This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

LSTM-MSNet: Leveraging Forecasts on Sets of Related Time Series With Multiple Seasonal Patterns

Kasun Bandara, Christoph Bergmeir, and Hansika Hewamalage

***Abstract*— Generating forecasts for time series with multiple seasonal cycles is an important use case for many industries nowadays. Accounting for the multiseasonal patterns becomes necessary to generate more accurate and meaningful forecasts in these contexts. In this article, we propose long short-term memory multiseasonal net (LSTM-MSNet), a decomposition-based unified prediction framework to forecast time series with multiple seasonal patterns. The current state of the art in this space is typically univariate methods, in which the model parameters of each time series are estimated independently. Consequently, these models are unable to include key patterns and structures that may be shared by a collection of time series. In contrast, LSTM-MSNet is a globally trained LSTM network, where a single prediction model is built across all the available time series to exploit the cross-series knowledge in a group of related time series. Furthermore, our methodology combines a series of state-of-the-art multiseasonal decomposition techniques to supplement the LSTM learning procedure. In our experiments, we are able to show that on data sets from disparate data sources, e.g., the popular M4 forecasting competition, a decomposition step is beneficial, whereas, in the common real-world situation of homogeneous series from a single application, exogenous seasonal variables or no seasonal preprocessing at all are better choices. All options are readily included in the framework and allow us to achieve competitive results for both cases, outperforming many state-of-the-art multiseasonal forecasting methods.**

***Index Terms*— Long short-term memory (LSTM), multiple seasonality, neural networks (NNs), recurrent neural network (RNN), time-series forecasting.**

I. Introduction

TIME series forecasting has become a key enabler of modern-day business planning by landscaping the short-, medium-, and long-term goals in an organization. As such, generating accurate and reliable forecasts is becoming a perpetual endeavor for many organizations, leading to significant savings and cost reductions. The complex nature of the properties present in a time series, such as seasonality, trend, and level, may bring numerous challenges to produce accurate forecasts. In terms of seasonality, a time series may exhibit complex behavior, such as multiple seasonal patterns, noninteger seasonality, and calendar effects.

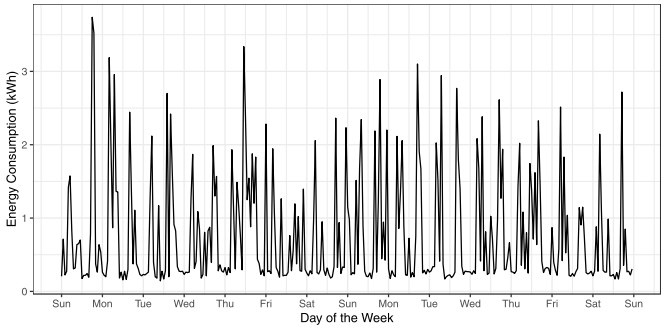

Fig. 1. Half-hourly energy consumption of a household over a two-week period of time, extracted from the AusGrid-Energy data set [1], displaying the interday (daily) and intraday (weekly) seasonal patterns.

As sensors and data storage capabilities advance, time series with higher sampling rates (subhourly, hourly, and daily) is becoming more common in many industries, e.g., in the utility demand industry (electricity and water usage). Fig. 1 illustrates an example of half-hourly energy consumption of an Australian household that exhibits both daily (period = 48) and weekly (period = 336) seasonal patterns. A longer version of this time series may even exhibit a yearly seasonality (period = 17532), representing seasonal effects, such as summer and winter. Particularly, in the energy industry, accurate short- and long-term load forecastings may lead to better demand planning and efficient resource management. In addition to the utility demand industry, the demand variations in the transportation, tourist, and healthcare industries can also be largely influenced by multiple seasonal cycles.

The current methods to handle multiple seasonal patterns are mostly statistical forecasting techniques [2], [3] that are univariate. Thus, they treat each time series as an independent sequence of observations and forecast it in isolation. The univariate time-series forecasting is not able to exploit any cross series information available in a set of time series that may be correlated and share a large number of common features. This is a common characteristic observed in the realm of "Big Data," where often large collections of related time series are available. Examples for these are sales demand of related

Manuscript received September 9, 2019; revised December 20, 2019 and March 9, 2020; accepted March 28, 2020. This work was supported in part by the Australian Research Council under Grant DE190100045, in part by the Facebook Statistics for Improving Insights and Decisions Research Award, in part by the Monash Institute of Medical Engineering Seed Funding, and in part by the MASSIVE—High Performance Computing Facility, Australia. *(Corresponding author: Kasun Bandara.)*

The authors are with the Faculty of Information Technology, Monash University, Melbourne, VIC 3800, Australia (e-mail: herath.bandara@monash.edu; christoph.bergmeir@monash.edu; hansika.hewamalage@monash.edu).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.2985720

2162-237X © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

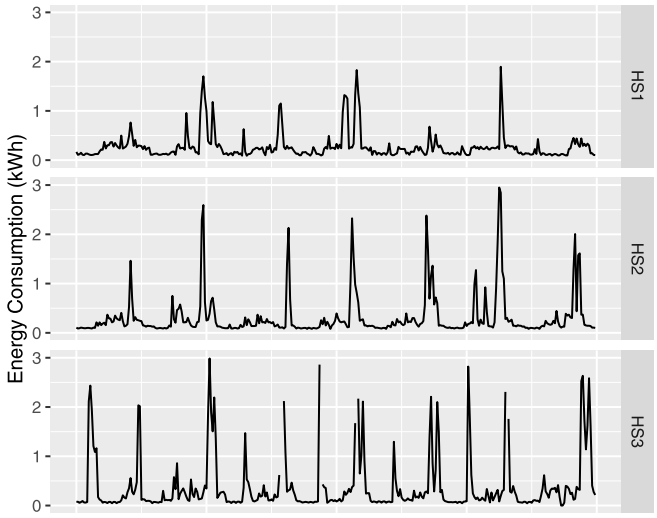

Fig. 2. Half-hourly energy consumption fluctuations of three different households in Australia [1] over a time period of one week.

product assortments in retail, server performance measures in computer centers, household smart meter data, and so on. This can be applied to the time series shown in Fig. 2, in which these energy consumption patterns of various households can be similar and may share key properties in common. As a result, efforts to build global models across multiple related time series are becoming increasingly popular, and these methods have achieved state-of-the-art performance in recent studies [4]–[9]. The recent success is mainly around recurrent neural networks (RNNs) and long short-term memory (LSTM) networks that are naturally suited in modeling sequence data.

Although several unified models have been proposed to learn better under these circumstances, how to handle multiple seasonal patterns in a set of time series has not yet been thoroughly studied. Moreover, the competitiveness of such global models highly relies on the characteristics of the time series. To this end, in this article, we propose the LSTM-multiseasonal net (MSNet), a novel forecasting framework using LSTMs that effectively accounts for the multiple seasonal periods present in a time series. Following the recent success, our model borrows the strength across a set of related time series to improve the forecast accuracy. This enables our model to untap the common seasonality structures and behaviors available in a collection of time series. As a part of the LSTM-MSNet architecture, we introduce a host of decomposition techniques to supplement the LSTM learning procedure, following the recommendations of [7], [10]–[12]. Nevertheless, the competitiveness of such global models can be affected by the homogeneous characteristics present in the collection of time series [7]. Therefore, LSTM-MSNet introduces two training paradigms to accommodate both homogeneous and inhomogeneous groups of time series. Our model is evaluated using several time-series databases, including a competition data set and real-world data sets, which contain multiple seasonal patterns, exhibiting different levels of seasonal homogeneity. The source code relevant to the LSTM-MSNet framework is available at *https://github.com/kasungayan/LSTMMSNet*.

The rest of this article is organized as follows. In Section II, we discuss the developments of statistical approaches and neural networks (NNs) in the multiseasonal time-series forecasting field. Next in Section III, we explore the proposed LSTM-MSNet forecasting framework in detail and outline the key learning paradigms employed in our architecture. Our experimental setup is presented in Section IV, where we demonstrate the results obtained by applying LSTM-MSNet to a variety of time-series data sets with multiple seasonal cycles. Finally, Section V concludes this article.

II. Background and Related Work

The traditional approaches to model time series with seasonal cycles are mostly state-of-the-art univariate statistical forecasting methods, such as exponential smoothing (ES) methods [13] and autoregressive integrated moving-average (ARIMA) models [3]. The basic forms of these algorithms are only suited in modeling a single seasonality and unable to account for multiple seasonal patterns.

Nonetheless, over the past decade, numerous studies have been conducted to extend the traditional statistical forecasting models to accommodate multiple seasonal patterns [14]–[18]. An early study developed by Harvey *et al.* [14] introduces a model to suit time series with two seasonal periods. Later, Taylor [15] adopts the simple Holt–Winters method to capture seasonalities by introducing multiple seasonal components to the linear version of the model. Gould *et al.* [16] proposed an innovation state-space approach to model multiple seasonalities, in which various forms of seasonal patterns can be incorporated, i.e., additive seasonality and multiplicative seasonality. Later, Taylor and Snyder [17] overcome the limitations of Gould *et al.* [16] by introducing a parsimonious version of the seasonal ES approach. However, the majority of these techniques suffer from overparameterization and optimization problems and are also unable to model complex seasonal patterns in a time series. For more detailed discussions of these weaknesses, we refer to De Livera *et al.* [18]. A more flexible and parsimonious version of an innovation state-space modeling framework was developed by De Livera *et al.* [18], aiming to address various challenges associated with seasonal time series, such as modeling multiple seasonal periods, noninteger seasonality, and calendar effects. Today, these proposed seasonal models, i.e., Box-Cox, ARMA, Trend, Seasonal (BATS) and Trigonometric BATS (TBATS), are considered as the state-of-the-art statistical techniques to model time series with multiple seasonal patterns.

Time-series decomposition is another popular strategy to handle time series with complex seasonal patterns [2], [19]. Here, the time series is decomposed into a trend, seasonal, and residual component. Each component is modeled separately so that the model complexity is less than forecasting the original time series as a whole. For example, this approach is applied by Nowicka-Zagrajek and Weron [19], where those authors initially decompose time series using a moving average technique. The seasonally adjusted time series is then modeled separately using an ARMA process. Moreover, Lee and Ko [2] used a lifting scheme, a different decomposition technique,

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

to separate the original time series at different load frequency levels. Afterward, individual ARIMA models are built to forecast each decomposed subseries separately.

