Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 12, No. 3, August 2021), Pages: 1-12
www.jacr.iausari.ac.ir

Multivariate Time Series Prediction Considering Intra-Time-Series and Inter-Time-Series Dependencies

Parinaz Eskandarian[1], Jamshid Bagherzadeh Mohasefi[1,2]✉, Habibollah Pirnejad[3], Zahra Niazkhani[4]
1) Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran
2) Department of Electrical and Computer Engineering, Urmia University, Urmia, Iran
3) Patient Safety Research Center, Clinical Research Institute, Urmia University of Medical Sciences, Urmia, Iran
4) Nephrology and Kidney Transplant Research Center, Clinical Research Institute, Urmia University of Medical Sciences, Urmia, Iran
p_eskandarian@yahoo.com; j.bagherzadeh@urmia.ac.ir; pirnejad.h@umsu.ac.ir; niazkhani.z@umsu.ac.ir

Received: 2021/03/17; Accepted: 2021/08/10

Abstract

A few artificial neural networks have been proposed so far for multivariate time series prediction and they used simple general-purpose neural networks. Therefore, they cannot achieve high prediction accuracy. In this paper, we propose an artificial neural network called DBMTSP (Dependency Based Multivariate Time Series Prediction) to predict the next element of a time series in the multivariate case. Compared to the existing methods, DBMTSP considers both intra-time-series dependencies and inter-time-series dependencies efficiently to achieve more accurate predictions. We propose a hierarchical encoder in DBMTSP to discover inter-time-series dependencies. The proposed hierarchical encoder is able to encode secondary time series into a single parameter that represents dependencies that exist between the main time series and the secondary time series. The hierarchical encoder has a scalable design such that it can accept a large number of secondary time-series. We have trained DBMTSP using 32760 data matrices. We evaluated DBMTSP using 8190 test data matrices. Our evaluations show that DBMTSP surpasses the existing methods in term of prediction accuracy.

Keywords: *Time Series, Multivariate, Prediction, Artificial Neural Network*

1. Introduction

In a system, there may be a variable output line that changes by time so that the sequentially-generated output values can be considered as a time-series. The time-series is not constant so that the system generates various output time-series every time we run it.

In the case of univariate time-series [1], there is a single time-dependent output variable. The system generates consecutive values for this output that lead to a time-series. Each value of this output depends on its past values. We call this characteristics as intra-time-series dependency according which the output depends on its past values. That means the output value at Time-Point t depends on the output values of Time-Point t-1,t-2,…,0.

In the case of multivariate time-series [2], there are more than one variable line (or time-series) in the system. In addition to intra-time-series dependency, there is also an inter-time-series dependency according which the main output variable has some dependencies on the other time-series (variables).

In such a system, we may have a few initial values of the output line up to Time-Point t and want to predict the next values that will be generated at Time-Point (t+1), Time-Point (t+2), and so on. We seek to solve this problem in this paper. As we reviewed the existing solutions in Section 3, the univariate case of this problem is simple and is explored well in the literature but the multivariate case is a complicated problem which is not explored well yet so that the existing solutions need improvement. The existing solutions for multivariate time-series prediction consider intra-time-series dependencies well but they are not efficient in considering inter-time-series dependencies.

In this paper, we design a machine called DBMTSP (Dependency-Based Multivariate Time-Series Prediction) that is an artificial neural network to predict the next element of a time-series in the multivariate case. DBMTSP learns the system's output time-series for a large number of previous runs. Then for a single run, the machine is able to get the first output values and predict the upcoming output values. DBMTSP considers both the intra-time-series and the inter-time-series dependencies in prediction. We propose a hierarchical encoder in DBMTSP to discover inter-time-series dependencies. The proposed hierarchical encoder is able to encode secondary time series into a single parameter that represents dependencies that exist between the main time series and the secondary time series. The hierarchical encoder has a scalable design such that it can accept a large number of secondary time-series. We have trained DBMTSP using 32760 data matrices. We evaluated DBMTSP using 8190 test data matrices. Our evaluations show that DBMTSP surpasses the existing solutions in term of prediction accuracy.

