Are Deep Learning Models Practically Good as Promised? A Strategic Comparison of Deep Learning Models for Time Series Forecasting

Zuokun Ouyang, Philippe Ravier, Meryem Jabloun
PRISME Laboratory, University of Orléans, France
{zuokun.ouyang,philippe.ravier,meryem.jabloun}@univ-orleans.fr

***Abstract*—Multivariate time series forecasting problem has attracted enormous research in recent years, and many deep learning models have been proposed and claimed to be effective in different tasks. We find that many of these models were tested in a simple one-step-ahead strategy, which does not apply to real scenarios requiring multistep forecasting. This paper compares the performance of three well-known deep learning models (DA-RNN, LSTNet, and TPA-LSTM) for multivariate time series forecasting under three forecasting strategies (One-Step-Ahead, Recursive, and MIMO) for multistep forecasting. We conducted our experiments on six datasets (NASDAQ 100 Stock Data, Beijing PM2.5 Data Set, Electricity, Exchange Rate, Solar Energy, and Traffic) with four forecasting horizons (3, 6, 12, and 24). Our results reveal that, under the Recursive strategy, these deep learning models constantly suffer from accumulated errors and thus cannot carry out real multistep forecasting tasks. However, combining them with the MIMO strategy can tackle this problem and thus enables one-step-ahead deep learning models for multistep forecasting.**

***Index Terms*—time series forecasting, multivariate, multistep forecasting, deep learning models, forecasting strategies**

I. Introduction

Time series forecasting is a subdomain of time series analysis in which the historical data are analyzed to develop a model describing its underlying characteristics and extrapolated into the future.

Time series analysis has been dominated for a few decades by statistical methods such as AutoRegressive Integrated Moving Average (ARIMA) [1] and ExponenTial Smoothing (ETS) [2]. Despite the fact that many statistical methods have demonstrated great forecasting ability in different tasks [3], [4], most of these works are conducted on univariate time series. In most cases, the data can be multivariate time series (MTS), where complex interdependencies must be captured to perform accurate predictions.

Over the last decade, the development of artificial intelligence has attracted much attention, and many machine learning models have been proposed to analyze time series [5], [6]. Many attempts at MTS forecasting problems have been made in recent years, especially with deep learning models based on recurrent neural networks (RNN), convolutional neural networks (CNN), and attention mechanisms [7]. Among these models, there are three crucial ones in the field of MTS forecasting, i.e. *Dual-stage Attention-based Recurrent Neural Network (**DA-RNN**)* [8] for introducing attention mechanism into MTS analysis for the first time, *Long- and Short-term Time-series network (**LSTNet**)* [9] for combining CNN and RNN for MTS data, and *Temporal Pattern Attention Long Short-Term Memory (**TPA-LSTM**)* [10] for introducing the *Temporal Attention Pattern* concept for selecting relevant variables.

Nevertheless, although multistep forecasting was claimed to be conducted in their original papers, only a one-step-ahead strategy was actually applied according to their descriptions for problem formulation. In this strategy, the authors generated a single-step-ahead forecast and fed the model with the new actual data to generate the following step. This strategy is intuitive but can only apply to limited cases where multistep forecasting is not required.

For multistep forecasting in real life where we do not possess future values, the *Recursive* and the *Multi-Input Multi-Output (MIMO)* strategies were often considered, and several machine learning models were proven to be applicable with these strategies to many tasks [11], [12]. To verify the applicability of deep learning models on multistep tasks, we conducted several experiments where we: 1) implemented these three models using multistep forecasting strategies, 2) evaluated and compared their performance over different horizons, and 3) tested their applicability for multistep forecasting.

The rest of this paper is organized as follows. In section II, we present a concise description of all the involved deep learning models and the forecasting strategies. Section III presents how we organized and conducted the experiments. We present the comparison results and discussions based on these results in section IV. Section V gives the conclusion.

II. Methods

This section reviews the three previously mentioned deep learning models and defines two forecasting strategies.