In parallel to these developments, NNs have been advocated as a strong alternative to traditional statistical forecasting methods in forecasting seasonal time series. The favorable properties toward forecasting, such as universal function approximation [20], [21], in theory position NNs as a competitive machine learning approach to model underlying seasonality in a time series. Though early studies postulate the suitability of NNs in modeling seasonal patterns [22], [23], more recent studies advise that deseasonalising the time series prior to modeling is useful to achieve better forecasting accuracy from NNs [10]–[12], [24], [25]. Here, deseasonalization refers to the process of removing the seasonal component from a time series. More specifically, Nelson *et al.* [10] and Ben Taieb *et al.* [12] empirically show the accuracy gains by including a deseasonalization process with NNs. Furthermore, Zhang and Qi [11] highlight that NNs are unable to model trend or seasonality directly; thus detrending or deseasonalization is necessary to produce accurate forecasts with NNs. Meanwhile, Dudek [26] developed a local learning-based approach to deal with multiple seasonal cycles. Though this obviates the need for time-series decomposition, the local learning procedure that matches similar seasonal patterns in a time series tends to weaken the global generalisability of the model.

More recently, deep NNs have drawn significant attention among forecasting practitioners. In particular, RNNs and convolutional NNs (CNNs) have exhibited promising results, outperforming many state-of-the-art statistical forecasting methods [4]–[8], [27], [28]. Nevertheless, despite the substantial literature available on deep learning in time-series forecasting, only a few attempts have been undertaken to explicitly handle multiple seasonal patterns in a time series [8], [29]. Lai *et al.* [8] introduced a combination of CNN and RNN architectures to model short- and long-term dependencies in a time series and employed a skipped connection architecture to model different seasonal periods. Bianchi *et al.* [29] implement a seasonal differencing strategy to select the most significant seasonal pattern, i.e., single seasonality present in a time series to forecast the short-term energy load. In order to capture temporal dependencies across both short- and long-term periods, Fernando *et al.* [27] developed a recursive memory network architecture that jointly models both long- and short-term relationships. Here, the proposed hierarchical memory structure attempts to jointly model both long- and short-term relationships in sequence-to-sequence mapping problems. Moreover, the winning submission of the recently concluded that M4 forecasting competition [30], ES-RNN, uses a hybrid approach to forecast the hourly time series category with two seasonalities. However, the original implementation of ES-RNN restricts the number of seasonalities to two, and also due to the limitations of the underlying models (Holt–Winters) that operate in this approach, the ES-RNN is not suitable to handle long term seasonalities in a time series (e.g., yearly seasonality in an hourly time series) [31].

III. LSTM-MSNet Framework

In this section, we first formally define the problem of forecasting with multiple seasonal patterns and then discuss the components of the proposed LSTM-MSNet architecture.

A. Problem Statement

Let $i \in \{1, 2, \ldots, n\}$ be the ith time series from n time series in our database. The past observations of the time series i are given by $X_i = \{x_1, x_2, \ldots, x_K\} \in \mathbb{R}^{K_i}$, where K_i represents the length of the time series i. We introduce the seasonality periods of time series i as $S_i = \{s_1, s_2, \ldots, s_P\} \in \mathbb{R}^P$, where P is the highest seasonal period present in the time series i. The primary objective of this study is to develop a global prediction model f, which uses previous observations of all the time series, i.e., $X = \{X_1, X_2, \ldots, X_n\}$, to forecast M number of future data points i, i.e., $X_i^M = \{x_t, x_{t+1}, \ldots, x_{t+M}\}$, while accounting for all the available seasonal periods $S = \{S_1, S_2, \ldots, S_P\} \in \mathbb{R}^{n \times P}$ present in the time series. Here, M is the intended forecasting horizon of time series i. The model f can be defined as follows:

$$X_i^M = f(X, S, \theta) \tag{1}$$

where θ is the model parameter of our LSTM-MSNet prediction model.

LSTM-MSNet is a forecasting framework designed to forecast time series with multiple seasonal patterns. The architecture of LSTM-MSNet is a fusion of statistical decomposition techniques and RNNs. The LSTM-MSNet has three layers, namely: 1) the preprocessing layer, which consists of normalization and variance stabilizing phase and a seasonal decomposition phase; 2) the recurrent layer, which consists of an LSTM-based stacking architecture to train the network; and 3) a postprocessing layer to denormalize and reseasonalize the time series to derive the final forecasts. The proposed framework can be used with any RNN variant, such as LSTMs, gated recurrent units (GRUs), and others. In this article, we select LSTMs, a promising RNN variant, as our primary network training module. In the following, we discuss each layer of the LSTM-MSNet in detail.

B. Normalization and Variance Stabilization Layer

The proposed LSTM-MSNet is a global model that is built across a group of time series. Therefore, performing a data normalization strategy becomes necessary as in a collection of time series; each time series may contain observations with different value ranges. Hence, we use the mean-scale transformation strategy that which uses the mean of a time series as the scaling factor. This scaling strategy can be defined as follows:

$$x_{i,\text{normalized}} = \frac{x_i}{\frac{1}{k}\sum_{t=1}^{k} x_{i,t}} \tag{2}$$

where $x_{i,\text{normalized}}$ represents the normalized observation, and k represents the number of observations of time series i.

After normalizing the time series, we stabilize the variance in the group of time series by transforming each time series to a logarithmic scale. Apart from the variance stabilization, the log transformation also enables the conversion of the

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
SUMMARY OF TECHNIQUES USED FOR MULTISEASONAL DECOMPOSITION

Technique	Package	Deterministic	Stochastic
MSTL (s.window = "periodic")	forecast [35]	✓	✗
MSTL (s.window = "7")	forecast [35]	✓	✓
AutoSTR	stR [32]	✓	✓
TBATS	forecast [35]	✓	✓
Prophet	prophet [39]	✓	✓

seasonality form in a given time series to an additive form. This is a necessary requirement for additive time series decomposition techniques employed in our decomposition layer. The transformation can be defined in the following way:

$$X_{i,\text{logscaled}} = \begin{cases} \log(X_i), & \min(X) > 0; \\ \log(X_i + 1), & \min(X) = 0 \end{cases} \tag{3}$$

where X denotes a time series, and $X_{i,\text{logscaled}}$ is the corresponding log transformed time series i.

C. Seasonal Decomposition

As highlighted in Section II, when modeling seasonal time series with NNs, many studies suggest applying a prior seasonal adjustment, i.e., deseasonalization to the time series [10]–[12]. The main intention of this approach is to minimize the complexity of the original time series, thereby reducing the subsequent effort of the NN's learning process. In line with these recommendations, LSTM-MSNet initially uses a deseasonalization strategy to detach the multiseasonal components from a time series. Here, seasonal components refer to the repeating patterns that exist in a time series and that may change slowly over time [37]. To accommodate this, we use a series of statistical decomposition techniques that support separating multiseasonal patterns in a time series. We also configure these methods to extract various forms of seasonality, i.e., deterministic and stochastic seasonalities to assess their sensitivity toward the forecast accuracy. Next, we briefly describe the different types of decomposition techniques used in our study. An overview of the methods is given in Table I.

1) Multiple STL Decomposition: Multiple seasonal-trend decomposition (STL) (MSTL) extends the original version of STL [33] to allow for decomposition of a time series with multiple seasonal cycles. The STL method additively decomposes a time series into trend, seasonal, and remainder components. In other words, the original series can be reconstructed by summing the decomposed parts of the time series. The additive decomposition can be formulated as follows:

$$x_t = \hat{S}_t + \hat{T}_t + \hat{R}_t \tag{4}$$

where x_t represents the observation at time t, and $\hat{S}_t$, $\hat{T}_t$, and $\hat{R}_t$ refers to the seasonal, trend, and the remaining components of the observation, respectively.

In MSTL, the STL procedure is used iteratively to estimate the multiple seasonal components in a time series. Thus, the original version of (4) can be extended to reflect the decomposition of MSTL as follows:

$$x_t = \hat{S^1}_t + \hat{S^2}_t + \cdots + \hat{S^n}_t + \hat{T}_t + \hat{R}_t \tag{5}$$

where n denotes the number of distinct seasonal patterns decomposed by the MSTL. In our study, we use the R [34] implementation of the MSTL algorithm, `mstl`, from the `forecast` package [35], [36]. MSTL also supports controlling the smoothness of the change of seasonal components extracted from the time series, i.e., configuring the `s.window` parameter. For example, by adjusting the `s.window` parameter to "periodic," the MSTL decomposition limits the change in the seasonal components to zero. This enables us to separate the deterministic seasonality from a time series. In Table I, we give the two `s.window` parameter values used in our experiments.

2) Seasonal-Trend Decomposition by Regression: The seasonal-trend decomposition by regression (STR) is a regression-based decomposition technique introduced by Dokumentov *et al.* [37]. The division is additive; hence, the decomposition accords with (5). In contrast to STL, STR is capable of incorporating multiple external regressors to the decomposition procedure while allowing to account for external factors that may influence the seasonal patterns in a time series. However, to make our comparisons unbiased, we use STR in the default mode, without including any exogenous regressors. In R, the STR algorithm is available through the `AutoSTR` function from the `stR` package [32].

3) Trigonometric, Box-Cox, ARMA, Trend, Seasonal: As highlighted in Section II, the TBATS model was developed to handle complex seasonal patterns present in a time series [18]. This method is currently established as a state-of-the-art technique to forecast time series with multiple seasonal cycles. Particularly, the inclusion of trigonometric expression terms has enabled TBATS to identify sophisticated seasonal terms in a time series (for details, see De Livera *et al.* [18]).

In our seasonal decomposition step, we use TBATS as a deseasonalization technique to extract the relevant seasonal components of a time series. We perform the seasonal extraction after fitting the TBATS model using the `tbats` function provided by the `forecast` package [35], [36] in R.

4) Prophet: The prophet is an automated forecasting framework developed by Taylor and Letham [38]. The main aim of this framework is to address the challenges involved in forecasting on Facebook by the employer of those authors at that time. The challenges include the task of forecasting time series with multiple seasonal cycles. The underlying model of the Prophet uses an additive decomposition layer similar to (5). However, this division introduces an additional term to model holidays as seasonal covariates. After including the holiday terms, (5) can be rewritten as follows:

$$x_t = \hat{S^1}_t + \hat{S^2}_t + \cdots + \hat{S^n}_t + \hat{T}_t + \hat{R}_t + \hat{H}_t \tag{6}$$

where $\hat{H}_t$ denotes the holiday covariates in the model that represents the effects of holidays. Likewise, in TBATS, we use Prophet in the decomposition layer to obtain the multiple seasonal components present in a time series. We achieve this by applying the Prophet algorithm available through the `prophet` package in R [39].

