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

Kasun Bandara, Christoph Bergmeir, Hansika Hewamalage,
Faculty of Information Technology
Monash University, Melbourne, Australia.
herath.bandara@monash.edu, christoph.bergmeir@monash.edu, hansika.hewamalage@monash.edu

***Abstract*—Generating forecasts for time series with multiple seasonal cycles is an important use-case for many industries nowadays. Accounting for the multi-seasonal patterns becomes necessary to generate more accurate and meaningful forecasts in these contexts. In this paper, we propose Long Short-Term Memory Multi-Seasonal 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 are 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 Long Short-Term Memory network (LSTM), 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 multi-seasonal decomposition techniques to supplement the LSTM learning procedure. In our experiments, we are able to show that on datasets from disparate data sources, like 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 multi-seasonal forecasting methods.**

***Index Terms*—Time Series Forecasting, Multiple Seasonality, Neural Networks, RNN, LSTM**

I. Introduction

Time series forecasting has become a key-enabler of modern day business planning by landscaping the short-term, medium-term and long-term goals in an organisation. As such, generating accurate and reliable forecasts is becoming a perpetual endeavour for many organisations, 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 behaviour such as multiple seasonal patterns, non-integer seasonality, calendar effects, etc.

As sensors and data storage capabilities advance, time series with higher sampling rates (sub-hourly, hourly, daily) are 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. Here, the variations among energy consumption patterns within a day, i.e., morning and evening, workday and weekday cause daily and weekly seasonal patterns in the time series. While the daily and weekly seasonality can be used to determine the short-term energy demand in households, the yearly seasonality may explain the seasonal effects towards the energy consumption, which can be beneficial in long-term energy planning. Therefore, accurate modelling of such multiple seasonal cycles is necessary to estimate the demand on various time horizons. Particularly in the energy industry, accurate short-term and long-term load forecasting 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.

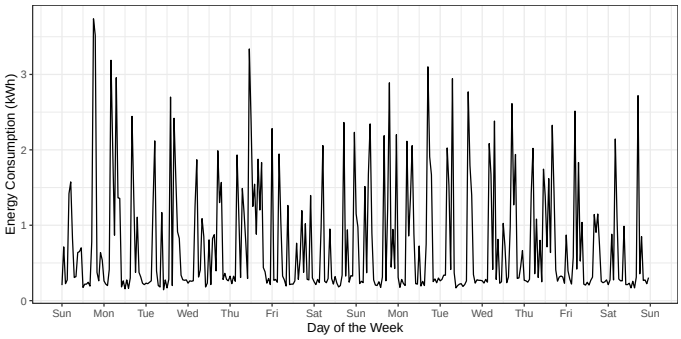

Fig. 1. Half-hourly energy consumption of a household over a two weeks period of time, extracted from the AusGrid-Energy Dataset [1], displaying the inter-day (daily) and intra-day (weekly) seasonal patterns.

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 amount 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

product assortments in retail, server performance measures in computer centres, household smart meter data, etc. 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 is 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 (RNN) and Long Short-Term Memory Networks (LSTM) that are naturally suited in modelling sequence data.

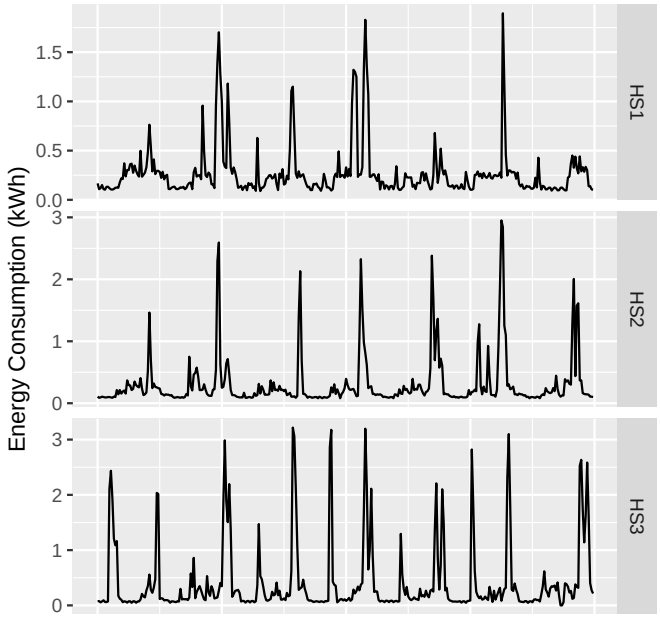

Fig. 2. Half-hourly energy consumption fluctuations of three different households in Australia [1] over a time period of one week.