The rest of this paper is organized as follows. In Section 2, we describe the multivariate time-series prediction problem in detail. In Section 3, we review the existing designs to solve the problem. We propose a new solution in Section 4 and evaluate it in Section 5. Section 6 finally concludes the paper.

2. Problem Description

In this section, we describe the problem we try to solve in this paper. We call it MTSP (Multivariate Time-Series Prediction).

Assume we have as many as M finite time-series in a system. All the time-series contain as many as N elements. In a single run of the system, we can represent the M time-series in a single MxN matrix as depicted in Fig.1. We call it the Multivariate Time-Series matrix. Row *i* of this matrix represents the elements of Time-Series i at various time points. Time-Series 1 is considered as the main time-series and the other (M-1) time-series of the matrix are considered as secondary input time-series.

$$X = \begin{array}{c} \begin{matrix} Time\ Point\ 1 & Time\ Point\ 2 & \cdots & Time\ Point\ N \end{matrix} \\ \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ \cdots & \cdots & \cdots & \cdots \\ x_{M,1} & x_{M,2} & \cdots & x_{M,N} \end{pmatrix} \end{array} \begin{array}{l} : \text{Time-Series1 (main)} \\ : \text{Time-Series2 (secondary)} \\ \cdots \\ : \text{Time-SeriesM (secondary)} \end{array}$$

Fig1. A MxN Multivariate Time-Series matrix

Every $x_{i,j}$ is the j-th element in Time-Series i that gets a single value for each run of the application but gets various values in different runs of the application. There may be a dependency between any $x_{i1,j1}$ and $x_{i2,j2}$ for any i1, i2, j1, and j2 where $i1 \in \{1,2,...,N\}$, $j1 \in \{1,2,...,N\}$, $i2 \in \{1,2,...,N\}$, and $j2 \in \{1,2,...,N\}$.

Problem MTSP: Let us consider Multivariate Time-Series matrices of a system in the form of Fig. 1. We are given a number of datasets containing train-set Multivariate Time-Series matrices. All the matrices have M rows and various numbers of columns. All columns of the train-set Multivariate Time-Series matrices are known. Now, consider an incomplete MxN Multivariate Time-Series matrix X where only its first k columns are known where k<N. According to the given train-set Multivariate Time-Series matrices, predict $x_{1,k+1}$ of X.

3. Related Works

In this section, we review the existing designs that try to predict dependent time-series.

3.1 Univariate Time Series Prediction

A very popular machine for univariate time series prediction is depicted in Fig. 2. We call it the Double-LSTMs machine. LSTM [3] (Long Short Term Memory) is a composite unit. LSTM is able to get the first elements of a time-series one by one as input and discover their similarity to one of the time-series with which the LSTM was trained previously.

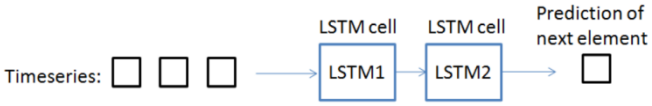

Fig2. The Double-LSTMs machine

The Double-LSTMs machine is not a simple neural network. It contains multiple internal subunits. If we unfold the machine through time (as described in [4]), it turns into a complicated neural network that can learn complicated time-series and sequences.

Many existing works on sequence/time-series prediction are based on the Double-LSTMs machine or combination of LSTM units. For example, authors in [4] combine LSTMs with autoencoders to forecast financial time-series. The Double-LSTMs machine is used in [5] for language translation which is a sequence prediction problem. Authors in [6] use unidirectional and bidirectional LSTMs to design an advanced neural network that predicts next element in time-series. Stacked LSTMs are used in [7] for load forecasting.

Since the Double-LSTMs machine was successful in the univariate case, we extend it to DBMTSP in Section 4 to work in the multivariate case.