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

5) Fourier Transformation: The Fourier terms are a flexible approach to model periodic effects in a time series [40]. For example, let x_t be an observation of time series X at time t. The seasonal terms relevant to x_t can be approximated by the Fourier terms as follows:

$$\sin\left(\frac{2\pi kt}{s_1}\right), \cos\left(\frac{2\pi kt}{s_1}\right), \ldots, \sin\left(\frac{2\pi kt}{s_n}\right), \cos\left(\frac{2\pi kt}{s_n}\right) \quad (7)$$

where s_n refers to the nth seasonal periodicity in the time series. Thereby, we can define an amount of n seasonal periodicities available in a time series. The parameter k in (7) is the number of sin, cos pairs used for the transformation process. This essentially controls the momentum of the seasonality, where a higher k allows to represent a seasonal pattern that changes more quickly compared with a lower k. In our case, for each seasonal periodicity in the time series, a separate k must be introduced. We generate these Fourier terms using the `fourier` function available in the `forecast` package. In our experiments, we use a parameter grid, which ranges from $k = 1$ to $k = s/2$, to determine the optimal k values in the Fourier terms. Moreover, we consider $k = 1$ (with least number of k) as a special use case in the Fourier terms and report separately as a variant of LSTM-MSNet.

The overall summary of the aforementioned methods is shown in Table I. Here, the Package column provides a reference to the software implementation used in our experiments. Table I, furthermore, indicates the type of seasonalities extracted by each method.

D. Recurrent Layer

The second layer, i.e., the recurrent Layer, is the primary prediction module of LSTM-MSNet, equipped with LSTMs. RNNs, and in particular LSTMs, have been embraced by many fields that involve sequence modeling tasks, such as natural language processing [41], speech recognition [42], and image generation [43], and, more recently, have received a great amount of attention in time-series research [5]–[7], [25], [44].

In the LSTM, the gating mechanism together with the self-contained memory cell enables the network to capture nonlinear long-term temporal dependencies in a sequence. We configure the input and forget gates of the LSTM network to include the previous state of the memory cell (C_{t-1}). This configuration is also known as "LSTM with peephole connections," in which the hidden state (h_t) at time t can be computed from the following equations:

$$i_t = \sigma(W_i \cdot h_{t-1} + U_i \cdot x_t + P_i \cdot C_{t-1} + b_i) \quad (8)$$

$$f_t = \sigma(W_f \cdot h_{t-1} + U_f \cdot x_t + P_f \cdot C_{t-1} + b_f) \quad (9)$$

$$\tilde{C}_t = \tanh(W_c \cdot h_{t-1} + U_c \cdot x_t + b_c) \quad (10)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (11)$$

$$o_t = \sigma(W_o \cdot h_{t-1} + U_o \cdot x_t + P_o \cdot C_t + b_o) \quad (12)$$

$$h_t = o_t \odot \phi(C_t) \quad (13)$$

where W_i, W_f, W_o, and W_c represent the weight matrices of input gate, forget gate, output gate, and memory cell gates, respectively, while x_t is the input at time t. Also, U_i, U_f, U_o, and U_c denote the corresponding input weight matrices, and P_i, P_f, and P_o are the respective peephole weight matrices. The biases of the gates are represented by b_i, b_f, b_o, and b_c. $\tilde{C}_t$ refers to the candidate cell state, which is used to update the state of the original memory cell C_t [see 11]. In these equations, $\odot$ represents the elementwise multiplication operation, σ represents the logistic sigmoid activation function, and ϕ stands for the hyperbolic tangent function, i.e., tanh. Furthermore, we use tanh as our hidden update activation function in (10).

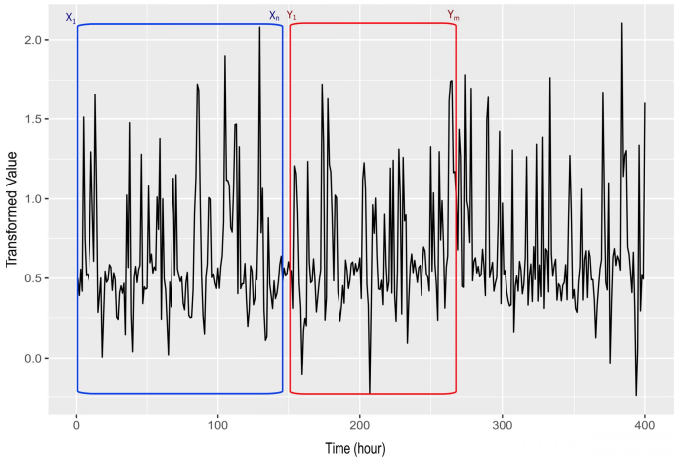

Fig. 3. Example of applying the MW approach to a preprocessed and seasonally adjusted time series X_i, i.e., after the normalization and variance stabilization and decomposition layers. Here, $\{x_1, x_2, \ldots, x_n\}$ represents the input window, and $\{y_1, y_2, \ldots, y_m\}$ corresponds to the output window.

1) Moving Window Transformation: As a preprocessing step, we transform the past observations of time series (X_i) into multiple pairs of input and output frames using a moving window (MW) strategy. Later, these frames are used as the primary training source of LSTM-MSNet.

In summary, the MW strategy converts a time series X_i of length K into $(K - n - m)$ records, where each record has an amount of $(m + n)$ observations. Here, m refers to the length of the output window, and n is the length of the input window. These frames are generated according to the multi-input–multi-output (MIMO) principle used in multistep forecasting, which directly predicts all the future observations up to the intended forecasting horizon X_i^M. Training the NNs in this way has the advantage of avoiding the potential error accumulation at each forecasting step [6], [12], [45]. Therefore, we choose the size of the output window m equivalent to the length of the intended forecast horizon M ($m = M$, the list of M values used for the benchmark data sets is summarized in Table II, under column M). Fig. 3 illustrates an example of applying the MW approach to the hourly time series T48 of the M4 data set.

To train the LSTM-MSNet, we use $(K - m)$ many observations from time series X_i and reserve the last output window for network validation and model hyperparameter tuning.

2) Training Paradigms: In this article, we propose to use the output of the decomposition layer in two different ways. These paradigms can be distinguished by the time-series components used in the MW process and later in the LSTM-MSNet training procedure. In the following, we provide a short overview of these two paradigms shown in Fig. 4.

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

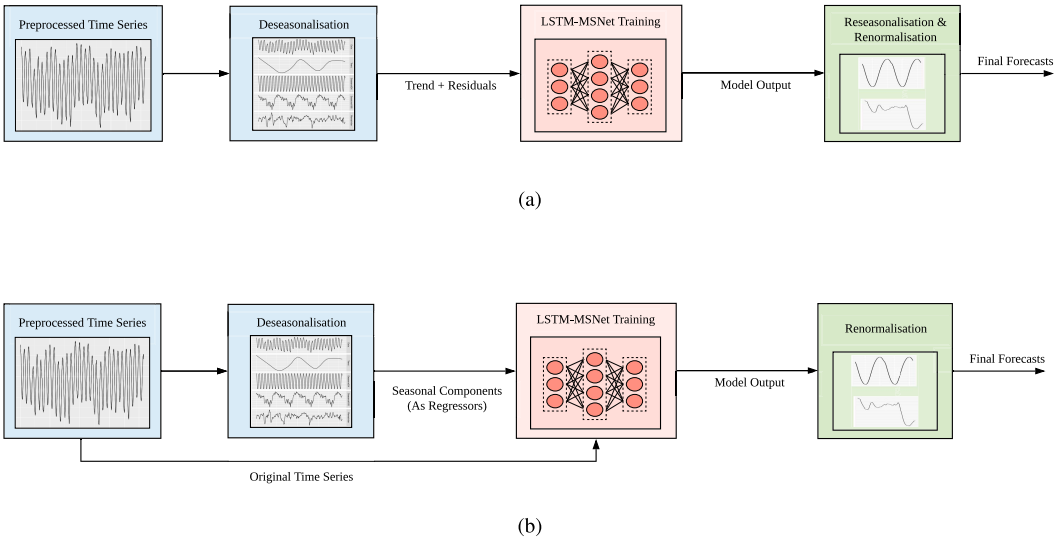

Fig. 4. Overview of the proposed LSTM-MSNet training paradigms. In the DS approach, the deseasonalized time series are used to train the LSTM-MSNet. Here, a reseasonalization phase is required as the target MW patches are seasonally adjusted, whereas, in the SE approach, the seasonal values extracted from the deseasonalization phase are employed as exogenous variables, along with the original time series to train the LSTM-MSNet. Here, a reseasonalization phase is not required as the target MW patches contain the original distribution of the time series. (a) Proposed DS training paradigm used to train the LSTM-MSNet. (b) Proposed SE training paradigm used to train the LSTM-MSNet.

TABLE II
DATA SET STATISTICS

Data set	N	K	T	S	M
M4-Hourly Dataset	414	700	hourly	(24, 168)	48
AusGrid-Energy Dataset (3 Months)	300	4704	half-hourly	(48, 336)	96
AusGrid-Energy Dataset (2 Years)	300	17600	hourly	(24, 168, 8766)	24
Traffic	963	700	hourly	(24, 168)	24

a) Deseasonalized approach: This approach uses seasonally adjusted time series as MW patches to train the LSTM-MSNet. Since the seasonal components are not included in deseasonalized approach (DS) for the training procedure, a reseasonalization technique is later introduced in the postprocessing layer of LSTM-MSNet to ascertain the corresponding multiple seasonal components of the time series.

b) Seasonal exogenous approach: This second approach uses the output of the preprocessing layer, together with the seasonal components extracted from the multiseasonal decomposition as external variables. Here, in addition to the normalized time series (without the deseasonalization phase), the seasonal components relevant to the last observation of the input window are used as exogenous variables in each input window. As the original components of the time series are used in the training phase of the seasonal exogenous approach (SE), the LSTM-MSNet is expected to forecast all the components of a time series, including the relevant multiseasonal patterns. Therefore, a reseasonalization stage is not required by SE.

