International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

# RECENT TRENDS IN TIME SERIES FORECASTING– A SURVEY

**Yatish H R[1], Dr. S.R Swamy[2]**

[1]*Student, Department of CSE, R. V College of Engineering, Bangalore Karnataka India.*
[2]*Professor, Department of CSE, R. V College of Engineering, Bangalore Karnataka India.*

---

**Abstract -** *Time series forecasting is one of the toughest problems which is very well-known in the field of data science and machine learning. It involves prediction of a set of continuous variables over various time intervals. Traditional techniques require the computation of regression factors analytically to fit the curve to the time series and produce forecasts. Such methods are time consuming, error prone and do not adequately represent the data. Due to the recent developments in the fields of deep learning and artificial intelligence time series forecasting has received a lot of traction especially among deep learning area. This paper presents a review of various deep learning techniques including Generative Adversarial Networks (GANS) and statistical techniques to solve the problem effectively. It compares various models and their performances. It also presents some of the key challenges faced by the models and gives an insight into possible applications of the models. This information will be key to solve the problem efficiently.*

***Key Words***: Time Series Forecasting, Big Data, Machine Learning, Deep Learning, Generative Adversarial Networks

## 1. INTRODUCTION

Recent advances in the field of big data and IoT has led to increased data influx which can be analyzed to enable better decision-making actions for any organization. One such type of data which occurs in abundance is the time series data. Analyzing and forecasting timeseries data has been one of the toughest challenges of modern-day data scientists and researchers.

The main aim of the time series forecasting is to capture the pattern in the data and model it accordingly. The model can then be used to extrapolate and obtain reliable forecasts for any application. A good model always tries to fit the data as accurately as possible with least error without loss of generality.

A time series data is a set of data points representing metrics varying over different steps of time. It can be defined as a set of vectors X(t); t = 0,1, 2…n. The time series is said to be univariate if the number of parameters changing with respect to time is one. If the number of parameters are more than one then it is termed as multivariate time series data.

Any timeseries will have the following four components:

1. Trend: This component affects the overall movement of average of the data across timeseries. Examples include variation of house rents with respect to time.
2. Cyclic: This component describes the medium-term cyclic changes in data. Examples include economic and financial data
3. Seasonal: This component occurs when there is a pattern across timeseries at a repeated interval. Examples include variation of temperatures across a year.
4. Irregular: This component contributes to randomness in the time series data. Real life ex. Include war, strikes, disasters.

Based on the variation of the four components two major models [1] can be deduced viz. Multiplicative and Additive models. The multiplicative and additive models for time series can be mathematically represented as below: -

$$\text{Multiplicative Model: } Y(t) = T(t) \times S(t) \times C(t) \times I(t).$$

$$\text{Additive Model: } Y(t) = T(t) + S(t) + C(t) + I(t).$$

In this T, S, C, I represent Trend, Seasonal, Cyclic and irregular components. Multiplicative model assumes that T, S, C, I variables are independent.

A timeseries is said to be stationary if the statistical process does not vary with respect to time. A stationary process is always easier to analyze. Some of the forecasting models such as ARIMA assume that the timeseries is stationary. The presence of trend and seasonality make the time series non stationary.

Some of the examples of traditional processes are ARIMA and exponential smoothening techniques. The ARIMA [2] model requires a stationary series as input. A non-stationary time series is converted to stationary series by removing non stationary components like trends (removed by differencing timeseries) and seasonality. It is also required to calculate the auto regressive and moving average components by analyzing ACF and PACF plots. This model is disadvantageous as these parameters have to be calculated for each time series which makes the process tedious, time consuming, and error prone.
The Auto Regressive (AR) component of ARIMA can be mathematically represented as:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_p Y_{t-p} + \epsilon_1$$

The moving average represents the overall trend of the time series. The moving average (MA) is represented as:

$$Y_t = \alpha + \epsilon_t + \phi_1\epsilon_{t-1} + \phi_2\epsilon_{t-2} + .. + \phi_q\epsilon_{t-q}$$

The overall ARIMA model is given by sum of AR and MA components. It is represented as:

$$Y^t = AR + MA$$

The equation can be expressed by expanding the terms as below:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p}\epsilon_t + \phi_1\epsilon_{t-1} + \phi_2\epsilon_{t-2} + .. + \phi_q\epsilon_{t-q}$$

The Exponential smoothening algorithm can work on non-stationary data but it has its own demerits such as the forecast lags behind as the trends when the data increases. It also fails to account for the dynamic changes in the real world.

