Semiconductor Demand Forecasting using Long Short-term Cognitive Networks

Isel Grau[1,2], Michiel de Hoop[1], Ana Glaser[3], Gonzalo Nápoles[4], and Remco Dijkman[1,2]

[1] Information Systems Group, Eindhoven University of Technology, The Netherlands
[2] Eindhoven Artificial Intelligence Systems Institute, Eindhoven University of Technology, The Netherlands
[3] Supply Chain Systems & Processes, NXP Semiconductors, The Netherlands
[4] Department of Cognitive Science & Artificial Intelligence, Tilburg University, The Netherlands
i.d.c.grau.garcia@tue.nl

Abstract. Demand forecasting plays a paramount role in effective supply chain management, giving a business the opportunity to optimize production and improve stock management and operation. Statistical techniques are widely and fittingly used in these prediction problems, however, recent advancements in machine learning techniques are worth exploring. In this paper, we use Long Short-term Cognitive Networks (LSTCN) for forecasting multivariate time-series data describing the demand for six different types of products of a semiconductor company. The results of the experiments show that LSTCN is able to outperform state-of-the-art techniques for three out of the six tested datasets. The results also show that LSTCN is able to leverage the inclusion of additional data and more accurately forecast peaks and valleys in demand. These results and the interpretability potential of LSTCN led to the integration of this algorithm into the suite of forecasting models available in the company's forecasting system.

Keywords: semiconductor demand · forecasting · long short-term cognitive networks · recurrent neural networks · multivariate time series

1 Introduction

In the last decades, companies have invested heavily in digitizing their historical data and processes. However, digitization should not be limited to archiving and transferring data in static information systems. The digital transformation of a company has the potential to change business models and business processes to create more efficiency, more value, or even new services [4]. In particular, there is still a lot of room for leveraging historical data to support the decision-making processes. For example, data analytics and forecasting can help businesses to better anticipate the demand and sales of their products, impacting their policies and plans. Accurately predicting the demand can help businesses to reduce costs,

reduce product cycle times, and optimize inventory management, giving them a competitive advantage [6].

Forecasting demand represented as time series data has been widely explored in the literature using several statistical approaches [18]. Some examples of successful applications include the use of the exponential smoothing method for predicting the short-term, mid-term, and long-term demand trends in the supply chain of the chemical industry [2], the demand for perishable products in food retails with Holt-Winters and autoregressive integrated moving average (ARIMA) models [3], and the consumer interest on a product based on web search traffic using ARIMA [5]. Among these techniques, ARIMA usually outperforms the rest in terms of precision and accuracy [14]. However, the demand for products is caused by several factors which are often too complex to describe by linear models based on univariate data.

More complex machine learning models such as neural networks offer an alternative for multivariate, highly non-linear data. For example, Silva et al. [15] predict the demand for consumer products in supermarkets using multilayer neural networks and support vector regression methods. Recurrent neural networks (RNN) such as the long-short term memory (LSTM) architecture or bidirectional RNN have also proven useful in demand forecasting [17] by capturing long-term dependencies in the time sequence through their feedback loops. Alternatively, tree-based regression models such as random forest (RF) and extra trees (ET) have shown significant performance, especially considering their potential for interpretability in the form of feature importance compared to deep learning models [1, 13]. The hybridization of neural networks with traditional statistical techniques is also a promising direction [16]. However, the use of RNN architectures is limited in practice due to their high computational complexity, their need for a large amount of data to learn accurate models, and their training time, which can increment financial and environmental costs [7, 8].

To mitigate the shortcomings of RNN, Long Short-term Cognitive Networks (LSTCNs) [10] were proposed as a computationally efficient and transparent alternative for forecasting multivariate time series. LSTCN is able to outperform state-of-the-art RNNs in terms of accuracy and training time for several case studies with multivariate time series. In this paper, we investigate whether LSTCN is a good candidate algorithm for solving the forecasting tasks of the semiconductor company NXP. The semiconductor company NXP is a major semiconductor designer and manufacturer headquartered in Eindhoven, Netherlands. NXP uses an ensemble of 52 statistical and machine learning models to generate forecasts for mass market customers served via their distribution channels. The forecasting team at NXP trains these models every month and selects the best-performing one for short-term and long-term predictions. Therefore, improving their predictions by integrating other efficient machine learning techniques into its ensemble of forecasting models is of great interest. This company estimates that for every 10% improvement in forecast accuracy, they obtain a cost reduction of 1 day of total inventory on hand and 15% in annual carrying

costs. Therefore, improved forecast accuracy results in significant cost savings and downstream benefits for the planning and strategy organization.