In summary, DS supplements the LSTM-MSNet by excluding the seasonal factors in the LSTM-MSNet training procedure. This essentially minimizes the overall training complexity of the LSTM-MSNet. In contrast, SE supplements LSTM-MSNet in the form of exogenous variables that assist in modeling the seasonal trajectories of a time series.

The DS paradigm can be seen as a boosting ensemble technique [46], where the deseasonalization process is a weak base learner that is subsequently supplemented by the LSTM, which is trained on the remainder of the base learner. Here, the complexity of the base learner, i.e., the different deseasonalization techniques, can affect the subsequent LSTM training procedure and may lead to different results. Thereby, the suitability of the base learner as a good preprocessing technique is more important than its overall forecasting accuracy. This is a conclusion that can also be drawn from the poor performance of our submission to the M4 competition [47], where we used ETS [13] and TBATS [18], which can be considered strong base learners, to fit the time series. Subsequently, an LSTM was trained on the residuals of those models. Therefore, in the DS approach, the equilibrium between the base learner and the LSTM determines the final performance of the models so that it seems worthwhile to test different such base learners.

On the other hand, the SE paradigm can be seen as an encompassing version of the DS case, where as a special case the LSTM could "learn" to subtract the exogenous input from the time series. However, during the LSTM learning phase, the LSTM only sees a particular input window at a time. This input, depending on how long a seasonal period is and where in the period of seasonality the current input comes from, may look dramatically different. Thus, learning such seemingly simple relationships from a limited set of input data may be in fact a difficult task for the LSTM. Consequently, this work empirically examines the capacity of the LSTM to learn in different practical situations.

3) LSTM Learning Scheme: As highlighted earlier, we use the past observations of time series X_i in the form of input and output windows to train the LSTM-MSNet. In this work, we follow the LSTM design guidelines recommended by Hewamalage *et al.* [25]. Fig. 5 illustrates the primary LSTM learning architecture of LSTM-MSNet. This consists of four components, namely, training input window layer, LSTM stacking layer, dense layer, and training output window layer.

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

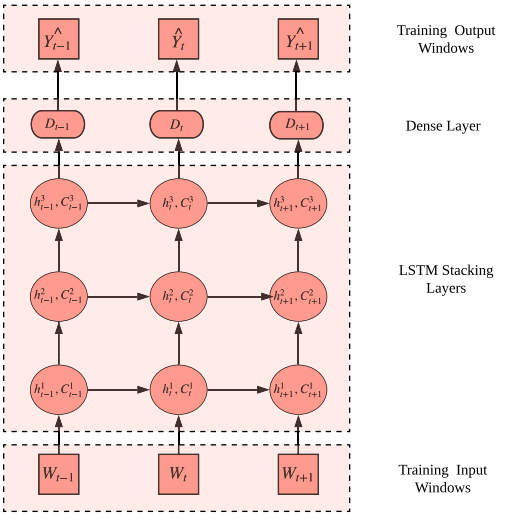

Fig. 5. Unrolled representation of a peephole connected LSTM in time, with the hidden state (h_t) and memory cell (C_t). In this architecture, h_t expects to capture the short-term dependencies in a sequence, while C_t accounts for the long-term dependencies. Here, the input window, dense layer, and projected LSTM output at time step t are denoted by W_t, D_t, and $\hat{Y}_t$, respectively.

Here, $W_t \in \mathbb{R}^n$ represents the input window at time step t. Also, the projected cell output of the LSTM at time step t is represented by $\hat{Y}_t \in \mathbb{R}^m$. Here, m represents the size of the output window, which is identical to the forecasting horizon M. Moreover, the hidden and the cell states of the LSTM are denoted by h_t and C_t. Here, h_t along with the C_t cell state provides a notion of memory to our learning scheme, while accounting for the dependencies that are longer than a given training input window. We use an affine neural layer (a fully connected layer, D_t), excluding the bias component to map each LSTM cell output h_t to the dimension of the output window m.

Before feeding these windows to the network for training, input and output windows are subjected to a local normalization process to avoid possible network saturation effects caused by the bounds of the network activation functions [7]. In the DS approach, we use the trend component of the last value of the input window as a local normalization factor, whereas in SE, we use the mean value of each input window as the normalization factor. Afterward, these factors are subtracted from each data point in the corresponding input and output windows. The abovementioned LSTM learning scheme is implemented using TensorFlow [48].

4) Loss Function: We use the L1-norm, as the primary learning objective function, which essentially minimizes the absolute differences between the target values and the estimated values. This has the advantage of being more robust to anomalies in the time series. The L1-loss is given by

$$\mathcal{L}_1 = \sum_{t \varepsilon \Omega_{\text{Train}}} \left| Y_t - \hat{Y}_t \right| + \underbrace{\psi \sum_{i=1}^{p} w_i^2}_{L2\text{regularization}} \tag{14}$$

where $Y_t \in \mathbb{R}^m$ refers to the actual observations of values in the output window at time step t. The cell output of the LSTM at time step t is defined by $\hat{Y}_t$. Also, Ω_{Train} is the set of time steps used for training. We include an L2-regularization term to minimize possible overfitting of the network. In (14), ψ is the regularization parameter, w_i refers to the network weights, and p is the number of weights in the network.

E. Postprocessing Layer

The reseasonalization and renormalization are the main components of the postprocessing layer in LSTM-MSNet. Here, in the reseasonalization stage, the relevant seasonal components of the time series are added to the forecasts generated by the LSTM. This is computed by repeating the last seasonal components of the time series to the intended forecast horizon. As outlined in Section III-D2, SE does not require this phase. Next, in the renormalization phase, the generated forecasts are backtransformed to their original scale by adding back the corresponding local normalization factor and taking the exponent of the values. The final forecasts are obtained by multiplying this vector by the scaling factor used for the normalization process.

IV. Experiments

In this section, we evaluate the proposed variants of the LSTM-MSNet framework on three time-series data sets. First, we describe the data sets, error metrics, hyperparameter selection method, and benchmarks used in our experimental setup. Then, we provide a detailed analysis of the results obtained.

A. Data Sets

We use three benchmark time-series data sets that present multiple seasonal cycles. Following is a brief overview of these data sets.

1) *M4-Hourly Data Set [30]:* Hourly data set from the M4 forecasting competition.
2) *AusGrid-Energy Data Set [1]:* Half-hourly data set, representing energy consumption of 300 households in Australia. We select general consumption (GC letter code in the data set) as the primary measure of energy consumption in households. First, we extract a subset of three months of half-hourly data (2012 July–2012 October). Then, to evaluate LSTM-MSNet on multiple seasonal patterns, we aggregate the original half-hourly time series to hourly time series and extend the extraction to two years (2010 July–2012 July), considering three seasonalities: daily, weekly, and yearly.
3) *Traffic Data Set [49]:* A collection of hourly time series, representing the traffic occupancy rate of different car lanes of San Francisco bay area freeways.

Table II summarizes the statistics of the data sets used in our experiments. Here, N denotes the number of time series, K denotes the length of each time series, T denotes the sampling rate of the time series, S represents the different seasonal cycles present in the time series, and M is the relevant forecast horizon. Moreover, we choose the size of the input window n equivalent to $M * 1.25$, following the heuristic proposed in [7] and [25].

We plot the seasonal components of our benchmark data sets to investigate the diversity of their seasonal distributions.

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

For simplicity, we use MSTL as the primary decomposition technique to extract the multiple seasonal components from a time series. From Fig. 6, it is evident that there exists a high variation of seasonality among the time series in the M4 data set. This can be attributed to a less homogeneous nature of the time series and different start/end calendar dates. On the other hand, it is clear that the distribution of the seasonal components is similar among the time series in the AusGrid-Energy and Traffic data sets, which shows less variation compared with the M4 data set, as the time series are homogeneous in the sense that they are all related to household energy consumption and follow identical calendar dates. This seasonal diversity present in our benchmark data sets enables us to assess the robustness of the LSTM-MSNet framework under different seasonality conditions, i.e., inhomogeneous and homogeneous seasonalities.

B. Error Metrics

To assess the accuracy of LSTM-MSNet against the benchmarks, we use two evaluation metrics commonly found in the forecasting literature, namely, the symmetric Mean Absolute Percentage Error (sMAPE) and the mean absolute scaled error (MASE) [50]. The sMAPE and MASE are defined as follows:

$$\text{sMAPE} = \frac{2}{m}\sum_{t=1}^{m}\left(\frac{|F_t - Y_t|}{|F_t| + |Y_t|}\right) \tag{15}$$

$$\text{MASE} = \frac{1}{m}\frac{\sum_{t=1}^{m}|F_t - Y_t|}{\frac{1}{n-S}\sum_{t=S+1}^{n}|Y_t - Y_{t-S}|} \tag{16}$$

where Y_t represents the observation at time t, and F_t is the generated forecast. Also, m denotes the number of data points in the test set, and n is the number of observations in the training set of a time series. We define S, as the frequency of the highest available seasonality in the given time series (e.g., $S = 168$, $S = 336$, $S = 8766$, and $S = 168$ for the M4-Hourly, AusGrid-Energy, and Traffic data sets, respectively). Furthermore, when calculating the sMAPE values for the energy data sets and the traffic data set, to avoid problems for zero forecasts and actual observations, we add a constant term of $\epsilon = 1$ to the denominator of (15). To provide a broader overview of the error distributions, we compute the mean and median of these primary error measures. This includes the mean of the sMAPEs (Mean sMAPE), median of the sMAPEs (Median sMAPE), mean of the MASEs (Mean MASE), and median MASEs (Median MASE).

C. Statistical Tests of the Results

We use the nonparametric Friedman rank-sum test to assess the statistically significant differences within the compared forecasting methods. Also, Hochberg's post-hoc procedure is used to further examine these differences [51].[1] The sMAPE error measure is used to perform the statistical testing, with a significance level of $\alpha = 0.05$.

[1]More information can be found on the thematic web site of SCI2S about Statistical Inference in Computational Intelligence and Data Mining http://sci2s.ugr.es/sicidm

TABLE III
RECURRENT LAYER HYPERPARAMETER GRID

Model Parameter	Minimum value	Maximum value
LSTM-cell-dimension	20	50
Mini-batch-size	20	80
Epoch-size	2	10
Maximum-epochs	10	40
Hidden Layers	1	2
Gaussian-noise-injection	10^{-4}	$8 \cdot 10^{-4}$
L2-regularisation-weight	10^{-4}	$8 \cdot 10^{-4}$

D. Hyperparameter Tuning and Optimization

The recurrent layer in LSTM-MSNet has various hyperparameters. This includes LSTM cell dimension, number of epochs, hidden layers, minibatch size, and model regularization terms. In the optimization framework, we use COntinuous COin Betting [52] as our primary learning algorithm to train the network, which does not require the tuning of the learning rate. In this way, we minimize the overall amount of hyperparameters to be tuned in the network.