3.2 Multivariate Time Series Prediction

There exist a few learning machines for non-linear multivariate time-series prediction. All of them consider a main time-series and a number of secondary time-series. It was first proposed in [8] to enter elements of not only the main time-series but also the secondary time-series into a neural network to predict the next element of the main time-series. Authors in [9] use Multi-Layer Perceptron (MLP) to predict daily solar

radiation as the main time-series. They use a number of other environment measurements as secondary time-series. A similar design is proposed in [10]. Authors in [11] and [12] utilize Twitter moods as secondary time-series to predict market price as the main time-series. A Self-Organizing Fuzzy Neural Network (SOFNN) was used in [11] for multivariate prediction. They showed that accuracy of the price predictions can be significantly improved by inclusion of specific Twitter mood time-series. DA-RNN [13] and EA-LSTM [14] were proposed for multivariate time-series prediction. They use attention networks to discover intra-time-series and inter-time-series dependencies.

R2N2 was proposed in [15] to predict both linear and nonlinear multivariate time-series.

A multi-head Attention scheme was proposed in [16] to predict multivariate energy time-series. Authors in [17] propose to calculate a separate influence parameter per secondary time-series. A convolutional neural network was proposed in [18] that discovers the input elements that were most influential for future outputs. A set of neural filters are used in [19] to extract time-invariant temporal patterns across multiple time points.

The existing multivariate schemes cannot efficiently extract inter-time-series dependencies. Therefore, as our evaluations in Section 5 shows, they cannot achieve a high prediction accuracy. In addition, most the existing multivariate schemes were designed with the assumption that there are a few secondary time-series in the system. Therefore, they did not provide a scalable design.

4. Methodology

In this section, we propose a novel learning machine called DBMTSP.

4.1 Overall Design

We design a learning machine that predicts the next element in Time-Series 1. As displayed in Fig.3, the machine accepts not only Time-Series 1 but also all the secondary time-series as input. This due to the fact that the machine is going to consider the dependencies between Time-Series 1 and the secondary time-series to be able to achieve a higher prediction accuracy.

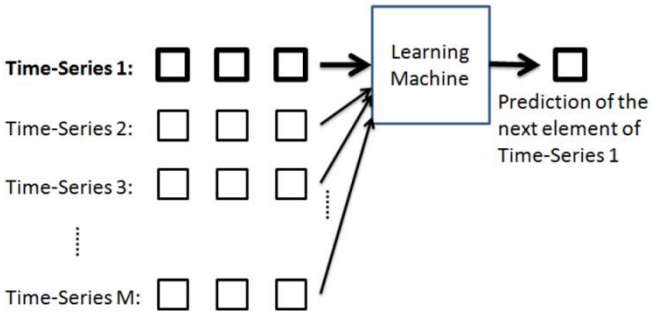

Fig3. Inputs and output of our machine

We design the learning machine depicted in Fig.4. At every time-point, a column of X enters the machine and passes the three components and finally $\hat{x}_{1,t+1}$ will be generated. $\hat{x}_{1,t+1}$ represents the predicted value for $x_{1,t+1}$.

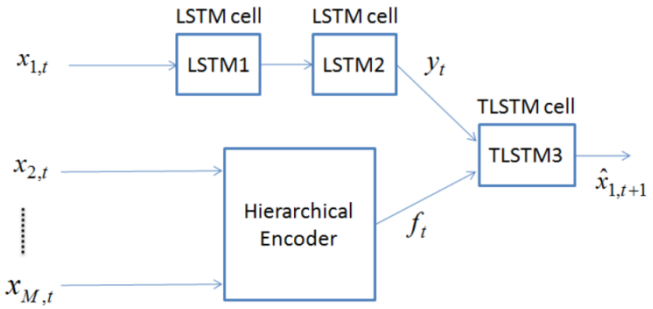

Fig4. Internal architecture of the learning machine in our design for Time-Point t

The machine contains two main data paths: one path through LSTM1 and LSTM2 that process Time-Series 1, the other path through the Hierarchical Encoder that processes the secondary time-series. Our machine has the following components:

LSTM1 and LSTM2: These two units compose a Double-LSTMs machine. They memorize the status of previous serial inputs and generate y_t which is an initial prediction for $\hat{x}_{1,t+1}$.

