation is done correctly. One last thing, the advantage of LSTMs is more highlighted when it comes to long-term forecasting. ARIMA remains quite important if it is a question of making accurate predictions for the next few periods to come.

VI. CONCLUSION

Overall, we have seen that neural networks provide more accuracy in long-term predictions. Since the error is not dramatically higher for the ARIMA model proposed by Box Jenkins, we see that its use remains an asset in time series prediction. We have also seen that the two models enjoy complementary differences: ARIMA is much more picky on seasonal peaks while LSTM better matches the stable part of the series. This leads us to consider switching to a hybrid model that would only retain the virtues of the two models. We can very well start by breaking down the series to retain only the trend. The seasonal component being periodic, it is easily duplicated. Then we can train our LSTM on this trend, as this model has proven its ability to predict the trend component. To finalize the study, we cross-reference the two components (by addition or multiplication, depending on the nature of the data) to have a robust prediction. The noise estimate will put the predicted values within a confidence interval. Last but not least, model parameterization is the most important part of model quality assessment. A well parameterized model could produce miraculous results with disconcerting accuracy. Unfortunately, this is a very time-consuming step.

REFERENCES

[1] M. Qi and G. P. Zhang, "Trend time-series modeling and forecasting with neural networks," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 19, 2008.

[2] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco, CA, USA: Holden-Day, Inc., 1990.

[3] K. Lawrence and M. Geurts, *Advances in Business and Management Forecasting*, vol. 19. Emerald Group Publishing, 2004.

[4] R. Adhikari and R. Agrawal, "An introductory study on time series modeling and forecasting,"

[5] G. Box and G. Jenkins, "Time series analysis : forecasting and control," *Ed. Holden-Day*, 1976.

[6] J. Z. Chris Chatfield and J. Lindsey, *An introduction to Generalized Linear Models*. New York, USA: Chapman and Hall/CRC, 2001.

[7] P. C. Phillips, "Testing for a unit root in time series regression," *Biometrika*, 1988.

[8] R. F., "The perceptron : a probabilistic model for information storage and organization in the brain," *Psychological Review*, 1958.

[9] K. Gurney, *An introduction to Neural Networks*. London and New York: UCL Press, 1997.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," 1997.

[11] K. A., "The unreasonable effectiveness of recurrent neural networks," 2015.

[12] K. C. Junyoung Chung, Caglar Gulcehre and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[13] R. Hyndman and A. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, 2005.

[14] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," 2001.

[15] R. Meese and K. Rogoff, "Empirical exchange rate models of the seventies : Do they fit out of samples?," *J. Int. Econ.*, 1983.

[16] T. S. Rao and M. Sabr, *An Introduction to Bispectral Analysis and Bilinear Time Series Models*. Springer-Verlag, 1984.

[17] K. Hipel and A. McLeod, "Time series modelling of water resources and environmental systems," *Elservier*, 1994.