Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 99 (2021) 604–609

www.elsevier.com/locate/procedia

14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME '20

Evaluation of deep learning with long short-term memory networks for time series forecasting in supply chain management

Massimo Pacella[a,*], Gabriele Papadia[a]

[a]Department of "Ingegneria dell'Innovazione", University of Salento, Piazza Tancredi 7, 73100 Lecce – ITALY

* Corresponding author. Tel.: +39 0832 297812; fax: +39 0832 297825. *E-mail address:* massimo.pacella@unisalento.it

Abstract

Performance analysis and forecasting the evolution of complex systems are two challenging tasks in manufacturing. Time series data from complex systems capture the dynamic behaviors of the underlying processes. However, non-linear and non-stationary dynamics pose a major challenge for accurate forecasting. To overcome statistical complexities through analyzing time series, we approach the problem with deep learning methods. In this paper, we mainly focus on the long short-term memory (LSTM) networks for demand forecasts in supply chain management, where the future demand for a certain product is the basis for the respective replenishment systems. This study contributes to the literature by conducting experiments on real data to investigate the potential of using LSTM networks for final customer demand forecasting, and hence for increasing the overall value generated by a supply chain. Both forward LSTM and bidirectional LSTM (forward-backward) for short- and long-term demand prediction in supply chain management are considered in this study.

Keywords: Time series analysis; Deep learning; Long short-term memory; Bidirectional long short-term memory

1. Introduction

A supply chain consists of all the organizations, or business units within an organization, which are involved, directly or indirectly, in fulfilling the final customer demand.

The extension of a supply chain goes from the final customers through retailers, wholesalers, and distributors, back to the manufacturers and their component and raw material suppliers. Within the chain, there are flows of goods, services, information, and finances moving from raw materials or parts supplier to manufacturer, wholesaler, retailer, and consumer. One of the main developments in the last decades has been the introduction of the supply chain management (SCM) system, which allows coordinating these flows. The entire area of SCM has many different aspects. In this paper, we focus on the topic related to the final customer demand forecasting, which sets the entire supply chain in motion. (For an extensive review on this topic, refer to [1]).

Each product involved in a supply chain drives decisions regarding products to be purchased, purchase time, and quantities to be purchased using demand information from its respective customers. Actions taken by retailing organizations to respond to such demand, by having the necessary products and services in place to satisfy customers, involve the generation of demand at the previous level in the supply chain, at wholesalers or distributors, who ultimately respond by placing requests on manufacturers, and so on. This upstream flow of requests constitutes the transmission of information from one supply chain member to another. This information flow is complemented by a flow of materials/products downstream of the supply chain to satisfy these requests.

If the final customer demand were known with certainty well in advance, because it is constant, then the operation of a supply chain would be a simple backward scheduling problem. However, in many real work applications, demand is not known. The uncertainty associated with it poses

10.1016/j.procir.2021.03.081

difficulties in the SCM system. Therefore, demand forecasting is required, which allows reliable operations at low inventory costs throughout the entire supply chain.

Many challenges that influence demand forecasting. One challenge is the 'bullwhip effect' [1], which is the distortion of true final customer's demand that comes from forecasts not completely correct. This implies that any prediction based on it will increase variability and further distort the demand anticipated by each agent of the chain. Another challenge is the required frequency for forecasting, which varies considerably depending on the decision-making process. Retail inventory replenishments, for example, rely upon frequent short term forecasts, whereas aggregate sales planning may take place in a long term period.

In the literature, much effort has been devoted to the development and improvement of demand forecasting models in SCM. Common statistical linear methods such as exponential smoothing, regression models, which include various driver variables, and time-series have the important advantage of easy interpretation and implementation [2].

A time series is a sequence of observations at regular intervals in chronological order over a specific time period. Common techniques for modeling sequential data involve estimating some parameters for fitting a given time series model, such as Autoregressive (AR), Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA) [2]. Due to the complex nature of nonlinear patterns in their constructs, the time series of final customer demand cannot be accurately captured by common linear methods. When linear models fail to perform well in both training (in-sample fitting) and testing (out-of-sample forecasting), more robust nonlinear models should be considered.

