energies

MDPI

Article

Stacking Ensemble Methodology Using Deep Learning and ARIMA Models for Short-Term Load Forecasting

Pedro M. R. Bento [1,2], Jose A. N. Pombo [1], Maria R. A. Calado [1,2] and Silvio J. P. S. Mariano [1,2,*]

[1] Department of Electromechanical Engineering, Faculty of Engineering, University of Beira Interior, 6201-001 Covilhã, Portugal; pedro.bento@lx.it.pt (P.M.R.B.); jose.pombo@ubi.pt (J.A.N.P.); rc@ubi.pt (M.R.A.C.)
[2] IT-Instituto de Telecomunicações, 6201-001 Covilhã, Portugal
* Correspondence: sm@ubi.pt; Tel.: +351-275329760

Abstract: Short-Term Load Forecasting is critical for reliable power system operation, and the search for enhanced methodologies has been a constant field of investigation, particularly in an increasingly competitive environment where the market operator and its participants need to better inform their decisions. Hence, it is important to continue advancing in terms of forecasting accuracy and consistency. This paper presents a new deep learning-based ensemble methodology for 24 h ahead load forecasting, where an automatic framework is proposed to select the best Box-Jenkins models (ARIMA Forecasters), from a wide-range of combinations. The method is distinct in its parameters but more importantly in considering different batches of historical (training) data, thus benefiting from prediction models focused on recent and longer load trends. Afterwards, these accurate predictions, mainly the linear components of the load time-series, are fed to the ensemble Deep Forward Neural Network. This flexible type of network architecture not only functions as a combiner but also receives additional historical and auxiliary data to further its generalization capabilities. Numerical testing using New England market data validated the proposed ensemble approach with diverse base forecasters, achieving promising results in comparison with other state-of-the-art methods.

Keywords: ARIMA models; correlation analysis; deep learning; deep neural networks; ensemble methods; ISO New England; load forecasting; short-term load forecasting

check for updates

Citation: Bento, P.M.R.; Pombo, J.A.N.; Calado, M.R.A.; Mariano, S.J.P.S. Stacking Ensemble Methodology Using Deep Learning and ARIMA Models for Short-Term Load Forecasting. *Energies* **2021**, *14*, 7378. https://doi.org/10.3390/en14217378

Academic Editors: Filipe Rodrigues and João M. F. Calado

Received: 14 September 2021
Accepted: 3 November 2021
Published: 5 November 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

The large-scale and complex nature of Power Systems Operation in a constantly changing environment poses a series of challenges for policy makers, regulators, market operators and participants (both the generation and demand sides) and transmission and distribution operators, among others [1]. One of these challenges is the task of electrical power load forecasting. This is crucial for the smooth operation of the power grid—which runs on a set of tight tolerance requirements—and the decision-making policies of utility companies, i.e., influencing the purchase and generation of electric power in the power market and consequently affecting operation costs, energy efficiency, load switching and infrastructural development [2]. In this regard, the increased level of variable renewable energy sources directly rises the required spinning reserve volume to offset the fluctuation of renewable generation [3]. Factors such as an expanded network of distributed energy resources and a higher share of demand-side management further add to the complexity of this prediction task [4], which is already influenced by several uncertain factors including climate and seasonal factors [5,6].

Given that a large part of market decisions and utilities operations have an hourly, daily or weekly basis, these types of forecasting lead times (from 1 h to 168 h) are of most interest. An interval that is well categorized in the electric power system literature is the "Short-Term Load Forecast (STLF)" [7], which has long been a very active field of research,

and which is the object of renewed interest given the advent of smart grids and microgrids [8]. This has led researchers to continuously follow and adapt the developments in data filtering, clustering and modeling in order to propose load forecasting methodologies able to support the decision-making process regarding crucial operation, maintenance and planning tasks. Typically, an underestimation of load demand results in unreliable supply, with penalty costs or even shortage of supply to customers, while an overestimation causes economic waste and results in curtailment periods [9].

Various methodologies have been used for the STLF problem, with the two main groups being statistical methods and machine learning methods. The former comprises methods specially tailored to handle the inherent linear correlation between the load and its lagged values, as well as weather and temporal variables. Among the many statistical methods, one can highlight the well-regarded autoregressive integrated moving average (ARIMA) models, exponential smoothing and generalized autoregressive conditional hetero-scedasticity (GARCH) models, among others [10].

An alternative path that has been widely adopted in recent years is using machine learning (ML) and deep learning (DL) given their ability to extrapolate and learn hard non-linear relationships through complex and dynamic structures [11], which are exactly the difficult aspects for statistical methods. Among the broad spectrum of methods, recurring options include Artificial Neural Networks [10] and Multi-Layer Perceptron (MLP), random forests [11], fuzzy systems (ANFIS) [12], support vector machines (SVMs) [13], recurrent neural networks (RNN) [14] and its modified counterparts, Long Short-Term Memory Networks (LSTM) [15,16] and gated recurrent units (GRU) [17], extreme learning machine (ELM) [18], deep belief networks (DBN) [19], echo state networks [20], convolutional neural networks (CNN) [21] and more recently deep stack of fully connected layers connected with forward and backward residual links (N-BEATS) [22].

In addition to the issue of the chosen forecaster (prediction method), critical aspects in STLF include selecting the proper input data and preprocessing (i.e., deciding which sets of variables to include in the model and its treatment). To illustrate this point, the authors in [23] analyzed the impact of the selection of exogenous variables on an ANFIS forecaster's prediction of load in the Polish power system (using the Hellwig method). In turn, the authors in [24] reinforced the importance of data preprocessing methods, specifically by employing the Variational Mode Decomposition (VMD) method. Other common options include empirical mode decomposition [12], wavelet transform [10] and principal component analysis among others.

A common approach in recent studies is to employ hybrid or ensemble approaches that combine several methods, extracting the best prediction accuracy from each one by primarily merging methods that model both linear and nonlinear components of the time-series [25]. For instance, a combined MLP-ANN with ANFIS was used for simultaneous load and price forecasting in [26]. In [27], a hybrid method was presented, where an ARIMA model was combined with SVMs. This generic approach of hybridizing an ARIMA-ANN method has been validated for time-series forecasting in [28].

Recognizing the growing role of ML methods with respect to solving the STLF task, a fusion algorithm was presented in [29] where SVM, Random Forest and LSTM forecasters were used to produce an enhanced prediction. Similarly, an ensemble approach was published in [30] using VMD to decompose the load data and 2D-CNN to extract the features that fed the Temporal Convolutional Network working as forecaster. Using another ensemble concept, a method combining three models, namely, Light Gradient Boosting Machine, eXtreme Gradient Boosting machine (XGB) and MLP was presented in [31]. Lastly, a recent approach made use of stacked boosters (forecasters) with different horizons coupled with a deep neural network to perform the STLF [32].