The above mentioned methods are some of the traditional methods which are employed to obtain the forecast. This paper discusses more about the recent developments in both univariate and multivariate time series (using techniques like GAN, Deep Learning, Machine Learning) to overcome some of the disadvantages of the traditional models.

The remainder of the paper is organized as follows. Section 2 analyses the various algorithms in the recent years in time series forecasting. Section 3 discusses some of the major applications of the time series forecasting, Section 4 provides a conclusion to the work.

## 2. TIMESERIES ALGORITHMS

In the univariate timeseries algorithms a single metric varies in different time steps. Some of the popular algorithms which can be used for automated time series analysis are auto arima and prophet. These algorithms are discussed as below:

### 2.1 Auto ARIMA

Auto arima [3] is an extension of the arima model. In ARIMA model a set of parameters (p, d, q) are calculated which are auto regressive, difference term and moving average respectively. The parameter d is determined by KPSS test. The parameter p denotes the order of autoregression, d denotes the difference order and q denotes the moving average order.

The KPSS test [4] for the time series can be represented by following model:

$$y_t = c_t + \delta t + u_{1t}$$
$$c_t = c_{t-1} + u_{2t},$$

Where, $\delta$ is the trend coefficient, $u_{1i}$ is a stationary process and $u_{2t}$ is an independent and identically distributed process having mean 0 and variance $\sigma^2$.

The KPSS test has a null hypothesis that the given time series is stationary. The p value for this hypothesis is set to 0.05. For different values of d the KPSS test is performed and the suitable d is returned.

For determining the Auto Regressor and Moving Average parameters the auto arima models four initial models with (0, d,0), (2, d, 2), (1, d, 0), (0, d, 1). The model with smallest AIC (Arkanine Information Coefficient) is selected and the model is further tuned with variations of p, q by +/- 1. The model with least AIC value is selected as the model. The authors also speak that higher order polynomials are avoided so that the model doesn't overfit the data.

The model has its own limitations such as it cannot be used to predict over a large interval. It is suitable for short term and medium-term predictions. It also needs a lot of data to build the model. Though auto arima is a traditional model it is still used for quick and reliable forecasts for short term in many applications such as weather predictions.

### 2.2 Facebook Prophet

Some of the shortcomings of the models like ARIMA and exponential smoothening are overcome by prophet. Prophet is an open source time series forecast library by Facebook. It is especially designed for modelling business timeseries.

The paper [5] highlights the fact that auto arima struggles to capture seasonality effectively. Auto arima also produces large trend errors at cutoff periods. Also, the exponential smoothening algorithm struggles to capture long term seasonality. Both exponential smoothening and auto arima overreact to end of year forecasts as the models do not adequately account for yearly seasonality.

The authors approach the problem by breaking the time series into three components namely - trend, seasonality, and holidays.

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t.$$

Here g(t) is the trend function, s(t) is seasonality and h(t) is holiday factor and $\varepsilon_t$ represents other parameters not accounted by the model. This parameter is assumed to be normally distributed.

The trend function g(t) represented by the non-linear function and a piecewise continuous linear function. The nonlinear function is given by:

$$g(t) = \frac{C(t)}{1 + \exp(-(k + \mathbf{a}(t)^\intercal\boldsymbol{\delta})(t - (m + \mathbf{a}(t)^\intercal\boldsymbol{\gamma})))}.$$

Where C is the carrying capacity, k is the growth rate, m an offset parameter, and a is a vector representing if at a particular time there is a change point or not. The carrying capacity denotes the external nonlinear factors that affect the time series

The piecewise linear function is given by:

$$g(t) = (k + \mathbf{a}(t)^\intercal \boldsymbol{\delta})t + (m + \mathbf{a}(t)^\intercal \boldsymbol{\gamma}),$$

