




## Article

# Deep LSTM-Based Transfer Learning Approach for Coherent Forecasts in Hierarchical Time Series

Alaa Sagheer <sup>1,2,\*</sup> , Hala Hamdoun <sup>2,3,†</sup>  and Hassan Youness <sup>3</sup> 

- <sup>1</sup> College of Computer Sciences and Information Technology, King Faisal University, Al-Ahsa 31982, Saudi Arabia
- <sup>2</sup> Center for Artificial Intelligence and Robotics (CAIRO), Aswan University, Aswan 81582, Egypt; hala@aswu.edu.eg
- <sup>3</sup> Department of Computers and Systems Engineering, Faculty of Engineering, Minia University, Al-Minia 61519, Egypt; hassan\_youness@mu.edu.eg
- \* Correspondence: asagheer@kfu.edu.sa
- † Equally contributed authors.

**Abstract:** Hierarchical time series is a set of data sequences organized by aggregation constraints to represent many real-world applications in research and the industry. Forecasting of hierarchical time series is a challenging and time-consuming problem owing to ensuring the forecasting consistency among the hierarchy levels based on their dimensional features. The excellent empirical performance of our Deep Long Short-Term Memory (DLSTM) approach on various forecasting tasks motivated us to extend it to solve the forecasting problem through hierarchical architectures. Toward this target, we develop the DLSTM model in auto-encoder (AE) fashion and take full advantage of the hierarchical architecture for better time series forecasting. DLSTM-AE works as an alternative approach to traditional and machine learning approaches that have been used to manipulate hierarchical forecasting. However, training a DLSTM in hierarchical architectures requires updating the weight vectors for each LSTM cell, which is time-consuming and requires a large amount of data through several dimensions. Transfer learning can mitigate this problem by training first the time series at the bottom level of the hierarchy using the proposed DLSTM-AE approach. Then, we transfer the learned features to perform synchronous training for the time series of the upper levels of the hierarchy. To demonstrate the efficiency of the proposed approach, we compare its performance with existing approaches using two case studies related to the energy and tourism domains. An evaluation of all approaches was based on two criteria, namely, the forecasting accuracy and the ability to produce coherent forecasts through through the hierarchy. In both case studies, the proposed approach attained the highest accuracy results among all counterparts and produced more coherent forecasts.

**Keywords:** deep long short-term memory; auto-encoder; hierarchical time series; coherent forecast; power generation; australian tourism



**Citation:** Sagheer, A.; Hamdoun, H.; Youness, H. Deep LSTM-Based Transfer Learning Approach for Coherent Forecasts in Hierarchical Time Series. *Sensors* **2021**, *21*, 4379. <https://doi.org/10.3390/s21134379>

Academic Editor: Kyandoghere Kyamakya

Received: 18 May 2021  
Accepted: 23 June 2021  
Published: 26 June 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Forecasting is one of the pillars for decision support systems in different domains such as weather, web traffic, demand and sales, and energy. It can be defined simply as the rational prediction of future events based on past and current events, where all events are represented as time series observations [1]. Time series forecasting (TSF) with reasonable accuracy is a necessary but quite tricky problem because it correlates many factors with an enormous amount of observations [2].

Many real-world applications exhibit naturally hierarchical data organization, with multiple time series on several levels based on dimensional attributes such as geography, products, or some other attributes. Forecasting through such a hierarchical time series (HTS) structure adds difficulty to the original TSF problem owing to ensuring the forecasting consistency among the hierarchy levels, a phenomenon called coherency [3]. Coherent

forecasts mean that the forecast of a time series in a higher level should equal to the sum of forecasts of the corresponding time series in the lower level. In other words, to ensure forecast coherency, we require the forecasts to add up in a manner that is consistent with the aggregation structure of the collection of time series [4]. Therefore, it is often necessary to carry out a reconciliation step to adjust the lower level forecasts and to make them coherent or consistent in the upper levels [5].

In the past two decades, several reconciliation procedures are presented to ensure coherent forecasts, including top-down, bottom-up, and a combination of both procedures called middle-out [6]. These procedures generate what we can call “base” forecasts in different ways by separately predicting individual time series. Then, the “base” forecasts are reconciled according to the inherent hierarchical structure of the selected procedure. The bottom-up procedure starts by estimating the “base” forecasts for the time series at the bottom level in the hierarchy. Then, it implements a simple aggregation way to obtain the “target” forecasts at the higher levels of the hierarchy. On the contrary, the top-down procedure involves estimating the “base” forecasts only for the top layer of the hierarchy and then disaggregates it based on historical proportions of time series at the lower levels. The middle-out procedure estimates the “base” forecasts for the time series at the intermediate level and then implements both bottom-up and top-down procedures to perform the prediction through the hierarchy [6].

Overall, it is not ultimately confirmed which procedure is more efficient than the other because every procedure focuses on a different aggregation level to yield forecasts [7]. Researchers have attributed such instability to the tendency of these procedures to minimize the forecast errors without questioning the coherence of consecutive predictions. As a result, some important information that exists in other levels will be ignored [8]. If we disregard the aggregation constraints, we could forecast all the time series in a group independently. Nevertheless, it is still doubtful that the produced set of forecasts is coherent [5]. Another drawback in the existing approaches is that none of them takes account of the relationships and inherent correlation among series. Therefore, it is not easy to identify prediction intervals for the forecasts from any of these approaches [6].

Toward optimal reconciliation, Hyndman et al. introduced an optimal combination approach that outperformed the approaches mentioned above [6]. This approach assumes that the forecasting error distribution is typical as the aggregated hierarchical structure [9]. It runs by estimating forecasts on all levels of the hierarchy independently. Then, it uses a regression model to optimally combine these forecasts to yield a set of coherent forecasts [6,9]. The generated forecasts add up appropriately across the hierarchy, are unbiased and have minimum variance amongst all combination forecasts [3]. However, this approach is computationally expensive, particularly for large hierarchies, because it should estimate forecasts for all levels in the hierarchy [10,11]. In addition, the authors did not provide any justification whether the reconciled forecasts work well compared to the most commonly implemented approaches [5].

As a common practice in all the approaches that mentioned above, the “base” forecasts are generated using either a statistical or empirical forecasting method depending on the aggregation level of focus [12]. Standard statistical methods include autoregression integrated moving average (ARIMA), Box and Jenkins method, and exponential smoothing (ES) method [13]. These statistical methods are static and often ignore the dynamics of the individual time series and the structure of grouped time series during computations. Consequently, they may fail to perform well if the corresponding groups of time series are subject to a time variation or any sudden change, as the case in energy load or supply chain applications [14]. In addition, it was demonstrated that these methods failed to exploit the complete available information in the hierarchy, which influences the efficiency of the overall forecasting. Moreover, they lack a straightforward mean to determine an optimal reconciliation approach, which is crucial for hierarchical aggregation [9].

Different machine learning algorithms were recently utilized to estimate the “base” and “target” forecasts and to derive the combination weights for the forecasts in the opti-

mal combination approach. For example, Mahdi et al. [14] used artificial neural networks (ANN), extreme gradient boosting (XGboost), and support vector regression (SVR) algorithms to estimate the proportions of time series through a middle-out approach. Spiliotis et al. [12] suggested a machine learning-based method that allows for a nonlinear combination of the “base” forecasts more general than other linear methods. It estimates the “base” forecasts directly without conditioning that the complete information is utilized to generate reconciled forecasts for each time series. Shiratori et al. [15] introduced an approach based on a forecasting model for each time series at the bottom level and then used a structured regularization step to combine the forecasts at the upper levels into the forecast of the bottom level. Mancuso et al. [16] represented the disaggregation method as a nonlinear regression procedure instead of relying on forecast or historical proportions. Namely, they proposed a deep neural network (DNN) model that learns how to share the top level forecasts to the time series at the bottom level of the hierarchy, taking into consideration the characteristics of the information of the individual series and the aggregated series.

To the best of our knowledge, some machine learning-based reconciliation approaches adopted the top-down approach and exploited information only from the parent node. They ignore the rest of the nodes that could be beneficial for obtaining more accurate results, such as the system shown in [14]. Some other machine learning-based approaches try to optimize the forecast accuracy in a linear coherence fashion, such as the system shown in [5], which adopted an in-sample error method for the baseline forecasting. This approach may not be representative of out-sample accuracy [12]. Overall, most machine learning-based reconciliation approaches are built to achieve coherency under particular assumptions to improve the overall prediction accuracy. An exception is the approach presented in [12], which solved the problem of forecasts reconciliation in a general nonlinear fashion to enhance the overall performance across all levels of the hierarchy. However, this approach has a reservation of bias since it selects the forecasts to be combined and neglect other forecasts that may be useful. Nevertheless, all of these approaches still use traditional methods, such as ARIMA and ES, to generate the “base” forecast.

In the last two years, the first author of this paper developed a Deep Long Short-Term Memory (DLSTM) approach that can solve various forecasting problems using either univariate or multivariate time series data [17,18]. The excellent empirical performance of DLSTM motivated us to extend it to treat the hierarchical time series forecasting (HTSF) problem. In this paper, we develop the DLSTM in an auto-encoder (AE) fashion; from here, on we call it DSLTM-AE, taking full advantage of the hierarchical architecture for better time series forecasting. DLSTM-AE can work as an alternative approach to traditional and machine learning approaches that have been used to manipulate the HTSF problem. However, as demonstrated in [18], training a DLSTM in hierarchical architectures requires updating the weight vectors for every LSTM cell, which is time-consuming and requires a large amount of data through several dimensions. Transfer learning strategy can mitigate the effect of this problem, particularly if we adopt it in a bottom-up procedure. The proposed approach runs as follows: First, we train the time series at the bottom level of the hierarchy using DLSTM-AE and generate the “base” forecasts. Then, we use (or transfer) the learned features of the “base” forecast to perform synchronous training to all time series at the upper levels of the hierarchy to estimate the “target” forecasts.

To demonstrate the efficiency of the proposed approach, we empirically compared its performance with several existing approaches using two different case studies using two different datasets. Namely, the Brazilian electrical power generation dataset [19] and the Australian visitor nights of domestic tourism dataset [20]. Using three different performance metrics, we compared our results with those reported by the authors of [19,20], respectively. The comparison among all approaches was based on two criteria: forecasting accuracy and the ability to produce coherent forecasts. Overall, the proposed approach attained the highest accuracy results among all reference approaches. Moreover, it produced more coherent forecasts through a straightforward scenario compared to reference approaches.

The DLSTM-AE procedure and its empirical results shown in the following sections highlight additional advantages of this approach. First, the DLSTM-AE utilizes all the information included in all time series in all levels of the hierarchy, contrary to reference approaches that may ignore useful information. Indeed, this is expected to enhance the overall forecasting performance of the approach. Second, the DLSTM-AE produces forecasts at each level of the hierarchy, which provides meaningful insights for decision-makers to make decisions at any moment during the operation, such as demand planning, production planning, energy crisis, etc. Third and in terms of the original LSTM properties, the DLSTM-AE inherently considers the relationship and temporal correlations among time series observations and automatically extracts many inherent useful features [21]. Fourth, as the computations at the higher levels are performed in synchronous mode, this undoubtedly reduces the whole process's computational cost.