A. Deep Learning Models

1) DA-RNN: DA-RNN has attracted much attention since it appeared on IJCAI 2017 [8]. It presents a sequence-to-sequence model [13] combined with the attention mechanism. Unlike the traditional attention models for natural language processing tasks, which include the attention mechanism only at the decoder stage, DA-RNN includes it at both the encoder and decoder stages. Inside the encoder, one attention weight measures the importance of the k-th input series at time t, while at the decoder stage, another attention weight measures the importance of the i-th hidden state from encoder output for the final forecasting. This two-stage attention mechanism enables the model to capture the interdependencies between relevant series and the long-term dependencies at the encoder and decoder stages, respectively.

2) LSTNet: LSTNet was presented by Lai et al. on SIGIR 2018 [9]. Firstly, LSTNet conducts a non pooling convolution on the preprocessed data to capture the dependencies between multivariate series. Then, a Rectified Linear Unit (ReLU) activated Gated Recurrent Unit (GRU) layer is added to capture the long-term interdependencies. A Recurrent-Skip component is elaborately designed to address the gradient vanishing problem for very long-term sequences. Specifically, skip-links are added into the information flow of the RNN, which allows the hidden cell to look at the state in adjacent periods. This procedure ensures that the network can better take care of the seasonality. A dense layer then integrates the outputs of the GRU layer and the skip layer. Lastly, the author includes an autoregressive component to deal with the violate scale changing in the series.

3) TPA-LSTM: Shih et al. proposed TPA-LSTM in 2019 [10]. It ameliorates the traditional attention mechanism for MTS forecasting by focusing on selecting the relevant series rather than the relevant time steps. Firstly, it uses LSTM to deal with the preprocessed time series to extract the hidden state matrix whose rows and columns represent the corresponding series and time steps. Then, a CNN layer detects the temporal patterns of every series by convolving the kernel with the row vector of the hidden state matrix output previously by LSTM. After that, a scoring function for the attention mechanism is applied, then the model calculates the corresponding attention weights using the sigmoid function and generates the context vector in which every row represents the temporal pattern of the corresponding series. Finally, combining the results from an autoregressive module as per LSTNet, the model integrates the hidden state and the context vector to yield the final forecasting.

B. Multistep Forecasting Strategies

A time series forecasting problem can be transformed into a supervised learning task that machine learning and deep learning methods can do. A commonly used approach is to formulate a training set by lagging and stacking the historical series several times.

For a one-step forecasting problem, we can construct a training set $\{X, Y\}$ of shape $[(N-n), n]$ and $[(N-n), 1]$ where N is the total length of the series and n is the number of times we lag the series, often referred to as the window length:

$$X = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \\ y_2 & y_3 & \cdots & y_{n+1} \\ \vdots & \vdots & \vdots & \vdots \\ y_{N-n} & y_{N-n+1} & \cdots & y_{N-1} \end{bmatrix}, Y = \begin{bmatrix} y_{n+1} \\ y_{n+2} \\ \vdots \\ y_N \end{bmatrix} \quad (1)$$

Each row in X represents a training example, while its target corresponds to the element in the same row in Y.

Nonetheless, as Y is a vector, (1) only describes the strategy for one-step-ahead forecasting. Two strategies extending the framework to tackle the multistep forecasting problem are discussed in this section.

1) Recursive: In *Recursive* strategy [11], a single model f is trained and used recursively to generate multistep forecasting by taking the predicted values as the input for future time steps.

$$y_{t+1} = f(y_t, ..., y_{t-n+1}) + w_{t+1} \quad (2)$$

with $t \in \{n, ..., N-1\}$. w_{t+1} is a noise term.

2) Multi-Input Multi-Output (MIMO): The *Recursive* strategy is intuitive but suffers from accumulated errors. To alleviate this problem, the *MIMO* strategy [12] was proposed.

The *MIMO* strategy learns a single multiple output model F:

$$[y_{t+H}, ..., y_{t+1}] = F(y_t, ..., y_{t-n+1}) + \mathbf{w} \quad (3)$$

with $t \in \{n, ..., N-H\}$ where H is the forecast horizon. $F : \mathbb{R}^n \to \mathbb{R}^H$ is a vector-valued function and $\mathbf{w} \in \mathbb{R}^H$ is a noise vector.