To address challenges related to forecasting models, deep learning algorithms can be considered. Since deep learning algorithms can be employed to perform prediction and classification operations based on highly complex training data, they show superior performance in many areas of applications such as signal processing, speech recognition, and image classification. Due to the advances in deep learning, many scientific fields exploit deep learning algorithms to build efficient solutions to different kinds of problems [3]. Recent studies have also shown that exploring deep learning algorithms in business analytics, operations research [4], financial time series prediction [5], and supply chain demand forecasting [6], is an emerging area of research and a recommended solution for actual applications.

Recurrent neural network (RNN) is one of the techniques employed in time series prediction. RNN can remember preceding data inputs while using current data to learn network weights. Long Short-Term Memory (LSTM) is a special case of RNN, which was initially introduced in [7] to deal with long input sequences to improve the network ability to preserve previous network states and capture longer-term dependencies. Another form of RNN is the bidirectional LSTM (BLSTM). The preceding and succeeding input sequences can be used to exploit all input data to satisfy the best learning process performance. A BLSTM is usually used to capture more complex patterns in the time series.

In this paper, we describe the development of an LSTM-based system for demand forecast, which helped to improve supply chain management. Both the LSTM and BLSTM architectures for short- and long-term demand prediction in supply chain management are considered.

The remainder of the paper is organized as follows. Section 2, presents the current state-of-the-art, related background, and preliminaries. In Section 3, the mathematical background of the LSTM methodology is introduced. Section 4 provides a comparison of the deep learning techniques and analyzes their respective advantages and weaknesses in forecasting data. Finally, conclusions are provided in Section 5.

2. Preliminaries and related work

Time series analysis is a research area whose aim is to study the path observations of time series, build a model to describe the structure of data, and predict future values. This field of research has a great number of applications in business, economics, finance, and computer science [2]. Due to the importance of time series forecasting in many branches of applied sciences and applications, it is essential to build an effective model to improve forecasting accuracy. A variety of time series forecasting models have been presented in the literature.

Statistical linear models used in demand forecasting range from simpler moving averages to the exponential smoothing family or the ARIMA approach [2]. All of these statistical linear models have been commonly used in supply chain modeling and forecasting (see reference [8]). Differently from statistical linear models, deep learning algorithms allow arbitrary non-linear approximation functions derived (learned) directly from the data. This increased generality improves the potential to provide more accurate forecasts (though with an increased danger of over-fitting).

Deep learning algorithms generalize neural networks. A neural network consists of at least three layers: 1) an input layer, 2) hidden layers, and 3) an output layer. The number of features of the data set determines the number of units in the input layer. These units are connected through links to the units created in the hidden layer(s). The links carry some weights for every unit in the input layer. The weights basically play the role of a decision-maker to decide which signal, or input, may pass through and which may not. A neural network basically learns by adjusting the weight for each link created in the hidden layer(s).

2.1. Recurrent Neural Network

A recurrent neural network (RNN) is a special case of a neural network where the objective is to predict the next step in the sequence of observations, for previous steps observed in the sequence. The idea behind RNNs is to make use of sequential observations and learn from the earlier stages to forecast future trends. In the earlier stages, data need to be remembered when guessing the next steps.

In RNNs, the hidden layers act as internal storage for storing the information captured in earlier stages of reading sequential data. RNNs are called ‘recurrent’ because they perform the same task for every element of the sequence, with the characteristic of utilizing information captured earlier to predict future unseen sequential data. The major challenge with a typical generic RNN is that these networks remember only a few earlier steps in the sequence and thus are not suitable for remembering longer sequences of data. This challenging problem is solved using the ‘memory line’ introduced in the LSTM recurrent network.

2.2. Long Short-Term Memory

LSTM is an RNN with additional features to memorize the sequence of data. An LSTM is a set of connected memory cells, where the data streams are captured and stored. Each memory cell connects out to another one conveying data from the past and gathering them for the present elaboration. Due to the use of some gates in each memory cell, data can be deleted, filtered, or added for the next cells. Each gate is a sigmoid unit and yields numbers in the range between zero and one. Three types of gates are involved in each memory cell to control the state of the cell.