The combination of the two functions can represent any nonlinear trends of a time series effectively.
The seasonality is obtained by representing s(t) as a Fourier series. This is because the seasonality is a periodic function which can be represented by the linear combination of harmonic functions and can be solved by the Fourier transform.

$$s(t) = \sum_{n=1}^{N} \left( a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right)$$

The holiday factor is modelled as

$$h(t) = Z(t)\boldsymbol{\kappa}.$$

Where k is assumed $\boldsymbol{\kappa} \sim \text{Normal}(0, \nu^2)$. The model also assumes data changes in the window of particular holidays and adjusts the trend factor accordingly. In order to determine the parameters in the final model, Stans L-BFGS [6] is used to find a maximum posterior model.

The advantage of this model is it works very well on data which have seasonal and trend patterns. The model can also be finetuned by analysts. The model is more intuitive even for naïve users. Some of the disadvantages of the model are that it needs a large number of data points. Also, long-horizon forecasting can be volatile with automatic changepoint selection.

### 2.3 Long Short-Term Memory

In the recent times the neural networks have been used widely to solve variety of problems. A neural network consists of a set of nodes each capable of learning non-linear functions. This learning transforms the input data into a required output data. The process of learning involves adjusting the weights of these neural networks to reduce the errors at the output.

A Recurrent neural network [7] consists of network of nodes which perform computations and predict at each instance. At every time step t the output of the network depends on the past data represented by the hidden variable, and the current input at time t. Each RNN unit can be represented as below.

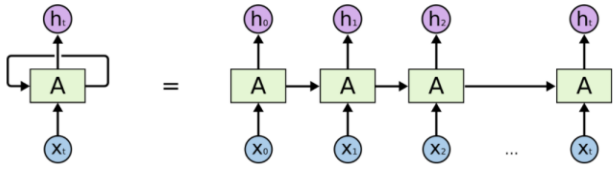
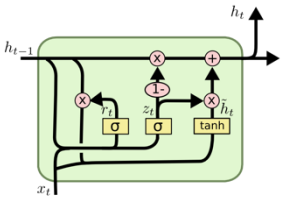



**Figure 1: Rolled out RNN at different time steps.**

Long Short-Term Memory [8] is one of the popular deep learning neural network architectures. LSTM is a special kind of Recurrent Neural Network with additional features which helps it to memorize the sequence of data better. Each LSTM is a set of cells or nodes, where the data stream is captured and stored. The cells resemble a transport line (the upper line in each cell) that connects out of one module to another helping the past data to be gathered into current data to make decisions. Due to the use of some gates in each cell, data in each cell can be modified (disposed, filtered, or added) for the next cells.

There are three types of gates namely –
Forget Gate: It is either 0 or 1 with 0 indicating ignore this data and 1 indicating keep the current data.
Memory Gate: It chooses which data has to be kept in the cells
Output Gate : This chooses what should be the output at the current time step




$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

**Figure 2: LSTM cell structure**

Hence, the gates, which are based on sigmoidal neural network layer, enable the cells either to let data pass through or disposed.
Since LSTM's can remember the data and the pattern it can be used for tasks like time series forecasting to forecast based on the trends and seasonality of the data.

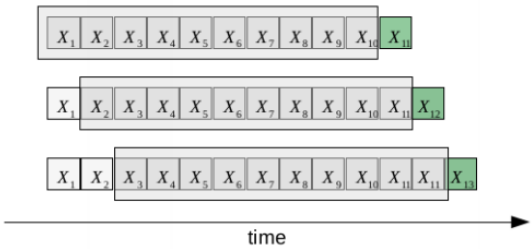



**Figure 3: Sliding window approach**

The authors in [9] found a sliding window method to train the LSTM networks for forecasting effectively. The output at every nth step depends on n-k states where k is regarded as a hyperparameter to be tuned.