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 rely on the characteristics of the time series. To this end, in this paper, we propose LSTM-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 behaviours 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, 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 dataset and real-world datasets, which contain multiple seasonal patterns, exhibiting different levels of seasonal homogeneity.

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 methods [13] and autoregressive integrated moving-average (ARIMA) models [3]. The basic forms of these algorithms are only suited in modelling 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] adapts the simple Holt-Winters method to capture seasonalities by introducing multiple seasonal components to the linear version of the model. Gould et al. [16] propose 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 exponential smoothing approach. However, the majority of these techniques suffer from over-parameterisation, optimisation 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 modelling framework was developed by De Livera et al. [18], aiming to address various challenges associated with seasonal time series, such as modelling multiple seasonal periods, non-integer seasonality, and calendar effects. Today, these proposed seasonal models, i.e., BATS and TBATS, are considered 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 modelled 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 modelled separately using an ARMA process. Moreover, Lee and Ko [2] use a lifting scheme, a different decomposition technique, to separate the original time series at different load frequency levels. Afterwards, individual ARIMA models are built to forecast each decomposed sub series separately.

In parallel to these developments, neural networks (NNs) have been advocated as a strong alternative to traditional statistical forecasting methods in forecasting seasonal time

series. The favourable properties towards 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 modelling seasonal patterns [22], [23], more recent studies advise that deseasonalising the time series prior to modelling is useful to achieve better forecasting accuracy from NNs [10]–[12], [24], [25]. Here, deseasonalisation 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 deseasonalisation process with NNs. Furthermore, Zhang and Qi [11] highlight that NNs are unable to model trend or seasonality directly, thus detrending or deseasonalisation is necessary to produce accurate forecasts with NNs. Meanwhile, Dudek [26] develops a local learning based approach to deal with multiple seasonal cycles. Though this obviates the need of 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 neural networks have drawn significant attention among forecasting practitioners. In particular, RNNs and convolutional neural networks (CNN) have exhibited promising results, outperforming many state-of-the-art statistical forecasting methods [4]–[8], [27]. Nevertheless, in spite of the substantial literature available on deep learning in time series forecasting, only few attempts have been undertaken to explicitly handle multiple seasonal patterns in a time series [8], [28]. Lai et al. in [8] introduce a combination of CNN and RNN architectures to model short and long term dependencies in a time series, and employ a skipped connection architecture to model different seasonal periods. Bianchi et al. [28] 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. Moreover, the winning submission of the recently concluded M4 forecasting competition [29], Exponential Smoothing-Recurrent Neural Network (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) [30].

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 shown in Fig. 3 in detail.

A. Problem Statement

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

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

Here, θ are the model parameters of our LSTM-MSNet prediction model.

Fig. 3 illustrates a schematic overview of the LSTM-MSNet forecasting framework. 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 recurrent neural networks. The LSTM-MSNet has three layers, namely: 1) the pre-processing layer, which consists of a normalisation and variance stabilising 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 post-processing layer to denormalise and reseasonalise 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 paper, we select LSTMs, a promising RNN variant, as our primary network training module. In the following sections, we discuss each layer of the LSTM-MSNet in detail.

B. Normalisation and Variance Stabilisation Layer

The proposed LSTM-MSNet is a global model that is built across a group of time series. Therefore, performing a data normalisation 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, which uses the mean of a time series as the scaling factor. This scaling strategy can be defined as follows:

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

Here, $x_{i,normalised}$ represents the normalised observation, and k represents the number of observations of time series i.