In our experiments, we use LSTCN for forecasting multivariate time-series data describing the demand for six different families of products. The results of the experiments show that LSTCN is able to outperform state-of-the-art techniques for three out of the six tested datasets. Furthermore, LSTCN outperforms NXP's current ensemble of models in one dataset. The results also show that LSTCN is able to leverage the inclusion of additional data about successful design opportunities and more accurately forecast peaks and valleys in demand. These results and the potential of LSTCN for interpretability and knowledge injection led to the integration of this algorithm into the suite of forecasting models available in the company's forecasting system.

The rest of this paper is organized as follows. Section 2 presents the theoretical aspects of the LSTCN method, including its construction and training processes. Section 3 presents the case study of NXP demand forecasting and compares the performance of LSTCN with other state-of-the-art techniques. Section 4 discusses the results and concludes the paper.

2 Long Short-term Cognitive Networks

Before introducing the LSTCN model, let us formalize the forecasting problem to be addressed. A multivariate time series can be defined as a sequence $\{X^{(t)}\}_{t=1}^{T} = \{X^{(1)}, X^{(2)}, \ldots, X^{(T)}\}$ of vectors of M continuous variables, such that $X^{(t)} = [x_1^{(t)}, x_2^{(t)}, \ldots, x_M^{(t)}]$ and $t \in \{1, 2, \ldots, T\}$ where $T \in \mathbb{N}$ is the number of observations. The forecasting problem consists of predicting the following $L < T$ steps ahead, from the known observations.

2.1 Obtaining the time patches

The LSTCN model operates on time patches, which can be defined as temporal pieces of data resulting from partitioning the time series. Let us assume that $X \in \mathbb{R}^{M \times T}$ is a dataset comprising a multivariate time series. Such time patches can be disjoint or overlapped. Next, we briefly explain how to obtain disjoint time patches given a multivariate time series.

- First, we encode the multivariate time series X as a set of Q tuples with the form $(X^{(t-L)}, X^{(t+L)})$, such that $t - L > 0, t + L \leq T$. This suggests that the LSTCN model will be symmetric, meaning that it will use the last L observations to predict the next L values.
- Secondly, each component in the tuple is flattened as a $Q \times (M(L + L))$ matrix from which we create a partition $P = \{P^{(1)}, \ldots, P^{(k)}, \ldots, P^{(K)}\}$ of time patches such that $P^{(k)} = (P_1^{(k)}, P_2^{(k)})$ is the k-th time patch involving two data pieces $P_1^{(k)}, P_2^{(k)} \in \mathbb{R}^{C \times (M \times L)}$ where C is the number of instances in a given time patch. It should be mentioned that we can trim the sequence by removing the earliest observations to ensure that the length and time patches to be created are compatible.

2.2 Construction and reasoning

An LSTCN model [10] can be defined as a recurrent neural network composed of a collection of STCN blocks that allow for sequential learning. In other words, each STCN block is a two-layer neural network that implements shallow learning to process a specific time patch. Which is more important, each block passes the knowledge learned in the previous iteration to the previous STCN model as prior knowledge defined by a weight matrix.

The STCN block was first introduced in [12] and later expanded in [11]. This neural computation system involves two primary processing gates: the *input gate* and the *output gate*. The input gate operates the prior knowledge matrix $W_1^{(k)} \in \mathbb{R}^{N\times N}$ with $P_1^{(k)} \in \mathbb{R}^{C\times N}$ and the prior bias matrix $B_1^{(k)} \in \mathbb{R}^{1\times N}$ such that $N = (M \times L)$. Both matrices $W_1^{(k)}$ and $B_1^{(k)}$ are transferred from the previous block and remain locked during the learning phase to be performed in that STCN block [9]. The result of the input gate is a temporal state $H^{(k)} \in \mathbb{R}^{C\times N}$ with the outcome that the block would have produced for $P_1^{(k)}$ if no further learning process would have been performed to obtain $P_2^{(k)}$. Such an adaptation is done in the output gate where the temporal state is operated with the learnable weight matrices $W_2^{(k)} \in \mathbb{R}^{N\times N}$ and $B_2^{(k)} \in \mathbb{R}^{1\times N}$. Equations (1) and (2) display the reasoning process of this model in the k-th iteration,