Consequently, we can summarize our contribution in this paper as follows:

1. We developed a novel deep neural network model runs in a transfer learning scenario that significantly simplifies the forecasting through hierarchical architectures.
2. Our approach attained the highest forecasting accuracies among existing approaches.
3. Our approach produced more coherent forecasts within the hierarchy levels using a straightforward procedure.
4. We made all source codes and a standalone implementation available on GitHub <https://github.com/Hala-Hamdoun/DLSTM-AE.git> accessed on 25 June 2021.

The rest of the paper is organized as follows. Section 2 briefly summarizes the standard time series forecasting problem and its related concepts. The elements and steps of the proposed approach are presented in Section 3. Section 4 shows the experimental settings and datasets used in this paper. The empirical results achieved using two case studies and related analysis are provided in Section 5. Finally, an overall discussion and the paper conclusions are presented in Section 6.

## 2. Time Series Forecasting

A Time Series Forecasting (TSF) system involves predicting the system performance in the future based on information from the current and past status of the system. It works by considering a sequence of observations ( $x$ ) measured in proper chronological order of a variable of interest at time ( $t$ ). A sequence of time series data can be described as:

$$X = (x_t; t = 1, \dots, N) \quad (1)$$

where  $X$  is the time series sequence and  $t$  refers to the observation's time for  $N$  observations. It is worth mentioning that the TSF problem is not limited to forecasting the observation of the next time step only, which is called single-step ahead forecast. There are also time series problems that involve forecasting a sequence of future values, which is known as multi-step ahead forecast [22].

### 2.1. Single-Step ahead Forecast (SSaF)

It is the simplest method in time series forecasting as it just forecasts only the next future value. Simply, SSaF at a time  $t + 1$  is performed by passing the current and the past observations  $t, t - 1, t - 2, \dots, t - N$  to a selected model [23],

$$F(t + 1) = M(o(t), o(t - 1), o(t - 2), \dots, o(t - N)) \quad (2)$$

where  $F(t + 1)$  is the forecast at time ( $t + 1$ ),  $M$  is the proper selected model, and  $o(t)$  is an observation at time  $t$ .

## 2.2. Multi-Step ahead Forecast (MSaF)

MSaF involves predicting the next  $H$  sequence of future time series values  $[y_1, y_2, \dots, y_N]$  with  $N$  observations and with  $H > 1$  being prediction horizon. The three main MSaF strategies adopted in the literature are recursive strategy, direct strategy, and multi-input multi-output strategy (MIMO) [22]. In the following, we describe these three strategies, where  $f$  and  $F$  denote the functional dependency between past and future observations, and  $d$  represents the number of past values used to forecast the future values.

1. Recursive strategy (also known as iterated or multi-stage) is the oldest and most intuitive strategy used to treat MSaF [24,25]. In this strategy, a single model  $f$  is trained to perform only a one-step ahead forecast. Then, the value that is just forecasted is used as part of the following input variable to forecast the next step with the same step ahead model. Accordingly, all forecasts for the entire horizon  $H$  are obtained as described in the above manner. Considering the trained one-step ahead model  $\hat{f}$ , the forecasts are given as follows:

$$\hat{y}_{N+h} = \begin{cases} \hat{f}(y_N, \dots, y_{N-d+1}), & \text{if } h=1 \\ \hat{f}(\hat{y}_{N+h-1}, \dots, \hat{y}_{N+1}, y_N, \dots, y_{N-d+1}), & \text{if } h \in 2, \dots, d \\ \hat{f}(\hat{y}_{N+h-1}, \dots, \hat{y}_{N-d+h}), & \text{if } h \in d+1, \dots, H \end{cases} \quad (3)$$

It is worth mentioning that the recursive strategy showed low performance in some multi-step ahead forecasting tasks due to the noise present in the time series and the forecasting horizon. In such a case, the produced prediction error propagates forward as long as the forecasting horizon increases. These forecasts are used as an input value for making subsequent forecasts [22].

2. Direct strategy (also known as independent strategy) depends on predicting each horizon independently from the others [24,25]. The forecasts are given as follows:

$$\hat{y}_{N+h} = \hat{f}_h(y_N, \dots, y_{N-d+1}) \quad (4)$$

The direct strategy does not use any approximated values to compute the forecasts, protecting them from the error accumulation formed by the recursive strategy. However, the  $H$  models are learned independently, inducing conditional independence of the  $H$  forecasts. This resulted in a negative effect on the forecasting accuracy. On the other hand, it is known that this strategy does not consider complex dependencies between the variables [22]. Therefore, it is an expensive computational strategy considering that it requires many models as the size of the adopted horizon [22].

3. Multi-input multi-output (MIMO) strategy learns the dependency between past observations and future values directly from data. In this case, the future values are a vector of values of a time series itself instead of scalar quantities [26,27]. The merits of the MIMO strategy is its capability to preserve the temporal stochastic dependency included in the predicted future values as it uses only one multi-output model. It constrains all the horizons to be predicted with a unified structure using the same learning algorithm. As a result, the forecasts are obtained in one-step at once by adopting the following multi-output model  $\hat{F}$ :

$$[\hat{y}_{t+H}, \dots, \hat{y}_{t+1}] = \hat{F}(y_N, \dots, y_{N-d+1}) \quad (5)$$

Regardless of the advantages and disadvantages of the first two strategies, it is clear that their original ideas still run as single-output functions but with multiple inputs. Indeed, prediction using a single-output function ignores the presence of stochastic dependencies among future values, which influence the overall prediction performance [28]. In contrast, many real-world applications have adopted this strategy to model and analyze univariate and multivariate TSF problems. In this paper, we extend the MIMO strategy to treat the HTSF problem summarized in the following subsection.

### 2.3. Hierarchical Time Series Forecasting

A hierarchical time series (HTS) is a collection of time series that follows a hierarchical aggregation structure with more than one level. Figure 1 demonstrates a simplified hierarchy, but a general hierarchical structure has  $K > 0$  levels, where level (0) contains the complete aggregated (top) time series. Each level from (1) to  $(K - 2)$  denotes a further disaggregation down to the level  $(K - 1)$ , which contains the most disaggregated (bottom) time series [5]. The structure of the hierarchy is determined by the summing matrix  $S$  that defines the aggregation constraints:

$$y_t = S y_{K,t} \quad (6)$$

where  $y_t$  is a vector of all observations in the hierarchy at time  $t$ ;  $S$  is the summing matrix of order  $m \times m_{K-1}$  that aggregates the bottom level series; and  $m$  and  $m_{K-1}$  denote the total number of series in the hierarchy and the number of series at the bottom level, respectively.

The hierarchy shown in Figure 1 can be expressed as follows:

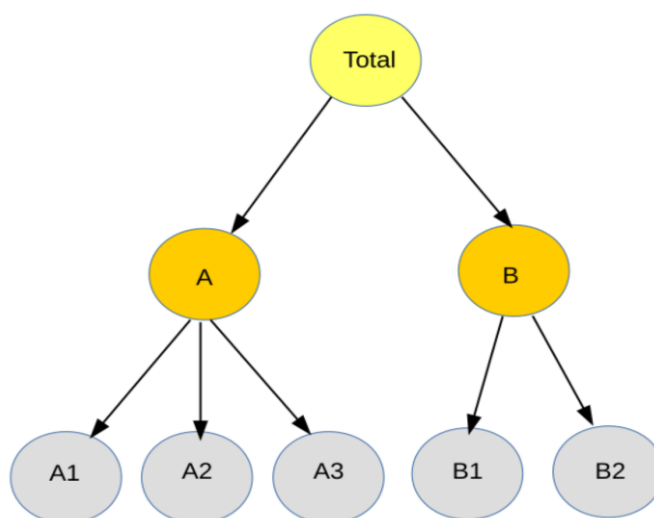


Figure 1. Two-level hierarchical time series structure.

$$\begin{bmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{A1,t} \\ y_{A2,t} \\ y_{A3,t} \\ y_{B1,t} \\ y_{B2,t} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ & & I_5 & & \end{bmatrix} X \begin{bmatrix} y_{A1} \\ y_{A2} \\ y_{A3} \\ y_{B1} \\ y_{B2} \end{bmatrix} \quad (7)$$

Compared to the original TSF problem, both forecasting accuracy of every series and aggregation consistency between levels are required in hierarchical forecasting. Aggregation consistency requires that the forecasts summation of the most disaggregated series are equal to the forecasts of the aggregated series; a constraint is known as coherence [3,5]. To ensure coherent forecasts, a two-step procedure is usually adopted in the literature. First, all of the time series at each level of the hierarchy are forecasted independently. These forecasts in this step are usually called “base” forecasts; however, they are usually not aggregated consistently. Therefore, it is often necessary to carry out a reconciliation approach in the second step to adjust the lower level forecasts and to make them coherent in the higher levels [5]. Many reconciliation approaches are introduced in the literature, including the bottom-up, top-down, and middle-out procedures [6,29].

It is demonstrated that the overall performance of these reconciliation approaches is not stable, and producing coherent forecasts is highly dependent on the application at hand. As an alternative, the optimal reconciliation approach using a linear regression algorithm is presented to overcome the limitations of the reconciliation approaches mentioned above, trying to produce more coherent forecasts [5,6,9]. Recently, different machine learning algorithms [12,14–16] are used to estimate the proportions of time series through the hierarchy as well as to derive the combination weights for the forecasts in the optimal combination approach. This paper develops a deep neural network-based transfer learning approach that runs in one-step, overcomes the limitations of state-of-the-art approaches, shows better prediction accuracy, and maintains coherency.

### 3. The Proposed Approach

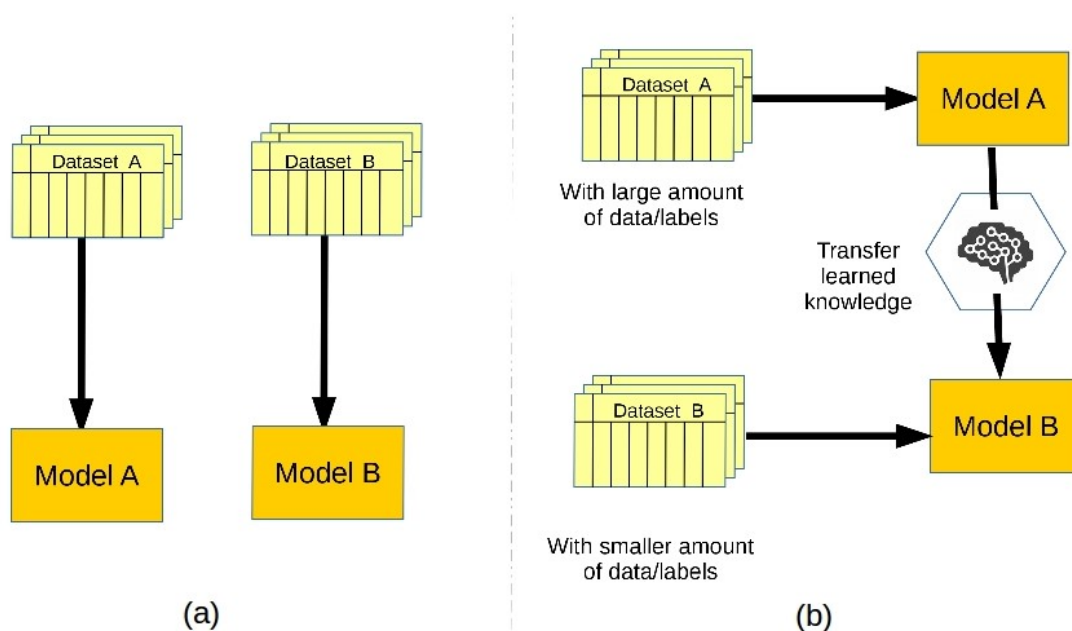
In this section, we present a novel forecasting approach that exploits the potential of our previous model DLSTM [17] in auto-encoder fashion. In the following subsections, we provide more details about the elements of the proposed approach.