In another formulation, the *MIMO* strategy extends (1) into the following format:

$$\begin{aligned} X &= \begin{bmatrix} y_1 & y_2 & \cdots & y_n \\ y_2 & y_3 & \cdots & y_{n+1} \\ \vdots & \vdots & \vdots & \vdots \\ y_{N-n-H+1} & y_{N-n-H+2} & \cdots & y_{N-H} \end{bmatrix}, \\ Y &= \begin{bmatrix} y_{n+1} & y_{n+2} & \cdots & y_{n+H} \\ y_{n+2} & y_{n+3} & \cdots & y_{n+H+1} \\ \vdots & \vdots & \vdots & \vdots \\ y_{N-H+1} & y_{N-H+2} & \cdots & y_N \end{bmatrix} \end{aligned} \quad (4)$$

The rationale of the *MIMO* strategy is that it outputs all future values in one vector during the forecasting stage. Meanwhile, it models the dependency between the values that characterizes the time series [14].

III. Experiments

A. Datasets and Setup

In our experiments, six datasets are selected to evaluate these deep learning models for MTS multistep forecasting. The statistics of these datasets are listed in Tab. I.

- Electricity[1]: Hourly electricity consumption of 321 clients from 2012 to 2014. Complex seasonal data.
- Exchange Rate[2]: Daily exchange rates of eight countries, i.e., Australia, British, Canada, China, Japan, New Zealand, Singapore, and Switzerland, from 1990 to 2016. Nonseasonal data.

[1]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[2]https://github.com/laiguokun/multivariate-time-series-data

TABLE I
DATASET DESCRIPTION

Dataset	Length	Dimension	Frequency	Seasonality
Electricity	26304	321	1 hour	Complex
Exchange-Rate	7588	8	1 day	Nonseasonal
NASDAQ-100	40560	82	1 minute	Nonseasonal
Solar-Energy	52560	137	10 minutes	Seasonal
Traffic	17544	862	1 hour	Seasonal
Beijing-PM$_{2.5}$	43800	8	1 hour	Seasonal

- NASDAQ 100 Stock Data[3]: Stock price by minute from July 26 to December 22, 2016, of 81 corporations under NASDAQ 100 which are used as the driving series and the NASDAQ Index 100 used as target series. Nonseasonal data.
- Solar Energy[4]: 10-minute-level solar power production data from photovoltaic plants in Alabama State in 2006. Seasonal data.
- Traffic[5]: Hourly data from the California Department of Transportation describing the road occupancy rates on San Francisco Bay area freeways from 2015 to 2016. Seasonal data.
- Beijing PM$_{2.5}$ Data[6]: Hourly data of the PM$_{2.5}$ data of US Embassy in Beijing from 2010 to 2014, which is used as the driving and target series. Meteorological data from Beijing Capital International Airport are also included as driving series as well. Seasonal data.

If not specified, all the series in the datasets are harnessed as target series. We split our datasets into training, validation, and test sets in chronological order by the ratio of 8:1:1.

B. Parameter Settings and Evaluation Metric

For simplicity, we took the same parameterization reported in [9] and [10] for LSTNet and TPA-LSTM on Electricity, Exchange-Rate, Solar-Energy, and Traffic. For DA-RNN, we followed the same parameter settings in [8] on NASDAQ-100. For other situations, the tunable parameters were selected based on the results from the validation set. The source codes of the aforementioned models are publicly available according to their papers [8]–[10].

Concretely, for LSTNet on NASDAQ-100 and Beijing PM$_{2.5}$, we set the window size *w* as 60 and 168, respectively. The periodicity pattern for *Recurrent-skip* was set to 30 and 24, and the AR components were both characterized to 24. The recurrent and convolution layer's hidden dimensions were set to 100, while the CNN kernel size was 6. Apart from the same parameterization of LSTNet, we set the number of the hidden state features to 12 on both NASDAQ-100 and Beijing PM$_{2.5}$ Dataset for TPA-LSTM.

[3]https://cseweb.ucsd.edu//~yaq007/NASDAQ100_stock_data.html
[4]https://www.nrel.gov/grid/solar-power-data.html
[5]https://pems.dot.ca.gov/
[6]https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data

For DA-RNN, on Beijing PM$_{2.5}$, we set the input window size to 10. On Electricity, Traffic, Exchange-Rate, and Solar-Energy, the window size was set to 24, 24, 10, and 144, respectively. Meanwhile, we fixed the encoder's and decoder's hidden dimensions to 64.