Furthermore, we automate the hyperparameter selection process by employing a Bayesian global optimization methodology that autonomously discovers the optimal set of parameters of an unknown function. Compared with other parameter optimization selection techniques, such as random search and grid search, the Bayesian optimization strategy is considered as a more systematic and/or more efficient approach. This is because, when determining the current pool of potential hyperparameters for validation, it takes into account the previously visited hyperparameter values in a nontrivial way [53]. In our experiments, we use the Python implementation of the sequential model-based algorithm configuration (SMAC) [54] that implements a version of the Bayesian hyperparameter optimization process. Table III summarizes the bounds of the hyperparameter values used throughout the LSTM-MSNet learning process, represented by the respective minimum and maximum values.

E. Benchmarks and LSTM-MSNet Variants

We compare our developments against a collection of current state-of-the-art techniques in forecasting with multiple seasonal cycles. This includes Tbats [18], Prophet [38], and FFORMA [55]. We also use two variants of Dynamic-Harmonic-Regression [35] as the benchmarks. In our experiments, Dynamic-Harmonic-Regression (T) represents the variant that uses the `tslm` function from the `forecast` package, whereas the Dynamic-Harmonic-Regression (A) variant uses the `auto.arima` function from the `forecast` package.

Based on the training paradigms defined in Section III-D2, we introduce variants of the LSTM-MSNet methodology for our comparative evaluation. These methods are summarized in Table IV, represented by the corresponding training paradigm and decomposition technique. Moreover, as the baseline, we use LSTM-MSNet, excluding the seasonal decomposition phase. In other words, we use original observations of the time series, without using DS or SE, to train the LSTM-MSNet. This is referred to as LSTM-Baseline in our experiments.

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

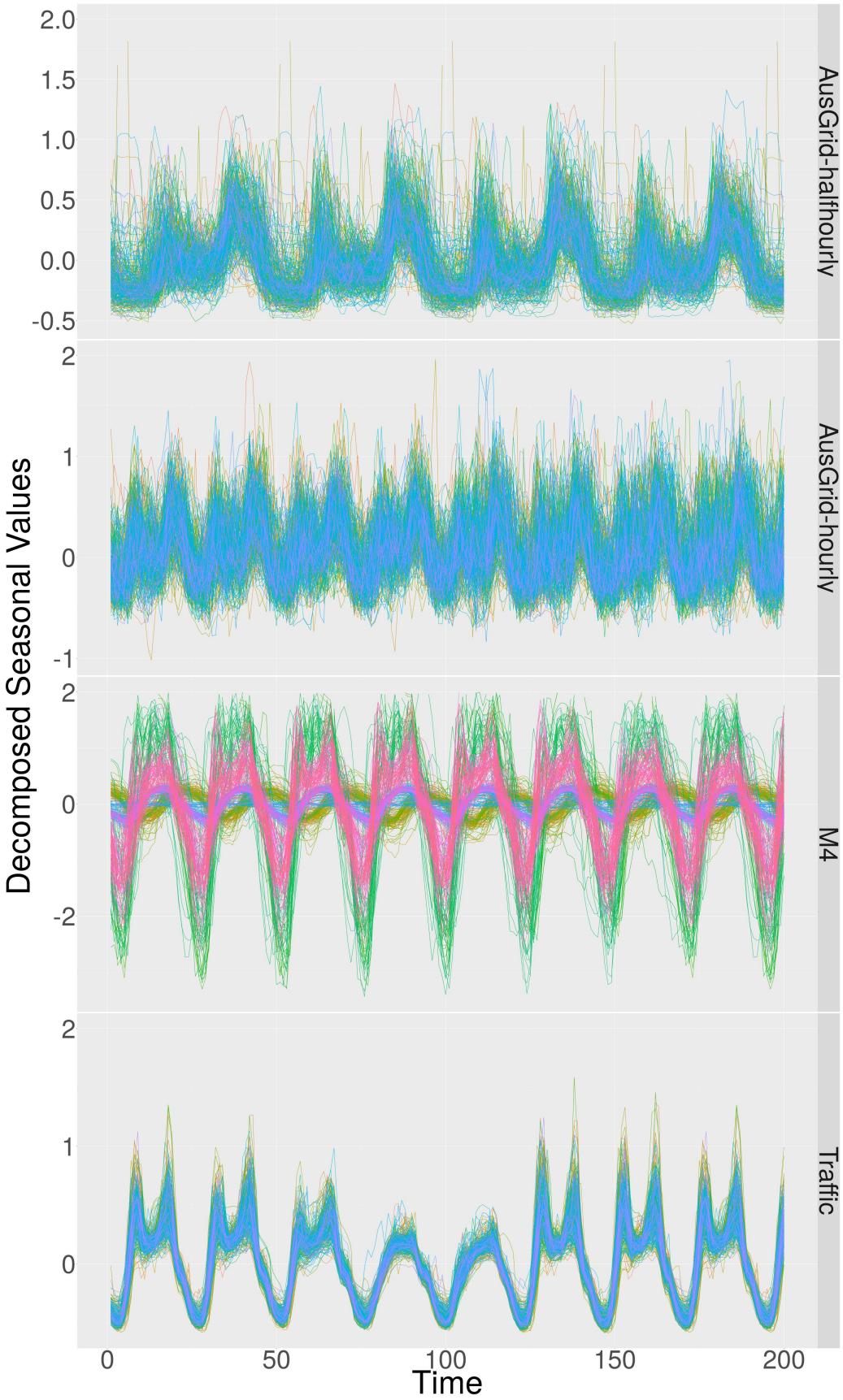

Fig. 6. Seasonal components' distributions of the sum of multiple seasonalities extracted from the AusGrid-Energy (half hourly), AusGrid-Energy (hourly), M4, and Traffic data sets, by applying the MSTL decomposition technique to the initial 200 data points of each time series.

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE IV
SUMMARY OF LSTM-MSNet VARIANTS

LSTM Variant	Training Paradigm	Decomposition Technique
LSTM-MSTL-DS	DS	MSTL (*s.window*=periodic)
LSTM-MSTL-7-DS	DS	MSTL (*s.window*=7)
LSTM-STR-DS	DS	STR
LSTM-Prophet-DS	DS	Prophet
LSTM-TBATS-DS	DS	TBATS
LSTM-Prophet-DS	DS	Prophet
LSTM-MSTL-SE	SE	MSTL (*s.window*=periodic)
LSTM-MSTL-7-SE	SE	MSTL (*s.window*=7)
LSTM-STR-SE	SE	STR
LSTM-Prophet-SE	SE	Prophet
LSTM-TBATS-SE	SE	TBATS
LSTM-Prophet-SE	SE	Prophet
LSTM-Fourier-SE	SE	Fourier Transformation
LSTM-Fourier-SE ($k = 1$)	SE	Fourier Transformation

TABLE V
RNN ARCHITECTURE COMPARISON

	M4 sMAPE		AusGrid-Energy (Half-hourly) sMAPE		AusGrid-Energy (Hourly) sMAPE		Traffic sMAPE	
Method	Mean	Median	Mean	Median	Mean	Median	Mean	Median
LSTM-MSTL-DS	**0.1069**	**0.0652**	0.1587	0.1498	0.3481	0.3415	0.0226	0.0188
GRU-MSTL-DS	0.1116	0.0723	0.1635	0.1558	0.3420	0.3371	0.0224	0.0188
ERNN-MSTL-DS	0.1109	0.0719	0.1633	0.1571	0.3421	0.3371	0.0221	0.0184
LSTM-Fourier-SE ($k = 1$)	0.1387	0.0566	0.1525	0.1428	**0.2590**	**0.2473**	**0.0151**	**0.0116**
GRU-Fourier-SE ($k = 1$)	0.1376	0.0582	**0.1513**	**0.1379**	0.2608	0.2513	0.0158	0.0121
ERNN-Fourier-SE ($k = 1$)	0.1381	0.0583	**0.1513**	0.1392	0.2611	0.2532	0.0182	0.0141

F. Computational Performance

We also report the computational costs in the execution time of our proposed LSTM-MSNet variants and the benchmark models on the aggregated hourly energy data set. The experiments are run on an Intel i7 processor (3.2 GHz), with two threads per core, six cores per socket, and 64 GB of main memory.

G. RNN Architecture Comparison

We perform a set of preliminary experiments to assess the compatibility of our proposed framework with other RNN architectures. To compare against the LSTM cell, we use popular RNN units, namely, the Elman RNN (ERNN) cell [56] and the GRU cell [57], which are commonly used for sequence modeling tasks. Table V shows the evaluation summary of these RNN architectures on the benchmark data sets. In our experiments, we use the MSTL-DS and the Fourier-SE ($k = 1$) variants of each RNN architecture to represent DS and SE training paradigms, respectively. For each data set, the results of the best performing methods are marked in boldface.

We observe that the overall p-value of the Friedman rank-sum test for the M4 data set is 8.57×10^{-11}, which is highly significant. The LSTM-MSTL-DS method performs best and achieves significantly better results than the other methods. The overall p-value for the AusGrid-Energy (Half-hourly) is 2.89×10^{-6}. Therefore, GRU-Fourier-SE ($k = 1$) performs significantly better than the rest of the methods. Also, for the AusGrid-Energy (Hourly) data set, the Friedman rank sum test gives an overall p-value of 2.16×10^{-10}, and LSTM-Fourier-SE ($k = 1$) is used as the control method, which performs best. Finally, the overall p-value for the Traffic data set is $p < 10^{-10}$. Therefore, the differences among the methods are significant. Also, according to Table V we see that the LSTM-Fourier-SE ($k = 1$) achieves significantly better results than the other methods. It is also evident that our framework can be employed with any other RNN architecture, such as the ERNN cell and the GRU cell. However, according to Table V, overall we see that the LSTM cell-based variants achieve competitive results compared with the ERNN and GRU cells. Therefore, we use the LSTM cell as the primary RNN architecture in our experiments.