1. Input Gate: chooses which new data need to be stored in the cell. Specifically, the input information will be stored in the cell when the input gate records high activation.

2. Output Gate: decides what will be yield out of each cell. The yielded value will be based on the cell state along with the filtered and newly added data. In practice, if the output gate records high activation, then it will release the stored information in the cell to the next one.

3. Forget Gate: outputs a number between 0 and 1, where 1 shows ‘completely keep this’; 0 implies ‘completely ignore this.’ In practice, the stored information will be cleared if the forget gate records high activation.

3. Mathematical Background

A memory cell has a state, say $c(t)$, at the time of index t. The information that flows in and out the cell is controlled by three gates. Each gate is characterized by a state, namely: input gate $i(t)$, output gate $o(t)$, and forget gate $f(t)$. Each gate receives the same input, the newly added vector of data $x(t)$ at time instant *t*, and the previous vector of hidden states $h(t-1)$. The three gates are sigmoid units according to the standard logistics sigmoid function defined as follows:

$$\sigma(z)=\left(1+\exp(-z)\right)^{-1}. \tag{1}$$

The input gate $i(t)$ controls the input information flowing into the memory cell, which derives the following:

$$i(t)=\sigma\left(W_{xi}x(t)+W_{hi}h(t-1)+W_{ci}c(t-1)+b_i\right). \tag{2}$$

Forget gate $f(t)$ controls the forgetting information of the cell, where:

$$f(t)=\sigma\left(W_{xf}x(t)+W_{hf}h(t-1)+W_{cf}c(t-1)+b_f\right). \tag{3}$$

The memory cell is updated by moderated input features and the partial forgetting of the previous cell, where the input features are calculated by combining newly added data $x(t)$ at time instant *t* and the previous hidden state $h(t-1)$ and by using a hyperbolic tangent layer. This yields to:

$$c(t)=f(t)c(t-1)+i(t)\tanh\left(W_{xc}x(t)+W_{hc}h(t-1)+b_c\right). \tag{4}$$

The output gate $o(t)$ controls the output information flowing out of the cell, which derives the following:

$$o(t)=\sigma\left(W_{xo}x(t)+W_{ho}h(t-1)+W_{co}c(t)+b_o\right). \tag{5}$$

Ultimately, the hidden output state $h(t)$ is calculated by output gate $o(t)$ and memory cell state $c(t)$, where:

$$h(t)=o(t)\tanh\left(c(t)\right). \tag{6}$$

Thus, the univariate output of LSTM $y(t)$ is computed as:

$$\hat{y}(t)=\sigma\left(W_{hy}h(t)+b_y\right). \tag{7}$$

The W_* terms denote weight values. In particular, W_{xi}, W_{xf}, W_{xo} and W_{xc} are the input weight values; W_{hi}, W_{hf}, W_{ho} and W_{hc} are the recurrent weight values; W_{hi} represent the hidden output weight value. Finally, b_* terms represent the corresponding bias values. The actual values for these parameters are defined during the training of the model.

The model of a baseline LSTM, which is described by previous equations from (1) to (7), can process data only in the forward direction of the time series. In many applications, we may need to consider dependencies/correlations in both forward and backward. In this case, bidirectional LSTM (BLSTM) is introduced, which can process data in both directions with two separate hidden layers. Both hidden layers are connected to the same output layer. The major difference to a baseline LSTM is that a BLSTM computes the forward hidden sequence and the backward hidden sequence separately, then the output layer is computed by iterating the backward layer from $t=T$ to $t=1$ and the forward layer from $t=1$ to $t=T$.

4. Experimental study

In this section, LSTM and BLSTM are both evaluated on the actual time series of final customer’s demand for 10 specific products in a market. We first describe some preliminary data elaboration and error measures used in our experiments.

4.1. Variance stabilization using power transformation

In our study, we considered a data set of 10 series, each related to the final customer's demand for a specific product in a market, and for three years 2015, 2016, and 2017. Each time series consisted of the monthly sales of the product. Data lengths are equal to 36 (35 in three specific cases, which presented one missing value each). Given such a short length, a preliminary step was required in data, a variance stabilization by a power transformation.