TLSTM3: This unit considers the effect of f_t on y_t and predicts $\hat{x}_{1,t+1}$.

Hierarchical Encoder: This unit gets the current values of secondary time-series ($x_{2,t} \cdots x_{M,t}$) and encodes them into a single feature f_t.

LSTM1 and LSTM2 are standard LSTM units described in [3]. TLSTM3 is a Child-Sum Tree-LSTM unit described in [20] that has only a left child LSTM and a right child LSTM.

4.2 Internal Design of Hierarchical Encoder

The Hierarchical Encoder is composed of Encoder units. The internal design of Encoder unit is depicted in Fig.5. Each Encoder unit is composed of two LSTM units that get as many as L inputs and encode them into a single output. The output represents all the L inputs as a single encoded value. L is an arbitrary parameter that can be set to an integer in the range of 10 to 100. This range is because of the fact that a single LSTM can memorize its inputs well if the number of its inputs is less than a few hundreds.

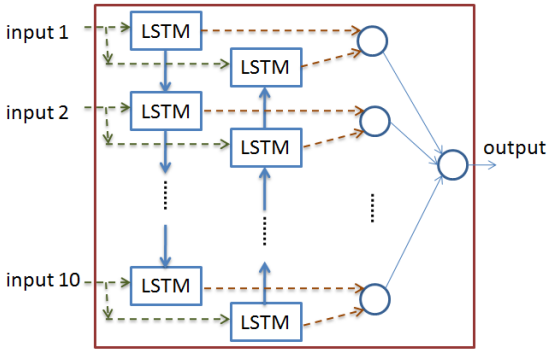

Fig 5. Internal Design of Encoder

Let us consider Time-Point t in the machine. All values on the t-th column of X enter the Hierarchical Encoder except for $x_{1,t}$. The internal design of our Hierarchical Encoder is depicted in Fig.6. It has as many as (M-1) inputs that include $\{x_{2,t}, x_{3,t}, ..., x_{M,t}\}$.

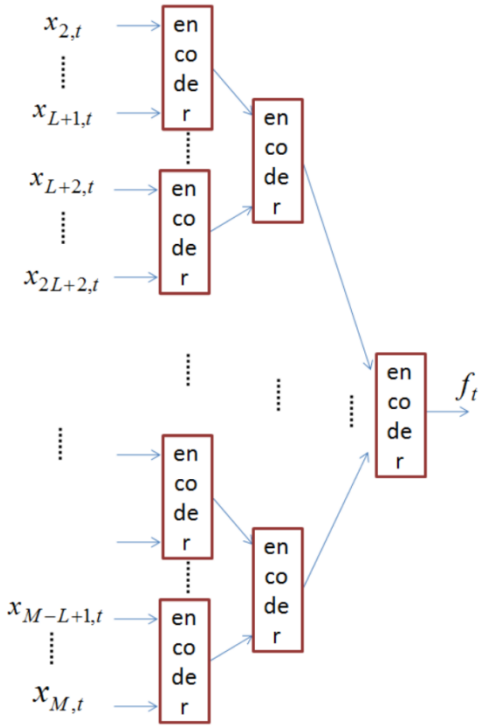

Fig 6. Internal Design of Hierarchical Encoder

The Hierarchical Encoder generates f_t. f_t can be considered as a factor in TLSTM3 that determines how much the value of y_t should be increased or decreased to generate $\hat{x}_{1,t+1}$. During training, the machine learns what to assign to f_t at every time-point for increasing or decreasing y_t. For example, a big f_t may lead to a big increase on y_t. By training, the Hierarchical Encoder learns some secondary time-series have no considerable effect on $x_{1,t}$ and assigns low weights to them. It also learns what weights to assign to the other secondary time-series that have stronger effects on $x_{1,t}$.