TABLE VI
M4 DATA SET RESULTS

Method	Mean sMAPE	Median sMAPE	Mean MASE	Median MASE
LSTM-MSTL-DS	**0.1069**	0.0652	0.7131	0.6340
LSTM-STR-DS	0.1099	0.0734	0.7528	0.6554
LSTM-Prophet-DS	0.1131	0.0623	0.8032	0.6511
FFORMA	0.1151	**0.0404**	0.7131	**0.4750**
LSTM-MSTL-7-DS	0.1187	0.0795	0.7840	0.7092
LSTM-TBATS-DS	0.1241	0.0589	0.8734	0.7036
Dynamic-Harmonic-Regression (A)	0.1253	0.0524	**0.6937**	0.5732
LSTM-MSTL-SE	0.1275	0.0518	0.9642	0.6986
LSTM-STR-SE	0.1285	0.0559	0.9308	0.6581
TBATS	0.1309	0.0477	0.7781	0.5390
Prophet	0.1334	0.0689	0.9685	0.7415
LSTM-Fourier-SE	0.1345	0.0555	0.9541	0.6606
LSTM-TBATS-SE	0.1368	0.0568	0.9675	0.6492
LSTM-MSTL-7-SE	0.1388	0.0631	0.9796	0.7334
LSTM-Prophet-SE	0.1393	0.0591	0.9737	0.6732
LSTM-Fourier-SE ($k = 1$)	0.1387	0.0566	1.0176	0.6581
LSTM-Baseline	0.1427	0.0569	1.0675	0.7243
Dynamic-Harmonic-Regression (T)	0.1611	0.1456	0.7821	0.6439

H. Results

Table VI summarizes the evaluation results of all the LSTM-MSNet variants and benchmarks for the 414 hourly series of the M4 competition data set, ordered by the first column, which is the Mean sMAPE. For each column, the results of the best performing method(s) are marked in boldface. According to Table VI, the proposed LSTM-MSTL-DS variant obtains the best Mean SMAPE, while FFORMA achieves the best Median SMAPE. We see that regarding the mean MASE, Dynamic-Harmonic-Regression with the auto.arima variant performs better than the rest of the benchmarks, whereas FFORMA outperforms the proposed LSTM variants, in terms of the median MASE. Also, on average, the LSTM-MSNet variants with the DS training paradigm achieve better accuracies compared with those of SE. Furthermore, the proposed LSTM-MSTL-DS variant consistently outperforms many state-of-the-art methods, such as TBATS, Prophet, and Dynamic-Harmonic-Regression variants in terms of Mean sMAPE. It is also noteworthy to mention that all the proposed LSTM-MSNet variants outperform our baseline model, i.e., LSTM-Baseline, in terms of mean sMAPE and mean MASE.

Table VII shows the results of the statistical testing evaluation. Adjusted p-values calculated from the Friedman test with Hochberg's post-hoc procedure are presented. A horizontal line is used to separate the methods that perform significantly worse than the best performing method. The overall result of the Friedman rank-sum test is a p-value of 2.91×10^{-10}, which is highly significant. The FFORMA method performs best and is used as the control method. Also, according to Table VII, we see that the FFORMA achieves significantly better results than the other methods.

Table VIII shows the evaluation summary for the 300 half-hourly series of the AusGrid-Energy data set. The NA values in the table represent the models that could not complete the execution within a time frame of six days. It can be seen

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE VII
SIGNIFICANCE TESTING FOR M4 DATA SET

Method	p_{Hoch}
FFORMA	-
LSTM-MSTL-DS	8.74×10^{-6}
LSTM-STR-DS	8.74×10^{-6}
Dynamic-Harmonic-Regression (A)	7.07×10^{-7}
TBATS	2.75×10^{-8}
LSTM-MSTL-7-DS	1.01×10^{-9}
LSTM-Prophet-DS	3.32×10^{-13}
LSTM-STR-SE	2.41×10^{-21}
LSTM-TBATS-SE	1.23×10^{-22}
LSTM-TBATS-DS	1.68×10^{-25}
LSTM-Fourier-SE	2.80×10^{-30}
LSTM-Fourier-SE ($k = 1$)	1.75×10^{-35}
LSTM-MSTL-SE	1.09×10^{-40}
LSTM-Prophet-SE	1.53×10^{-51}
Prophet	5.10×10^{-61}
LSTM-MSTL-7-SE	6.38×10^{-65}
Dynamic-Harmonic-Regression (T)	1.64×10^{-66}
LSTM-Baseline	1.93×10^{-79}

TABLE VIII
AUSGRID-ENERGY (HALF-HOURLY) DATA SET RESULTS

Method	Mean sMAPE	Median sMAPE	Mean MASE	Median MASE
LSTM-MSTL-SE	**0.1475**	**0.1369**	0.7461	0.5900
LSTM-Prophet-DS	0.1478	0.1397	**0.7350**	**0.5809**
LSTM-TBATS-SE	0.1479	0.1374	0.7471	0.5984
LSTM-Prophet-SE	0.1511	0.1397	0.7589	0.6071
LSTM-MSTL-7-SE	0.1523	0.1422	0.7651	0.6115
LSTM-Fourier-SE ($k = 1$)	0.1525	0.1428	0.7694	0.6052
LSTM-Fourier-SE	0.1527	0.1414	0.7676	0.6130
LSTM-Baseline	0.1561	0.1430	0.7838	0.6263
LSTM-MSTL-DS	0.1587	0.1498	0.7729	0.6458
TBATS	0.1597	0.1467	0.8221	0.6506
LSTM-MSTL-7-DS	0.1620	0.1538	0.7932	0.6599
FFORMA	0.1808	0.1708	0.9615	0.6953
Prophet	0.1848	0.1766	0.8935	0.7346
Dynamic-Harmonic-Regression (A)	0.1919	0.1808	0.9138	0.7599
Dynamic-Harmonic-Regression (T)	0.2059	0.1847	0.9773	0.8567
LSTM-TBATS-DS	0.3092	0.3014	1.3011	1.0601
LSTM-STR-DS	NA	NA	NA	NA
LSTM-STR-SE	NA	NA	NA	NA

TABLE IX
SIGNIFICANCE TESTING FOR AUSGRID-ENERGY (HALF-HOURLY) DATA SET

Method	p_{Hoch}
LSTM-MSTL-SE	-
LSTM-TBATS-SE	0.600
LSTM-Prophet-DS	0.233
LSTM-Prophet-SE	8.89×10^{-4}
LSTM-MSTL-7-SE	7.62×10^{-6}
LSTM-Fourier-SE	6.69×10^{-7}
LSTM-Fourier-SE ($k = 1$)	4.56×10^{-7}
TBATS	7.48×10^{-8}
LSTM-MSTL-DS	2.76×10^{-9}
LSTM-Baseline-DS	5.35×10^{-15}
LSTM-MSTL-7-DS	1.19×10^{-22}
FFORMA	5.96×10^{-55}
Prophet	4.65×10^{-62}
Dynamic-Harmonic-Regression (A)	5.54×10^{-64}
Dynamic-Harmonic-Regression (T)	3.32×10^{-64}
LSTM-TBATS-DS	1.94×10^{-91}

TABLE X
AUSGRID-ENERGY (HOURLY) DATA SET RESULTS

Method	Mean sMAPE	Median sMAPE	Mean MASE	Median MASE
LSTM-Fourier-SE ($k = 1$)	**0.2590**	**0.2473**	**0.7189**	**0.6455**
LSTM-MSTL-SE	0.2638	0.2626	0.7286	0.6652
LSTM-Prophet-SE	0.2653	0.2575	0.7332	0.6819
LSTM-TBATS-SE	0.2665	0.2676	0.7346	0.6753
LSTM-Fourier-SE	0.2672	0.2572	0.7399	0.6790
LSTM-Baseline	0.2685	0.2660	0.7408	0.6852
FFORMA	0.2692	0.2613	0.7360	0.6573
LSTM-Prophet-DS	0.2749	0.2761	0.7526	0.6913
LSTM-MSTL-7-SE	0.2884	0.2785	0.7930	0.7219
LSTM-MSTL-7-DS	0.3172	0.3059	0.8250	0.7790
TBATS	0.3177	0.3072	0.8179	0.7861
Prophet	0.3390	0.3369	0.8757	0.8156
LSTM-TBATS-DS	0.3414	0.2971	1.0640	0.7458
LSTM-MSTL-DS	0.3480	0.3415	0.9123	0.8550
Dynamic-Harmonic-Regression (T)	0.3530	0.3460	0.9015	0.8692
Dynamic-Harmonic-Regression (A)	NA	NA	NA	NA
LSTM-STR-DS	NA	NA	NA	NA
LSTM-STR-SE	NA	NA	NA	NA

that the proposed LSTM-MSTL-SE variant outperforms all the benchmarks in terms of the Mean sMAPE and Median sMAPE. Meanwhile, the proposed LSTM-Prophet-DS variant achieves the best accuracy with respect to Mean MASE and Median MASE. Here, in the majority of the cases, the LSTM variants that use SE as the training paradigm obtains better forecasts, which is contrary to our previous findings from the M4 competition data set. Also, several LSTM-MSNet variants with the DS training paradigm display poor performance compared with the LSTM-Baseline. Most importantly, we observe that the proposed LSTM variants LSTM-MSTL-SE and LSTM-Prophet-DS consistently surpass the current state of the art in all performance metrics.

The Friedman rank sum test gives an overall p-value of $p < 10^{-10}$. Therefore, the differences among the benchmarks are highly significant. According to Table IX, we see that the LSTM-MSTL-SE performs best and is used as the control method. Moreover, the LSTM-MSNet variants, LSTM-TBATS-SE, and LSTM-Prophet-DS do not perform significantly worse than the control method.

Table X provides the results for the evaluations on the aggregated hourly time series of the AusGrid-Energy data set. We see that the proposed LSTM-Fourier-SE ($k = 1$) variant achieves the best results on each performance metric and outperforms the rest of the benchmarks. Also, among the proposed variants, we observe that the LSTM-MSNet variants with the SE training paradigm outperform their counterparts, i.e., the LSTM-MSNet variants with the DS training paradigm. Furthermore, the LSTM-Baseline method performs better than the LSTM-MSNet variants with the DS training paradigm. Nevertheless, consistent with our previous findings from Table VIII, the proposed LSTM-MSNet variants outperform the current state-of-the-art techniques: FFORMA, TBATS, and Prophet.