Given these works and their merits, an ensemble approach will be introduced in this paper where a new enhanced method to accurately forecast the one-day ahead loads of national or sizable regional (as is the case of New England in the US) electric grids is proposed. We address the need to further develop data-driven forecasting techniques that

are compact in terms of user-dependence and computationally cost-efficient. To fulfill this objective, we chose an ensemble method based on a deep neural network, which due to its scalable and non-linearity traits is a great tool for enhancing feature extraction and better mapping the input space [33]. Thus, an ensemble approach has the potential to improve generalization and increase model stability (less output variance). In turn, different ARIMA models contribute to increasing input diversity through a stacked generalization framework. The straightforward formulation and the relatively small number of hyperparameters favors its robustness to model time-series based on the autocorrelation between past samples (only requires prior data), making it one of the preferred methods to benchmark proposed STLF models.

A brief description of the several methodological stages and its testing is listed: (i) a data selection process is used to assemble batches of training data with different sizes, ranging from recent to longer-term trends; (ii) a unit-root test is used to determine the need to include integration (number of time-series differences) in the ARIMA model parameters; (iii) after using the Box-Jenkins method, i.e., correlation analysis, a number of different seasonal and non-seasonal ARIMA models are fitted to the training data; (iv) the most promising models are hand-picked based on the Bayesian Information Criteria; (v) this process occurs automatically, a windowing technique is applied and the ARIMA model is rolled (updated) with the recently tested data; (vi) these models, which are able to capture most of the linear components in the time-series, work as diversified forecasters fed to the ensemble method in a stacking manner; (vii) the deep neural network is trained offline with not only the forecasters but also with additional load and exogenous variables with significant levels of Pearson correlation; (viii) the proposed methodology was tested using the well known aggregated load data from the New England region; (ix) the numerical results showed the validity of the proposed ensemble methodology, outperforming the standalone shallow neural network without and with the ARIMA forecasters and the exogenous inputs, and the standalone (different base learners) ARIMA models.

The content of this paper is summarized below. The most relevant theoretical background and formulation regarding the ARIMA models (forecasters) and the deep neural network architecture (assembler), including its non-linear computations, are given in Section 2; a detailed explanation of the different stages of the proposed methodology, as well as the considered parameter selection, is provided in Section 3; Section 4 introduces the case study and the underlying key statistics, which supported not only the design of the proposed methodology but also the analysis of results; Section 5 presents the most relevant numerical results and their interpretation, which includes a graphical analysis. Lastly, a summary of the main conclusions of this work is presented in Section 6.

2. Background

This section introduces the theoretical fundamentals behind each stage and building block considered in the proposed methodology.

2.1. Box-Jenkins Method

To study and model a generic time-series data X with a total of n_{length} sequentially spaced time observations, such as $X = (\chi_1, \chi_2, \ldots, \chi_{n_t-1}, \chi_{n_t})$, a simple and common choice is to use the Box-Jenkins method to identify and fit the most suitable Autoregressive Integrated Moving Average (ARIMA) time-series models. These statistical models constitute a good approach to tackle the forecasting task, as mentioned in the Introduction.

These models translate a linear relationship with the observed output data and rely on the assumption that future observations can be approximated based on past observations. Models that can be written using a single equation as expressed in Equation (1), with an autoregressive term, AR(p), where p is the autoregressive polynomial order and a moving average term, MA(q), where q is the moving average polynomial order, hence translating a linear combination of the preceding error terms up to lag q. Thus, the polynomial orders or degrees, $p, q \in \mathbb{Z}^+$, represent the maximum value for the AR and MA terms, respectively.

In turn, the integrated part is commonly represented by the letter d, which stands for the differencing order. In the case of $d = 0$, the ARIMA model is simply an ARMA model [34]:

$$\chi_t = \mu + \varepsilon_t + \sum_{i=1}^{p} \varphi_i \chi_{t-i} + \sum_{j=1}^{q} \theta_i \varepsilon_{t-j}, \quad (1)$$

where μ denotes the constant term (vertical translation), χ_{t-i} denotes the lagged (past) time-series (independent variable) and ε_{t-j} represents the unknown lagged error terms, i.e., the random error term that accounts for the difference between the actual observation and the theoretical value given by the regression-based model, often referred as white Gaussian noise. Therefore, the actual observation, χ_t, is influenced by factors beyond the independent variable. In turn, the autoregressive and moving average coefficients are denoted by φ_i and θ_i, respectively. Typically, when using the Box–Jenkins notation, a backshift operator, B, is used to represent the lagged terms such that $B\,\chi_t = \chi_{t-1}$. Thus, the model in Equation (1) is re-written as demonstrated in Equation (2).

$$\begin{gathered} \varphi_p(B)\chi_t = \mu + \theta_q(B)\varepsilon_t \Leftrightarrow \\ \left(1 - \varphi_1 B - \ldots . - \varphi_p B^p\right)\chi_t = \mu + \left(1 - \theta_1 B - \ldots . - \theta_q B^q\right)\varepsilon_t \end{gathered}, \quad (2)$$

In addition, most of the time-series encountered in engineering problems, including traditional energy load forecasting, reveal nonstationary characteristics, meaning that the unconditional joint probability distribution changes with time (a non-constant mean and variance across observations). These traits in the time-series reveal the presence of trends and/or seasonal variations. Consequently, since the Box-Jenkins models assume stationarity, one approach is to obtain the first differences of the time-series, i.e., $\chi_t - \chi_{t-1} = (1 - B)\chi_t$, which results in a general version of the ARIMA(p, d, q) model with an added generic difference term, as shown in Equation (3).

$$\begin{gathered} \varphi_p(B)(1 - B)^d\chi_t = \mu + \theta_q(B)\varepsilon_t \Leftrightarrow \\ \left(1 - \varphi_1 B - \ldots . - \varphi_p B^p\right)\left(\varphi_1 B + \ldots . + \varphi_p B^p\right)^d\chi_t = \mu + \left(1 - \theta_1 B - \ldots . - \theta_q B^q\right)\varepsilon_t \end{gathered}, \quad (3)$$

To determine the order at which the original time-series must be integrated to acquire stationarity, authors often resort to unit root tests to assess the stationarity hypothesis. For the vast range of time-series, when the unit root test determines that the data needs to be integrated, due to variance in the time-series, the first and second differences are usually ruled sufficient for removing the trends [10]. In this scenario, where the time-series reveals clear features of seasonality, e.g., daily, monthly and semesterly cycles, among others, it is advantageous to use a seasonal ARIMA$(p, d, q) \times (P, D, Q)$ (SARIMA) model [35]. The same letters but capitalized, P, D and Q, describe the seasonal orders of the autoregressive, moving average and differentiation terms; conversely, the lower-case letters represent the non-seasonal term orders. As such, the generic abbreviated form of the multiplicative model can be written as follows Equation (4):

$$\varphi_p(B)\Phi_P(B^s)(1 - B)^d(1 - B^s)^D\chi_t = \mu + \theta_q(B)\Theta_Q(B^s)\varepsilon_t, \quad (4)$$

where the superscript s distinguishes the seasonal terms from the non-seasonal, whereas the seasonal autoregressive and moving average coefficients are denoted by Φ_P and Θ_Q, respectively. All the remaining parameters were already defined in Equations (1) and (2).

These statistical models are especially tailored to handle linear problems, but the same cannot be said about hard non-linear problems, i.e., they do not follow the normal distribution assumption [36].