Although power transformations may alter the original non-linearity in a time series, in our preliminary experiments, we found that such a step is required to enable the deep learning algorithm to learn dynamics in data, as trend or periodicity, which otherwise could be difficult to identify in such small-length time series. A common type of transformation for variance stabilization is the power family of transformations [2] defined in the following equation (8).

$$w(t)=\begin{cases} \log\left(y(t)\right) & \lambda=0 \\ \left(y^{\lambda}(t)-1\right)/\lambda & \lambda\neq 0 \end{cases}. \tag{8}$$

A difficulty with this transform is the choice of a proper value for the parameter λ, which could be suitable for variance stabilization. The procedure implemented in our study to choose λ automatically is the procedure in reference [9].

However, in preliminary experiments, we found that this procedure has its shortcomings, and the parameter λ is difficult to choose in practice. From equation (8), it can be observed that the transform resembles, depending on its parameter λ, the logarithm or the identity in its most extreme case ($\lambda=0$ or $\lambda=1$, respectively). In particular, the logarithm is a strongly non-linear transformation that should be used with caution, as small differences in log space may result in large differences in the original space, and therewith the training phase can yield sub-optimal results.

Therefore, we use the power transform in (8) through a more conservative approach, consisting in forcing λ to be equal to $\lambda=1$, i.e., avoiding the transformation of data, when the optimal value of λ, which resulted from the procedure in [9] for variance stabilization, was close to zero (specifically, when $\lambda\leq 0.005$).

4.2. Error measures

Different choices to evaluate forecasts exists in the forecasting literature. Overviews are provided in references [10] and [11]. In our study, to compare time series modeling performance, we calculate 4 measures.

The forecast error is $e(t)=y(t)-\hat{y}(t)$, regardless of how the forecast was produced. Here, $y(t)$ denotes the observation at the time of index t and $\hat{y}(t)$ is the respective forecast. This forecast error is on the same scale as the data. Hence, accuracy measurements based on $e(t)$ is scale-dependent.

The most commonly used scale-dependent metrics are based on absolute errors or squared errors are the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE). They are defined respectively as follows:

$$MAE=\frac{1}{h}\sum_{t=1}^{h}\left|e(t)\right|, \tag{9}$$

$$RMSE=\sqrt{\frac{1}{h}\sum_{t=1}^{h}\left|e^{2}(t)\right|}, \tag{10}$$

where h denotes the number of data points in the test set.

To compare forecast performance between different data series, percentage errors should be used instead, having the advantage of being scale independent. The most commonly used metric is the symmetric Mean Absolute Percentage Error (*sMAPE*) defined as follows.

$$sMAPE=\frac{200}{h}\sum_{t=1}^{h}\left(\frac{\left|e(t)\right|}{\left|y(t)\right|+\left|\hat{y}(t)\right|}\right). \tag{11}$$

However, if the actual value $y(t)$ is zero, the forecast $\hat{y}(\mathrm{t})$ is likely to be close to zero too. In this case, the measure in (1) will involve division by a number close to zero. Additional shortcomings of measures such as the *sMAPE* are that they are skewed, they have lack of robustness, lack of interpretability. To address some of these issues, we use as a second evaluation metric in our experiments the Mean Absolute Scaled Error (*MASE*), as proposed in [11].

MASE is a scale-independent error measure, which also offers interpretability, as it measures the forecasting accuracy relative to the seasonal *'naïve'* forecast error. We use *MASE* in the following definition

$$MASE=\frac{1}{h}\frac{\sum_{t=1}^{h}\left|e(t)\right|}{\frac{1}{n-M}\sum_{t=M+1}^{n}\left|y(t)-y(t-M)\right|}, \tag{12}$$

where n denotes the number of data points in the training set of a time series. The seasonal period of a time series is represented by M. That is, assuming the time series is seasonal, $MASE<1$ means that on average the method performs better than the *'naïve'* seasonal forecast computed on the training data while $MASE>1$ indicates that the method performs worse.