$$\hat{P_2}^{(k)} = f\left(H^{(k)}W_2^{(k)} \oplus B_2^{(k)}\right) \tag{1}$$

and

$$H^{(k)} = f\left(P_1^{(k)}W_1^{(k)} \oplus B_1^{(k)}\right) \tag{2}$$

such that

$$f(x) = \frac{1}{1+e^{-x}} \tag{3}$$

where $\hat{P_2}^{(k)}$ is the predicted output, while $\oplus$ performs a matrix-vector addition by operating each row of a given matrix with a vector. Of course, these matrices should have the same number of columns. Moreover, for the sake of simplicity, we assume that outputs should be in the $[0, 1]$ interval.

Now, we are in a position to introduce the LSTCN architecture, which consists of a sequence of STCN blocks, each processing a time patch and passing the learned weights to the following block as prior knowledge. Figure 1 portrays this idea for a time series involving three time patches.

In this model, we use an aggregation operator to merge the knowledge characterizing the current STCN block when building the prior knowledge matrix. This research will study the aggregation operators formalized below to produce the knowledge to be transferred to the next block.

– *Linear*. This operator transfers the learned weights in the current block to the next one without further modification.

$$W_1^{(k)} = W_2^{(k-1)}, k-1 \geq 0$$

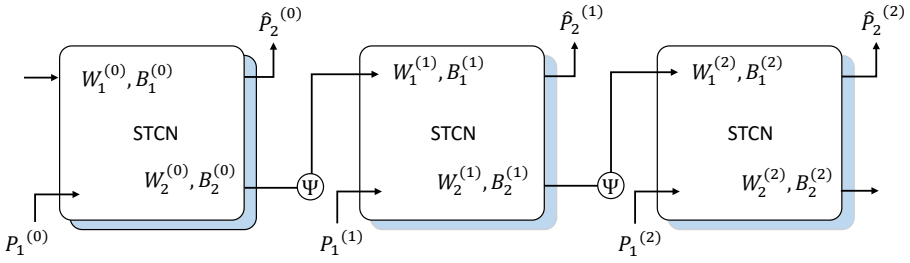


Fig. 1: LSTCN model with three STCN blocks. The weights learned in the current block are transferred to the following as a prior knowledge matrix.

$$B_1^{(k)} = B_2^{(k-1)}, k-1 \geq 0$$

– *Nonlinear*. This operator applies a nonlinear transformation to $W_2^{(k-1)}$ and $B_2^{(k-1)}$ before passing them to the next block.

$$W_1^{(k)} = \Psi(W_2^{(k-1)}), k-1 \geq 0$$

$$B_1^{(k)} = \Psi(B_2^{(k-1)}), k-1 \geq 0$$

– *Average linear*. This operator aggregates $W_1^{(k-1)}$ and $W_2^{(k-1)}$, and $B_1^{(k-1)}$ and $B_2^{(k-1)}$ before passing them to the next block.

$$W_1^{(k)} = W_1^{(k-1)} + W_2^{(k-1)}, k-1 \geq 0$$

$$B_1^{(k)} = B_1^{(k-1)} + B_2^{(k-1)}, k-1 \geq 0$$

– *Average nonlinear*. This operator is similar to the previous one; however, it applies a nonlinear transformation to the averaged matrices.

$$W_1^{(k)} = \Psi(W_1^{(k-1)} + W_2^{(k-1)}), k-1 \geq 0$$

$$B_1^{(k)} = \Psi(B_1^{(k-1)} + B_2^{(k-1)}), k-1 \geq 0$$

such that $\Psi(x) = tanh(x)$ is the nonlinear aggregation function. The best operator will be decided through hyper-parameter tuning.

2.3 Shallow parameter learning

Similarly to other gated recurrent neural networks, the learning process used by LSTCN models takes place inside each STCN block, even considering the frozen weights input as prior knowledge.