The authors then built the architecture using keras and trained on the various datasets. They found that the outputs of LSTM were more accurate than the outputs of the ARIMA model. Also, it was observed that as the number of hidden nodes were increased the error rates came down. The authors in [10] also found that there was a high reduction in error rates (84% - 87% reduction in RMS Error) compared to ARIMA model. However, the LSTM model requires larger amount of data in order to produce accurate results. Hence LSTM models are best suited for larger datasets.

## 2.4 Generative Adversarial Neural Networks (GANS)

Generative Adversarial Neural Networks are a recent trend in deep learning community in recent days. GANS consists of pairing two neural network architectures both competing with each other in order to classify data as fake or real.

A GANS [11] architecture consists of two components: (a) A generator network and, (b)A discriminator. A generator network generates data at random based on the unsupervised data on which it is trained on. The data generated could be real or a fake. The discriminator network inputs a real data set and tries to classify the output of the generated network as real or fake. The main objective is to train the generator and discriminator in such a way the discriminator is unable to distinguish the fake and the real data.

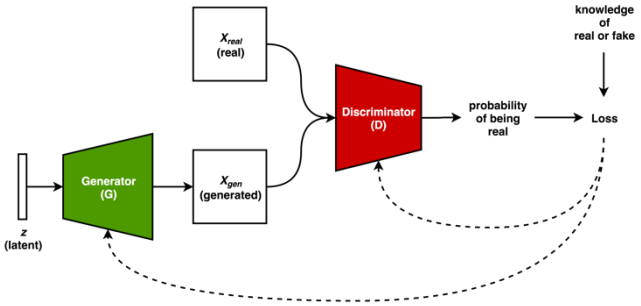



**Figure 4 GANS Architecture**

Mathematically the training GANS involves minimising the following loss function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim\rho_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim\rho_{\text{noise}}(z)}[\log(1 - D(G(z)))].$$

Here G is the generator, D is the discriminator. The generator G learns to transform a known probability distribution $\rho_z$ to the generator's distribution $\rho_G$ which resembles $\rho_{data}$

The authors in [12] solve the time series problem by using the GAN architecture. The authors use LSTM as a generator and RNN for a discriminator. Using LSTM as a generator helps to generate a time series close to the true model. The RNN is used as a discriminator as it is better in classification of features in time series data.

Using the generator network the LSTM predicts $A_{n+1}$ from values $[A_{n-k}, A_{n-k+1}, ....A_n]$ . The predicted value is then sent to the RNN.

discriminator. The discriminator classifies it as a fake or real value based on the actual real world data and the losses are back propagated.

It is important to note that training GANS is not easy as the model may not reach a global minima. Also the generator and discriminator should be coupled tightly i.e.., it should not be too easy for discriminator to classify the generator output and vice versa.

The authors found that by using this method they were able to achieve a very high accuracy after training for sufficient number of epochs on publicly available datasets. The model's error rate (RMSE) was found to be just 4.06% . The limitations of the model are that it requires high computations to be performed. Also the dataset has to be large enough to capture the patterns.

## 2.5 Multivariate timeseries using GRU

A multivariate timeseries is a timeseries which has more than one time-dependent variables. Each variable can be dependent on the other. More formally a multivariate data can be represented as:

$$X = (x_1, x_2, x_3, \dots x_T)^T$$

Where,
$X \in R^{T \times D}$, where $x_t \in R^D$ represents the $t^{th}$ observations of all variables and $x_t^d$ denotes the measurement of $d^{th}$ variable of $x_t$.

One of the major problems arising with multivariate data is missing values. As the number of dimensions are more the probability of a missing data of any variable increase and the entire row of data for that time stamp has to be removed. This reduces the multivariate data available for training. Hence multivariate time series models should be less susceptible to missing values and take them into consideration.

A GRU (Gated Recurrent Unit) [13] is one of the popular variants of the Recurrent Neural Network Unit. A GRU has two gates unlike an LSTM which has three gates. The GRU gates are:
Update gate: This gate is responsible for updating at time $Z^t$ at time step t.

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

Reset gate: This gate determined how much of the past information can be forgot in the current time step. The reset gate can be given by:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

Here each of the W's represent the weight matrix associated with the corresponding gate at timestep t and U's represent the weight at time t-1th step (hidden state).