Each Encoder accepts as many as L input time-series. If we connect as many as (M-1) secondary time-series to the Hierarchical Encoder, then the number of required Encoders in the first layer of the Hierarchical Encoder equals $(M-1)/L$. Then, it requires as many as $(M-1)/L/L$ Encoders in the second layer. Finally, it requires one Encoder in the third layer. The total number of Encoders is:

$$\frac{M-1}{L}+\frac{(M-1)/L}{L}+1 \qquad (3)$$

Therefore, the Hierarchical Encoder has a scalable design that can accept a large number of secondary time-series.

4.3 Prediction

For prediction as defined in MTSP, we have a Multivariate Time-Series matrix X and we should predict $x_{1,t+1}$. For prediction, the machine works once per Time-Point t for t=1,2,...,k where k<=N. At Time-Point t, values of the t-th column of X enter the machine and the machine generates $\hat{x}_{1,t+1}$. If $x_{1,t+1}$ exists in X, then $x_{1,t+1}$ is used and $\hat{x}_{1,t+1}$ is ignored at Time-Point (t+1).

4.4 Training Algorithm

To train the machine, only complete Multivariate Time-Series matrices are extracted from the datasets. For training the machine using a MxN Multivariate Time-Series matrix, the machine works once per Time-Point t for t=1,2,...,N. At Time-Point t, values of Column t of the matrix enter the machine along with $x_{1,t+1}$, and then the machine adjusts its internal weights and bias values.

We used the truncated BPTT algorithm [21] for training our machine with a learning rate of 10^{-5} and a momentum of 0.9. We chose the initial weights randomly from interval [-0.1; 0.1].

5. Evaluations

In this section, we evaluate our method in two scenarios.

5.1 Comparison

To compare to our machine, we have chosen SOFFN and DA-RNN (Section 3) which are two of the best existing learning machines that consider multivariate time-series for next-element prediction. Although SOFFN was used for stock price prediction, it can be used for all problems that match MTSP. In addition, we compare our machine with the Double-LSTMs machine (Section 3) to understand the improvement of considering a multivariate design compared to a univariate one.

5.2 Metrics

We choose the Root Mean Squared Errors (RMSE) and Mean Absolute Errors (MAE) as the evaluation metrics to represent prediction error. They are calculated by (4) and (5).

$$RMSE = \sqrt{\frac{1}{H}\sum_{s=1}^{H}(P_s - R_s)^2} \quad (4)$$

$$MAE = \frac{1}{H}\sum_{s=1}^{H}|P_s - R_s| \quad (5)$$

where s indicates a test matrix, P_s is prediction for Matrix s, R_s is real value in Matrix s, and H is the number of test matrices.

5.3 Datasets

To evaluate our machine, we have chosen the Gas sensor array temperature modulation Dataset (https://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+temperature+modulation). It contains as many as 4095000 data instances. We broke it into 40950 Multivariate Time-Series matrices so that each one of these matrices contained as many as N=100 rows.

That means each one of the time-series had 100 sequential elements. We used 32760 (80%) of the matrices for training and 8190 (20%) for evaluations.

The original dataset contains 20 time-series. We have selected as many as M=10 dependent time-series to include in our evaluations as described in Table 1. One of the time-series was selected as the main time-series and the others were selected as the secondary time-series.

Table 1: The time-series used in our evaluations

	Description	Name	Unit
Main Time-Series	Resistance of the first gas sensor	R1	Mega Ohm
Secondary Time-Series	CO concentration	CO	Parts Per Million
	Relative Humidity	RH	%
	Temperature	T	ºC
	Flow rate	FR	MiliLitter per minute
	Heater voltage	HV	Volt
	Resistance of the second gas sensor	R2	Mega Ohm
	Resistance of the third gas sensor	R3	Mega Ohm
	Resistance of the forth gas sensor	R4	Mega Ohm
	Resistance of the fifth gas sensor	R5	Mega Ohm

We have calculated the Pearson correlation coefficient between the main time-series and each one of the secondary time-series in the datasets. Table 2 displays the results. Since the coefficients are bigger than 0.5, we can conclude that there are high dependencies between the time-series we have selected in the datasets. If a learning machine discovers these dependencies, it will be able to achieve a high prediction accuracy.