Overall, the learning task can be summarized as follows. Given a temporal state $H^{(k)}$ resulting from the input gate and the block's expected output $P_2^{(k)}$, we

need to compute the matrices $W_2^{(k)} \in \mathbb{R}^{N\times N}$ and $B_2^{(k)} \in \mathbb{R}^{1\times N}$. These matrices are estimated using shallow learning by solving a linear equation system that adapts the temporal state to the expected output. Equation (4) displays the deterministic learning rule solving this regression problem,

$$\begin{bmatrix} W_2^{(k)} \\ B_2^{(k)} \end{bmatrix} = \left(\left(\Phi^{(k)} \right)^\top \Phi^{(k)} + \lambda \Omega^{(k)} \right)^{-1} \left(\Phi^{(k)} \right)^\top f^- \left(P_2^{(k)} \right) \tag{4}$$

where $\Phi^{(k)} = (H^{(k)}|A)$ such that $A_{C\times 1}$ is a column vector filled with ones, $\Omega^{(k)}$ denotes the diagonal matrix of $(\Phi^{(k)})^\top \Phi^{(k)}$, while $\lambda \geq 0$ denotes the ridge regularization penalty [10]. This deterministic learning rule assumes that the neuron's activation values inner layer are standardized. If needed, the predicted values can be adjusted back into their original scale.

We need to specify $W_1^{(0)}$ and $B_1^{(0)}$ in the first STCN block. These matrices can be obtained from a previous learning process, or they can be provided by domain experts. Since this information is not available, we fit a single STCN block without an intermediate state (i.e., $H^{(0)} = P_1^{(0)}$) on a smoothed representation of the whole (available) time series. The smoothed time series is obtained using the moving average method for a given window size.

3 Forecasting experiments

In this section, we will explore the performance of LSTCN in forecasting the demand for six groups of products from the semiconductor company NXP. First, we describe the characteristics of the datasets and the forecasting system currently used at the company. Next, we describe the experiments performed using LSTCN, the comparison with other state-of-the-art techniques, and discuss the results obtained.

3.1 Data description

For forecasting demand, the semiconductors company NXP divides its customers into two channel groups named Distribution Mass Market (DMM) and Distribution Fulfillment (DF), based on the size and frequency of their orders. The DMM channel contains the sales to customers that generally request relatively small and more frequent orders, while the DF channel contains the sales to bigger customers that request larger orders. Within these channel groups, two product aggregation levels are relevant for this research, the Major Article Group (MAG) and the 6TG levels. The 6TG level is the lowest level of aggregation besides the individual products, and the MAG level is the highest level of aggregation. The demand data is predicted at the 6TG level, i.e., a prediction is made for each product group. Afterward, the results are aggregated, and the performance is measured at the MAG level. The MAG level includes three article groups for each channel, resulting in six independent forecasting datasets, which were selected for this case study based on the availability of data. Table 1 summarizes

the organization, the number of products, and the total requested orders (in millions) for the datasets considered in the experiments.

Table 1: Characteristics of the datasets considered in the case study.

Channel Group	MAG level	6TG level	6TG products	Total Requested Orders
DMM	Group 1	Dataset 1	740	1152.86 M
	Group 2	Dataset 2	270	1081.28 M
	Group 3	Dataset 3	99	112.62 M
DF	Group 1	Dataset 4	368	619.29 M
	Group 2	Dataset 5	166	439.15 M
	Group 3	Dataset 6	59	108.37 M

The current forecasting system of the company is based on 52 models, including techniques such as ARIMA, moving average, and extra trees regression. Each month the models are trained on historical data and the best-performing ones are used to predict the short-term demand for the next four months and long-term demand for the months four to 18. The forecasting for the DF channel is conducted manually, using expert knowledge within the company, based on customer and sales information. The historical data comprises five time-distributed variables making the prediction problem a multivariate forecasting task. The number of requested orders (RO) is the target variable to be predicted and represents the demand of a given product group, at the 6TG level of aggregation, for a given month. Other time-dependent variables considered are the sales information from the distributors (POS), the inventory level of the distributors (DI), a record of the estimated number of orders that would be requested in the future 18 months (FB), and a record of the design opportunities with an estimated revenue from the distributors. For the last variable, when the estimated revenue surpasses a given quantity, these design opportunities are considered design wins (DW). However, DW data is not currently considered in the company's forecasting system. In the next section, we report the performance for each dataset, when excluding and including the DW variable. Table 2 displays the mean and standard deviation of each variable for the six datasets studied, negative values are possible in this context. None of the series display seasonalities or trends, although a considerable decrease in requested orders is observed during 2020 and later recovered in 2021 due to the influence of the Covid19 pandemic.

3.2 Experiments and results

In order to measure the performance of LSTCN for the six datasets described before, we compute the volume weighted mean absolute percentage error (wMAPE). This measure computes the mean absolute percentage error of the forecasting for each 6TG product group of the dataset, and weights the result based on the volume of orders with regard to the total orders of each 6TG product group (see Table 1).