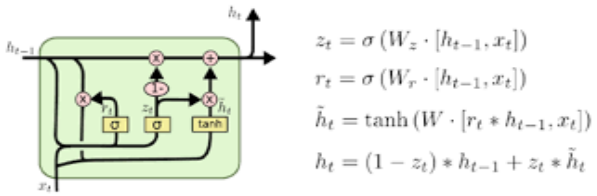



**Figure 5 GRU cell structure**

The authors in [14] conducted several models for forecasting multivariate data. Initially they three models were designed GRU-mean, GRU-forward, GRU-simple.

In GRU-mean the missing values are just replaced by the means. In GRU-forward the missing value is the same as the data in the previous time step. In GRU-simple the data was input into the network as it is, with additional data - a one hot vector mask representing if the data at that position was valid or not, and how long the given data is missing.

The authors used the aforesaid models as a baseline for comparison of GRU-D model. The GRU-D model, unlike the previous three models, has a decay factor which is a trainable parameter. Each of the variable has its own decay rate. The decay rate depends on the hidden state as well as the input data. The decay rates γ is modelled as

$$\gamma_t = \exp\{-\max(\mathbf{0}, \boldsymbol{W}_\gamma \boldsymbol{\delta}_t + \boldsymbol{b}_\gamma)\}$$

Here Wγ and bγ are model parameters that are trained jointly with all the other parameters of the GRU. The decay factor converges to the mean of the variable at infinity. Mathematically it can be represented as below:

$$x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d)\gamma_{x_t^d} x_{t'}^d + (1 - m_t^d)(1 - \gamma_{x_t^d})\tilde{x}^d$$

Where,

$X_t^d$ is the last observation of the $d^{th}$ variable (t`< t) and $x\tilde{}_d$ is the empirical mean of the $d^{th}$ variable and γ is the rate of decay.

The model also computes the decay for the hidden state in order to get a richer meaning out of the missing state. Intuitively, this has the effect of decaying the extracted features (GRU hidden states) rather than input variables directly.

It is represented below

$$\boldsymbol{h}_{t-1} \leftarrow \boldsymbol{\gamma}_{\boldsymbol{h}_t} \odot \boldsymbol{h}_{t-1},$$

Where $h_{t-1}$ represents the hidden state at t-1th state.

The overall GRU-D update operations can be represented as:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + V_z m_t + b_z) \qquad r_t = \sigma(W_r x_t + U_r h_{t-1} + V_r m_t + b_r)$$
$$\tilde{h}_t = \tanh(W x_t + U(r_t \odot h_{t-1}) + V m_t + b) \qquad h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

The GRU-D was then experimented with publicly available datasets as well as synthetic datasets. It was found that GRU-D performs better than the GRU-mean, GRU-simple and GRU-forward. The accuracy of GRU-D was found to be ~85% while the GRU-simple, GRU-forward, and GRU-mean were found to be ~82% on various multivariate datasets.

## 3. APPLICATIONS

Some of the important applications of time series forecasting [15] are elucidated below:

**Time series forecasting for trend and seasonal patterns:** Most of the business data have a certain trends and seasonality in them. These are critical for taking any business decisions of the organisation. Hence time series forecast plays an important role for businesses helping them to estimate accurately of the future needs.

**Electricity Forecasting:** Electricity load forecasting is an important and crucial parameter to plan and operate various electrical needs. The magnitude of the load and the geographical location of electricity over different time periods are predicted. There can be three different time periods for load forecasting: Short-Term Forecasting (Duration : < 1 day ), Mid-Term Load Forecasting (Duration: 1 day to 1 year) and Long-Term Forecasting (Duration: 1 year to 10 years ). Such analysis helps to allocate or take some actions beforehand and mitigate the problems. Such analysis requires accurate forecasting of the needs.

**Stock Selection and Portfolio management:** In the recent era, the stock market and stock selection has become crucial in investments and decision making. The main task here is to predict the future of the stock market and manage the portfolios accordingly. Using the historical stock market data, many forecasting techniques can be used for predicting financial time series to mitigate the risk of reduction, and make expected gains on the investment and to construct portfolios accordingly. Many approaches such as the Neural Networks, Support vector Machines and Fuzzy Logic have been used for this purpose of stock selection and portfolio using the financial time series data. Recently deep learning techniques are used more for obtaining better results.