#### 3.1. Transfer Learning in TSF

Recently, machine learning algorithms have been used widely to model and analyze different time series problems, including univariate, multivariate, and hierarchical architectures. In this concern, traditional machine learning algorithms, including ANNs, require the training and testing data to be sampled from the same domain, therefore having the same data distribution to perform a specific task. Nevertheless, we know that training ANNs require expensive resources and high computational costs, mainly when a similar process is performed for different tasks. The computational cost worsens and increases exponentially when the training model uses the DNN algorithm as a learning model. Transfer learning can address this problem and decrease the computation cost drastically.

In the general transfer learning approach, we first train a “base” model and learn meaningful features of a large-scope dataset to achieve a specific task. Then, we reuse the learned features, i.e., transfer them, onto another “target” network to be trained on a target dataset to perform a new, or similar, task [30]. In this way, transfer learning can overcome the limitations of traditional machine learning algorithms for training multiple deep learning models. Figure 2 simplifies a comparison between traditional machine learning and transfer learning algorithms. The traditional machine learning algorithms (Figure 2a) learn from a single dataset, and each traditional machine learning model operates separately. While transfer learning algorithms (Figure 2b) utilizes the knowledge gained from many source datasets, from a specific domain, and transfer them into the target domain. In this way, transfer learning significantly reduces the computational cost along with enhances the overall performance.

In our treatment of hierarchical forecasting in this paper, transfer learning is used to train the most disaggregated series, i.e. the bottom level, to produce the “base” forecasts. Then, leveraging the learned features in the bottom level to estimate the forecasts at the upper, or target, levels. Estimating the upper layers’ forecasts is executed in parallel, which reduces the computation costs, particularly in large hierarchies. In addition, it reduces the amount of data required for training the models through the hierarchy levels. Previously, transfer learning has applied for regression and classification problems in machine learning [31]. Recently, transfer learning have been applied for time series forecasting for real-world problems [32–34]. However, according to the best of our knowledge, this is the first time that transfer learning is employed in hierarchical forecasting problems.



**Figure 2.** Representation of (a) traditional machine learning (b) transfer Learning.

### 3.2. Long Short-Term Memory (LSTM)

Sequence-to-sequence (seq2seq) learning is used in language translation, speech recognition, and recently in time series forecasting [35]. Conventional neural networks assumed that all observations of the input vectors are independent of each other. As a result, the traditional neural network cannot utilize the sequential information included in time series data. Contrary to the conventional neural networks, the recurrent neural network (RNN) approaches are used to generate a sequence of data such that each observation is supposed to be dependent on the previous ones [36]. As an elegant variation of RNN, LSTM is a recurrent neural network approach that can be applied to model sequential data as well [21]. RNN and its variants are trained to map an input sequence into an output sequence, where the network's delay recursion enables it to represent the dynamic performance of sequential systems [37,38].

As shown in Figure 3, the LSTM memory cell comprises four gates, namely, the input gate, the output gate, the forget gate, and the candidate (input modulation) gate; each has a different purpose. Precisely, the input gate controls whether to write data to the cell state. The output gate controls what data to pass as the output hidden state. The forget gate controls whether to erase data from the cell state. Finally, the candidate gate controls what data to write to the cell state. The following recursive equations show how the LSTM cell works.

$$i_t = \sigma(W_{xi}X_t + W_{hi}h_{t-1} + b_i) \quad (8)$$

$$o_t = \sigma(W_{xo}X_t + W_{ho}h_{t-1} + b_o) \quad (9)$$

$$f_t = \sigma(W_{xf}X_t + W_{hf}h_{t-1} + b_f) \quad (10)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}X_t + W_{hc}h_{t-1} + b_c) \quad (11)$$

$$h_t = o_t \odot \tanh(c_t) \quad (12)$$

where  $i_t$ ,  $o_t$ , and  $f_t$  are the LSTM gating for the cell state to input, output, and forget information.  $c_t$  and  $h_t$  are the cell memory state vector and the hidden state vector, respectively.  $\sigma$  represents the sigmoid function, and  $x_t$  is the input vector.  $Ws$  are linear transformation matrices whose parameters need to be learned for each gate and cell memory, while  $bs$



is the corresponding bias vector. For simplicity, *LSTM* could be represented in one form as follows:

$$h_t, c_t = LSTM(x_t, h_{t-1}, c_{t-1}) \quad (13)$$

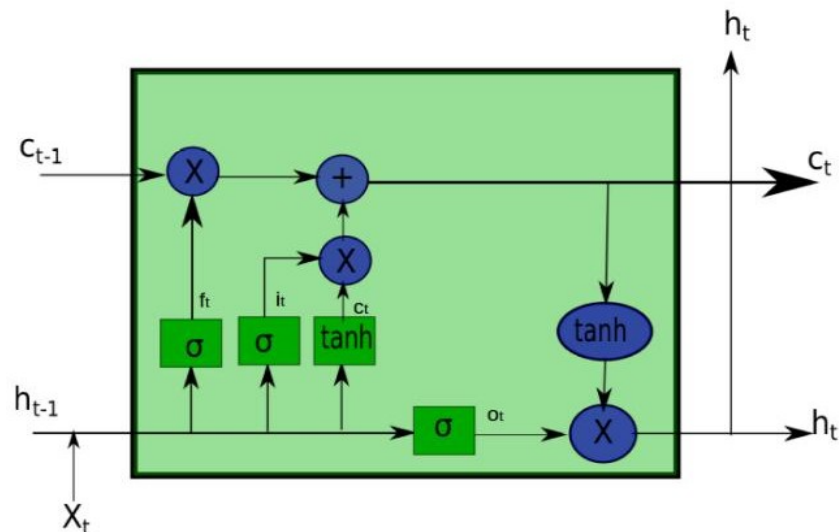


Figure 3. The internal structure of one LSTM cell.

Recently, the first author of this paper introduced a deep LSTM (DLSTM) approach [17] as an extension of the shallow LSTM and used it to model various time series problems. The presented model includes a stack of LSTM blocks (or layers), one after another and interconnected in a deep architecture to combine the advantages of every single LSTM [39]. It runs as follows, each LSTM block operates at a different time scale and, therefore, processes a specific part of the desired task and, subsequently, passes it onto the following block until finally the last block that generates the network's output [17].

The goal of stacking multiple LSTM blocks in such a hierarchical architecture is to build the features at the lower levels that disentangles the factors of variations in the input data and then combines these representations at the higher levels. In [17], we empirically showed that this deep architecture ensures the recovery of the limitations of shallow neural network architectures, particularly when long interval time series datasets are used. In addition, the empirical assessment showed that the DLSTM could efficiently represent the nonlinear relationship between the system inputs and outputs. These advantages can meet the requirements of generating stable and dynamic “base” forecasts, where the learned knowledge, or features, are transferred to the upper layers to obtain coherent forecasts.

### 3.3. Auto-Encoder (AE)

The AE is a neural network model often used for data generation. It consists of three layers, namely, the input layer (encoder), hidden layer, and output layer (decoder), as shown in Figure 4. The AE training procedure comprises two phases; encoding and decoding. In the encoding phase, the model learns a compressed representation (or latent variables) of the input. In contrast, in the decoding phase, the model reconstructs the target from the compressed representation during the encoding phase. The AE algorithm provides state-of-the-art results in seq2seq based applications, such as language translation, and sentiment analysis. Hence, the multi-input multi-output TSF problem can also be treated in a seq2seq fashion, as we will show in this paper.

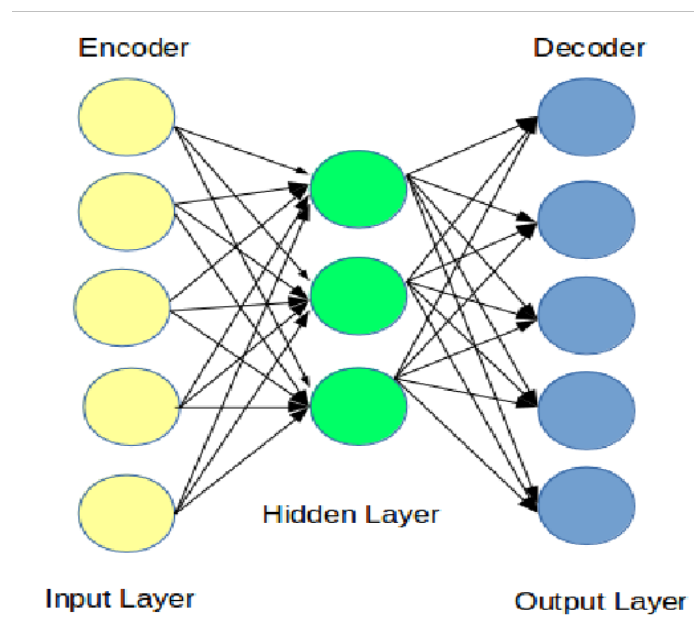


Figure 4. The auto-encoder architecture.

The AE works by encoding the input  $x(i)$  and maps it into  $h(x)$  following to Equation (14). Next, the decoder phase maps  $h$  into the output  $y(x)$  following to Equation (15).

$$h(x) = f(W_1X + b_1) \quad (14)$$

$$y(x) = g(W_2h + b_2) \quad (15)$$

where  $W_1$  and  $b_1$  are the weight matrix the bias vector or the encoder, respectively. Whereas, the  $W_2$  and  $b_2$  are the same but for the decoder. The training of AE includes finding the parameters  $W_1$ ,  $W_2$ ,  $b_1$ , and  $b_2$  that minimize the reconstruction error after back-propagating it through the network. The training objective is to minimize the reconstruction errors using the following squared errors,  $\mathcal{L} = \|x - y\|$ .

### 3.4. DLSTM-Based Auto-Encoder

The DLSTM-AE model is a novel architecture with the same structure as the original AE with some exceptions. As shown in Figure 5, we replace both the encoder layer with a DLSTM and the decoder layer with another DLSTM, which described in Section 3.2. The encoder layer converts the given input sequence into a fixed-length vector that acts as a summary of the input sequence. Usually, this fixed-length vector is called the context vector [40]. This context vector is fed as input to the decoder layer and the final encoder state as an initial decoder state to predict the output sequence. To use this architecture in a multi-step ahead forecasting, we add two layers, namely, repeat vector layer and time distributed dense layer. The repeat vector layer repeats the context vector, which we obtained from the encoder, and passes it as an input to the decoder. We repeat this for  $n$ -steps, where  $n$  is the number of future steps subject to forecast [41].