Table 2: Mean and standard deviation of each variable for the six datasets studied.

	RO		POS		DI		FB		DW	
6TG level	mean	std	mean	std	mean	std	mean	std	mean	std
Dataset 1	10,472	61,994	2,819	28,433	7,769	121,930	45,577	176,181	6,247	14,516
Dataset 2	25,883	65,781	9,598	29,566	17,182	81,414	160,771	301,615	8,635	15,597
Dataset 3	15,354	41,274	4,158	15,530	13,929	81,557	60,946	136,109	5,142	10,178
Dataset 4	58,952	407,053	19,266	139,614	8,815	137,014	70,026	281,306	16,352	55,351
Dataset 5	61,817	218,275	25,171	87,684	19,175	109,648	104,416	315,045	10,554	18,798
Dataset 6	67,771	137,681	33,020	64,269	16,301	92,914	78,597	155,765	20,556	132,682

We compare LSTCN against linear regression (LR), a multivariate implementation of ARIMA (ARIMAX), extra trees regression (ET), and a vanilla RNN. We also report the performance of the best of 52 models of the forecasting system as provided by NXP. For the validation, we use a nested walk-forward cross-validation with a grid search for optimizing hyperparameters. For the ARIMAX method, we varied the order of the autoregressive term (AR) between one and 10 and the order of the moving average term (MA) between one and eight, both with steps of one. For the ET regressor, we varied the number of trees between 50 and 100 with steps of 10, the maximum depth of the tree between two and 10 with steps of one, the minimum number of samples required to split an internal node between two and six with steps of one, and the minimum number of samples for leaf nodes between two and eight with steps of one. The RNN model was tested with learning rate values between 0.001 and 10 with steps power of 10, and it was trained using mean square error and mean absolute error as loss functions. Finally, LSTCN considers three hyperparameters: the number of time patches (see Figure 1) with values one, two, and three, the regularization penalty with values between 0.0001 and 1.0, with intermediate steps 0.001, 0.01, 0.1 and 0 (see λ in Equation 4), and the aggregation operators linear, nonlinear, average linear and average nonlinear (see Section 2.2). The best hyperparameters for each model are reported in Table 3.

The wMAPE of the selected models and NXP forecasting system for all dataset configurations are presented in Table 4. These results are based on the walk-forward cross-validation with data from a period of six months. For each dataset, the selected model that performed best compared to the other selected models is highlighted in the table. Furthermore, NXP refers to the performance of the current forecasting system at the company. As explained in the previous section, NXP's performance for Datasets 1, 2 and 3 is derived from the best prediction of a pool of 52 models in their forecasting system, while the forecasts for Datasets 4, 5 and 6 are performed and evaluated manually. It is important to clarify that NXP selects the best forecasting method for each month from a pool that includes the techniques considered in the comparison, except for RNN. Therefore, the monthly NXP forecast is at least as good as the best forecasting method for that month, such that on average, it will always perform better than all forecasting methods included in the NXP system.

Table 3: Best hyperparameter values for each method.