4.3. Implementation

LSTM and BLSTM were both implemented in MATLAB using the DEEP LEARNING TOOLBOX. We trained LSTM and BLSTM models using the stochastic gradient descent with momentum algorithm, learning rate equal to 0.008, and learning rate drop factor equal to 0.2 applied each 40 training epochs. We used a mini-batch size of 36, and normalization for each vector of a sequence by using the mean and standard deviation computed from the training set. Training and testing were done on a single CPU (Intel Core I7, 2.5 GHz, 16 GB memory). The training time for each time series of 35-36 data averages 7 sec. in the case of LSTM and 11 sec. in the case of BLSTM. The maximum number of epochs was equal to 360. An example of a MATLAB window showing training progress for deep learning is reported in Fig. 1.

Table 1 summarizes the parameters of the LSTM [BLSTM] implemented. According to our experiments, multiple layers should be preferred rather than one layer. In our study, we implemented a double-layer stacked LSTM [BLSTM] network to guarantee the predictive effect.

The model adopts the traditional input-hidden-output structure of the neural network. In the input layer, the number of units equals the data dimension of each input. The hidden layer has the number of units that are arbitrarily determined (200 units for each hidden layer, in our case study). Since the problem solved is a single-valued prediction problem, the output layer was set to a single neuron. An additional regression layer, which computes the half-mean-squared-error loss for regression problems, was added in the model to normalize the response and hence stabilizing and speeding up the training of the LSTM algorithm.

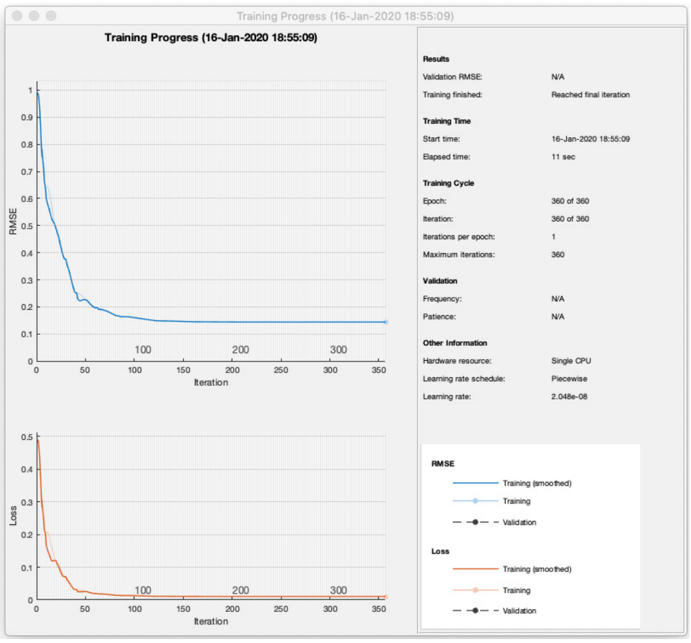
LSTM networks require sequences of input features for training. We opted for a sequence of 10 input features. Hence the number of units in the input layer is equal to 10. Specifically, at each time of index t, we used as input a sequence of 10 values of time series with indexes $\left[(t-14),\cdots,(t-11),(t-6),\cdots,(t-1)\right]$. The target value at each time of index t was set equal to $x(t)$. As a consequence, the LSTM algorithm provided a forecast $\hat{y}(t)$ only for $t>14$.

Training performance (in-sample fitting) are in Table 2. The benchmark method here is a common Exponential Moving Average (ExpMA) approach [2] of lag 12. From Table 2, the three methods implemented, ExpMA, LSTM, and BLSTM, have a $MASE<1$ meaning that each of them performs better than the *'naïve'* seasonal forecast computed on the dataset. Given that each time series represented monthly sales, in computing $MASE$ the seasonal period was set equal to $M=12$ From Table 2, it can be observed that BLSTM presents better training results, which in some cases resulted in a null forecast error (products no. 2, 3, 4, 6, 8).

Testing performance (out-of-sample forecasting) is in Table 3, where only LSTM and BLSTM methods are included in the comparison for brevity. Both LSTM and BLSTM have $MASE<1$ for each data set (for testing data, the seasonal period was set to $M=1$). Results in Table 3 show that LSTM and BLSTM algorithms do not over-fit training data set, despite the high number of parameters of them. For testing, the BLSTM algorithm presents better performance when compared to a baseline LSTM one. Therefore, BLSTM appears as the preferred approach for consumers' demand modeling and forecasting.