Table 2: Pearson correlation coefficient between the main time-series and the secondary time-series

	CO	RH	T	FR	HV	R2	R3	R4	R5
R1	0.81	0.79	0.72	0.86	0.78	0.84	0.83	0.89	0.86

5.4 Implementation

We implemented our design in the Python 3 language using the Keras library using TensorFlow. For LSTM1 and LSTM2, we used the standard LSTM implemented in Keras. We implemented a new LSTM unit for TLSTM3.

5.5 Results in Scenario I

We evaluated our machine in Scenario I in which various numbers (k) of initial columns in the under-prediction Multivariate Time-Series matrix are known. If more columns are known, the machine should be able to understand better what states/conditions the system is and then has a more accurate prediction of $x_{1,k+1}$.

In Fig.7 and Fig.8, we compare DBMTSP with Double-LSTMs, DA-RNN, and SOFNN. The highest curve represents the Double-LSTMs machine which does not consider the secondary time-series. The lowest curve represents the results of our proposed DBMTSP, which considers both the main and the secondary time-series while emphasizing the main time-series.

In Fig.7, we observe that in most of the time, DBMTSP outperforms the other methods. Considering RMSE in Scenario I, DBMTSP outperforms Double-LSTMs, SOFNN and DA-RNN by 22.7%, 19.4%, and 8.1% in average, respectively. This result

proves DBMTSP is able to capture some valuable information which might be ignored by SOFNN and DA-RNN. Both DA-RNN and DBMTSP perform better than Double-LSTMs because they consider inter-time-series dependencies. The above observations also apply to the MAE results in Fig.8.

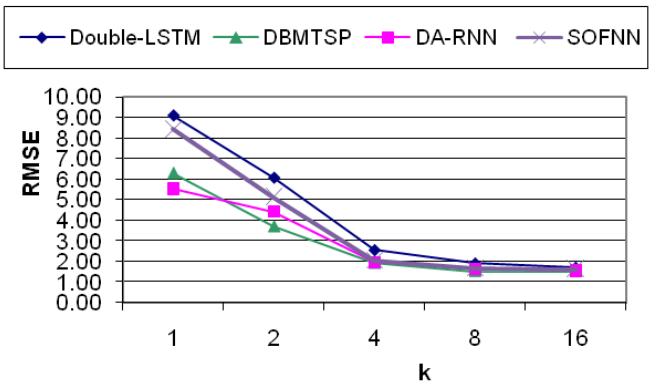

Fig 7. RMSE versus k for one time-point prediction

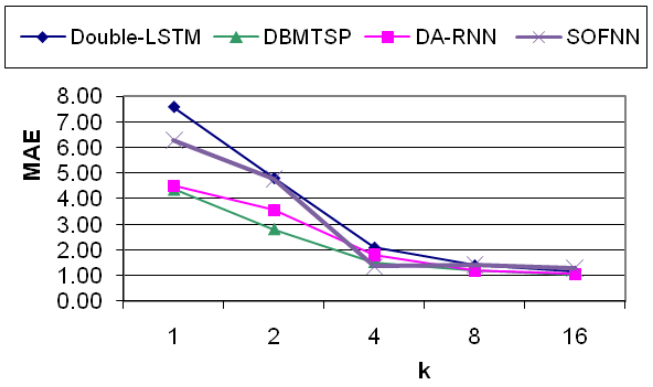

Fig 8. MAE versus k for one time-point prediction

In addition, we have evaluated the four machines on the training datasets. Fig.9 presents the results. It shows that prediction error depends on k. The machines achieve lower prediction error with a bigger k. All the four machines had prediction errors less than 1% for k>1. For k=1, it is acceptable to have RMSE=1.5 because of the fact that the machine cannot discover the under-prediction input time-series by reading only its first time point.