After normalising the time series, we stabilise the variance in the group of time series by transforming each time series to a logarithmic scale. Apart from the variance stabilisation, the log transformation also enables the conversion of the 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:

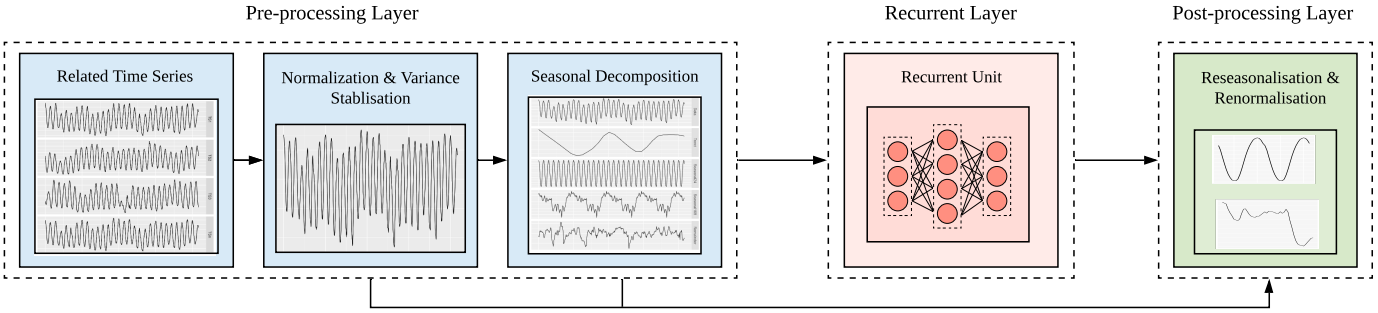

Fig. 3. An overview of the proposed forecasting framework (LSTM-MSNet), which consists of three layers, namely Pre-processing layer, Recurrent layer, and Post-processing layer.

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

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

C. Seasonal Decomposition

As highlighted in Section II, when modelling seasonal time series with NNs, many studies suggest applying a prior seasonal adjustment, i.e., deseasonalisation to the time series [10]–[12]. The main intention of this approach is to minimise the complexity of the original time series, and thereby reducing the subsequent effort of the NN's learning process. In line with these recommendations, LSTM-MSNet initially uses a deseasonalisation strategy to detach the multi-seasonal components from a time series. To accommodate this, we use a series of statistical decomposition techniques that support separating multi-seasonal patterns in a time series. We also configure these methods to extract various forms of seasonality, i.e., deterministic and stochastic seasonality to assess their sensitivity towards 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 (MSTL): MSTL extends the original version of Seasonal-Trend Decomposition (STL) [31], 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:

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

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

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

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

Here, n denotes the number of distinct seasonal patterns decomposed by the MSTL. In our study, we use the R [32] implementation of the MSTL algorithm, `mstl`, from the `forecast` package [33], [34]. 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 (STR): STR is a regression based decomposition technique introduced by Dokumentov et al. [35]. The division is additive, hence the decomposition accords with Equation 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 [36].

3) Trigonometric, Box-Cox, ARMA, Trend, Seasonal (TBATS): 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 Livera et al. [18])).

In our seasonal decomposition step, we use TBATS as a deseasonalisation 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 [33], [34] in R.

4) Prophet: Prophet is an automated forecasting framework developed by Taylor and Letham [37]. The main aim of this framework is to address the challenges involved in forecasting at Facebook, 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 Prophet uses an additive decomposition layer similar to Equation 5. However, this division introduces an additional term to model holidays as seasonal covariates. After including the holiday terms, Equation 5 can be rewritten as follows:

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

Here, $\hat{H}_t$ denotes the holiday covariates in the model that represent 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 [38].

5) Fourier Transformation: Fourier terms are a flexible approach to model periodic effects in a time series [39]. 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 Fourier terms as follows:

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

Here, 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 Equation 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 to 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 Fourier terms. Moreover, we consider $k = 1$ (with least number of k) as a special use case in 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. The table furthermore indicates the type of seasonalities extracted by each method.

TABLE I
SUMMARY OF TECHNIQUES USED FOR MULTI-SEASONAL DECOMPOSITION

Technique	Package	Deterministic	Stochastic
MSTL (s.window = "periodic")	forecast [33]	✓	✗
MSTL (s.window = "7")	forecast [33]	✓	✓
AutoSTR	stR [36]	✓	✓
TBATS	forecast [33]	✓	✓
Prophet	prophet [38]	✓	✓

D. Recurrent Layer

The second layer, 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 modelling tasks, such as Natural Language Processing [40], speech recognition [41], image generation [42], and more recently have received a great amount of attention in time series research [5]–[7], [25], [43].