We performed a 128-minibatch training and a dropout after each layer as per LSTNet with a dropout rate of 0.2. The Adam optimizer [15] was used for all models with a learning rate of 0.0003. Furthermore, unlike the original normalization settings reported in LSTNet and TPA-LSTM, which causes information leakage, we normalized the training, validation, and test sets using the max-min values on their own.

We used the Root Relative Squared Error (RSE) as our evaluation metric with a slight difference with the one in [9], concentrated more on the errors of each series:

$$\text{RSE} = \frac{1}{K}\sum_{i=1}^{K}\frac{\sqrt{\sum_{t=1}^{H}(y_{t,i}-\hat{y}_{t,i})^2}}{\sqrt{\sum_{t=1}^{H}(y_{t,i}-\overline{y_{1:H,i}})^2}} \quad (5)$$

where H is the forecasting horizon, and K is the number of series in the datasets. y_t is the ground truth at time t. $\hat{y}_t$ is the forecast produced by the model, and $\overline{y}$ represents the mean of y.

IV. Results and Discussion

We present our results in Tab. II with the best results highlighted in bold and the following contents.

Tab. II represents the forecasting errors of each model on six datasets with different seasonalities, under two strategies, i.e. *MIMO* and *Recursive*. The results are displayed with those of the original *One-Step-Ahead*, which uses the new actual data as the inputs. We use it as a baseline.

The first question is whether the *MIMO* strategy can be applied to deep learning models for multistep forecasting. Fig. 1 illustrates the average RSEs v.s. different strategies. As the figure shows, the *Recursive* strategy performs the worst while the *MIMO* strategy performs better with tolerable errors

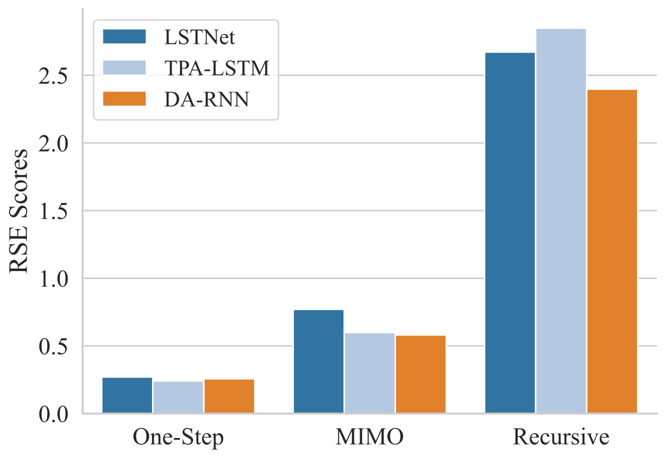

Fig. 1. Average of RSEs over horizon for different strategies

TABLE II

Forecasting RSEs for different models on different forecast horizons with different strategies