With the groundwork notation introduced, the next step concerns the question of how to identify ARIMA(p, d, q) order levels. To this end, the typical choice is to apply the well known Box and Jenkins method, which relies on the autocorrelation function (ACF), a measure of the mutual dependence between distinct lags, and the partial autocorrelation

function (PACF) to draw inferences about an adequate model [25]. This last function is a measure of the relationship between the current observation and its lags excluding the effects of intermediate observations, which is useful to specify AR terms.

2.2. Deep Neural Networks

A Deep Neural Network (DNN) is a type of machine learning approach, with the particularity of having multiple (hidden) layers between the input and output layers (at least two). The term "deep" is used to symbolize the more complex structure of layers and nodes, which significantly increases the number of weights and bias terms, deriving a more abstract and high-dimensional feature mapping from the provided input and output data [37,38]. In the same fashion as its classic shallow ANN counterpart, it draws inspiration from human brain functioning, particularly the information flow in biological neurons [39]. Moreover, its ability to cope with non-linearities renders it a suitable tool to handle a vast range of engineering problems from pattern recognition and data classification to time-series modeling, etc.

A common architecture for the DNN is the Deep Feedforward Neural Network (DFNN), forming a directed acyclic graph as represented in Figure 1 meaning that the flow of information only proceeds forward, simplifying the backpropagation training process. In each DFNN layer, every neuron output is evaluated by the activation function f_j, with a cost given by a biased weighted sum a_j, which works as a threshold. Hence, one can state that for any two consecutive layers $l-1$ and l, the neuron (node) response can be formulated as follows:

$$y_j = f_j(a_j) = f_j\left(\sum_{i=1}^{n} \omega_{ij}\, x_i + b_j\right), \tag{5}$$

where $j \in]0, m]$ is an index representing an arbitrary neuron response in layer l; and the $i \in]0, n]$ index denotes an arbitrary input node from the previous layer $l-1$ (notice the flexibility of this formulation where the previous layer can either be an input layer or an adjacent hidden layer). Accordingly, n is the total number of neurons/input nodes in layer $l-1$, and m is the number of neurons in the connecting layer l, with n and $m \in \mathbb{Z}^+$. y_j is the output for neuron j; x_i is the input signal from neuron i; b_j is the respective neuron bias; and lastly ω_{ij} is the weight of the synaptic connection between neurons i and j. This output response (5) is then one of the inputs for all the nodes of the proceeding layer in sequential form. Thus, one can compute the output response of all of the hidden layers in a DFNN, with an arbitrary number of L hidden layers, as a composite function g of the original input x (input layer), as denoted in Equation (6) [40]:

$$g(x) = g\left[a_{(L+1)}\left(h_{(L)}\left(a_{(L)}\left(\ldots\ \left(h_{(2)}\left(a_{(2)}\left(h_{(1)}\left(a_{(1)}(x)\right)\right)\right)\right)\right)\right)\right)\right], \tag{6}$$

where $h_{(l)}(x)$ denotes the output of an arbitrary hidden layer l (superscript), and $a_{(l)}(x)$ is an abbreviated form for the biased weighted sum for an arbitrary hidden layer l (pre-activation cost).

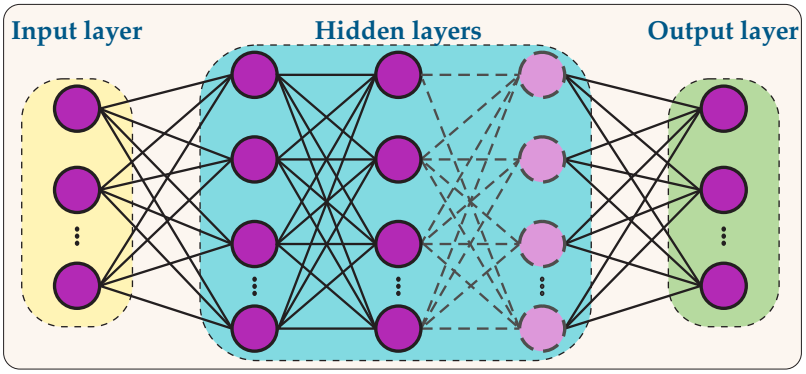

Figure 1. Deep Forward Neural Network (DFNN).

In the considered training process, the DFNN weights are updated by using an offline supervised strategy, where the weights are adjusted to minimize the error in the output layer with known target samples. The scaled conjugate gradient backpropagation algorithm was chosen to perform the training task given its balance between performance and speed, which is particularly important for deep networks.

3. Proposed Methodology

This section presents a comprehensive description of the proposed load forecasting methodology and discusses its implementation details. First, the proposed methodology is guided by the ensemble methods (learning) concept, where an improved predictive performance is expected when employing/combining multiple methods versus any of the individual methods. More specifically, this work is centered on a stacking ensemble type approach, which in a broad sense involves training a machine learning method to merge the predictions of some other methods [41]. To reinforce this point, the literature in the forecasting field documents report that when dealing with an unstable and varying data pattern, it is convenient to use dissimilar models to improve time-series forecasting accuracy [42]. Much of the reasoning behind the wide-adoption (as shown the Introduction section) of ensemble methods resides in the increased diversity among the base classifiers/forecasters. When using a stacking strategy, the meta-learning model (assembler) is used to integrate the output of base models [43].

Following this line of thought, a stacking ensemble approach is proposed in this work where a DFNN combines of the Box-Jenkins predictors. For better context and for learning purposes, additional input data from the load time-series and relevant related exogenous variables are also fed to the DFNN. An overview of the most relevant stages and main blocks that constitute the proposed methodology is given in Figure 2. A more thorough account of these stages and blocks is provided in the subsections below. Additionally, the pseudo code of the main implementation stages is given in Appendix A.

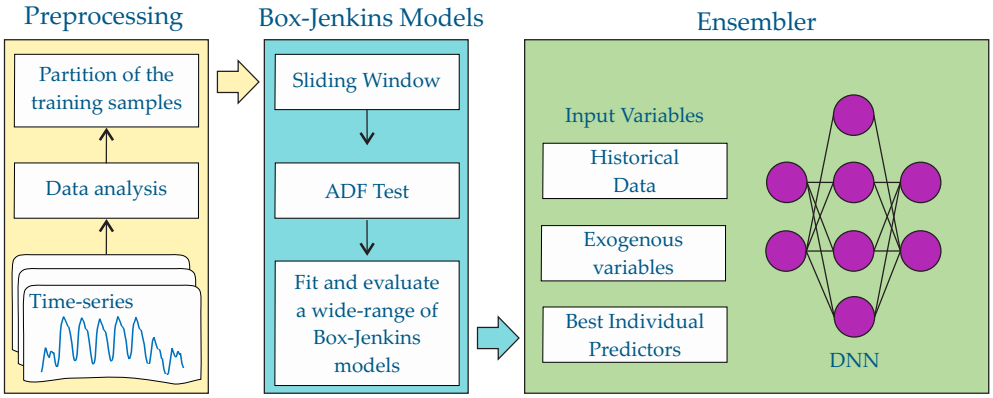

Figure 2. Proposed methodology: overview.