As the output received from the decoder concerning each time step is mixed, the time distributed densely applies a fully connected dense layer on each time step and separates the output for each time step [41]. Moreover, in the experiments of this paper, we used the proposed AE architecture in forecasting time series instead of reconstructing the input data. This deep architecture enables the model to learn complex and dynamic input sequential data from adjacent time intervals using multiple memory cells to remember long-term sequential data.

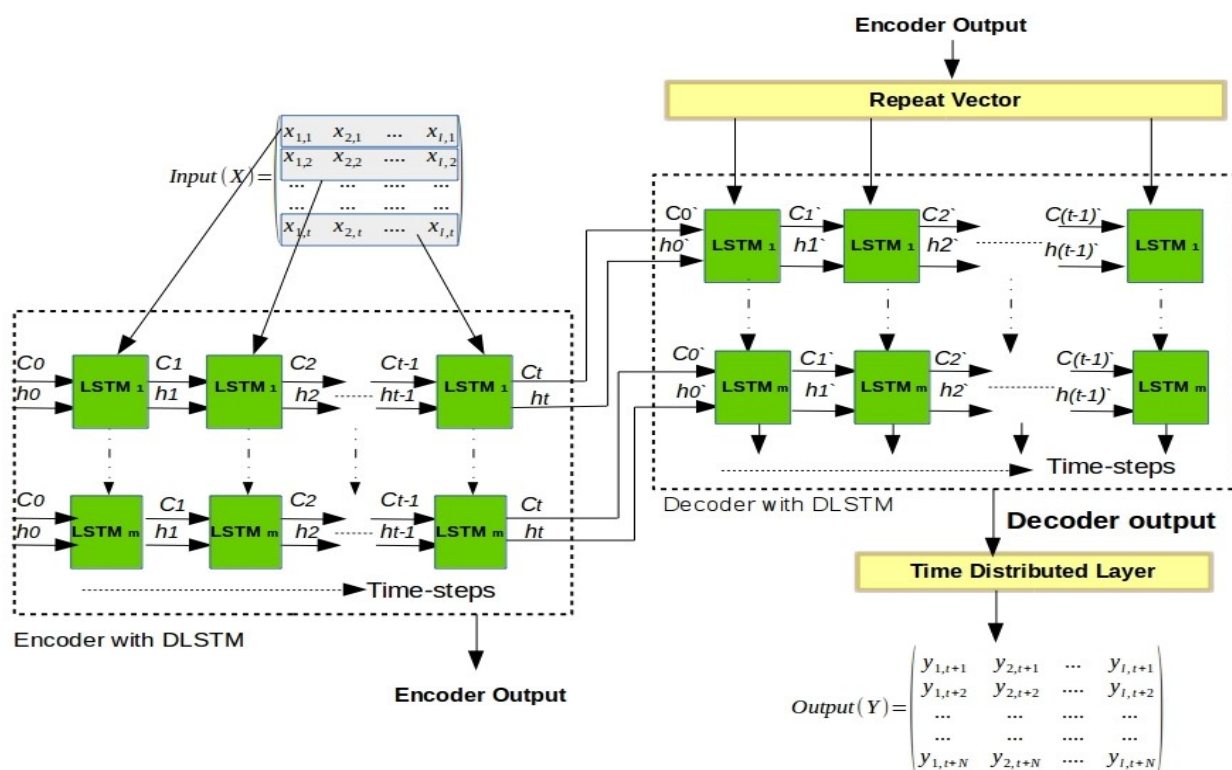


Figure 5. The DLSTM-based Auto-Encoder architecture.

It is worth mentioning that the proposed DLSTM-AE is different from the LSTM-based Stacked Auto-Encoder (LSTM-SAE) model presented recently by Sagheer et al. [18]. In the LSTM-SAE, the AE uses two LSTMs, one as the encoder and the other as a decoder. Then, we built several models similar to this in a stacked fashion. However, the AE in the proposed approach maintains the original AE structure but with two DLSTMs: one DLSTM as an encoder and one DLSTM as a decoder, such that each DLSTM contains several hidden layers, typically as shown Section 3.2.

### 3.5. Methodology

The proposed methodology to manipulate the hierarchical forecasting uses the DLSTM-AE model and the transfer learning strategy in a straightforward and simple procedure. According to its properties, the DLSTM enables the proposed approach to extract complex features and to capture more information as long as the LSTM is going deeper in the bottom level [17]. Meanwhile, the generated forecasts in the upper levels are coherent and reconciled successfully by adopting the transfer learning strategy.

As shown in Figure 6, the proposed methodology can be summarized in the following steps.

1. The encoder layer of the DLSTM-AE, which located at the bottom level of the hierarchy, receives as an input a sequence  $X$  of multiple time series that is treated jointly all at once as multivariate time series. This sequence can be represented as  $[X_{i,j}]$ , where  $i = 1, \dots, l$  terms to time series in the  $j = 1, \dots, t$  time step, as shown in Figure 5.
2. The encoder DLSTM recursively handles the input sequence ( $X$ ) of length  $t$  and updates the cell memory state vector  $C_t$  and hidden state vector  $h_t$  at each time step  $t$  using Equation (13). After ( $t$ ) time steps, the encoder summarizes the whole input sequence into the final vectors  $C_t$  and  $h_t$ .
3. A repeat vector layer takes the encoder output and passes it, as an input, to the decoder layer. Here, the function of the repeat vector is to repeat the decoder inputs to a number equals to future time steps that needed to perform the forecasting.

4. Then, the DLSTM at the decoder layer, at the bottom level, uses the final vectors  $C_t$  and  $h_t$  that passed from the encoder as initial cell memory state vector and initial hidden state vector  $C_{0'}$  and  $h_{0'}$  for  $t'$ -length time step.
5. The decoder DLSTM learns the features that included in the original input data and yields multiple outputs with  $N$ -time step ahead in a single shot following the MIMO strategy that described in Section 2.2.
6. The time distributed densely applies a fully connected dense layer on each time step and separates the output for each time-step. In this step, we can evaluate the performance of the DLSTM-AE by estimating the prediction accuracy at the bottom level using different performance metrics.
7. Once the bottom level is trained, we maintain (or freeze) the neurons' weight vectors of the DLSTM-AE model because they carry the features (or knowledge) of the input data. In the transfer learning terminology, this model is referred to as the "base" model.
8. We convey, or transfer, the trained DLSTM-AE model, or "base" forecast, to be used in the upper layers to learn the output layer of each upper level separately to generate the corresponding or "target" forecasts.

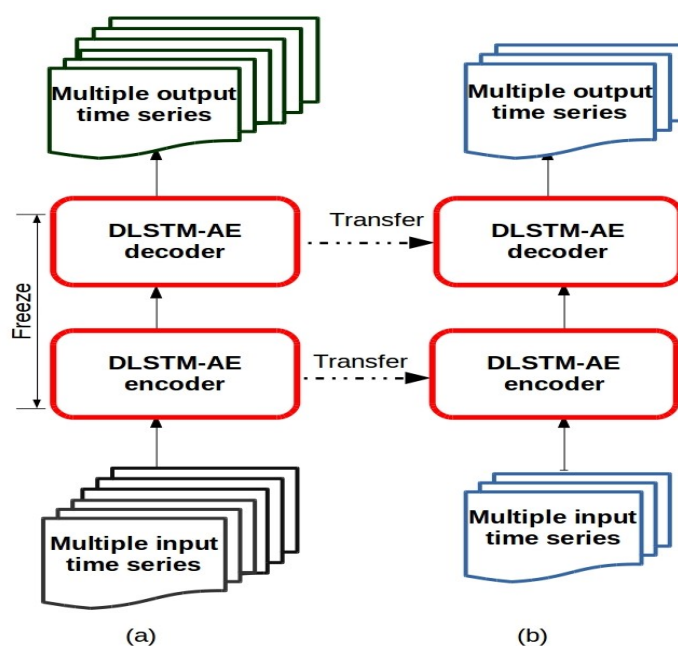


Figure 6. The DLSTM-AE model for (a) the bottom level (b) the upper levels

It is worth mentioning that taking the input (in steps 1–4) as a sequence of time series and producing multi-output in a single shot effectively reduces the overall computation. Additionally, based on the dataset and application in hands, we can implement the MIMO strategy (in step 5) either in one-step ahead or multi-step ahead prediction. We adopted nine-steps ahead for the experiments conducted in this paper, as shown in the empirical results section. As depicted in Figure 6, the trained models (in steps 1–6) are frozen before the transfer learning strategy that implemented in the higher levels in order to maintain, or not destroy, any of the information they contain [42].

As depicted in Figure 6, we utilize the "base" model to train the output layer in each higher level. At any higher level, it is possible that the length of the output sequences is smaller than the length of the input sequences included in the transferred "base" model. To solve this problem, zeros are inserted to each small-size sequence up to the established typical length in a process called zero-padding [43]. For auxiliary materials that may help to reproduce the proposed methodology, please refer to the GitHub page.

#### 4. Experimental Setting

This section presents the details of datasets, performance metrics used, and implementation platforms.

##### 4.1. Dataset-I: Brazilian Power Generation System

The first dataset utilized to assess the proposed approach corresponds to the amounts of power generation by each Brazilian electrical region, namely north, northeast, southeast/midwest, and south. Due to their reliability, the dataset was obtained from the Brazilian National Electric System Operator [44] as hourly power generation (GWh) between 1/2018 and 1/2020. The hierarchical aggregation structure of this dataset is demonstrated in Figure 7 and listed in Table 1. As we may notice, the structure of the dataset hierarchy allows us to model and analyze the hourly power generation in Brazil during the corresponding period.

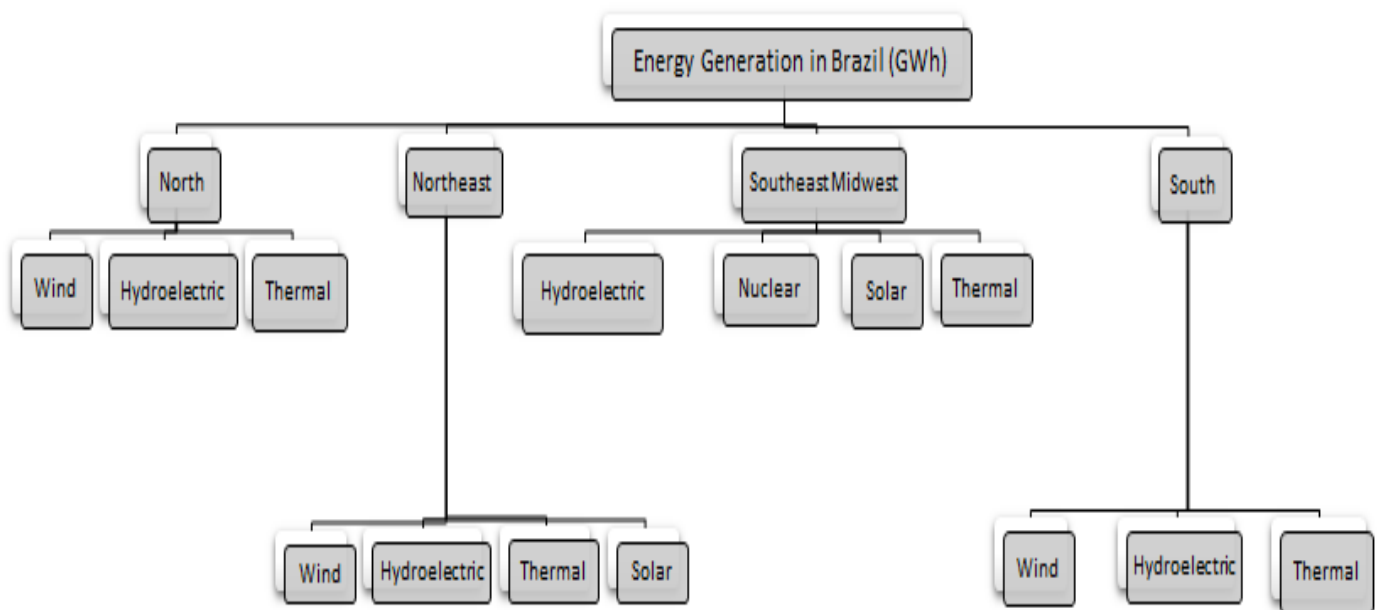


Figure 7. The Brazilian power generation system.