5. Conclusions

Nowadays, large quantities of time series data are available in many application cases. One promising approach in these applications is LSTM, which is a special type of RNN, for modeling a time series and forecasting future values.



Results
Validation RMSE: N/A
Training finished: Reached final iteration

Training Time
Start time: 16-Jan-2020 18:55:09
Elapsed time: 11 sec

Training Cycle
Epoch: 360 of 360
Iteration: 360 of 360
Iterations per epoch: 1
Maximum iterations: 360

Validation
Frequency: N/A
Patience: N/A

Other Information
Hardware resource: Single CPU
Learning rate schedule: Piecewise
Learning rate: 2.048e-08

Fig. 1. An example of a MATLAB window for deep learning progress.

Table 1. LSTM [BLSTM] implemented by the Deep Learning toolbox.

Level	*Type*	*Learnable*	*States*
1	Sequence Input with 10 dimensions	-	-
2	LSTM [BLSTM] with 200 hidden units	Input Weights 800[1600]x10 Recurrent Weights 800 [1600]x200 Bias 800[1600]x1	Hidden State 200[400]x1 CellState 200[400]x1
3	LSTM [BLSTM] with 200 hidden units	Input Weights 800[1600]x10 Recurrent Weights 800[1600]x200 Bias 800[1600]x1	Hidden State 200[400]x1 CellState 200[400]x1
4	1 fully connected layer	Weights 1x200[400] Bias 1x1	-
5	Regression Output	-	-

Table 2. Training performance of LSTM and BLSTM for 10 demand time series of length 36 and 35 data. In bold the outperforming method.

Exp.	*Method*	*RMSE*	*MAE*	*sMAPE*	*MASE*
Prod. 1	ExpMA	3,046	2,560	34,545	0,723
	LSTM	0,853	0,545	8,457	0,176
n = 36	**BLSTM**	**0,564**	**0,318**	**5,023**	**0,103**
Prod. 2	ExpMA	2,072	1,625	40,824	0,719
	LSTM	0,369	0,136	3,838	0,055
n = 35	**BLSTM**	**0,000**	**0,000**	**0,000**	**0,000**
Prod. 3	ExpMA	2,425	1,800	32,462	0,559
	LSTM	0,769	0,409	6,272	0,141
n = 35	**BLSTM**	**0,000**	**0,000**	**0,000**	**0,000**
Prod. 4	ExpMA	2,366	2,080	25,059	0,846
	LSTM	0,879	0,591	7,392	0,191
n = 36	**BLSTM**	**0,000**	**0,000**	**0,000**	**0,000**
Prod. 5	ExpMA	5,396	4,720	29,086	0,558
	LSTM	2,680	2,091	13,001	0,188
n = 36	**BLSTM**	**2,477**	**2,045**	**13,180**	**0,184**
Prod. 6	ExpMA	2,577	2,160	41,272	0,720
	LSTM	0,213	0,045	0,826	0,014
n = 36	**BLSTM**	**0,000**	**0,000**	**0,000**	**0,000**
Prod. 7	ExpMA	3,027	2,520	22,341	0,720
	LSTM	0,477	0,227	1,709	0,058
n = 36	**BLSTM**	**0,213**	**0,045**	**0,293**	**0,012**
Prod. 8	ExpMA	3,240	2,750	45,390	0,821
	LSTM	0,816	0,476	9,773	0,165
n = 35	**BLSTM**	**0,000**	**0,000**	**0,000**	**0,000**
Prod. 9	ExpMA	2,209	1,680	42,760	0,568
	LSTM	0,213	0,045	1,010	0,017
n = 36	**BLSTM**	**0,213**	**0,045**	**1,010**	**0,017**
Prod. 10	ExpMA	3,644	2,960	20,525	0,646
	LSTM	1,477	1,182	8,767	0,303
n = 36	**BLSTM**	**1,000**	**0,818**	**5,795**	**0,210**

In this study, LSTM has experimented with demand forecasts using 10 actual data sets related to a market. Performances of both training and testing phases have been evaluated. The results indicate that LSTM is a competitive method, it effectively models the non-linearity of the time series and therewith it appears being able to outperform state-of-the-art linear forecasting method. The BLSTM, which can process data in both forward and backward directions of the time series, augments the accuracy of the baseline LSTM approach in all of the cases considered in our study.