In the LSTM, the gating mechanism together with the self-contained memory cell enables the network to capture non-linear 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:

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

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

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

$$C_t = f_t \odot C_{t-1} \ + \ i_t \odot \tilde{C_t} \tag{11}$$

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

$$h_t = o_t \odot \phi(C_t) \tag{13}$$

Here, 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, 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 Equation 11). Moreover, $\odot$ is the element-wise multiplication operation, while σ represents the sigmoid activation function. We use a hyperbolic tangent function, i.e., *tanh* as our hidden update activation function in Equation 10.

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 multi-step 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], [44]. Therefore, we choose the size of the output window m equivalent to the length of the intended forecast horizon M. Fig. 4 illustrates an example of applying the MW approach to the hourly time series T48 of the M4 dataset.

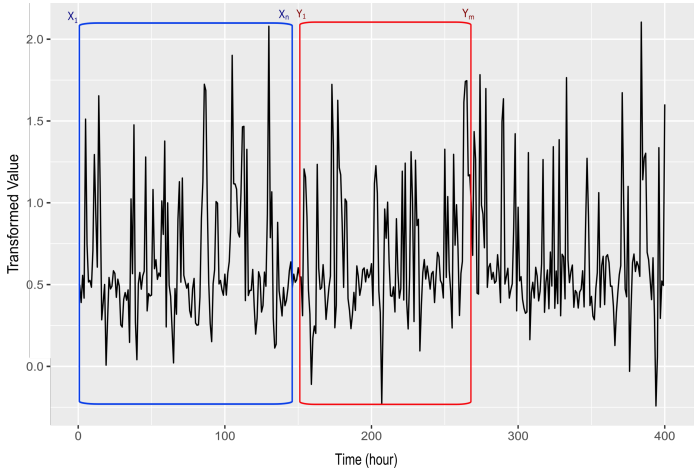

Fig. 4. An example of applying the moving window approach to a preprocessed and seasonally adjusted time series X_i, i.e., after the normalisation & variance stabilisation and decomposition layers. Here, $\{x_1, x_2, ..., x_n\}$ represents the input window, and $\{y_1, y_2, ..., y_m\}$ corresponds to the output window.

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.

2) Training Paradigms: In this study, 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.

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

b) Seasonal Exogenous Approach (SE): This second approach uses the output of the pre-processing layer, together with the seasonal components extracted from the multi-seasonal decomposition as external variables. Here, in addition to the normalised time series (without the deseasonalisation 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 SE, the LSTM-MSNet is expected to forecast all the components of a time series, including the relevant multi-seasonal patterns. Therefore, a reseasonalisation 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 minimises the overall training complexity of the LSTM-MSNet. In contrast, SE supplements LSTM-MSNet in the form of exogenous variables that assist modelling the seasonal trajectories of a time series.

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 our 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. 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. 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.

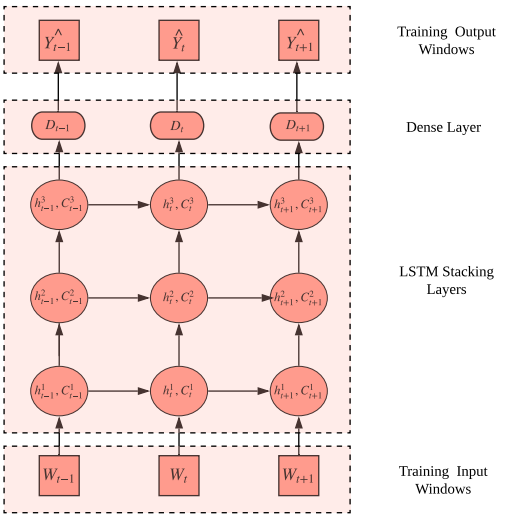

Fig. 5. The unrolled representation of a peephole connected LSTM in time, with the hidden state (h_t) and memory cell (C_t). In this architecture, h_t expects to capture the short-term dependencies in a sequence, while C_t accounts for the long-term dependencies. Here, the input window, dense layer,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

Before feeding these windows to the network for training, each input and output window is subjected to a local normalisation 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 normalisation factor, whereas in SE we use the mean value of each input window as the normalisation factor. Afterwards, these factors are subtracted from each data point in the corresponding input

and output window. The above LSTM learning scheme is implemented using TensorFlow [45].