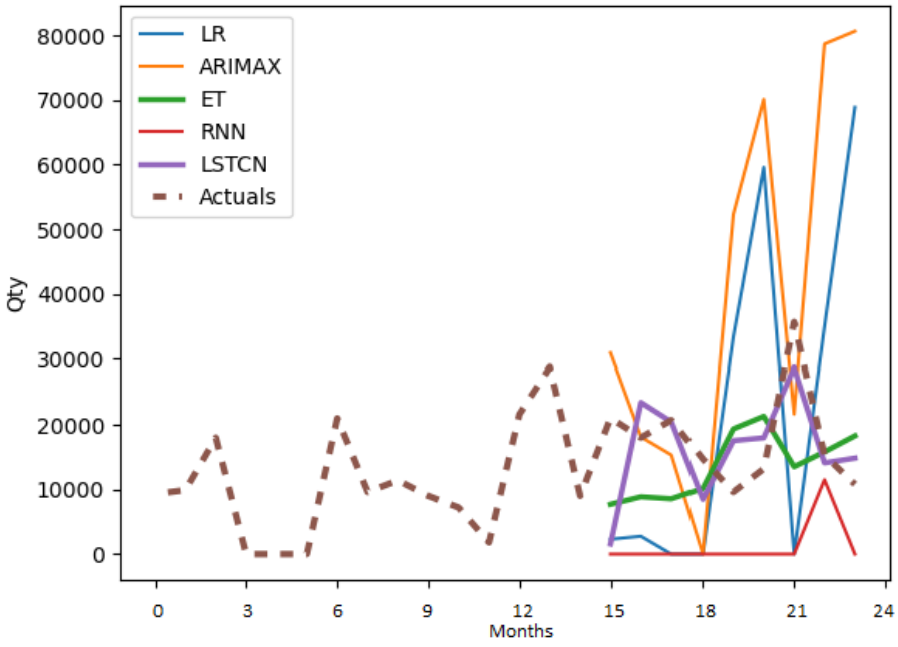
6TG level	ARIMAX		ET				RNN		LSTCN		
	AR	MA	trees	depth	split	leaf	lrate	loss	patches	λ	aggregation
Dataset 1	5	2	110	9	4	2	0.1	MAE	3	0.001	average linear
+ DW	5	2	60	7	4	2	0.01	MAE	3	0.001	average linear
Dataset 2	5	2	80	6	4	3	0.01	MSE	1	1	average linear
+ DW	1	3	60	9	4	2	0.01	MAE	2	0.1	average linear
Dataset 3	1	2	100	6	3	2	0.01	MSE	3	0.001	nonlinear
+ DW	5	4	100	7	4	2	0.1	MAE	2	0.1	average linear
Dataset 4	2	1	70	9	3	2	0.01	MSE	3	0.1	average nonlinear
+ DW	1	1	120	6	3	2	0.01	MAE	3	0.1	average nonlinear
Dataset 5	1	3	80	6	4	2	0.01	MSE	3	0.001	average nonlinear
+ DW	1	2	70	6	4	2	0.1	MAE	3	0.1	average linear
Dataset 6	2	5	80	6	4	2	0.1	MAE	3	0.1	average linear
+ DW	1	1	80	6	5	2	0.01	MAE	3	1	average linear

Table 4: Volume weighted MAPE for each method and the NXP forecasting system.

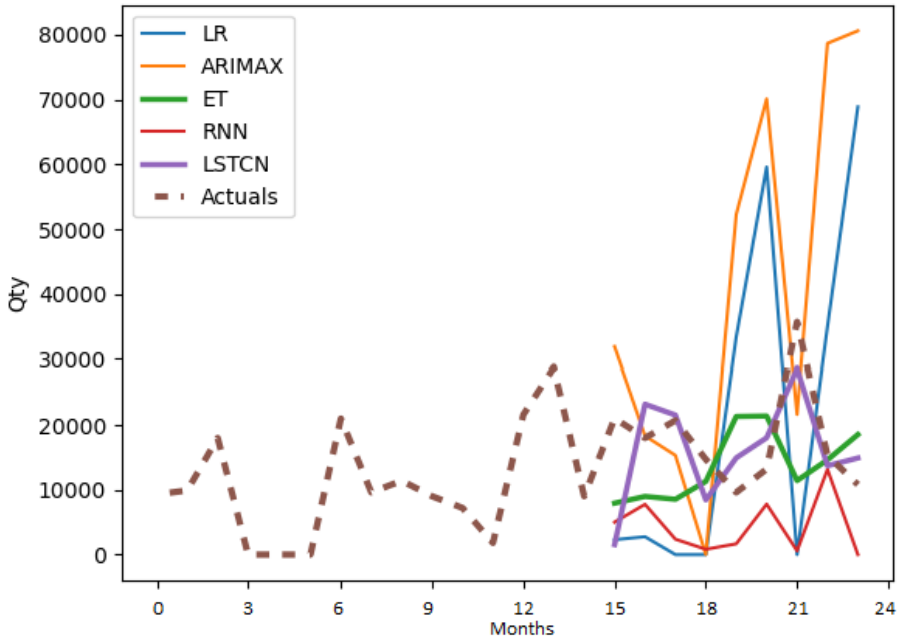
Channel Group	MAG level	6TG level	NXP	LR	ARIMAX	ET	RNN	LSTCN
DMM	Group 1	Dataset 1	40.2	42.6	44.8	**41.6**	63.2	42.1
		+ DW		44.6	47.1	**39.7**	45	41.4
	Group 2	Dataset 2	35.9	42.6	46.3	40.9	42.5	**35.5**
		+ DW		42.2	45.6	37.3	40.1	**33.9**
	Group 3	Dataset 3	36.6	44.4	38.9	46	63	**38.5**
		+ DW		47	42.1	40.7	61.4	**36.7**
DF	Group 1	Dataset 4	38.8	46	42.2	**41.3**	63	45.6
		+ DW		45.6	44.3	**39.6**	61.8	45
	Group 2	Dataset 5	30.8	42.6	41.2	39.4	35.7	**35.4**
		+ DW		43.4	37.6	38.7	55.5	**34.6**
	Group 3	Dataset 6	19.1	**28.5**	35	31.6	54.9	36.3
		+ DW		33.5	36	**30.7**	40.9	33.4