The overall result of the Friedman rank-sum test is a p-value of 2.76×10^{-10}, which means that the results are highly significant. According to Table XI, the LSTM-Fourier-SE ($k = 1$) performs best and is used as the control method. We see that the LSTM-MSNet variants, LSTM-MSTL-SE, LSTM-TBATS-SE, LSTM-Prophet-SE, LSTM-Fourier-SE, and FFORMA do not perform significantly worse than the control method.

Table XII provides the results for the evaluations on the hourly time series of the San Francisco traffic data set. We observe that the results are similar to the previous findings from Table X, where the proposed LSTM-Fourier-SE ($k = 1$) variant outperforms the rest of the benchmarks. Moreover, the LSTM-MSNet variants with the SE training paradigm achieve better results compared with the LSTM-MSNet variants with the DS training paradigm. Also, consistent with our

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE XI
SIGNIFICANCE TESTING FOR AUSGRID-ENERGY (HOURLY) DATA SET

Method	p_{Hoch}
LSTM-Fourier-SE ($k = 1$)	-
LSTM-MSTL-SE	0.265
LSTM-TBATS-SE	0.128
LSTM-Prophet-SE	0.128
LSTM-Fourier-SE	0.128
FFORMA	0.084
LSTM-Prophet-DS	0.041
LSTM-Baseline	0.016
LSTM-TBATS-DS	2.70×10^{-9}
LSTM-MSTL-7-SE	2.99×10^{-15}
TBATS	3.27×10^{-23}
LSTM-MSTL-7-DS	2.25×10^{-23}
Prophet	1.98×10^{-44}
Dynamic-Harmonic-Regression (T)	3.46×10^{-45}
LSTM-MSTL-DS	1.40×10^{-57}

TABLE XII
TRAFFIC DATA SET RESULTS

Method	Mean sMAPE	Median sMAPE	Mean MASE	Median MASE
LSTM-Fourier-SE ($k = 1$)	**0.0151**	**0.0116**	**0.6102**	**0.5035**
LSTM-Fourier-SE	0.0179	0.0143	0.7213	0.6178
LSTM-MSTL-SE	0.0177	0.0142	0.7107	0.6159
LSTM-TBATS-SE	0.0186	0.0151	0.7467	0.6495
LSTM-Prophet-SE	0.0194	0.0157	0.7828	0.6744
LSTM-Baseline	0.0193	0.0158	0.7853	0.6777
LSTM-MSTL-7-SE	0.0213	0.0179	0.8760	0.7801
LSTM-Prophet-DS	0.0219	0.0191	0.8951	0.7928
LSTM-MSTL-DS	0.0232	0.0196	0.9529	0.8321
FFORMA	0.0245	0.0190	0.9177	0.7582
LSTM-STR-DS	0.0249	0.0209	0.9918	0.8710
LSTM-MSTL-7-DS	0.0303	0.0273	1.2619	1.1576
TBATS	0.0247	0.0215	1.0412	0.8861
LSTM-STR-SE	0.0251	0.0217	1.0377	0.9334
Dynamic-Harmonic-Regression (A)	0.0303	0.0259	1.1987	1.0555
Prophet	0.0332	0.0298	1.3248	1.2061
Dynamic-Harmonic-Regression (T)	0.0335	0.0286	1.1570	1.1812
LSTM-TBATS-DS	0.0789	0.0773	3.1593	2.9352

TABLE XIII
SIGNIFICANCE TESTING FOR TRAFFIC DATA SET

Method	p_{Hoch}
LSTM-Fourier-SE ($k = 1$)	-
LSTM-MSTL-SE	1.44×10^{-15}
LSTM-Fourier-SE	4.32×10^{-21}
LSTM-TBATS-SE	6.06×10^{-42}
LSTM-Baseline	6.79×10^{-71}
LSTM-Prophet-SE	8.77×10^{-84}
LSTM-MSTL-7-SE	$p < 10^{-100}$
FFORMA	$p < 10^{-100}$
LSTM-MSTL-DS	$p < 10^{-100}$
LSTM-Prophet-DS	$p < 10^{-100}$
TBATS	$p < 10^{-100}$
LSTM-STR-DS	$p < 10^{-100}$
LSTM-STR-SE	$p < 10^{-100}$
Dynamic-Harmonic-Regression (A)	$p < 10^{-100}$
Dynamic-Harmonic-Regression (T)	$p < 10^{-100}$
LSTM-MSTL-7-DS	$p < 10^{-100}$
Prophet	$p < 10^{-100}$
LSTM-TBATS-DS	$p < 10^{-100}$

observations from the AusGrid-Energy data sets, the LSTM-Baseline method performs better than the LSTM-MSNet variants with the DS training paradigm. Furthermore, the majority of the LSTM-MSNet variants outperform the statistical benchmarks, FFORMA, TBATS, and Prophet.

The Friedman rank sum test gives an overall p-value of $p < 10^{-10}$, which means that the results are highly significant.

TABLE XIV
COMPUTATIONAL SUMMARY—AUSGRID-ENERGY DATA SET: HOURLY (IN min)

Method	Pre-processing	Model-Training & Post-processing	Total-time
Dynamic-Harmonic-Regression (T)	-	4	4
Prophet	-	90	90
LSTM-Baseline	10	140	150
LSTM-MSTL-DS	88	120	208
LSTM-MSTL-7-DS	88	120	208
LSTM-Fourier-SE ($k = 1$)	30	180	210
LSTM-Fourier-SE	34	180	214
LSTM-MSTL-SE	74	180	254
LSTM-MSTL-7-SE	74	180	254
LSTM-Prophet-DS	210	120	330
LSTM-Prophet-SE	200	180	380
TBATS	-	2304	2304
LSTM-TBATS-DS	2332	120	2452
LSTM-TBATS-SE	2318	180	2498
FFORMA	-	4320	4320

According to Table XIII, the LSTM-Fourier-SE ($k = 1$) performs best and is used as the control method. Also, we see that the LSTM-Fourier-SE ($k = 1$) achieves significantly better results than the other methods.

Also, with respect to the computational cost of the proposed variants and benchmarks, according to Table XIV, except for LSTM-TBATS-DS and LSTM-TBATS-SE, which use TBATS as the decomposition technique, we see that the proposed LSTM-MSNet variants have a lower execution time compared with TBATS and FFORMA. The Dynamic-Harmonic-Regression (T) variant and Prophet are more computationally efficient than the LSTM-MSNet variants. However, we notice that both these methods do not display competitive results compared with the LSTM-MSNet variants according to Tables VI, VIII, and X. On the contrary, LSTM-MSNet promises to deliver better performance than TBATS and FFORMA, with respect to both accuracy and computation time.

I. Discussion

It can be seen that the LSTM-Baseline results for the M4 and real-world data sets are contradictory. Even though the LSTM-Baseline model cannot outperform both DS and SE learning paradigms of LSTM-MSNet in the M4 data set, it obtains better results than the DS learning paradigm in AusGrid-Energy and Traffic data sets. These results can be interpreted by the seasonal characteristics present in these data sets (see Fig. 6). As discussed in Section IV-A, the seasonal components of the M4 data set are less homogeneous. Thus, in this scenario, learning the seasonality directly from the original time series becomes difficult for LSTM-MSNet. Hence, removing the seasonal components prior to training (DS learning paradigm) has positively contributed toward the LSTM-MSNet results in the M4 data set. This also explains the poor results of the LSTM-Baseline variant, which attempts to learn seasonality directly from the time series, without any assistance of DS and SE. We also note that the lengths of the series are likely to have an effect here, in the sense that for longer series, the amount of data is sufficient to learn seasonal patterns directly, whereas, for shorter series, deseasonalization is beneficial. However, we do not analyze this in our current study. Furthermore, we observe the highly competitive results of FFORMA in the M4 data set. FFORMA was the

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

second-best performing method in the overall M4 competition, so it can be arguably seen as optimized for this particular data set. Also, due to its nature as being an ensemble of simpler univariate techniques, it is inherently more suitable for this situation of a data set of inhomogeneous series, whereas, on the AusGrid-Energy and Traffic data sets, it is less performant. Due to the high presence of homogeneous seasonality, exclusive learning of the seasonality becomes viable for LSTM-MSNet. As a result, the LSTM-Baseline achieves better results compared with LSTM-MSNet with the DS learning paradigm. However, in this scenario, we observe that the variants of LSTM-MSNet with the SE learning paradigm achieve better results than the LSTM-Baseline model. This suggests that the extracted seasonal components have supplemented the LSTM-MSNet training procedure, in the form of external variables that assist to determine the trajectory of the multiple seasonal cycles.

In terms of decomposition techniques, MSTL, STR, and Prophet are the best-performing methods in the M4 data set, whereas, on the AusGrid-Energy and Traffic data sets, we see that the LSTM-MSNet variants with MSTL and Prophet decomposition techniques give better results. Also, the STR-based LSTM-MSNet variants are unstable on the AusGrid-Energy data sets, which can be attributed to the longer lengths of the time series. Furthermore, according to Table XIV, the MSTL and Prophet methods are computationally efficient decomposition techniques compared with TBATS.

These results indicate that our proposed LSTM-MSNet forecasting framework can be easily adapted, depending on the seasonal characteristics in a group of time series. The LSTM-MSNet with the DS learning paradigm is more suitable for situations where the origins of the time series are unknown and time series are inhomogeneous, whereas its counterpart, i.e., SE, is better if the time series are homogeneous and share similar shapes of the seasonal components. Another important finding from this study is that the RNNs are competitive in situations where groups of time series are highly homogeneous. As discussed in Section I, this is the case in many real-world applications. However, in situations such as in the M4 data set, where the groups of time series exhibit highly heterogeneous characteristics, better competitiveness of the RNNs can be achieved by applying additional preprocessing steps, such as deseasonalization (DS learning paradigm) that supplements the RNN training procedure. Apart from these observations, we also see that for longer time series, the majority of the LSTM-MSNet variants are computationally more efficient than the state-of-the-art univariate forecasting techniques, such as TBATS and FFORMA.

V. Conclusion

In this article, we have presented the LSTM-MSNet methodology, a novel, three-layered forecasting framework that is capable of forecasting a group of related time series with multiple seasonal cycles. Our methodology is based on time-series decomposition and LSTM RNNs to overcome the limitations of the current univariate state-of-the-art models by training a unified model that exploits key structures, behaviors, and patterns common within a group of time series.