3.1. Prediction Models: ARIMA Forecasters

Given this ensemble approach to perform STLF, ARIMA models were chosen to serve as reference inputs for the combiner DFNN. This approach of coupling soft computing methods with ARIMA model has proven to increase accuracy, as reported in [36,42]. While ARMA models are unable to capture the non-linearity of the series, they can capture the linear traits of the time-series, which is worthy information to be fed as inputs to the assembler (non-linear) DFNN [44].

Before identifying and fitting the ARIMA models, a significant question concerns the suitable number of observations to estimate the models. In this regard, a common-sense rule is to have more observations than model parameters. This implies that, for example, in a SARIMA model as defined in Equation (4), a number of $p + d + q + P + D + Q + 1$ observations is required to estimate the model. However, when dealing with data

exhibiting substantial random variation, this minimum requirement is insufficient, and observations must exceed the parameters several fold [45]. Hence, a hard rule can often be misleading given the underlying variability of the data. As such, instead of following a narrow approach, in this work, a conscious decision was made to assemble a pool of ARIMA models.

These differed from each other not only in parameters but also in the training sample window (number of observations used to fit the model). This procedure results in very different ARIMA models, where some will better capture the most recent trends (when the training window is shorter), while others with a longer training window (more samples) will provide a fuller picture of the past time-series behavior. Hence, in this work, five different training windows (in months) were selected, i.e., the number of past observations (training samples) used to fit the model was given by a specific number of prior months defined in the variable $\delta = [1, 2, 3, 6, 12]$.

For each selected training window, stationarity was evaluated through the Unit Root hypothesis using the Augmented Dickey Fuller (ADF) test. Based on the test's p-value, the methodology decides the order of integration. In our case study, the first differences proved to be sufficient, but with a 12-month training window, in some instances, the unit-root null hypothesis was rejected in favor of the alternative hypothesis (ARMA model is suitable). Having sorted the integration order d, the ACF and PACF functions were built, proceeding far as 15 days prior, and a list of the most promising AR and MA terms was established. With this task completed, all the parameter combinations were tested, and the respective ARIMA models were assembled together with their mirroring SARIMA models, where the seasonal P and Q orders depict daily and weekly seasonalities. This process of automatic model selection, based on the information provided by Unit Root Tests and correlation analysis, is frequently referred as "Automatic ARIMA" [46].

Having identified the several models identified, a Maximum likelihood estimation (MLE) algorithm was used to estimate the model coefficients, i.e., the coefficients that maximize the log-likelihood function. Subsequently, the Bayesian Information Criterion (BIC) was used to select the three best models for each training window (for a total of 15 ARIMA forecasters), by determining the set of model parameters $\left[\varphi_p, \Phi_P, \mu, \theta_q, \Theta_q\right]$ that minimize the BIC value, as expressed in Equation (7):

$$BIC = k \ln\left(n_{length}\right) - 2 \ln\left(\frac{RSS}{n_{length}}\right), \tag{7}$$

where k denotes the number of model parameters, n_{length} is the number of observations and RSS is the residual sum of square, i.e., a measure of the variance in the error term of the model [47]. Notice that while all combinations were tested, this does not imply that if, for example, $p = 24$, all AR terms between 1 and p have non-null coefficients. On the contrary, since the model complexity is penalized, as observed in Equation (7), this reinforces the need to select models with strong correlations.

This offline process is performed iteratively in an automatic manner, and its different stages are illustrated in Figure 3. The selected training window is only valid for testing in the subsequent week, and the forecasting performance is stored upon test completion. Thereafter, the most recent data (testing data) are added to the training window (batch), while the oldest data are removed, similarly to a Last In First Out procedure. Therefore, it ensures a training window (batch) with a fixed size. This windowing process is carried out until all testing data are evaluated and the respective ARIMA Forecasters are stored. The process concludes after analyzing all the training windows and their correspondent prediction models, producing the key prediction inputs for the ensemble method.

This offline process was performed iteratively in an automatic manner, and its different stages are illustrated in Figure 3. The selected training window is only valid for testing in the subsequent week, and the forecasting performance is stored upon test completion. Thereafter, the most recent data (testing data) were added to the training window (batch),

while the oldest data were removed, as in a Last In First Out procedure. This ensures a training window (batch) with a fixed size. This windowing process was carried out until all the testing data were evaluated and the respective ARIMA Forecasters were stored. The process concluded after analyzing all the training windows and their correspondent prediction models, producing the key prediction inputs for the ensemble method.

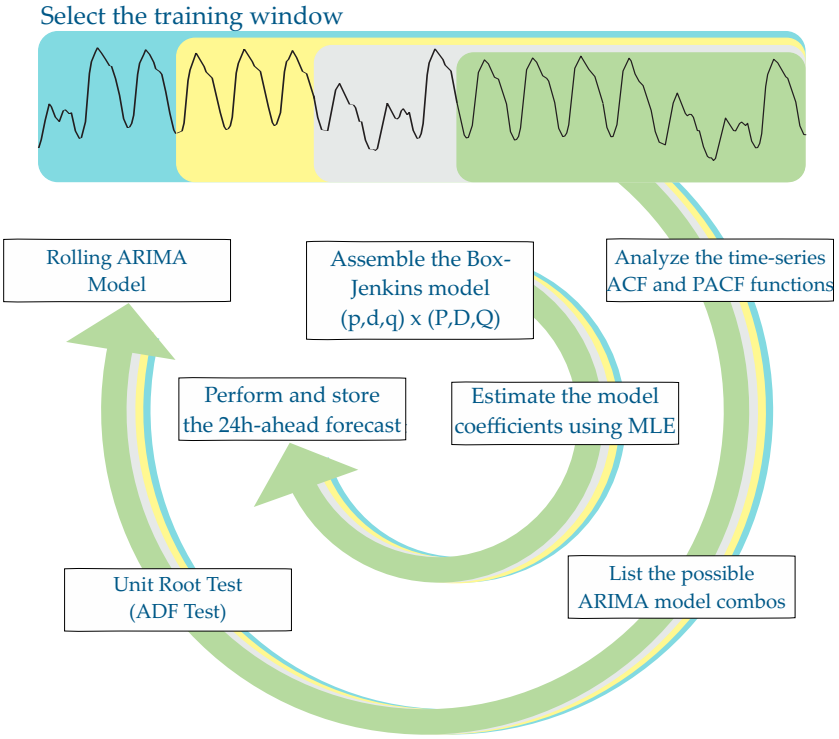

Figure 3. Predictions models: outline of the different stages from data selection to ARIMA Forecasters.

3.2. Ensembler DFNN

As previously discussed in the Introduction and Section 2.2, DNNs are a great solution for solving a wide range of problems given their ability to explore and model deep nonlinear relationships.