Strategy		One-Step-Ahead			MIMO			Recursive		
Dataset	Horizon	LSTNet	TPA-LSTM	DA-RNN	LSTNet	TPA-LSTM	DA-RNN	LSTNet	TPA-LSTM	DA-RNN
Electricity	3	0.0852	**0.0823**	0.0858	0.9020	**0.4310**	0.4748	**1.9427**	2.2879	2.3300
	6	0.0896	0.0920	**0.0882**	1.1232	0.5387	**0.5039**	**1.9981**	2.2502	2.2161
	12	0.0951	0.0945	**0.0923**	1.2349	0.6626	**0.5631**	2.8920	**2.7278**	2.7953
	24	0.1022	**0.1011**	0.1019	1.3857	0.9764	**0.7004**	3.7038	4.0929	**3.2050**
Exchange	3	0.0233	0.0184	**0.0173**	0.2469	**0.1224**	0.1248	1.5907	3.1972	**1.0445**
	6	0.0295	0.0244	**0.0233**	0.2929	0.1404	**0.1203**	1.6755	4.2372	**1.0606**
	12	0.0370	0.0342	**0.0338**	0.3984	0.1616	**0.1441**	2.0769	4.5510	**1.2025**
	24	0.0452	0.0452	**0.0429**	0.4023	0.1855	**0.1718**	2.2096	4.7817	**1.5694**
NASDAQ-100	3	0.2580	**0.1266**	0.1301	0.9425	**0.4761**	0.4867	7.8994	5.4117	**5.2139**
	6	0.2618	**0.1327**	0.1480	0.8904	**0.4888**	0.5159	7.9054	6.7827	**6.7231**
	12	0.2915	**0.1493**	0.1505	0.9340	0.5289	**0.4683**	8.1166	**6.8759**	6.9342
	24	0.3266	**0.1622**	0.1627	1.0992	**0.5809**	0.6408	9.2435	**7.3209**	8.7724
Solar	3	0.1900	0.1815	**0.1590**	0.2955	0.2723	**0.2502**	0.3940	0.3893	**0.3263**
	6	0.2601	0.2417	**0.2309**	0.3705	0.3363	**0.3299**	0.4707	0.5135	**0.4166**
	12	**0.3129**	0.3336	0.4233	**0.3630**	0.4950	0.4729	**0.5466**	0.6701	0.7990
	24	**0.4525**	0.4609	0.5752	**0.4450**	0.5033	0.6903	**0.7218**	0.7620	0.8921
Traffic	3	0.4923	0.4609	**0.4348**	**0.7032**	0.7123	0.8329	**0.7417**	1.2104	1.1012
	6	0.5003	**0.4855**	0.5016	**0.7805**	0.8166	0.8849	1.2999	**1.2282**	1.3874
	12	0.5125	**0.4960**	0.6285	**0.8181**	1.1805	1.0014	**1.2037**	1.3951	1.5951
	24	0.5299	**0.5201**	0.5922	**0.9947**	1.2482	1.1108	1.3574	**1.2777**	1.7765
Beijing PM2.5	3	0.2868	**0.2691**	0.2722	0.5544	**0.4527**	0.4997	0.7581	**0.6940**	0.7004
	6	0.3533	0.3480	**0.3363**	0.7246	0.7192	**0.6203**	1.2652	1.2198	**1.0327**
	12	0.4418	**0.4332**	0.4333	1.0816	**0.9236**	0.9941	1.6615	**1.6505**	1.7788
	24	0.5019	**0.4972**	0.5001	1.4984	1.4232	**1.3388**	**2.4471**	2.8298	2.6719
Winning count		2	12	10	6	7	11	7	7	10

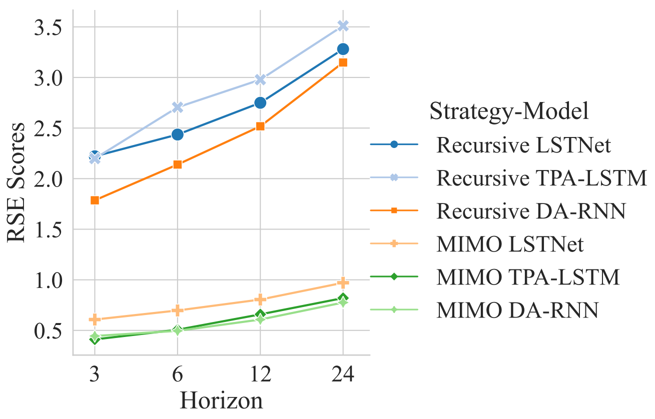

Fig. 2. Average of RSEs for different forecasting horizons

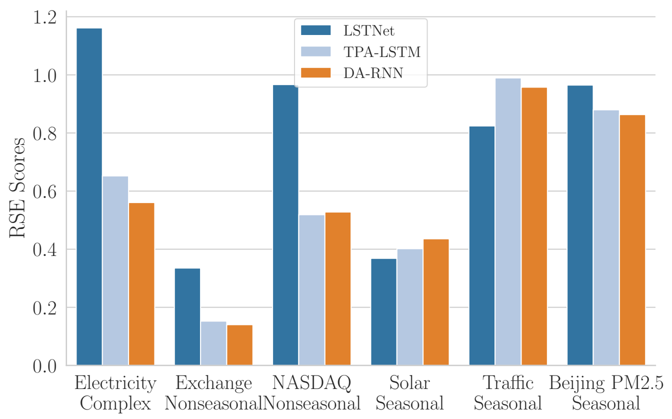

Fig. 3. Average of RSEs for different datasets in *MIMO* strategy

compared to the *One-Step-Ahead* strategy. Less accumulated errors for the *MIMO* strategy are reported in Fig. 2 as the slopes of the *MIMO* curves at the bottom are much smaller than that of the *Recursive* strategy on top.