We have utilized a series of decomposition techniques from the literature to extract the various forms of seasonal components in time series. Moreover, we have also discussed two training paradigms, the deseasonalized approach and the Seasonal Exogenous approach, highlighting how these decomposition techniques can be used to supplement the LSTM learning procedure. We have identified that the choice of these learning paradigms can be determined by the characteristics of the time series, where a deseasonalized approach is more suitable for situations where the series have different seasonal patterns, and the start/end dates of the series are different, whereas its counterpart, i.e., the seasonal exogenous approach, is better if the time series are homogeneous and share similar shapes of the seasonal components. In general, MSTL and Prophet are stable decomposition techniques for both shorter and longer time series and achieve competitive results with a lower computational cost. We have evaluated the proposed forecasting framework using a competition data set, two energy consumption time-series data sets, and Traffic data set that contain multiple seasonal patterns. Also, with respect to accuracy and computational time, we observe that our framework can be a competitive approach among the current state of the arts in his research space. Furthermore, somewhat contrary to widespread beliefs, the globally trained LSTM-MSNet can be computationally more efficient than many univariate forecasting methods.

As a possible future work, a hybrid version of this approach can be introduced to handle seasonalities in longer time series. Here, the deseasonalized approach can be used to model shorter seasonalities, whereas the seasonal exogenous approach can be applied to address the longer seasonalities.

References

[1] AusGrid. (2019). *Innovation and Research—Ausgrid*. Accessed: May 16, 2019. [Online]. Available: https://www.ausgrid.com.au/Industry/Innovation-and-research/

[2] C.-M. Lee and C.-N. Ko, "Short-term load forecasting using lifting scheme and ARIMA models," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5902–5911, May 2011.

[3] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. Hoboken, NJ, USA: Wiley, 2015.

[4] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," 2017, *arXiv:1703.04691*. [Online]. Available: http://arxiv.org/abs/1703.04691

[5] D. Salinas, V. Flunkert, and J. Gasthaus, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," 2017, *arXiv:1704.04110*. [Online]. Available: http://arxiv.org/abs/1704.04110

[6] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A multi-horizon quantile recurrent forecaster," 2017, *arXiv:1711.11053*. [Online]. Available: http://arxiv.org/abs/1711.11053

[7] K. Bandara, C. Bergmeir, and S. Smyl, "Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach," *Expert Syst. Appl.*, vol. 140, Feb. 2020, Art. no. 112896.

[8] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, New York, NY, USA, 2018, pp. 95–104.

[9] K. Bandara, P. Shi, C. Bergmeir, H. Hewamalage, Q. Tran, and B. Seaman, "Sales demand forecast in E-commerce using a long short-term memory neural network methodology," 2019, *arXiv:1901.04028*. [Online]. Available: http://arxiv.org/abs/1901.04028

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[10] M. Nelson, T. Hill, W. Remus, and M. O'Connor, "Time series forecasting using neural networks: Should the data be deseasonalized first?" *J. Forecasting*, vol. 18, no. 5, pp. 359–367, Sep. 1999.

[11] G. P. Zhang and M. Qi, "Neural network forecasting for seasonal and trend time series," *Eur. J. Oper. Res.*, vol. 160, no. 2, pp. 501–514, Jan. 2005.

[12] S. Ben Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 7067–7083, Jun. 2012.

[13] R. Hyndman, A. B. Koehler, J. Keith Ord, and R. D. Snyder, *Forecasting with Exponential Smoothing: The State Space Approach*. Springer, 2008.

[14] A. Harvey, S. J. Koopman, and M. Riani, "The modeling and seasonal adjustment of weekly observations," *J. Bus. Econ. Statist.*, vol. 15, no. 3, pp. 354–368, Jul. 1997.

[15] J. W. Taylor, "Short-term electricity demand forecasting using double seasonal exponential smoothing," *J. Oper. Res. Soc.*, vol. 54, no. 8, pp. 799–805, Aug. 2003.

[16] P. G. Gould, A. B. Koehler, J. K. Ord, R. Snyder, R. J. Hyndman, and F. Vahid-Araghi, "Forecasting time series with multiple seasonal patterns," *Eur. J. Oper. Res.*, vol. 191, no. 1, pp. 207–222, Nov. 2008.

[17] J. W. Taylor and R. D. Snyder, "Forecasting intraday time series with multiple seasonal cycles using parsimonious seasonal exponential smoothing," *Omega*, vol. 40, no. 6, pp. 748–757, Dec. 2012.

[18] A. M. De Livera, R. J. Hyndman, and R. D. Snyder, "Forecasting time series with complex seasonal patterns using exponential smoothing," *J. Amer. Stat. Assoc.*, vol. 106, no. 496, pp. 1513–1527, Dec. 2011.

[19] J. Nowicka-Zagrajek and R. Weron, "Modeling electricity loads in california: ARMA models with hyperbolic noise," *Signal Process.*, vol. 82, no. 12, pp. 1903–1915, Dec. 2002.

[20] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[21] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[22] Z. Tang, C. de Almeida, and P. A. Fishwick, "Time series forecasting using neural networks vs. box- jenkins methodology," *SIMULATION*, vol. 57, no. 5, pp. 303–310, Nov. 1991.

[23] M. Marseguerra, S. Minoggio, A. Rossi, and E. Zio, "Neural networks prediction and fault diagnosis applied to stationary and non stationary ARMA modeled time series," *Prog. Nucl. Energy*, vol. 27, no. 1, pp. 25–36, Jan. 1992.

[24] W. Yan, "Toward automatic time-series forecasting using neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1028–1039, Jul. 2012.

[25] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," 2019, *arXiv:1909.00590*. [Online]. Available: http://arxiv.org/abs/1909.00590

[26] G. Dudek, "Forecasting time series with multiple seasonal cycles using neural networks with local learning," in *Artificial Intelligence and Soft Computing*. Berlin, Germany: Springer, 2013, pp. 52–63.

[27] T. Fernando, S. Denman, A. McFadyen, S. Sridharan, and C. Fookes, "Tree memory networks for modelling long-term temporal dependencies," *Neurocomputing*, vol. 304, pp. 64–81, Aug. 2018.

[28] M. Han and M. Xu, "Laplacian echo state network for multivariate time series prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 238–244, Jan. 2018.

[29] F. Maria Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, "An overview and comparative analysis of recurrent neural networks for short term load forecasting," 2017, *arXiv:1705.04378*. [Online]. Available: http://arxiv.org/abs/1705.04378

[30] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: Results, findings, conclusion and way forward," *Int. J. Forecasting*, vol. 34, no. 4, pp. 802–808, Oct. 2018.

[31] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *Int. J. Forecasting*, vol. 36, no. 1, pp. 75–85, Jan. 2020.

[32] A. Dokumentov and R. J. Hyndman. (2018). *stR: STR Decomposition, R Package Version 0.4*. [Online]. Available: https://CRAN.R-project.org/package=stR

[33] R. B. Cleveland, W. S. Cleveland, and I. Terpenning, "STL: A seasonal-trend decomposition procedure based on loess," *J. Off. Stat.*, vol. 6, no. 1, p. 3, 1990.

[34] R Core Team. R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing. Vienna, Austria, 2013. [Online]. Available: http://www.R-project.org/

[35] R. J. Hyndman *et al.*, "Forecast: Forecasting functions for time series and linear models," *R Package Version*, vol. 6, no. 6, p. 7, 2015.

[36] Y. Khandakar and R. J. Hyndman, "Automatic time series forecasting: The forecast package for R," *J. Stat. Softw.*, vol. 27, no. 3, pp. 1–22, 2008.

[37] A. Dokumentov and R. J. Hyndman, "STR: A seasonal-trend decomposition procedure based on regression," Dept. Econometrics Bus. Statist., Monash Univ., Melbourne, VIC, Australia, Tech. Rep., 2015.

[38] S. Taylor and B. Letham, "Forecasting at scale," *Amer. Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[39] S. Taylor and B. Letham. (2019). *Prophet: Automatic Forecasting Procedure. R Package Version 0.5*. [Online]. Available: https://CRAN.R-project.org/package=prophet

[40] A. C. Harvey and N. Shephard, "10 structural time series models," in *Handbook of Statistics*, vol. 11. Amsterdam, The Netherlands: Elsevier, Jan. 1993, pp. 261–302.

[41] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Interspeech*, vol. 2, 2010, p. 3.

[42] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[43] K. Gregor *et al.*, "DRAW: A recurrent neural network for image generation," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, 2015, pp. 1462–1471.

[44] H.-G. Zimmermann, C. Tietz, and R. Grothmann, "Forecasting with recurrent neural networks: 12 tricks," in *Neural Networks: Tricks Trade* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2012, pp. 687–707.

[45] S. B. Taieb and A. F. Atiya, "A bias and variance analysis for multistep-ahead time series forecasting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 1, pp. 62–76, Jan. 2016.

[46] R. E. Schapire, "The boosting approach to machine learning: An overview," in *Nonlinear Estimation Classification*, D. D. Denison, M. H. Hansen, C. C. Holmes, B. Mallick, and B. Yu, Eds. New York, NY, USA: Springer, 2003, pp. 149–171.

[47] M4 Competition. (2018). *M4 Submission Repository*. Accessed: Nov. 12, 2019. [Online]. Available: https://bit.ly/2Q1BIWA

[48] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[49] California Department of Transportation. (2019). *San Francisco Traffic*. Accessed: Nov. 20, 2019. [Online]. Available: https://github.com/rofuyu/exp-trmf-nips16s

[50] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *Int. J. Forecasting*, vol. 22, no. 4, pp. 679–688, Oct. 2006.

[51] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci.*, vol. 180, no. 10, pp. 2044–2064, May 2010.

[52] F. Orabona and T. Tommasi, "Training deep networks without learning rates through coin betting," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 2157–2167.

[53] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, 2012, pp. 2951–2959.

[54] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proc. 5th Int. Conf. Learn. Intell. Optim.*, 2011, pp. 507–523.

[55] P. Montero-Manso *et al.*, "FFORMA: Feature-based forecast model averaging," *Int. J. Forecast.*, vol. 36, no. 1, pp. 86–92, Jan. 2020.

[56] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[57] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.

Authorized licensed use limited to: University College London. Downloaded on May 23,2020 at 23:55:07 UTC from IEEE Xplore. Restrictions apply.