With respect to the DFNN architecture, its input layer was built with 26 nodes that account for the following: 15 ARIMA forecasters (three best forecasts for each of the five training windows), $\{\hat{y1}_t{}^{\delta_1}, \ldots, \hat{y1}_t{}^{\delta_5}, \hat{y2}_t{}^{\delta_1}, \ldots, \hat{y2}_t{}^{\delta_5}, \hat{y3}_t{}^{\delta_1}, \ldots, \hat{y3}_t{}^{\delta_5}\}$, which are stacked and fed to the ensemble method providing information about the expected linear trends; the six past lagged values of the actual load data $(x_{t-24}, x_{t-25}, x_{t-26}, x_{t-32}, x_{t-48}, x_{t-168})$; four date/time related exogenous variables, namely hour, weekday, binary holiday indicator and season of the year; information about the time-sample of the 24 h ahead target load $\left(exog_t{}^{hour}, exog_t{}^{weekday}, exog_t{}^{holiday}, exog_t{}^{season}\right)$; and finally, the lagged relative humidity information was also fed to the DFNN $\left(exog_{t-24}{}^{relHum}\right)$. The addition of exogenous variables to forecast the load in the ensemble method is of crucial importance since electricity demand does not depend exclusively on the autocorrelation of its own data (endogenous). In fact, a comprehensive assortment of exogenous factors has an impact on the load [44].

Concerning the hidden layer(s) configuration and after performing a few trial simulations, it was found that a three hidden layer DFNN (meaning a total of five layers, including the output and input layers), with 20, 10 and 5 neurons in each hidden layer, respectively, was found to be a good fit. These hidden sizes ensured a good level of complexity and abstraction without using too many neurons and, consequently, weights, which can often result in overfitting. In this regard, note that an optimal paradigm for the definition of the best model does not exist and likewise for the number of neurons in the hidden layer(s). As such, the conventional path is to rely on past experiences and trial-and-error procedures.

The hyperbolic tangent sigmoid transfer function, $\sigma(x) = \frac{2}{1+\exp(-2x)} - 1$, was chosen as the activation function for the hidden layers. The DFNN flow ends in the output layer, with a single neuron, producing the 24 h ahead forecasting, $\hat{y}_t$. In addition, a linear transfer function, $out(x) = \alpha x$, was used in this layer, which is a very common choice for approximation and regression tasks.

With the architecture defined, the next stage was preprocessing (8) the input and target data, ensuring that the data always fell within the range $[-1, 1]$ by using the equation below:

$$x_{norm} = -1 + 2\left(\frac{x - x_{\min}}{x_{\max} - x_{\min}}\right), \tag{8}$$

where x_{norm} is the preprocessed input sample; x is the original input sample; and $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of all input samples.

All the training samples are employed to form the learning batch; as such, a Batch Gradient Descent paradigm is considered in this work. This implies that all samples contribute to training and validation errors before updating DFNN weights and biases. The input and target data spanned over the period of 1 year prior to the testing of the trained ensemble methodology in the subsequent month. These datasets were divided randomly into training set and validation set by using a typical 70% to 30% ratio in order to improve the generalization capabilities.

This validation set provides a separate (unseen) set of representative data-samples that are used to perform an unbiased evaluation of the model fit (validation error), different from the training error, during the training phase (weights and biases update). When the validation error starts to decouple from the training error, i.e., does not decrease for several iterations, further continuation of the training process may invariably result in overfitting. This decoupling, therefore, constituted the main stopping criterion for DNN training even though a maximum number of learning iterations was also defined. Next, the essential training process of the ensemble DFNN was carried out. The scaled conjugate gradient (SCG) algorithm [48] was chosen to perform this supervised learning task given that no critical user-dependent parameters are needed, making it a fully automated algorithm. In addition, its step size scaling mechanism improves the speed of convergence by avoiding a time-consuming line search per iteration, which is a common calculation of other second order algorithms [10,49].

Furthermore, in order to improve generalization during the training process, a modified version was given by Equation (9), modPerfF, from the standard performance function using the mean squared error (MSE), and it was chosen by adding the mean of the sum of squares of the network weights and biases (MSW). This forces the network to have smaller weights:

$$\text{modPerfF} = (1-\eta)\text{MSE} + \eta\,\text{MSW} = (1-\eta)\frac{\sum_{i=1}^{N_S}(y_i - \hat{y}_i)^2}{N_S} + \eta\frac{\sum_{j=1}^{N_W} w_j^2}{N_W}, \tag{9}$$

where η is the generalization ratio (a good compromise was achieved with its value set to 0.3), N_S is the number of training samples, N_w is the number of weights in the network, $\hat{y}_i$ is the DFNN output and y_i is the desired (target) output response.

4. Case Study

In order to evaluate the proposed methodology's forecasting accuracy, the well known real case study from the New England region system is considered. This region in Northeastern US spreads across six states, namely Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island and Vermont. The hourly system load samples are provided by the New England Independent System Operator (ISO-NE) and are available in [50]. In this work, data from 2014 to 2015 were selected for training different forecasters. The data from 2016 were applied to test the models, and Figure 4 illustrates its hourly and monthly behavior.

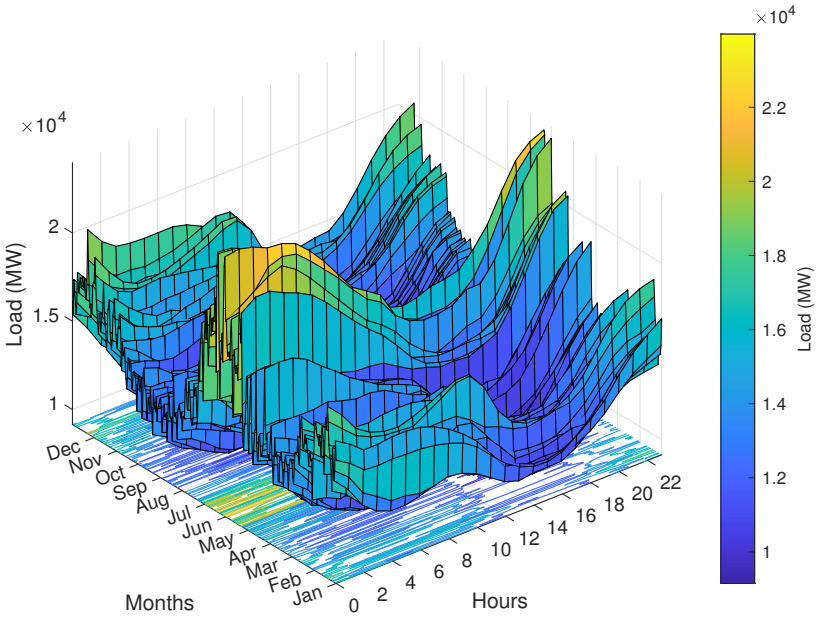

Figure 4. New England region (US) 2016 Load (MW).

A brief data analysis revealed a mean load value of 13.504 GW and significant amount of standard deviation, 2.426 GW (∼18% of the mean value), with a moderately positively skewed and Leptokurtic distribution (skewness = 0.779 and kurtosis = 3.870). This implies a distribution with longer tail and more observations located to the right (higher) of the mean value. In addition, as illustrated in Figure 4, the summer months registered not only higher load values but also greater variabilities largely due to the cooling related loads. By contrast, the spring months had a smaller mean and standard deviation. Accordingly, the month of June had the highest mean and standard values, 16.700 GW and 2.9269 GW, respectively, as well as the maximum recorded load value of 23.973 GW, which took place on 3 June. The month of August recorded the lowest mean value of 11.480 GW and also the minimum observed load value of 9.149 GW (7 August), while the minimum standard deviation occurred in the spring month of April with a value of 1.3383 GW.