Fig. 3 presents the average of RSEs over four horizons for different datasets in *MIMO* strategy. One obvious finding from Fig. 3 is that although all the three models can capture the seasonal pattern, the performance of LSTNet falls behind TPA-LSTM and DA-RNN when facing series with zero or complex seasonalities. One explanation could be that the Recurrent-Skip component in LSTNet dedicated to capturing seasonal patterns is not applicable in this situation, as the periodicity pattern needs to be specified as a hyperparameter. Inversely, for seasonal data, periodicity specification is favorable for LSTNet in the case of Solar and Traffic, while less promising for Beijing $PM_{2.5}$ whose seasonality is less evident.

While TPA-LSTM uses a CNN to capture the temporal patterns, DA-RNN uses the attention mechanism in its decoder to put more importance on relevant time steps. So it is also interesting to note that DA-RNN performs slightly better in many cases than TPA-LSTM. This means the CNN component in TPA-LSTM is weaker than the attention mechanism in DA-RNN's decoder in capturing temporal patterns in long sequences. Furthermore, we noticed that a well-designed attention mechanism might help with the input scale variation. This accords with our observations that although DA-RNN does not include the AR component to respond to the changing scale, which LSTNet and TPA-LSTM both use, it still gives, in general, the best results.

V. Conclusion

This investigation aimed to determine whether the deep learning methods are suitable for dealing with the real MTS multistep forecasting problem. The results give a positive answer, which is that by combining with the *MIMO* strategy, deep learning models are competent to carry out real multistep forecasting tasks. In the meantime, our experiments also revealed several interesting findings on their performances dealing with data's seasonality. These could help us select the proper deep learning models for different tasks. Besides, due to the limited length of the paper, we did not include the statistical methods for comparison. Furthermore, other strategies dedicated to multistep forecasting [14] and the recently brought transformer models for time series forecasting [16] are worthy of future research as well.

Acknowledgment

The authors would like to thank the National Research and Technology Association (ANRT) for partially providing the funding (N° 2019/0551) for this research.

References

[1] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. John Wiley & Sons, Inc.,, 2015.

[2] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *Int. J. Forecast.*, vol. 20, no. 1, 2004.

[3] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 3rd ed. OTexts, 2021.

[4] Z. Ouyang, P. Ravier, and M. Jabloun, "STL Decomposition of Time Series Can Benefit Forecasting Done by Statistical Methods but Not by Machine Learning Ones," *Eng. Proc.*, vol. 5, no. 1, p. 42, 2021.

[5] T. Lin, B. G. Horne, P. Tiiio, C. L. Giles, and S. Member, "Learning long-term dependencies in NARX recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1329–1351, 1996.

[6] A. Bouchachia and S. Bouchachia, "Ensemble Learning for Time Series Prediction," in *INDS*, 2008.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All you Need," in *NeurIPS*, 2017.

[8] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction," in *IJCAI*, 2017.

[9] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks," in *SIGIR*, 2018.

[10] S.-Y. Shih, F.-K. Sun, and H.-y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Mach. Learn.*, vol. 108, no. 8-9, pp. 1421–1441, 2019.

[11] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, and A. Lendasse, "Methodology for long-term prediction of time series," *Neurocomputing*, vol. 70, no. 16-18, pp. 2861–2869, 2007.

[12] G. Bontempi, "Long term time series prediction with multi-input multi-output local learning," in *ESTSP*, 2008.

[13] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *NeurIPS*, 2014.

[14] G. Bontempi, S. B. Taieb, and Y.-A. Le Borgne, "Machine learning strategies for time series forecasting," in *eBISS*, 2012.

[15] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, 2017.

[16] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in Time Series: A Survey," *arXiv:2202.07125 [cs, eess, stat]*, 2022.