In the experiment of this case study, we adhered to the experimental setting that adopted in other research papers that manipulate this dataset [19]. Specifically, the data are categorized based on source type of power energy (wind, hydroelectric, thermal, solar, and nuclear) for each region or subsystem. The observations of hourly power generation (in GWh) were collected from January 2018 to January 2020, making 17,521 h.

Table 1. Dataset-I: Brazilian power generation hierarchy data structure.

Level (No.)	Group	No. of Series Per Level
(0)	Power energy generation system in Brazil	1
(1)	Power energy generation subsystems (regions)	4
(2)	Power energy generation sources	14

##### 4.2. Dataset-II: Australian Domestic Tourism

In this experiment, we utilized the Australian domestic tourism dataset samples obtained from the National Visitor Survey, managed by TRA [45]. This dataset was collected annually by computer-assisted telephone interviews from more than 120,000 Australians aged 15 years. In this experiment, we followed the same procedure adopted in the literature that utilized the number of visitor nights as a sign of tourism activities. Then,

we disaggregated the data based on a few purposes of travel: holiday, visiting friends and relatives, business, and others. The included observations span between 1998 to 2006, in a total of 36 quarterly observations. The structure of this hierarchy time series is displayed in Table 2.

**Table 2.** Dataset-II: Australian domestic tourism hierarchy data structure.

Level (No.)	Group	No. of Series Per Level
(0)	Australia	1
(1)	Purpose of Travel	4
(2)	States and Territories	28
(3)	Capital city versus others	56

In our experiments, we perform an out-of-sample forecast assessment to estimate all models using the first 12 observations (i.e., from 1998:Q1 to 2000:Q4) and then produced from 1-step ahead up to 8-step ahead forecasts. It is worth highlighting that we adhered to the experimental setting described in the original article to compare our proposed approach with it [20].

#### 4.3. Performance Metrics

To ensure a fair assessment, several kinds of performance metrics are used widely in literature to calculate two main performance errors: scale-dependent errors and percentage-dependent errors. In all performance metrics, the error is defined as actual (or observed) value minus the forecasted value.

- Scale-dependent errors that are on the same scale as the data itself. The most common scale-dependent metric is the root mean square error (RMSE) [46], which calculates the square root of the mean of the squares of all errors. It can be given as follows:

$$RMSE = \sqrt{(1/n \sum_{t=1}^n (y_t - \hat{y}_t)^2)} \quad (16)$$

The RMSE is the most used metric in the prediction assessment; however, it can not ensure coherence between the observations and their prediction within the same series. Therefore, we enhance the assessment by calculating the dRMSE, which is the RMSE but applied for two consecutive prediction values. In other words, dRMSE calculates the difference, or variation, between two consecutive forecasts and their corresponding observations within the same series. Intuitively, the smaller the value of dRMSE, the better the performance is [47]. In this case, the dRMSE can be given as follows:

$$dRMSE = \sqrt{(1/n \sum_{t=1}^n (\delta y_t - \delta \hat{y}_t)^2)} \quad (17)$$

- Percentage-dependent errors are more efficient in estimating the prediction precision since they have the advantage of being scale-independent. They are easy to understand because they provide the error in terms of percentages. Therefore, they are frequently used to assess the forecast performance among different scaled time series [17]. The most commonly used metric in this category is the mean absolute percentage error (MAPE), where percentage errors are summed regardless of the sign. MAPE can be given as follows:

$$MAPE = 1/n \sum_{t=1}^n (| (y_t - \hat{y}_t) / y_t |) * 100 (\%) \quad (18)$$

#### 4.4. Hardware and Software Platforms

All experiments in this paper were implemented on an HP workstation-PC with an Intel Core™ i7-6700 CPU at 3.40 GHz, 8.00 GB RAM, ×64 based processor, equipped with an Ubuntu 16.04 operating system with python 3.7 software environment. The proposed approach was developed using the Keras library with an open-source TensorFlow [48] library in the back-end.

### 5. Empirical Results and Assessment

To assess the proposed approach, we investigated its performance in two case studies using the two datasets described in the previous section. In this assessment, we were concerned with checking two criteria: first, the forecasting accuracy and, second, the forecast coherency within the hierarchy. To demonstrate the efficiency, we compared the forecasting accuracy of the proposed approach with other reference methods in solving the same forecasting problems. It is worth highlighting that all results demonstrated in this section are based on the testing (or unseen) data in each case study. The last column in each table labeled “Average” shows the average values across all the forecast horizons for each approach.

#### 5.1. Case Study I: Brazilian Electrical Power Generation

As depicted in Figure 7, this hierarchical dataset contains three levels. Therefore, the results displayed in Table 3 were estimated considering the hierarchical structure for the following levels,

- Level 0 (top): total power generation in Brazil
- Level 1 (middle): total power generation by four electrical subsystems (regions).
- Level 2 (bottom): total power generation by different fourteen sources.

For the first assessment criterion, which is related to forecasting accuracy, Table 3 shows the MAPE results of the proposed approach and other reference approaches. The tabulated results are computed from single-step up to nine-step ahead for all levels, where the average values are calculated and included in the last column in the table. The reference approaches are (BU) Bottom-up; (TDGSA) Top-down Gross-Sohl method A; (TDGSF) Top-down Gross-Sohl method F; (TDFP) Top-down forecast proportions; (OLS) Ordinary least-square; WLSs, Weighted least squares (structure scaling); WLSv, Weighted least squares (variance scaling); MinT (Sample), Minimum trace reconciliation; and MinT (Shrink), Minimum trace reconciliation (for more information about these models, please refer to [19]). All of the reference approaches generated the “base” forecasts using the ARIMA method, whereas the proposed approach generated the “base” forecast using the DLSTM-AE model.

It is easy to notice that the proposed approach attains a minor percentage error than other reference approaches presented in the table. Regarding the reference approaches, which are ARIMA-based, the results obtained using the top-down (TD) procedures show the highest percentage errors in all levels. Both the bottom-up (BU) and top-down (TD) approaches have another drawback; namely, they do not consider the correlation among the time series at each level. The MinT (Sample) approach attains the second-best performance after our approach, particularly for the one-step ahead forecast, among the ARIMA-based approaches. The MAPE values tend to be stable for the last three hours of the forecasting horizon. Figure 8 displays visual representations of the comparison between the proposed and reference approaches based on the average values.

Besides the MAPE performance metric, we performed other assessments using the RMSE and dRMSE metrics. Table 4 shows the results of these two metrics for the proposed approach only because the authors in [19] did not use these two metrics. The RMSE values are the difference between the predicted value and the corresponding observation for each time series on the same level. We can notice the small values of this metric, particularly in the earlier horizons or low-step ahead.

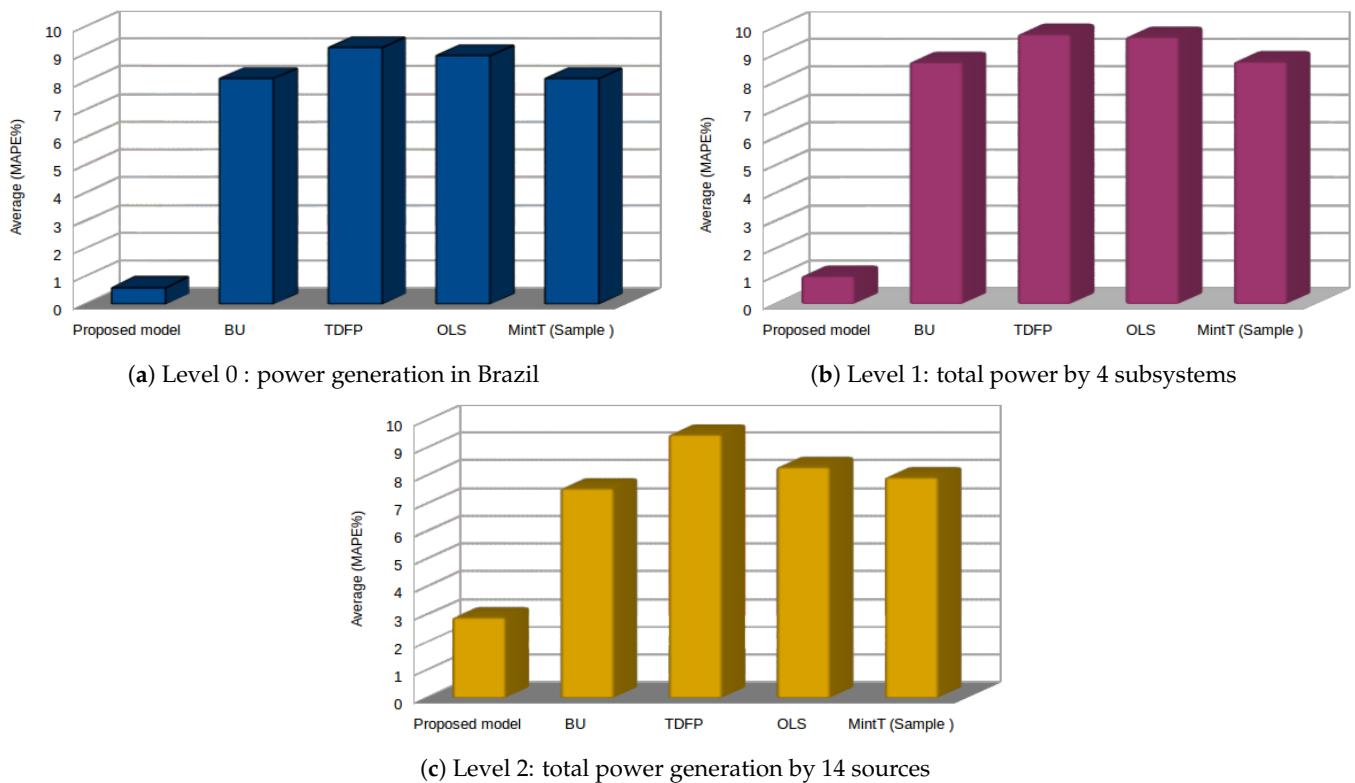
**Table 3.** Performance comparison based on the MAPE performance metric between the proposed approach and the reference approaches to solve the Brazilian power generation problem as reported in [19].