5. Results

In order to demonstrate the proposed methodology's behavior in a representative manner for the entire year, four different scenarios (out-of-sample scenarios) were considered, comprising four months corresponding to four different meteorological seasons, including holidays (typically similar to weekends in terms of load profile).The following months were selected to serve as testing data from the New England region: (i) winter month (February 2016); (ii) spring month (April 2016); (iii) summer month (August 2016); and (iv) fall month (October 2016). In order to ensure the consistency, the results of the ensemble DFNN were the mean results of chosen error metrics over 50 simulations, thereby avoiding skewed inferences based on specific random data divisions and random DFNN weight initialization.

In order to gauge the forecasting accuracy of the 24 ahead predictions, the common scale-invariant error metrics MAPE(%) (mean absolute percentage error) and RMSE (MW) (root mean square error) were used, respectively, Equations (10) and (11), which are defined as follows:

$$MAPE\ (\%) = \frac{100}{N_{test}} \sum_{i=1}^{N_{test}} \left| \frac{load_i - \hat{load_i}}{load_i} \right|, \tag{10}$$

$$RMSE = \sqrt[2]{\frac{100}{N_{test}} \sum_{i=1}^{N_{test}} \left(load_i - \hat{load_i} \right)^2}, \tag{11}$$

where N_{test} is the number of testing samples (length of the target and forecast vector); and $load_i$ and $\hat{load}_i$ are the actual and predicted load value at each time-step i, respectively.

The obtained load forecasting error metrics are displayed in Tables 1 and 2. Notably, in terms of MAPE (%) (Table 1), the ensemble method was able to outperform the ARIMA forecasters (input Predictions Models) for all the testing periods. This translates the sizeable monthly accuracy gains of MAPE (%) in comparison with the mean accuracy of all the ARIMA forecasters of ∼23%, ∼14%, ∼12% and ∼26% for the months of February, April, August and October, respectively, and accuracy gains of ∼16%, ∼2%, ∼8% and ∼14% when comparing the ensemble methodology versus the best ARIMA Forecaster in the same testing months. Table 2 indicates a similar pattern for the RMSE, where one can see monthly error improvements of ∼22%, ∼11%, ∼13% and ∼28% in comparison with the mean accuracy of all the ARIMA forecasters and RMSE accuracy gains of 12%, ∼1%, ∼10% and ∼18%, respectively, for the four different testing months. In addition, we can oberve that the methodology's accuracy is not specifically dependent on the time-series scale and variance characteristics, i.e., the largest MAPE and RMSE values took place in the month of April, which as observed previously is not the month with the highest mean or standard deviation.

Table 1. One day-ahead NE Load Forecasting results: MAPE (%).

Prediction Model/Test Month	16 February	16 April	16 August	16 October
ARIMA Forecaster 1	4.36	6.34	5.26	4.37
ARIMA Forecaster 2	4.79	7.28	5.25	5.37
ARIMA Forecaster 3	4.87	7.40	5.51	5.32
ARIMA Forecaster 4	4.93	7.49	5.73	5.20
ARIMA Forecaster 5	4.93	7.55	5.48	5.14
Ensemble DFNN	3.67	6.19	4.82	3.75

Table 2. One day-ahead NE Load Forecasting results: RMSE (MW).

Prediction Model/Test Month	16 February	16 April	16 August	16 October
ARIMA Forecaster 1	737.3	1266.6	983.7	774.5
ARIMA Forecaster 2	840.7	1443.5	978.6	937.7
ARIMA Forecaster 3	859.8	1446.4	1034.6	916.7
ARIMA Forecaster 4	857.6	1454.9	1067.2	891.7
ARIMA Forecaster 5	860.5	1454.6	1034.1	915.9
Ensemble DFNN	649.4	1254.1	883.3	634.7

Figure 5 illustrates the forecasting performance of the proposed ensemble methodology in comparison with the selected ARIMA forecasters (inputs) for two indiscriminately selected fortnights, i.e., the period from 1 to 14 February (winter) and the period from 14 to 27 August (summer). This figure also underlines the ensemble methodology's good performance in resembling the real load profile. More importantly, these examples reveal the ability of DFNN to generalize beyond the provided ARIMA forecasters and extrapolate further, thereby enabling the ensemble forecast tomatch the real load more closely. This behavior is illustrated in the zoomed secondary axis in both plots, where one can clearly observe the ensemble forecast outside of the region mapped by the 15 selected ARIMA forecasters (input prediction models). In addition, it is also clear that, as expected, the ARIMA forecasters are a suitable option to capture the linear components of the time-series but struggle to reproduce the faster changing variations of the morning and evening peak loads. This confirms the need for a stacking ensemble methodology where a deep network

with nonlinear mapping abilities, fed with additional data, compensates this shortcoming and provides relevant error accuracy gains.

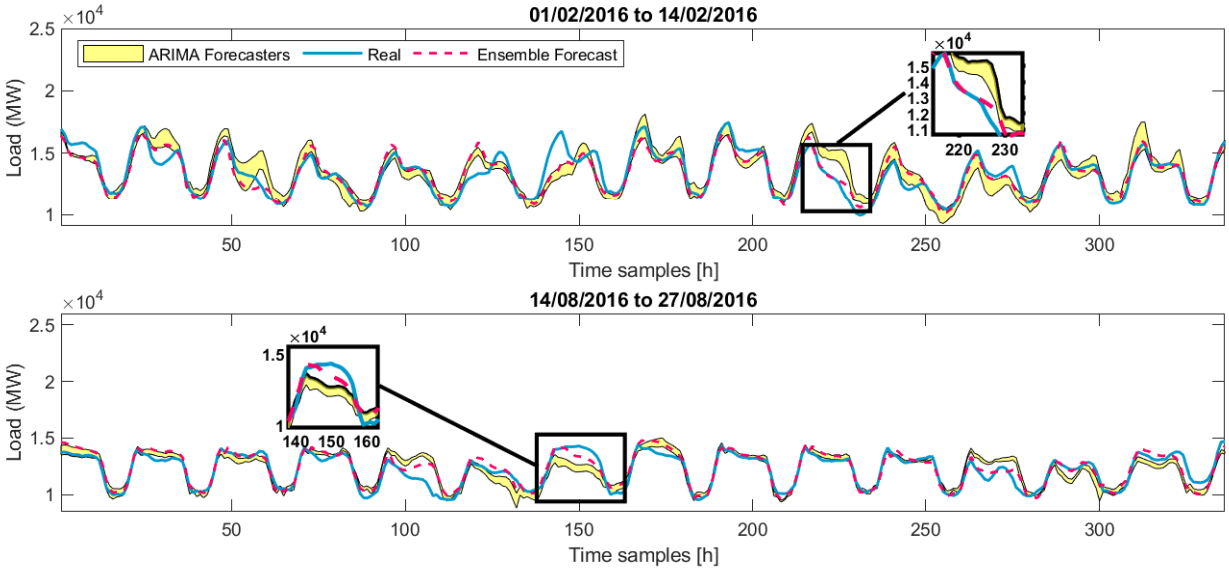

Figure 5. Load forecasts vs. real load (New England region), where the ARIMA forecasters area in yellow represents the region mapped by the forecasts of 15 different ARIMA models, the stacking ensemble forecast is illustrated in a dashed magenta line and the real (target) load is illustrated in a solid blue line.