4) Loss Function: We use the L1-norm, as the primary learning objective function, which essentially minimises 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:

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

Here, $Y_t \in \mathbb{R}^m$ 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-regularisation term to minimise possible overfitting of the network. In Equation 14, ψ is the regularisation parameter, w_i refers to the network weights and p is the number of weights in the network.

E. Post-processing Layer

The reseasonalisation and renormalisation is the main component of the post processing layer in LSTM-MSNet. Here, in the reseasonalisation 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 renormalisation phase, the generated forecasts are back-transformed to their original scale by adding back the corresponding local normalisation factor, and taking the exponent of the values. The final forecasts are obtained by multiplying this vector by the scaling factor used for the normalisation process.

IV. Experiments

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

A. Datasets

We use three benchmark time series datasets which present multiple seasonal cycles. Following is a brief overview of these datasets:

- M4-Hourly Dataset [29]: Hourly dataset from the M4 forecasting competition.
- AusGrid-Energy Dataset [1]: Half-hourly dataset, representing energy consumption of 300 households in Australia. We select general consumption (GC letter code in the dataset) as the primary measure of energy consumption in households. Firstly, we extract a subset of 3 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 2 years (2010 July - 2012 July), considering three seasonalities: daily, weekly, and yearly.

TABLE II
DATASET 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

Table II summarises statistics of the datasets 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], [25].

We plot the seasonal components of our benchmark datasets to investigate the diversity of their seasonal distributions. 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 dataset. 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 datasets, and shows less variation compared to the M4 dataset, 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 datasets 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) [46]. The sMAPE and MASE are defined as follows:

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

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

Here, A_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, and S = 8766 for

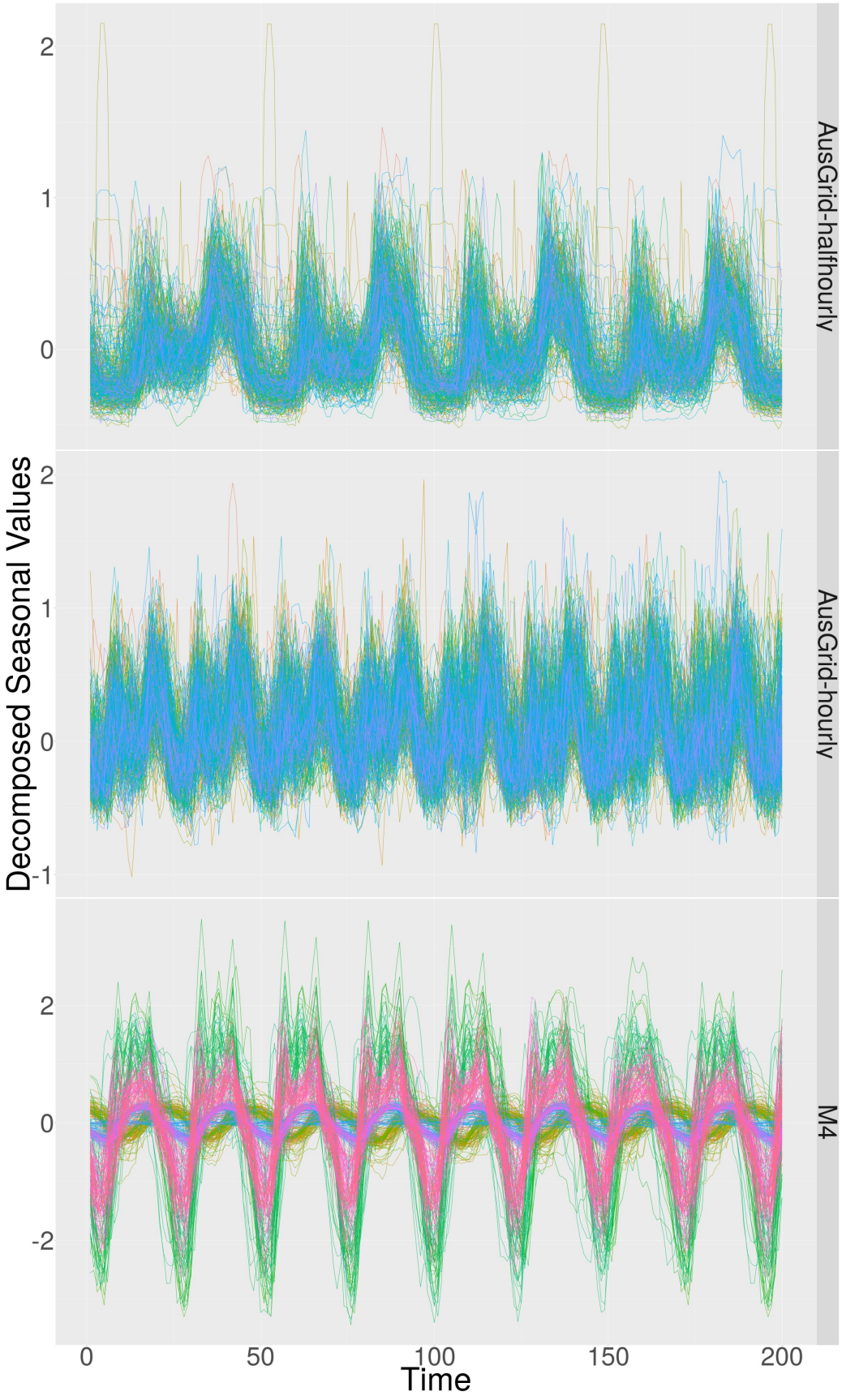

Fig. 6. The seasonal components distribution of the AusGrid-Energy (half hourly), AusGrid-Energy (hourly), and M4 datasets

the M4-Hourly and AusGrid-Energy datasets, respectively). Furthermore, when calculating the sMAPE values for the energy datasets, to avoid problems for zero forecasts and actual observations, we add a constant term of $\epsilon = 1$ to the denominator of Equation 15. To provide a broader overview of the error distributions, we compute the mean and median of these primary error measures. This includes, 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 non-parametric 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 [47][1]. The sMAPE error measure is used to perform the statistical testing, with a significance level of α = 0.05.

D. Hyper-parameter Tuning and Optimisation

The recurrent layer in LSTM-MSNet has various hyper-parameters. This includes LSTM cell dimension, number of epochs, hidden-layers, mini-batch-size and model regularisation terms. In the optimisation framework, we use COntinuous COin Betting [48] as our primary learning algorithm to train the network, which does not require the tuning of learning rate. In this way, we minimise the overall amount of hyper-parameters to be tuned in the network.

Furthermore, we automate the hyper-parameter selection process by employing a Bayesian global optimisation methodology that autonomously discovers the optimal set of parameters of an unknown function. Compared to other parameter optimisation selection techniques, such as Random Search and Grid Search, the Bayesian optimisation strategy is considered as a more systematic and/or more efficient approach. This is because, when determining the current pool of potential hyper-parameters for validation, it takes into account the previously visited hyper-parameter values in a non-trivial way [49]. In our experiments, we use the python implementation of the sequential model-based algorithm configuration (SMAC) [50] that implements a version of the Bayesian hyper-parameter optimisation process. Table III summarises the bounds of the hyper-parameter 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* [37], and *FFORMA* [51]. We also use two variants of *Dynamic-Harmonic-Regression* [33] 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.

[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 HYPER-PARAMETER 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}	$8 \cdot 10^{-4}$

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

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 summarised 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.

F. Computational Performance

We also report the computational costs in execution time of our proposed LSTM-MSNet variants and the benchmark models on the aggregated hourly energy dataset. The experiments are run on an Intel(R) i7 processor (3.2 GHz), with 2 threads per core, 6 cores per socket, and 64GB of main memory.

G. Results

Table V summarises the evaluation results of all the LSTM-MSNet variants and benchmarks for the 414 hourly series of the M4 competition dataset, 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 V, 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

TABLE V
M4 DATASET 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

TABLE VI
SIGNIFICANCE TESTING FOR M4 DATASET

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 VII
AUSGRID-ENERGY (HALF-HOURLY) DATASET 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 VIII
SIGNIFICANCE TESTING FOR AUSGRID-ENERGY (HALF-HOURLY) DATASET

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}

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 to 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, LSTM-Baseline, in terms of mean sMAPE and mean MASE.