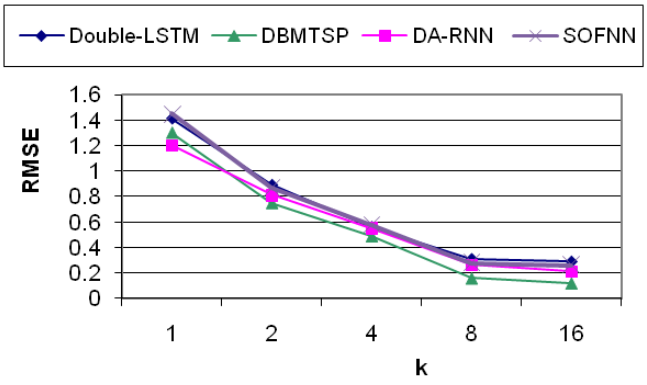

Fig 9. RMSE versus k for one time-point prediction in the training datasets

5.6 Results in Scenario II

In Scenario II, we evaluated the power of our machine by checking how many time-points forward it can predict correctly where the first k columns are known in the under-prediction Multivariate Time-Series matrix. We expect that prediction accuracy drops as more additional time-points are predicted. But it depends on k too. If an enough number of initial columns are not known, the machine cannot obtain a proper understanding of in what state/condition the system is. Therefore, the prediction accuracy drops dramatically as more additional time-points are predicted. Thus, we have chosen k=5 in this evaluation.

We report the prediction errors of the evaluated machines in Fig.10 and Fig.11. Considering RMSE in Scenario II, DBMTSP outperforms Double-LSTMs, SOFNN and DA-RNN by 26.9%, 18.5%, and 9.3% in average, respectively. Considering DBMTSP only, we could see that RMSE increases from 1.54 to 2.81 while MAE increases from 1.08 to 2.48. That means when the predicted time-point gets farther from Time-Point k as far as one time-point, the prediction accuracy goes down by 12.9% in average.

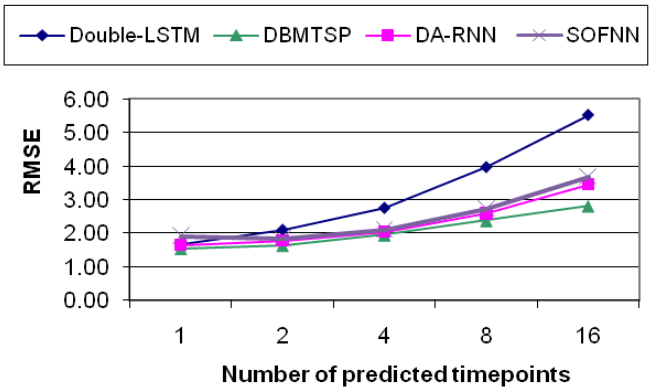

Fig 10. RMSE versus number of predicted time-points after Time-Point k

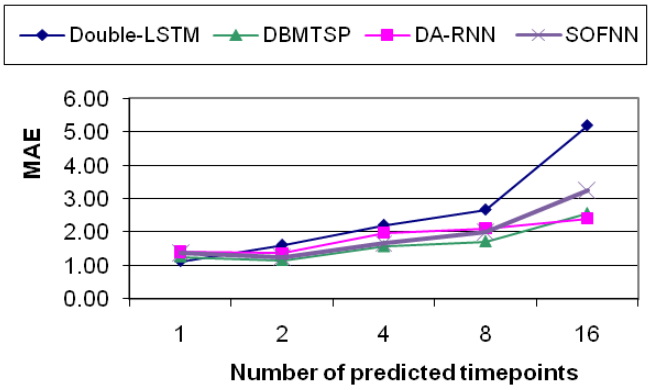

Fig 11. MAE versus number of predicted time-points after Time-Point k

6. Conclusion

A few practical designs have been proposed for prediction in the multivariate case (Section 3) and they used simple general-purpose learning machines. Therefore, they cannot achieve high prediction accuracy. Considering the deficiencies of the existing designs in the multivariate case, we designed the DBMTSP machine that has more

power to extract features of input time-series. DBMTSP is based on the Double-LSTMs machine and extends it to take effect of secondary time-series on the main time-series into account. The hierarchical encoder generates the effect as a feature, then TLSTM3 combines this feature and LSTM2's output into a single prediction. LSTM1 and LSTM2 do the job of memorizing the previous state of their inputs.