To complement these results, a comparative analysis was made against a common shallow NN architecture with a single hidden layer with 35 nodes, also trained using the SCG algorithm, for benchmarking purposes and also for evaluating the effect of the added exogenous input variables in the input dataset (the same exogenous variables were fed to the ensemble methodology). Ergo, two different input datasets are considered. Likewise, a support vector machine regression with a Gaussian kernel function and input standardization is used to map the input domain, building the decision boundaries to achieve the hyperplane that allow us to predict continuous load outputs. The average error metrics over the course of 50 runs are shown in Tables 3 and 4, highlighting not only the superior performance of the the ensemble method for all the testing periods but also the clear benefit of including related exogenous information in the input datasets when we compare the errors between the two NNs. A brief statistical analysis reveals gains in terms of RMSE between ∼1.9% to ∼17.5% versus the NN with exogenous inputs; between ∼10.5% to ∼32.9% versus the NN without exogenous inputs; and between ∼3.4% to ∼33.7% versus the Regression-SVM. An identical account is also depicted by the MAPE (%) values, revealing considerable monthly accuracy improvements (between ∼11.5% to ∼33.8%) for all the months versus the other three forecasting approaches, with the exception being the mean monthly MAPE (%) of August where the gain was more moderate (∼1.5%).

Table 3. Comparison of one day-ahead NE Load Forecasting results: MAPE (%).

Method/Test Month	16 February	16 April	16 August	16 October
R-SVM	4.71	8.15	5.39	5.67
NN wo/ Exog	4.87	6.96	5.85	5.16
NN w/ Exog	4.22	6.99	4.89	4.66
Ensemble DFNN	3.67	6.19	4.82	3.75

Table 4. Comparison of one day-ahead NE Load Forecasting results: RMSE (MW).

Method/Test Month	16 February	16 April	16 August	16 October
R-SVM	769.5	1492.7	914.6	958.2
NN wo/ Exog	860.1	1400.5	1066.8	945.2
NN w/ Exog	717.8	1278.6	923.5	769.2
Ensemble DFNN	649.4	1254.1	883.3	634.7

Another important feature of the proposed methodology and its comparative NN peers was the relatively low levels of standard variation across the 50 runs, as one can notice in Figure 6, by the narrow range of the interquartile distribution in the box-plot and the small number of outliers, both for the MAPE and RMSE error metrics. More specifically, the ensemble methodology even with a large input dataset is in line with the NN with exogenous inputs variance, and its median line "sits" lower in the box-plots for all four testing months. In the months of February, April, August and October, the recorded values of standard deviation were, respectively, 0.132%, 0.252%, 0.217% and 0.113% in terms of MAPE (%) and 26.5 MW, 60.8 MW, 36.3 MW and 21.4 MW in terms of RMSE. These numbers validate the robustness of the presented ensemble methodology with respect to consistently performing STLF in an improved manner.

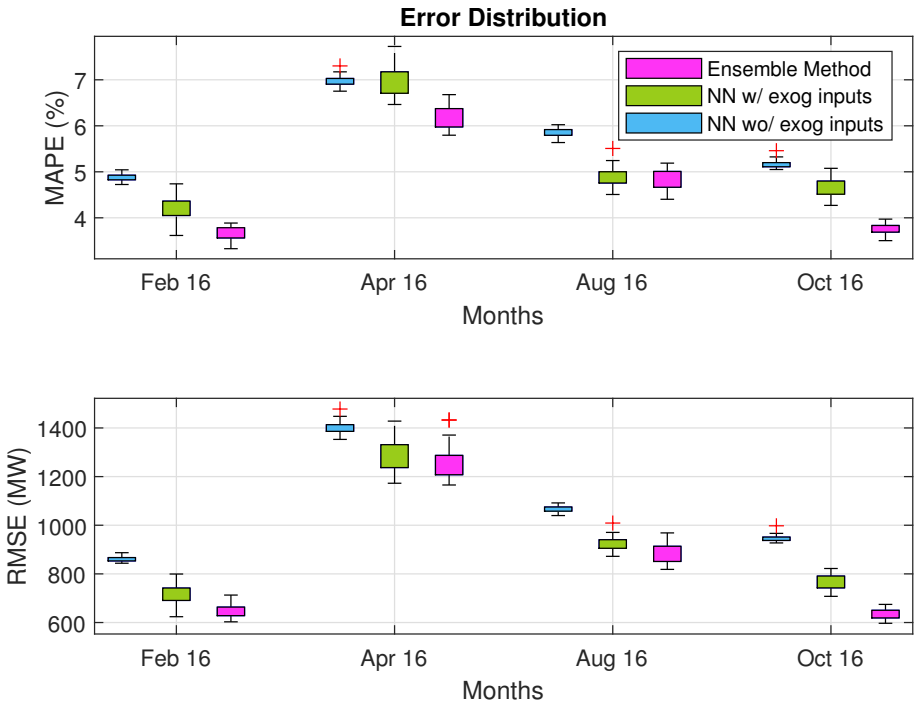

Figure 6. Box-plots of the MAPE (%) and RMSE (MW) error distributions (NN without exogenous inputs-blue, NN with exogenous inputs-green and the proposed ensemble methodology—magenta over the course of 50 runs).

Last but not least, Figures 7 and 8 are presented to illustrate two different weeks (first week of April and third week of October, respectively) of forecasting loads versus the real load, as well as the individual hourly deviation (signed error) of each comparative method and the proposed approach below (in the form of bar plots). These bars allow a better understanding of the difference between predicted and real hourly loads, and the smaller (better) magnitudes seen in the Ensemble DNN reinforce the inferences made by not only analyzing the error tables but also the error distribution in Figure 6, with a slightly positive bias in terms of error signal for all the considered methods.