MAPE	Forecast Horizon ( <i>h</i> )									
	1	2	3	4	5	6	7	8	9	Average
Level 0 (Top): Total Electrical Power in Brazil										
<b>Proposed</b>	<b>0.04</b>	<b>0.87</b>	<b>0.80</b>	<b>0.85</b>	<b>0.83</b>	<b>0.83</b>	<b>0.76</b>	<b>0.74</b>	<b>0.48</b>	<b>0.60</b>
BU [19]	2.00	3.53	5.62	8.04	10.17	11.45	11.17	10.76	10.48	8.14
TDGSA [19]	2.07	3.76	6.10	8.79	11.25	12.87	12.95	12.72	12.63	9.24
TDGSF [19]	2.07	3.76	6.10	8.79	11.25	12.87	12.95	12.72	12.63	9.24
TDFP [19]	2.07	3.76	6.10	8.79	11.25	12.87	12.95	12.72	12.63	9.24
OLS [19]	1.98	3.65	5.94	8.59	10.99	12.54	12.55	12.23	12.06	8.95
WLSv [19]	1.91	3.51	5.70	8.24	10.51	11.93	11.79	11.32	10.99	8.43
WLSs [19]	1.88	3.46	5.63	8.15	10.39	11.77	11.59	11.09	10.76	8.30
MintT (Sample ) [19]	1.68	3.29	5.50	8.02	10.24	11.57	11.31	10.85	10.60	8.12
MinT (Shrink) [19]	1.74	3.36	5.59	8.12	10.35	11.69	11.44	10.94	10.69	8.21
Level 1—Electrical Subsystems										
<b>Proposed</b>	<b>0.17</b>	<b>0.98</b>	<b>1.2</b>	<b>1.31</b>	<b>1.21</b>	<b>1.16</b>	<b>1.09</b>	<b>1.06</b>	<b>1.06</b>	<b>1.02</b>
BU [19]	1.97	3.64	6.12	8.75	10.78	11.93	11.90	11.70	11.88	8.74
TDGSA [19]	31.97	31.74	30.37	28.93	28.12	27.49	26.71	26.04	25.36	28.53
TDGSF [19]	32.38	32.14	30.71	29.21	28.21	27.46	26.71	26.06	25.41	28.70
TDFP [19]	1.86	3.88	6.68	9.89	9.89	12.52	14.19	14.45	14.34	9.75
OLS [19]	1.90	3.55	6.30	9.20	11.70	13.36	13.64	13.56	13.66	9.65
WLSv [19]	1.77	3.35	5.84	8.62	10.84	12.38	12.56	12.40	12.41	8.91
WLSs [19]	1.81	3.41	5.92	8.74	11.00	12.57	12.79	12.68	12.75	9.07
MintT (Sample ) [19]	1.64	3.20	5.66	8.50	10.76	12.23	12.40	12.21	12.20	8.76
MinT (Shrink) [19]	1.66	3.28	5.75	8.57	10.84	12.33	12.50	12.31	12.28	8.83
Level 2—Sources										
<b>Proposed</b>	<b>0.60</b>	<b>1.20</b>	<b>3.89</b>	<b>4.26</b>	<b>3.76</b>	<b>3.42</b>	<b>3.01</b>	<b>3.01</b>	<b>3.02</b>	<b>2.91</b>
BU [19]	2.66	5.05	6.53	7.71	8.88	9.46	9.40	9.22	9.11	7.56
TDGSA [19]	46.33	44.34	41.72	40.35	39.87	39.29	38.44	37.52	36.58	40.49
TDGSF [19]	47.66	45.70	42.87	41.24	40.42	39.64	38.80	37.90	36.96	41.24
TDFP [19]	2.83	5.51	7.53	9.45	9.45	11.46	12.79	13.20	13.33	9.50
OLS	2.51	5.07	6.78	8.29	9.78	10.62	10.73	10.63	10.56	8.33
WLSv [19]	2.60	5.11	6.74	8.09	9.42	10.18	10.21	10.07	9.97	8.04
WLSs [19]	2.56	4.98	6.64	8.00	9.31	10.00	9.95	9.70	9.63	7.86
MintT (Sample ) [19]	2.48	4.96	6.58	7.91	9.27	10.10	10.22	10.10	10.00	7.96
MinT (Shrink) [19]	2.52	5.04	6.68	8.02	9.38	10.20	10.30	10.19	10.10	8.05

**Table 4.** Hierarchical forecasting for Brazilian power generation using other performance metrics.

Metric	Forecast Horizon ( <i>h</i> )									
	1	2	3	4	5	6	7	8	9	
Level 0 : Total—Brazil										
RMSE	$5.2 \times 10^{-4}$	$8.8 \times 10^{-3}$	$9.2 \times 10^{-3}$	$9.8 \times 10^{-3}$	$9.9 \times 10^{-3}$	$9.7 \times 10^{-3}$	$9.2 \times 10^{-3}$	$8.2 \times 10^{-5}$	$8.04 \times 10^{-3}$	
dRMSE	0.299	0.312	0.35	0.33	0.35	0.42	0.43	0.45	0.24	
Level 1 : Electrical subsystems										
RMSE	$2.6 \times 10^{-3}$	0.012	0.014	0.014	0.014	0.014	0.014	0.014	0.02	
dRMSE	9.3	6.68	6.71	5.67	5.81	6.30	6.53	7.50	1.68	





**Figure 8.** Performance comparison based on the MAPE metric for each level in case study I between the proposed approach and reference approaches.

As a special case from the RMSE, the dRMSE metric calculates the difference between two consecutive predictions and their corresponding values. In this way, we can ensure that the dRMSE refers to the accuracy of the predicted variations. Accordingly, this performance metric identifies the relationship and inherent correlation included in a time series. From Table 4, it is clear that the results values range from 0.2 and 0.4 for the top level and range from 1.7 and 9.3 for the middle level. However, the error values tend to decrease as the horizon increases. Moreover, dRMSE attains minor error values at the top level of the hierarchy compared to the lower level. This means that, at the higher levels of hierarchy, the forecast values tend to be more correlated. Overall, the dRMSE values are not optimum to some extent; a phenomenon may be attributed to the diversity or divergence that exists among the original data [19]. We attributed this attitude to the diversity of original data because we do not note this phenomenon in case study II.

The second criterion we consider in our assessment is the ability to produce coherent forecasts more simply than that adopted in reference approaches. Table 5 shows the values of the MSE performance metric for each level, after the bottom-level. The computations are performed in two steps: First, the forecasts for every series at the bottom level were summed to generate forecasts for all higher levels in the hierarchy. Second, we calculated the difference between these forecast values and the corresponding forecast values produced by the proposed approach for the same level. Intuitively, as long as this difference is tiny, the proposed approach yields coherent forecasts. It is easy to notice that most differences in all horizons are very tiny, particularly as we are going deeper in the hierarchy. This indicates that the proposed approach maintains forecast coherency through the hierarchy.

**Table 5.** Hierarchical forecasting for Brazilian power generation: Coherence performance metric.

Metric	Forecast Horizon (h)								
	1	2	3	4	5	6	7	8	9
Level 0 : Total—Brazil									
MSE	$1.15 \times 10^{-6}$	0.001	0.002	0.003	0.005	0.006	0.008	0.009	0.09
Level 1 : Electrical subsystems									
MSE	$4.2 \times 10^{-6}$	0.003	0.0003	0.0003	0.0002	0.0002	0.0001	0.0001	0.0002

### 5.2. Case Study II: Australian Visitor Nights of Domestic Tourism

This hierarchical dataset contains four levels, therefore, the results displayed in Table 6 were estimated considering the hierarchical structure for the following levels:

- Level 0 (top): total visitors in Australia;
- Level 1 (middle): total visitors according to the purpose of travel, with total four travel purposes;
- Level 2 (middle): total visitors according to a state visit, with a total of 28 state visits;
- Level 3 (bottom): total visitors according to city visit, with a total of 56 visits.

**Table 6.** Performance comparison based on the MAPE performance metric between the proposed approach and the reference approaches to solve the Australian tourism visitor nights problem as reported in [20].

MAPE	Forecast Horizon (h)								
	1	2	3	4	5	6	7	8	Average
Level 0 : Australia									
<b>Proposed</b>	<b>3.17</b>	<b>1.12</b>	<b>1.55</b>	<b>1.90</b>	<b>2.41</b>	<b>2.47</b>	<b>2.89</b>	<b>2.61</b>	<b>2.27</b>
Bottomup [20]	3.48	3.30	3.81	4.04	3.90	4.56	4.53	4.58	4.03
Top-down HP1 [20]	3.89	3.71	3.41	3.90	3.91	4.12	4.27	4.27	3.93
Top-down HP2 [20]	3.89	3.71	3.41	3.90	3.91	4.12	4.27	4.27	3.93
Top-down FP [20]	3.89	3.71	3.41	3.90	3.91	4.12	4.27	4.27	3.93
Optimal [20]	3.80	3.64	3.48	3.94	3.85	4.22	4.34	4.35	3.95
Level 1: Purpose of travel									
<b>Proposed</b>	<b>4.98</b>	<b>4.51</b>	<b>4.57</b>	<b>3.65</b>	<b>3.38</b>	<b>3.50</b>	<b>4.51</b>	<b>5.00</b>	<b>5.26</b>
Bottom-up [20]	6.15	6.22	6.49	6.99	7.80	8.15	8.21	7.88	7.24
Top-down HP1 [20]	9.83	9.34	9.34	9.67	9.81	9.52	9.88	9.81	9.65
Top-down HP2 [20]	10.01	9.56	9.55	9.84	9.98	9.71	10.06	9.97	9.84
Top-down FP [20]	5.73	5.78	5.58	6.15	6.80	7.28	7.56	7.68	6.57
Optimal [20]	5.63	5.71	5.74	6.14	6.91	7.35	7.57	7.64	6.59
Level 2: States									
<b>Proposed</b>	<b>8.47</b>	<b>7.98</b>	<b>6.54</b>	<b>6.58</b>	<b>6.02</b>	<b>5.14</b>	<b>4.91</b>	<b>5.07</b>	<b>6.34</b>
Bottom-up [20]	21.34	21.75	21.81	22.39	23.76	23.26	23.01	23.31	22.58
Top-down HP1 [20]	32.63	30.98	31.49	31.91	32.23	30.11	30.51	30.91	31.35
Top-down HP2 [20]	32.92	31.23	31.72	32.13	32.47	30.32	30.67	31.01	31.56
Top-down FP [20]	22.15	21.96	21.94	22.52	23.79	23.18	22.96	23.07	22.70
Optimal [20]	22.17	21.80	22.33	23.53	24.26	23.15	22.76	23.90	22.99
Level 3: Capital city versus other									
<b>Proposed</b>	<b>9.51</b>	<b>9.00</b>	<b>7.71</b>	<b>8.70</b>	<b>8.51</b>	<b>7.42</b>	<b>6.91</b>	<b>6.82</b>	<b>8.07</b>
Bottom-up [20]	31.97	31.65	31.39	32.19	33.93	33.70	32.67	33.47	32.62
Top-down HP1 [20]	42.47	40.19	40.57	41.12	41.71	39.67	39.87	40.68	40.79
Top-down HP2 [20]	43.04	40.54	40.87	41.44	42.06	39.99	40.21	40.99	41.14
Top-down FP [20]	32.16	31.30	31.24	32.18	34.00	33.25	32.42	33.22	32.47
Optimal [20]	32.31	30.92	30.87	32.41	33.92	33.35	32.47	34.13	32.55

For the first assessment of the forecasting accuracy, Table 6 shows the MAPE measurements of the proposed approach and other reference models. The results are shown for all levels from a single-step up to eight-steps ahead, where the average values are calculated and included in the last column of the table. The reference models in this case study are bottom-up, top-down based on historical proportion (HP1 and HP2), top-down based on forecast  $p$  (FP), and optimal combination (for more information about the setting of these models, please refer to [20]). All the reference models generated the “base” forecast using the exponential smoothing statistical technique, whereas the proposed approach generated the “base” forecast using the proposed DLSTM-AE model.