In DBMTSP, predictions can be performed for multiple time-points. Prediction accuracy goes down as we predict more time-points after Time-Point k. Prediction accuracy depends on training quality and number of known columns in the Multivariate Time-Series matrix.

References

[1] Mills, T.C., 1990. Time Series Techniques for Economists. Cambridge University Press, 0521343399.

[2] D. Asteriou, and S. G. Hall, "Vector Autoregressive (VAR) Models and Causality Tests", Applied Econometrics, Second ed., London: Palgrave MacMillan, 2011, pp.319–333.

[3] A. Graves, A. Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks", in Proc. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, May 2013.

[4] Bao, W. et al. 2017. "A deep learning framework for financial time series using stacked autoencoders and long-short term memory", PloS one. 12, 7 (2017), e0180944.

[5] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, "Sequence to Sequence Learning with Neural Networks", Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, Pages 3104-3112, Montreal, Canada — December 08 - 13, 2014.

[6] Cui, Z., Ke, R., & Wang, Y. , "Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction", UrbCom'17, Aug. 14, 2017, Halifax, Nova Scotia, Canada.

[7] H. Zheng, J. Yuan, and L. Chen, "Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation", Energies, vol.1, issue.8, p.1168, 2017.

[8] K. Chakraborty, K. Mehrotra, C..K. Mohan, and S. Ranka. (1992, Nov.). Forecasting the behavior of multivariate time series using neural networks. Neural Networks. 5(6). pp.961–970.

[9] C. Voyant , M. Muselli , C. Paoli , and M. Nivet. (2011, Jan.). Optimization of an artificial neural network dedicated to the multivariate forecasting of daily global radiation. Energy. 36(1). pp.348–359.

[10] V. Jimenez, A. Barrionuevo, A. Will, and S. Rodríguez. (2016, Jan.). Neural Network for Estimating Daily Global Solar Radiation Using Temperature, Humidity and Pressure as Unique Climatic Input Variables. Smart Grid and Renewable Energy. 7(3). pp.94-103.

[11] J Bollen, H Mao, X Zeng. (2011, March). Twitter mood predicts the stock market. Journal of computational science. 2(1), pp.1-8.

[12] T. Rao, S. Srivastava, "Modeling Movements in Oil, Gold, Forex and Market Indices using Search Volume Index and Twitter Sentiments", in proc. the 5th Annual ACM Web Science Conference, 2013, pp.336-345.

[13] Y. Qin, et al, "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction", in proc. the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), Melbourne, Australia, August 2017, pp.2627-2633.

[14] Y. Li, Z. Zhu, D. Kong, H. Han, and Y. Zhao. (2018). EA-LSTM: Evolutionary Attention-based LSTM for Time Series Prediction. CoRR abs/1811.03760.

[15] H. Goel, I. Melnyk, and A. Banerjee. (2017). R2N2: Residual Recurrent Neural Networks for Multivariate Time Series Forecasting. arXiv:1709.03159.

[16] Seok-Jun Bu, and Sung-Bae Cho, "Time Series Forecasting with Multi-Headed Attention-Based Deep Learning for Residential Energy Consumption", Energies 2020, 13, 4722; doi:10.3390/en13184722.

[17] Jun Hu, Wendong Zheng, "Multistage attention network for multivariate time series prediction", Neurocomputing, Volume 383, 28 March 2020, Pages 122-137.

[18] Pantiskas, L.; Verstoep, C.; Bal, H. Interpretable Multivariate Time Series Forecasting with Temporal Attention Convolutional Neural Networks. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence, Canberra, Australia, 1–4 December 2020.

[19] Shun-Yao Shih, Fan-Keng Sun, Hung-yi Lee, "Temporal pattern attention for multivariate time series forecasting", Machine Learning volume 108, pages 1421–1441, 2019.

[20] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. ACL, pages 1556–1566.

[21] P. Werbos, "Backpropagation through time: what it does and how to do it", in proc. the IEEE, 78(10), 1990, pp.1550–1560.