In the results, we observe that LSTCN provides the best results for the datasets of groups 2 and 3 in the DMM channel and group 2 in the DF channel, compared to the other tested methods. LSTCN and ET provide the closest performance to NXP's forecasting system, with an average difference of 4.65 and 5.39 percentage points, respectively. We also observe that LSTCN is even able to outperform in 2% the NXP forecasting system for Dataset 2 while using the DW data. It is noticeable that LSTCN and ET were able to leverage the inclusion of DW data for all the channels and groups analyzed, however, this is not always the case for LR, ARIMAX and RNN models.

Figure 2 shows an example forecast for a period of six months, of a randomly selected 6TG level product group from Dataset 2, when excluding and including the DW data. In this example, we illustrate how LR, ARIMAX, and RNN models produce poor forecasting of the last months of the dataset. The results of LR



(a) excluding DW data



(b) including DW data

Fig. 2: Forecasts for requested orders of a randomly selected 6TG product group from Dataset 2.

and ARIMAX do not improve when including DW data, while for the RNN, we can observe a slight improvement. The ET and LSTCN models produce the best forecasts, as reflected in the general experimental results. For this particular instance, we can observe that LSTCN makes the closest forecasting to the actual values (brown dashed series) since it adequately captures the peaks and valleys of the series. In contrast, the LR and ARIMAX models overestimate the peaks and valleys of the actual values. Even though the forecasts do not seem to differ significantly, when including DW, the ET and LSTCN forecasts are more precise in Figure 2b. The latter observation is especially visible in the LSTCN forecast for the 19th month.

Overall, in the experiments, LSTCN shows its suitability as a candidate for inclusion in the forecasting system of the company. Our method obtains the best performance in three out of six tested datasets, complementing the ET regression predictions already available in the forecasting system. LSTCN also outperformed the vanilla RNN while requiring considerably smaller training time [10]. As a result, the company decided to integrate LSTCN as one of the models in its forecasting system. Given the potential of the method for including prior knowledge in the weight matrix and the possibility of obtaining explanations in the form of feature attribution, the company is interested in continuing the collaboration to leverage further the characteristics of LSTCN.

4 Conclusions

In this paper, we studied the forecasting capabilities of the Long Short-term Cognitive Networks using a real case study concerning multivariate time-series data describing the demand for six different groups of products of a semiconductor company. These novel gated recurrent neural networks use chained learning blocks, each processing a specific time patch in the time series. More importantly, the knowledge learned in a given block is transferred to the next one, thus enabling shallow learning implemented via a fast pseudo-inverse learning rule. One of the main contributions of this paper is that it studies different aggregation operations used when building the matrices to be transferred to the next neural processing system.

The results of numerical simulations have shown that the LSTCN model outperformed well-established methods such as linear regression, ARIMAX, extra tree regression, and vanilla RNN for three of the six tested datasets. Remarkably, LSTCN outperformed NXP's current ensemble of 52 models in one dataset while reporting very low training times. As for the optimal parameter settings, the average linear and the average non-linear aggregation operators combined with small penalization values reported the best results. This provides strong evidence that the model is not overfitting the data; otherwise, larger penalization values would have been preferred. It is reasonable to assume that the average operation is an effective mechanism to prevent overfitting in this case study.

The promising forecasting results, the short training time, the potential for including prior knowledge, and the interpretability of LSTCNs [10] led to the

integration of this algorithm into the suite of models available in the company's forecasting system. Similarly, the possibility of including additional knowledge in the form of a prior knowledge matrix defined by human experts brings a significant advantage over other forecasting models. Such a possibility will be investigated in our future research endeavours.