Table VI 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 VI, we see that the FFORMA achieves significantly better results than the other methods.

Table VII shows the evaluation summary for the 300 half-hourly series of the AusGrid-Energy dataset. The *NA* values in the table represent models that could not complete the execution within a time frame of 6 days. It can be seen 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 obtain better forecasts, which is contrary to our previous findings from the M4 competition dataset. Also, several LSTM-MSNet variants with the DS training paradigm display poor performance compared to 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 VIII, we see that the LSTM-MSTL-SE performs best and is used as the control method. Moreover, the LSTM-MSNet variants, LSTM-TBATS-SE, LSTM-Prophet-DS do not perform significantly worse than the control method.

Table IX provides the results for the evaluations on the aggregated hourly time series of the AusGrid-Energy dataset. We see that the proposed LSTM-Fourier-SE ($k = 1$) variant

TABLE IX
AUSGRID-ENERGY (HOURLY) DATASET 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-DS	NA	NA	NA	NA

TABLE X
SIGNIFICANCE TESTING FOR AUSGRID-ENERGY (HOURLY) DATASET

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}

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, the LSTM-MSNet variants with the DS training paradigm. Furthermore, the LSTM-Baseline method performs better than the LSTM-MSNet variants with DS training paradigm. Nevertheless, consistent with our previous findings from Table VII, 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 the results are highly significant. According to Table X, 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.

Also, with respect to computational cost of the proposed variants and benchmarks, according to Table XI, 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 to 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

TABLE XI
COMPUTATIONAL SUMMARY - AUSGRID-ENERGY DATASET: HOURLY (IN MINUTES)

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

these methods do not display competitive results compared to the LSTM-MSNet variants, according to Tables V, VII, and IX. In contrary, LSTM-MSNet promises to deliver better performance than TBATS and FFORMA, with respect to both accuracy and computation time.

H. Discussion

It can be seen that the LSTM-Baseline results for the M4 and AusGrid-Energy datasets are contradictory. Even though the LSTM-Baseline model cannot outperform both DS and SE learning paradigms of LSTM-MSNet in the M4 dataset, it obtains better results than the DS learning paradigm in both AusGrid-Energy datasets. These results can be interpreted by the seasonal characteristics present in these datasets (Fig. 6). As discussed in Section IV-A, the seasonal components of the M4 dataset are less homogeneous. So, 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 towards the LSTM-MSNet results in the M4 dataset. 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 lengths of the series are likely to have an effect here, in the sense that for longer series, we assume that the patterns can be learned easier, whereas for shorter series, deseasonalisation is beneficial. However, we do not analyse this in our current study. Furthermore, we observe the highly competitive results of FFORMA in the M4 dataset. FFORMA was the second-best performing method in the overall M4 competition, so it can be arguably seen as optimised for this particular dataset. Also, due to its nature as being an ensemble of simpler univariate techniques, it is inherently more suitable for this situation of a dataset of inhomogeneous series, whereas on the AusGrid-Energy datasets, it is less performant. There, due to the high presence of homogeneous seasonality, exclusive learning of the seasonality becomes viable for LSTM-MSNet. As a result, LSTM-Baseline achieves better results compared to 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 dataset. Whereas, on the AusGrid-Energy datasets, 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 datasets, which can be attributed to the longer lengths of the time series. Furthermore, according to Table XI, the MSTL and Prophet methods are computationally efficient decomposition techniques compared to 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 counter part, 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 like in the M4 dataset, 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 deseasonalisation (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 paper, 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 recurrent neural networks, to overcome the limitations of the current univariate state-of-the-art models by training a unified model that exploits key structures, behaviours, and patterns common within a group of time series.

We have utilised 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 Deseasonalised 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 deseasonalised 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 counter part, 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 dataset, and two energy consumption time series datasets 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-art 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 deseasonalised approach can be used to model shorter seasonalities, whereas the seasonal exogenous approach can be applied to address the longer seasonalities.