In Table 6, we can easily notice that the proposed approach attains a minor percentage error compared to other reference approaches displayed in the table. Regarding the reference models, the results obtained using the top-down procedure based on historical proportions (top-down HP1 and top-down HP2) show the highest percentage errors in all the hierarchy levels; however, they attain good results for forecasting at the top level of the hierarchy. The top-down procedure based on historical proportions (top-down FP) and optimal combination procedure attain the second-best performances after our proposed approach. In contrast, the bottom-up procedure performs better at the two top levels of the hierarchy and the worst at the two bottom levels. Figure 9 displays visual representations of this comparison, where we can notice the superiority of the proposed approach’s performance, particularly at the bottom level and the two middle levels.

Besides the MAPE performance metric, we conducted another assessment using the RMSE and dRMSE performance metrics. Table 7 shows the results of these two metrics for the proposed approach only because the authors in [20] did not consider these two metrics. Again, the RMSE value is the difference between the predicted value and the corresponding observation for each time series on the same level. We can notice the tiny values of this metric attained by the approach, particularly in the top layers of the hierarchy.

**Table 7.** Hierarchical forecasting for Australian tourism visitor nights using other performance metrics.

Measure	Forecast Horizon ( $h$ )							
	1	2	3	4	5	6	7	8
Hierarchical Level 0: Total—Australia								
RMSE	0.03	0.014	0.014	0.014	0.02	0.03	0.03	0.03
dRMSE	0.69	0.67	0.45	0.52	0.37	0.37	0.33	0.05
Hierarchical level 1: Purpose of travel								
RMSE	0.05	0.05	0.04	0.04	0.04	0.04	0.05	0.07
dRMSE	1.05	0.82	0.59	0.54	0.49	0.40	0.28	0.12
Hierarchical level 2: States								
RMSE	0.14	0.14	0.12	0.14	0.13	0.13	0.13	0.13
dRMSE	0.53	0.27	0.26	0.18	0.17	0.16	0.14	0.19

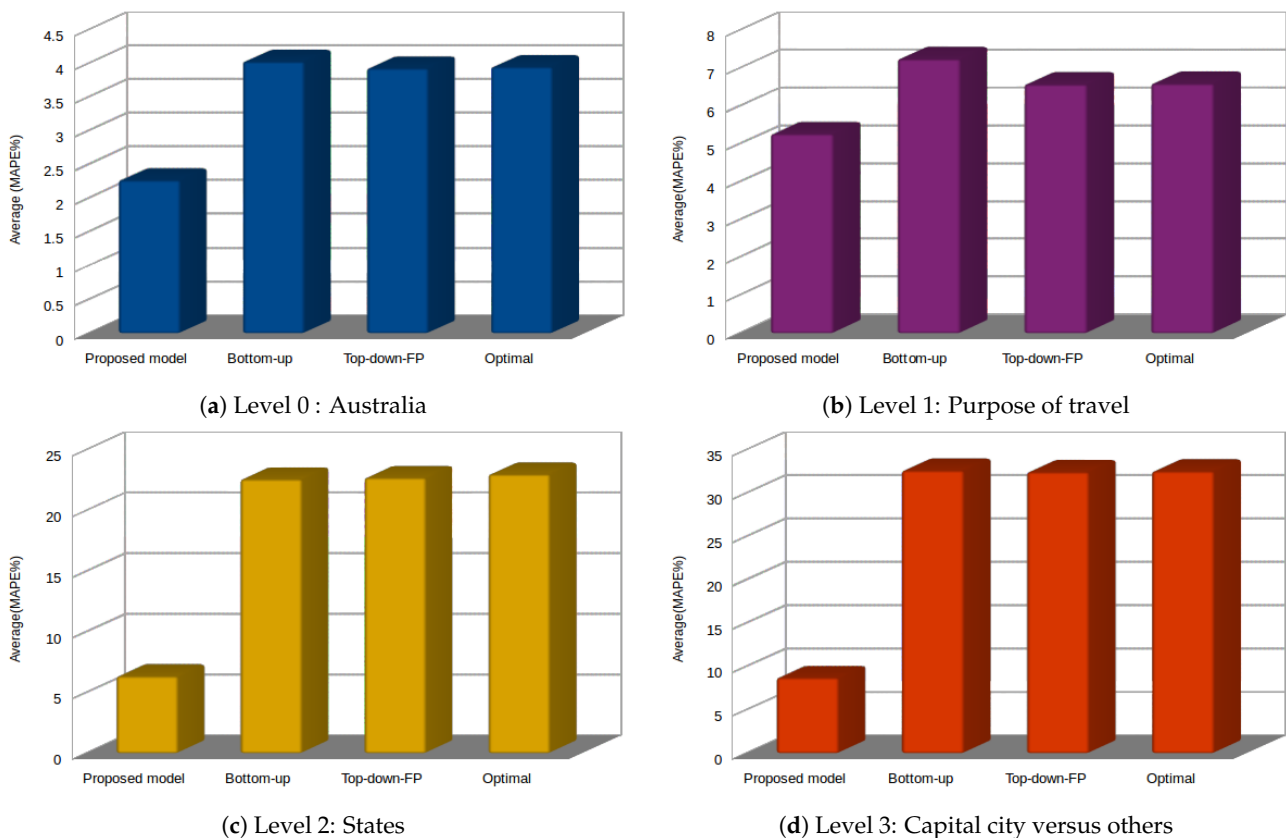
As defined before, the dRMSE metric calculates the difference between two consecutive predictions and their corresponding values. By this metric, we can identify the relationship and inherent correlation that exist in a time series. In Table 7, it is easy to notice that the error values tend to decrease as the horizon increases for all levels. As in case study I, the dRMSE showed more minimal values at the top level of the hierarchy than the lower levels. Again, this means that the forecast values tend to be more correlated at higher hierarchy levels. In contrast to case study I, the dRMSE values here are still ideal and attained lower values, which confirm our conclusion that the high values of dRMSE in case study I may be attributed to the heterogeneity of the original data samples.

In this case study, we continue to investigate the proposed approach’s capability to produce coherent forecasts in a straightforward way compared to that utilized in other

approaches. Table 8 shows the values of the MSE performance metric for each level in the same way that described in case study I. As we explained in the previous case study, as long as this difference is very small, the proposed approach produces coherent forecasts. It is clear that most differences in all horizons are very tiny, particularly for earlier horizons or low-step ahead. Newly, this confirms that the proposed approach maintains forecast coherency for all levels in the hierarchy.

**Table 8.** Hierarchical forecasting for Australian tourism visitor nights: Coherence performance metric.

Metric	Forecast Horizon ( <i>h</i> )							
	1	2	3	4	5	6	7	8
Hierarchical level 0: Total—Australia								
MSE	0.03	0.016	0.015	0.02	0.023	0.021	0.021	0.025
Hierarchical level 1: Purpose of travel								
MSE	0.0009	0.001	0.0019	0.0014	0.0011	0.0011	0.0019	0.0026
Hierarchical level 2: States								
MSE	0.00061	0.0008	0.9	0.1	0.1	0.03	0.05	0.1



**Figure 9.** Average performance comparison for each level in Case Study II: Australian visitor nights of domestic tourism.

## 6. Overall Discussion and Conclusions

Forecasting through hierarchical time series has always been a challenging task for traditional approaches and various machine learning approaches. The challenge here is twofold; the first is to attain high prediction accuracy through the hierarchical architecture. The second is to ensure the forecasting consistency among hierarchy levels based on their dimensional features, a phenomenon called coherency. However, existing approaches exhibit good results addressing these two challenges, often showing instability in the overall

prediction performance. This is because these approaches aim to minimize the prediction errors and to obtain good accuracy without questioning the coherence of consecutive predictions. As the granularity at which forecasts are needed increases, existing approaches may not scale well.

This paper addressed the hierarchical time series forecasting problem by extending our recent achievement, namely, the DLSTM model. Here, we reproduced the DLSTM model in auto-encoder (DLSTM-AE) fashion and implemented it in a transfer learning scenario. The proposed approach runs as follows: we first train the time series of the bottom level using DLSTM-AE to generate the “base” forecasts. Then, we freeze the weight vector of the “base” models and transferred the learned features to achieve synchronous training to the time series of the upper levels in the hierarchy.

Toward a fair evaluation, we compared the performance of the proposed approach with several existing approaches using two case studies belonging to different domains. The evaluation was based on two criteria: forecasting accuracy and the ability to produce coherent forecasts. The performance of all contenders was evaluated using three different performance metrics through multi-step ahead prediction mode. In both case studies, the proposed approach attained the highest accuracy results compared to other counterparts. We can attribute the goodness of the proposed approach to the use of DLSTM in generating the “base” forecasts at the bottom level compared to the existing approaches that use traditional statistical techniques, such as ARIMA and ES, to generate it. In this way, the individual forecasts at the bottom level are utilized on higher hierarchy levels to rapidly generate global forecasts without conducting a time-consuming parameter estimation as in existing approaches.

Moreover, the DLSTM-AE considered the relationship and temporal correlations among time series data and automatically extracted many inherent useful features that helped to generate the “target” forecasts. In contrast, most traditional techniques are static and ignore more helpful information. With the help of transfer learning, we significantly reduced the time for calculating the forecasts and substantially increase the forecasting efficiency for the higher level entities. The superiority of the proposed approach was not noticed only for the forecasting accuracy but also for the production of coherent forecasts. Using a standard performance metric, we empirically found that the difference between the forecast of a time series at a higher level and the sum of the corresponding time series forecasts at a lower level is tiny. This means that the proposed approach generated more coherent and consistent forecasts compared to reference models.

For future work, we plan to ensure forecast coherency at all levels of the hierarchy by running the proposed approach in a cross-temporal framework. Besides coherency, this framework will reduce the effect of the outliers and enhance the signal-to-noise ratio at aggregated lower frequencies of the time series while mitigating loss of information. In addition, despite describing our approach exemplarily through the tourism and energy domains, it can be easily adapted to other domains such as supply chain, sales promotion, and retail.

**Author Contributions:** Conceptualization: H.H., A.S. and H.Y.; methodology: H.H. and A.S.; software: H.H.; validation: H.H. and A.S.; installations and measurements: H.H.; formal analysis: H.H. and A.S.; investigation: H.H., A.S. and H.Y.; resources: A.S.; data curation: H.H. and A.S.; visualization: H.H.; writing—original draft preparation: H.H. and A.S.; writing—review and editing: A.S. and H.Y.; funding acquisition: A.S.; supervision: H.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This paper was funded by the Deputyship for Research and Innovation, Ministry of Education, Saudi Arabia, through the project number IFT20190.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The dataset of Australian tourism visitor nights is publicly available at: [https://robjhyndman.com/data/hier1\\_with\\_names.csv](https://robjhyndman.com/data/hier1_with_names.csv) accessed on 25 June 2021.