References

1. Ahmad, M.W., Reynolds, J., Rezgui, Y.: Predictive modelling for solar thermal energy systems: A comparison of support vector regression, random forest, extra trees and regression trees. Journal of Cleaner Production **203**, 810–821 (2018). https://doi.org/10.1016/j.jclepro.2018.08.207
2. Blackburn, R., Lurz, K., Priese, B., Göb, R., Darkow, I.L.: A predictive analytics approach for demand forecasting in the process industry. International Transactions in Operational Research **22**(3), 407–428 (2015). https://doi.org/10.1111/itor.12122
3. Da Veiga, C.P., Da Veiga, C.R.P., Catapan, A., Tortato, U., Da Silva, W.V.: Demand forecasting in food retail: A comparison between the holt-winters and arima models. WSEAS Transactions on Business and Economics **11**(1), 608–614 (2014)
4. Dijkman, R.: Digital process transformation. Inaugural lecture, Technische Universiteit Eindhoven (2022)
5. Jun, S.P., Park, D.H., Yeom, J.: The possibility of using search traffic information to explore consumer product attitudes and forecast consumer preference. Technological Forecasting and Social Change **86**, 237–253 (2014). https://doi.org/10.1016/j.techfore.2013.10.021
6. Ma, S., Fildes, R.: Retail sales forecasting with meta-learning. European Journal of Operational Research **288**(1), 111–128 (2021). https://doi.org/10.1016/j.ejor.2020.05.038
7. Makridakis, S., Spiliotis, E., Assimakopoulos, V.: The m4 competition: Results, findings, conclusion and way forward. International Journal of Forecasting **34**(4), 802–808 (2018). https://doi.org/10.1016/j.ijforecast.2018.06.001
8. Makridakis, S., Spiliotis, E., Assimakopoulos, V.: Statistical and machine learning forecasting methods: Concerns and ways forward. PloS one **13**(3), e0194889 (2018). https://doi.org/10.1371/journal.pone.0194889
9. Morales-Hernández, A., Nápoles, G., Jastrzebska, A., Salgueiro, Y., Vanhoof, K.: Online learning of windmill time series using long short-term cognitive networks. Expert Systems with Applications **205**, 117721 (2022). https://doi.org/10.1016/j.eswa.2022.117721
10. Nápoles, G., Grau, I., Jastrzebska, A., Salgueiro, Y.: Long short-term cognitive networks. Neural Computing and Applications (2022). https://doi.org/10.1007/s00521-022-07348-5
11. Nápoles, G., Vanhoenshoven, F., Falcon, R., Vanhoof, K.: Nonsynaptic error backpropagation in long-term cognitive networks. IEEE Transactions on Neural Networks and Learning Systems **31**(3), 865–875 (2020). https://doi.org/10.1109/TNNLS.2019.2910555
12. Nápoles, G., Vanhoenshoven, F., Vanhoof, K.: Short-term cognitive networks, flexible reasoning and nonsynaptic learning. Neural Networks **115**, 72–81 (2019). https://doi.org/10.1016/j.neunet.2019.03.012

13. Ramachandra, H.V., Balaraju, G., Rajashekar, A., Patil, H.: Machine learning application for black friday sales prediction framework. In: 2021 International Conference on Emerging Smart Computing and Informatics (ESCI). pp. 57–61 (2021). https://doi.org/10.1109/ESCI50559.2021.9396994
14. Seyedan, M., Mafakheri, F.: Predictive big data analytics for supply chain demand forecasting: methods, applications, and research opportunities. Journal of Big Data **7**(1), 1–22 (2020). https://doi.org/10.1186/s40537-020-00329-2
15. Silva, J., Mojica Herazo, J.C., Rojas Millán, R.H., Pineda Lezama, O.B., Gamero, W.M., Varela, N.: Early warning method for the commodity prices based on artificial neural networks: Smes case. Procedia Computer Science **151**, 1243–1248 (2019). https://doi.org/10.1016/j.procs.2019.04.179
16. Smyl, S.: A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. International Journal of Forecasting **36**(1), 75–85 (2020). https://doi.org/10.1016/j.ijforecast.2019.03.017
17. Tang, X., Dai, Y., Wang, T., Chen, Y.: Short-term power load forecasting based on multi-layer bidirectional recurrent neural network. IET Generation, Transmission & Distribution **13**(17), 3847–3854 (2019). https://doi.org/10.1049/iet-gtd.2018.6687
18. Wang, G., Gunasekaran, A., Ngai, E.W., Papadopoulos, T.: Big data analytics in logistics and supply chain management: Certain investigations for research and applications. International Journal of Production Economics **176**, 98–110 (2016). https://doi.org/10.1016/j.ijpe.2016.03.014