VI. Acknowledgements

This research was supported by the Australian Research Council under grant DE190100045, Facebook Statistics for Improving Insights and Decisions research award, Monash Institute of Medical Engineering seed funding, and MASSIVE - High performance computing facility, Australia.

References

[1] AusGrid, "Innovation and research - ausgrid," https://www.ausgrid.com.au/Industry/Innovation-and-research/, 2019, accessed: 2019-5-16.

[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*. John Wiley & Sons, 2015.

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

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

[6] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A Multi-Horizon quantile recurrent forecaster," Nov. 2017. [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.

[8] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and Short-Term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, ser. SIGIR '18. New York, NY, USA: ACM, 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," Jan. 2019. [Online]. Available: https://arxiv.org/abs/1901.04028

[10] M. Nelson, T. Hill, W. Remus, and M. O'Connor, "Time series forecasting using neural networks: should the data be deseasonalized first?" *J. Forecast.*, vol. 18, no. 5, pp. 359–367, 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, 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 Science & Business Media, 2008.

[14] A. Harvey, S. J. Koopman, and M. Riani, "The modeling and seasonal adjustment of weekly observations," *J. Bus. Econ. Stat.*, vol. 15, no. 3, pp. 354–368, 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. 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. Snyder, "Forecasting time series with complex seasonal patterns using exponential smoothing," *J. Am. 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 Processing*, vol. 82, no. 12, pp. 1903–1915, Dec. 2002.

[20] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Systems*, 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, Jan. 1991.

[22] Zaiyong 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, 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. Nuclear Energy*, vol. 27, no. 1, pp. 25–36, 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, 2012.

[25] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *arXiv [cs.LG]*, 2019. [Online]. Available: https://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*. Springer Berlin Heidelberg, 2013, pp. 52–63.

[27] 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.

[28] F. M. 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," May 2017. [Online]. Available: http://arxiv.org/abs/1705.04378

[29] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 competition: Results, findings, conclusion and way forward," *Int. J. Forecast.*, vol. 34, no. 4, pp. 802–808, 2018.

[30] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *Int. J. Forecast.*, Jul. 2019.

[31] 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.

[32] 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/

[33] R. J. Hyndman, G. Athanasopoulos, S. Razbash, D. Schmidt, Z. Zhou, Y. Khan, C. Bergmeir, and E. Wang, "forecast: Forecasting functions for time series and linear models," *R package version*, vol. 6, no. 6, p. 7, 2015.

[34] Y. Khandakar and R. J. Hyndman, "Automatic time series forecasting: the forecast package for R," *J. Stat. Softw.*, vol. 27, no. 03, 2008.

[35] A. Dokumentov, R. J. Hyndman, and Others, "STR: A seasonal-trend decomposition procedure based on regression," Monash University, Department of Econometrics and Business Statistics, Tech. Rep., 2015.

[36] A. Dokumentov and R. J. Hyndman, "str: STR decomposition," 2018.

[37] S. J. Taylor and B. Letham, "Forecasting at scale," PeerJ Preprints, Tech. Rep. e3190v2, Sep. 2017.

[38] S. Taylor and B. Letham, "prophet: Automatic forecasting procedure," 2018.

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

[40] T. Mikolov, M. Karafiát, L. Burget, J. Cernock\'y, and S. Khudanpur, "Recurrent neural network based language model," in *Interspeech*, vol. 2. fit.vutbr.cz, 2010, p. 3.

[41] A. Graves, A. r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. ieeexplore.ieee.org, 2013, pp. 6645–6649.

[42] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "DRAW: A recurrent neural network for image generation," Feb. 2015.

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

[44] S. Ben 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.

[45] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale machine learning on heterogeneous distributed systems," Mar. 2016.

[46] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *Int. J. Forecast.*, 2006.

[47] 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.

[48] F. Orabona and T. Tommasi, "Training deep networks without learning rates through coin betting," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, USA, 2017, pp. 2157–2167.

[49] 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, Inc., 2012, pp. 2951–2959.

[50] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, 2011, pp. 507–523.

[51] P. Montero-Manso, G. Athanasopoulos, R. J. Hyndman, and T. S. Talagala, "FFORMA: Feature-based forecast model averaging," Tech. Rep. 19/18, 2018.