**Acknowledgments:** The authors extend their appreciation to the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia for funding this research through the project number (IFT20190).

**Conflicts of Interest:** The authors declare that there are no conflict of interest.

## References

1. Tealab, A. Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Comput. Inform. J.* **2018**, *3*, 334–340. [[CrossRef](#)]
2. Kotu, V.; Deshpande, B. *Chapter 12—Time Series Forecasting*, 2nd ed.; Kotu, V., Deshpande, B., Data Science, Eds.; Morgan Kaufmann: Burlington, MA, USA, 2019; pp. 395–445.
3. Hyndman, R.J.; Lee, A.J.; Wang, E. Fast computation of reconciled forecasts for hierarchical and grouped time series. *Comput. Stat. Data Anal.* **2016**, *97*, 16–32 [[CrossRef](#)]
4. Hyndman, R.J.; Athanasopoulos, G. *Forecasting: Principles and Practice*, 2nd ed.; OTexts: Melbourne, Australia, 2018.
5. Wickramasuriya S.L.; Athanasopoulos G.; Hyndman R.J. Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *J. Am. Stat.* **2019**, *114*, 804–819. [[CrossRef](#)]
6. Hyndman, R.J.; Ahmed, R.A.; Athanasopoulos, G.; Shang, H.L. Optimal combination forecasts for hierarchical time series. *Comput. Stat. Data Anal.* **2011**, *55*, 2579–2589. [[CrossRef](#)]
7. Zellner, A.; Tobias, J. A note on aggregation, disaggregation and forecasting performance. *J. Forecast.* **2000**, *19*, 457–469. [[CrossRef](#)]
8. Pennings, C.L.; van Dalen, J. Integrated hierarchical forecasting. *Eur. J. Oper. Res.* **2017**, *263*, 412–418. [[CrossRef](#)]
9. Hyndman, J.; George A. Optimally Reconciling Forecasts in a Hierarchy. *Foresight Int. J. Appl. Forecast.* **2014**, *35*, 42–48
10. Dannecker, L.; Lorenz, R.; Rösch, P.; Lehner, W.; Hackenbroich, G. Efficient forecasting for hierarchical time series. In Proceedings of the 22nd ACM international conference on Information & Knowledge Management (CIKM '13) 2013, San Francisco, CA, USA, 27 October–3 November 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 2399–2404. [[CrossRef](#)]
11. Almeida, V.; Ribeiro, R.; Gama, J. Hierarchical Time Series Forecast in Electrical Grids 2016. In *Information Science and Applications (ICISA) 2016*; Kim, K., Joukov, N., Eds.; Lecture Notes in Electrical Engineering; Springer: Singapore, 2016; Volume 376.
12. Spiliotis, E.; Abolghasemi, M.; Hyndman, J.; Petropoulos, F.; Assimakopoulos, V. Hierarchical forecast reconciliation with machine learning. *arXiv* **2020**, arXiv:2006.02043
13. Montgomery, D.; Jennings, C.; Kulahci, M. *Introduction to Time Series Analysis and Forecasting*, 2nd ed.; Wiley Series in Probability and Statistics; John Wiley and Sons: Hoboken, NJ, USA, 2015.
14. Abolghasemi, M.; Hyndman, J.; Tarr, G.; Bergmeir, C. Machine learning applications in time series hierarchical forecasting. *arXiv* **2019**, arXiv:1912.00370.
15. Shiratori, T.; Kobayashi, K.; Takano, Y. Prediction of hierarchical time series using structured regularization and its application to artificial neural networks. *PLoS ONE* **2020**, *15*, e0242099.
16. Mancuso, P.; Piccialli, V.; Antonio, M. A machine learning approach for forecasting hierarchical time series. *Expert Syst. Appl.* **2021**, *182*, 115102. [[CrossRef](#)]
17. Sagheer, A.; Kotb, M. Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing* **2019**, *323*, 203–213. [[CrossRef](#)]
18. Sagheer, A.; Kotb, M. Unsupervised Pre-training of a Deep LSTM-based Stacked Autoencoder for Multivariate Time Series Forecasting Problems. *Sci. Rep. Nat.* **2019**, *9*, 19038. [[CrossRef](#)]
19. Gontijo, S.; Costa, A. Forecasting Hierarchical Time Series in Power Generation. *Energies* **2020**, *13*, 3722. [[CrossRef](#)]
20. Athanasopoulos, G.; Ahmed, R.; Hyndman, J. Hierarchical forecasts for Australian domestic tourism. *Int. J. Forecast.* **2009**, *25*, 146–166. Corrigendum to *Int. J. Forecast.* **2015**, *31*, 585. [[CrossRef](#)]
21. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [[CrossRef](#)]
22. Ben Taieb, S.; Bontempi, G.; Atiya, A.; Sorjamaa, A. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Syst. Appl.* **2012**, *39*, 7067–7083. [[CrossRef](#)]
23. Masum, S.; Liu, Y.; Chiverton, J. Multi-step Time Series Forecasting of Electric Load Using Machine Learning Models. In *Artificial Intelligence and Soft Computing. ICAISC 2018. LNCS*; Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J., Eds.; Springer: Cham, Switzerland, 2018; Volume 10841.
24. Sorjamaa, A.; Hao, J.; Reyhani, N.; Ji, Y.; Lendasse, A. Methodology for long-term prediction of time series. *Neurocomputing* **2007**, *70*, 2861–2869. [[CrossRef](#)]
25. Hamzaebi, C.; Akay, D.; Kutay, F. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert Syst. Appl.* **2009**, *36*, 3839–3844. [[CrossRef](#)]
26. Bontempi, G. Long term time series prediction with multi-input multi-output local learning. In Proceedings of the 2nd European Symposium on Time Series Prediction (TSP), ESTSP08, Helsinki, Finland, 17–19 September 2008 ; pp. 145–154.

27. Chen, J.; Wang, X. Multi-innovation Generalized Extended Stochastic Gradient Algorithm for Multi-Input Multi-Output Nonlinear Box-Jenkins Systems Based on the Auxiliary Model 2010. In *Life System Modeling and Intelligent Computing. ICSEE 2010, LSMS 2010*; Li, K., Fei, M., Jia, L., Irwin, G.W., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6328.
28. Li, I.-H.; Lee, L.-W. A hierarchical structure of observer-based adaptive fuzzy-neural controller for MIMO systems. *Fuzzy Sets Syst.* **2011**, *185*, 52–82. [[CrossRef](#)]
29. Widiarta, H.; Viswanathan, S.; Piplani, R. Forecasting aggregate demand: An analytical evaluation of top-down versus bottom-up forecasting in a production planning framework. *Int. J. Prod. Econ.* **2009**, *118*, 87–94. [[CrossRef](#)]
30. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 9. [[CrossRef](#)]
31. Torrey, L.; Shavlik, J. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*; Olivás, E.S., Guerrero, J.D.M., Martínez-Sober, M., Magdalena-Benedito, J.R., López, A.J.S., Eds.; IGI Global: Hershey, PA, USA, 2010; Volume 1, pp. 242–264.
32. Zeng, P.; Sheng, C.; Jin, M. A learning framework based on weighted knowledge transfer for holiday load forecasting. *J. Mod. Power Syst. Clean Energy* **2019**, *7*, 329–339. [[CrossRef](#)]
33. Cai, L.; Gu, J.; Jin, Z. Two-Layer Transfer-Learning-Based Architecture for Short-Term Load Forecasting. *IEEE Trans. Ind. Inform.* **2020**, *16*, 1722–1732. [[CrossRef](#)]
34. Cao, L.; Wang, L.; Huang, C.; Luo, X.; Wang, J.H. A Transfer Learning Strategy for Short-term Wind Power Forecasting 2018. In Proceedings of the 2018 Chinese Automation Congress (CAC), Xi'an, China, 30 November–2 December 2018; pp. 3070–3075.
35. Du, S.; Li, T.; Horng, S. Time Series Forecasting Using Sequence-to-Sequence Deep Learning Framework 2018. In Proceedings of the 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), Taipei, Taiwan, 26–28 December 2018; pp. 171–176. [[CrossRef](#)]
36. Das, A.; Patra, G.R.; Mohanty, M.N. A Comparison Study of Recurrent Neural Networks in Recognition of Handwritten Odia Numerals 2021. In *Advances in Electronics, Communication and Computing*; Lecture Notes in Electrical Engineering; Mallick, P.K., Bhoi, A.K., Chae, G.S., Kalita, K., Eds.; Springer: Singapore, 2021; Volume 709, 26. [[CrossRef](#)]
37. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.R.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [[CrossRef](#)]
38. Sutskever, I. Training Recurrent Neural Networks. Ph.D. Thesis, University of Toronto, Toronto, ON, Canada, 2012.
39. Pascanu, R.; Gulcehre, C.; Cho, K.; Bengio, Y. How to construct deep recurrent neural networks. In Proceedings of the Second International Conference on Learning Representations ICLR, Banff, AB, Canada, 14–16 April 2014.
40. Wong, T.; Luo, Z. Recurrent Auto-Encoder Model for Large-Scale Industrial Sensor Signal Analysis. *arXiv* **2018**, arXiv:1807.03710v1.
41. Lane, H.; Hapke, H.; Howard, C. *Natural Language Processing in Action: Understanding, Analyzing, and Generating Text with Python*, 1st ed.; Simon and Schuster: New York, NY, USA, 2019.
42. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks. In Proceedings of the Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3320–3328.
43. Dwarampudi, M.; Reddy, S. Effects of padding on LSTMs and CNNs. *arXiv* **2019**, arXiv:1903.07288.
44. National System Operator. Operation History (Report of Power Generation). 2020. Available online: <http://www.ons.org.br/paginas/resultados-da-operacao/historico-da-operacao> (accessed on 15 May 2020).
45. Athanasopoulos, G.; Hyndman, J. Modelling and forecasting Australian domestic tourism. *Tour. Manag.* **2008**, *29*, 19–31. [[CrossRef](#)]
46. Hyndman, R.J.; Koehler, A.B. Another look at measures of forecast accuracy. *Int. J. Forecast.* **2006**, *22*, 679–688. [[CrossRef](#)]
47. De Bois, M.; Yacoubi M.A.E.; Ammi, M. Prediction-Coherent LSTM-Based Recurrent Neural Network for Safer Glucose Predictions in Diabetic People 2019. In *Neural Information Processing. ICONIP 2019*; Lecture Notes in Computer Science; Gedeon, T., Wong, K., Lee, M., Eds.; Springer: Cham, Switzerland, 2019; Volume 11955.
48. Martín, A.; Ashish, A.; Paul, B.; Eugene, B.; Zhifeng, C.; Craig, C.; Greg, C.; Andy, D.; Jeffrey, D.; Matthieu, D.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems 2015. Available online: [tensorflow.org](https://www.tensorflow.org) (accessed on 5 February 2021).