**Oil and petroleum forecasting**: Oil and petroleum products form the important sources of energy for various applications such as transportation. As the economies grow it is important to forecast [16] the needs of the population to help make decisions such as the amount of petroleum to be extracted, processed and exported. Such decisions are important for managing supply chains effectively.

## 4. CONCLUSION

The amount of time series data produced is growing exponentially every day. Using time series analysis helps various businesses to keep a check on their operations by enabling them to take better decisions. This article provides various algorithms that are popular in the recent era to solve the time series forecasting effectively. The paper also talks about few key areas where time series is important and is used widely. It also highlights various new algorithms in deep learning which solve the forecasting efficiently. The paper also provides shortcomings of various models which helps to choose a better model according to the requirements.

## 5. REFERENCES

[1] 1. Adhikari, Ratnadip & Agrawal, R.. (2013). An Introductory Study on Time series Modeling and Forecasting. 10.13140/2.1.2771.8084.

[2] 2. Hillmer, S. & Tiao, G.. (1982). An ARIMA-model-based approach to seasonal adjustment. Journal of The American Statistical Association - J AMER STATIST ASSN. 77. 63-70. 10.1080/01621459.1982.10477767.

[3] 3. Mélard, Guy & Pasteels, J.-M. (2000). Automatic ARIMA modeling including interventions, using time series expert software. International Journal of Forecasting. 16. 497-508. 10.1016/S0169-2070(00)00067-4.

[4] 4. Kokoszka, Piotr & Young, Gabriel. (2016). KPSS test for functional time series. Statistics. 50. 1-17. 10.1080/02331888.2015.1128937.

[5] 5. Taylor, Sean & Letham, Benjamin. (2017). Forecasting at Scale. The American Statistician. 72. 10.1080/00031305.2017.1380080.

[6] 6. Ibrahim, Mohd Asrul Hery & Mamat, Mustafa & Leong, Wah. (2014). BFGS method: A new search direction. Sains Malaysiana. 43. 1591-1597.

[7] 7. L. C. Jain and L. R. Medsker. 1999. Recurrent Neural Networks: Design and Applications (1st. ed.). CRC Press, Inc., USA.

[8] 8. Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735..

[9] 9. Azzouni, Abdelhadi & Pujolle, Guy. (2017). A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction.

[10] 10. Siami Namini, Sima & Tavakoli, Neda & Siami Namin, Akbar. (2018). A Comparison of ARIMA and LSTM in Forecasting Time Series. 1394-1401. 10.1109/ICMLA.2018.00227.

[11] 11. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair‡ , Aaron Courville, Yoshua Bengio Generative Adversarial Nets. 2014 NIPS

[12] 12. Koochali, Alireza & Schichtel, Peter & Dengel, Andreas & Ahmed, Sheraz. (2019). Probabilistic Forecasting of Sensory Data with Generative Adversarial Networks – ForGAN. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2915544.

[13] 13. Chung, Junyoung & Gulcehre, Caglar & Cho, KyungHyun & Bengio, Y.. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.

[14] 14. Che, Zhengping & Purushotham, Sanjay & Cho, Kyunghyun & Sontag, David & Liu, Yan. (2016). Recurrent Neural Networks for Multivariate Time Series with Missing Values. Scientific Reports. 8. 10.1038/s41598-018-24271-9.

[15] 15. G. Mahalakshmi, S. Sridevi and S. Rajaram, "A survey on forecasting of time series data," 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16), Kovilpatti, 2016, pp. 1-8.

[16] 16. López-Yáñez, Itzamá & Sheremetov, Leonid & González-Sánchez, Arturo & Ponomarev, Andrew. (2013). Time series forecasting: Applications to the upstream oil and gas supply chain. IFAC Proceedings Volumes (IFAC-PapersOnline). 10.3182/20130619-3-RU-3018.00526.