Acknowledgments

This work has been funded by Puglia Region (Italy) – Project ‘Cooperative Supply Chain’.

Table 3. Testing performance of LSTM and BLSTM for 10 demand time series of length 12 data. In bold the outperforming method.

Exp.	*Method*	*RMSE*	*MAE*	*sMAPE*	*MASE*
Prod. 1	LSTM	1,190	0,750	17,631	0,196
h = 12	**BLSTM**	**0,816**	**0,500**	**12,611**	**0,131**
Prod. 2	LSTM	0,707	0,500	12,381	0,262
h = 12	**BLSTM**	**0,408**	**0,167**	**3,492**	**0,087**
Prod. 3	LSTM	1,414	1,000	18,792	0,478
h = 12	**BLSTM**	**0,408**	**0,167**	**2,797**	**0,080**
Prod. 4	LSTM	1,443	1,250	13,791	0,550
h = 12	**BLSTM**	**0,764**	**0,583**	**5,875**	**0,257**
Prod. 5	LSTM	3,651	2,167	8,486	0,681
h = 12	**BLSTM**	**2,291**	**1,917**	**7,057**	**0,602**
Prod. 6	LSTM	0,645	0,417	11,886	0,127
h = 12	**BLSTM**	**0,289**	**0,083**	**5,556**	**0,025**
Prod. 7	LSTM	1,080	0,833	6,673	0,655
h = 12	**BLSTM**	**0,500**	**0,250**	**2,185**	**0,196**
Prod. 8	LSTM	0,707	0,500	8,757	0,108
h = 12	**BLSTM**	**0,577**	**0,333**	**3,762**	**0,072**
Prod. 9	LSTM	1,041	0,750	18,479	0,516
h = 12	**BLSTM**	**0,645**	**0,417**	**9,498**	**0,286**
Prod. 10	LSTM	1,414	1,167	10,103	0,377
h = 12	**BLSTM**	**1,155**	**1,000**	**8,656**	**0,324**

References

[1] Syntetos AA, Babai Z, Boylan JE, Kolassa S, Nikolopoulos K. Supply chain forecasting: Theory, practice, their gap and the future. European J. of Operational Research 2016;252:1-26.

[2] Montgomery DC, Jennings CL, Kulahci M. Introduction to Time Series Analysis and Forecasting. 2nd edition. Hoboken: Wiley; 2015.

[3] Hatcher WG, Yu W. A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends. IEEE Access 2018;6:24411-32.

[4] Fischer T, Krauss C. Deep learning with long short-term memory networks for financial market predictions. European J. of Operational Research 2018;270:654-69.

[5] Kraus M, Feuerriegel S, Oztekin A. Deep learning in business analytics and operations research: models, applications and managerial implications. European . of Operational Research 2020;281:628-41.

[6] Weng T, Liu W, Xiao J. Supply chain sales forecasting based on lightGBM and LSTM combination model. Industrial Management and Data Systems 2019;120:265-279.

[7] Hochreiter S, Schmidhuber J. Long Short-Term memory. Neural Computation 1997;9:1735-80.

[8] Svetunkov I, Boylan JE. State-space ARIMA for supply-chain forecasting. Int. J. of Production Research 2020;58:818-27.

[9] Guerrero VM, Perera R. Variance stabilizing power transformation for time series. J. of Modern Applied Statistical Methods 2004;3:357-69.

[10] Hyndman RJ, Koehler AB. Another look at measures of forecast accuracy. Int. J. of Forecasting 2006;22:679-88.

[11] Davydenko A, Fildes R. Measuring forecasting accuracy: the case of judgmental adjustments to SKU-level demand forecasts. Int. J. of Forecasting 2013;29:510-22.