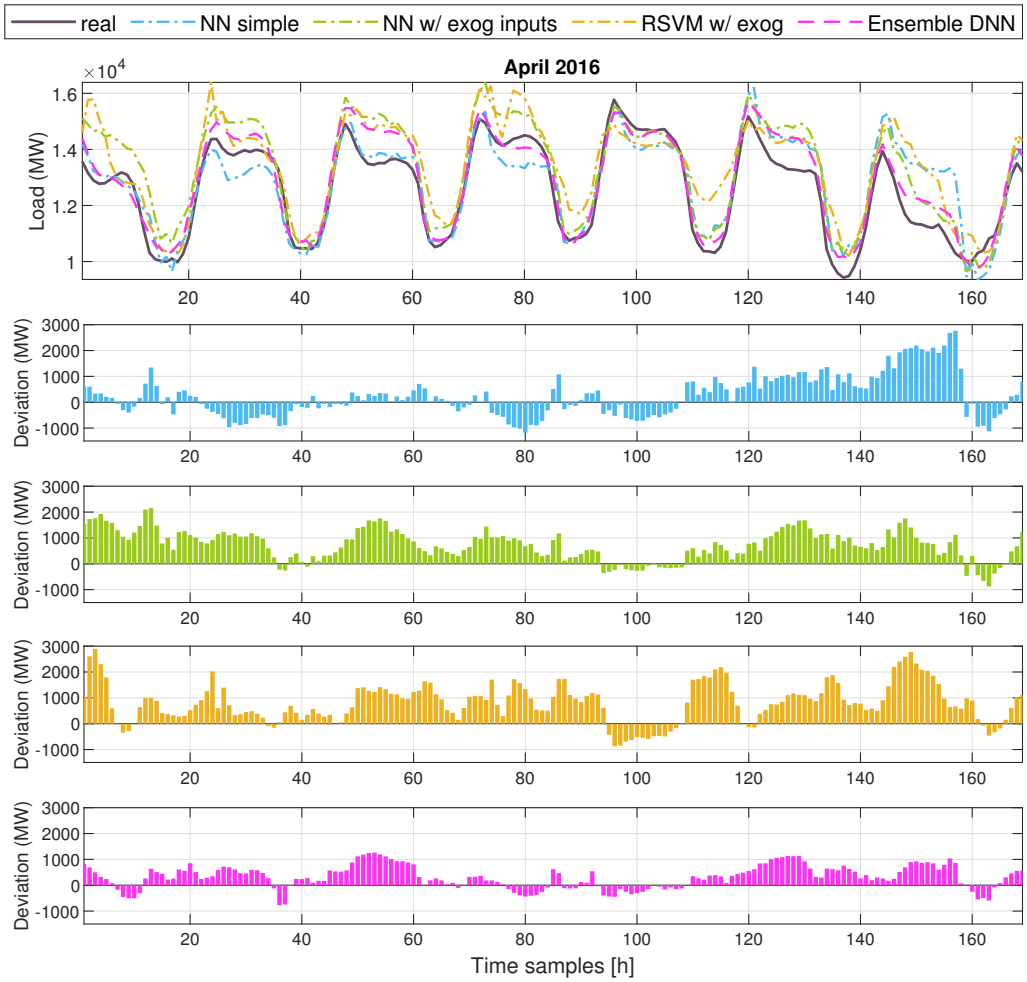

Figure 7. Predicted loads vs. real load and the respective mean deviation (signed error) of each method: 1st week of April 2016.

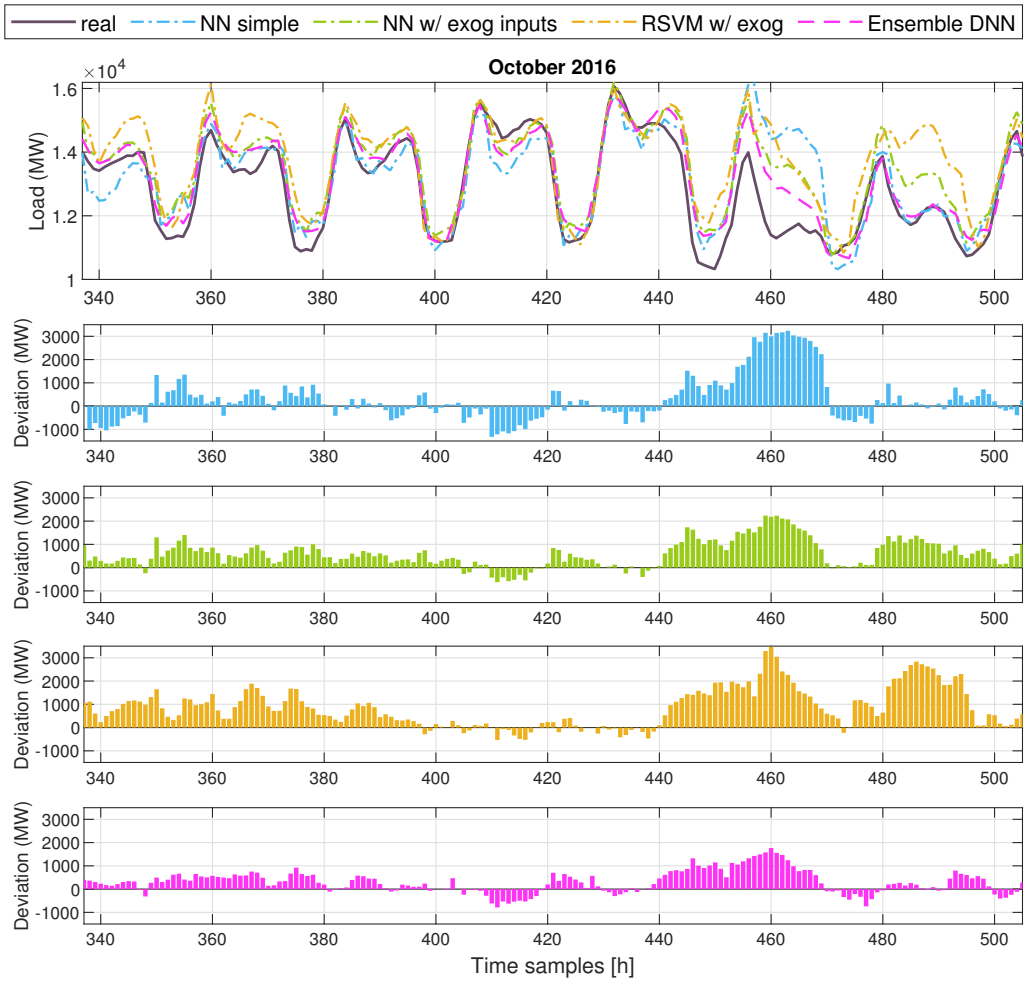

Figure 8. Predicted loads vs. real load and the respective mean deviation (signed error) of each method: 3rd week of April 2016.

6. Conclusions

The task of short-term load forecasting plays a crucial part for a better functioning electric power system, enabling better scheduling, lower generation costs, better planning and better count of load flows. This task has received new attention, given the increasing adoption of self-consumption and demand response mechanisms. A review of the latest developments in the STLF problem revealed a broad field of study, with researchers trying to explore the significant breakthroughs in ML and DL in order to reduce forecasting error. An increasingly consolidated trend is the use of ensemble or fusion methods (particularly black-box type models) in order to explore the different generalization capabilities of dissimilar methods and to increase the diversity of solutions (among the base classifiers/inputs).

By considering all these aspects, this study proposes a new stacking ensemble methodology that uses a pool of different ARIMA forecasters that not only differ in the orders of autoregressive and moving average terms but also in terms of the batch (training) sizes in which they are adjusted. These prediction models are fitted according to their stationarity hypotheses in an automatic manner, and they are mainly tailored for modelling the linear components of the load time-series; their information is fed as forecaster inputs to a DFNN given its well-known ability to map hard non-linear relationshipsby using a gradient based training algorithm. This deep black-box architecture was designed to include another set of endogenous (lagged load data) and exogenous information in order to allow the network to generalize beyond ARIMA forecasters. To improve the ANN learning process,

an early stop mechanism is not only employed but a modified performance function is also employed, which improves generalization.

This implementation philosophy will ensure a reduction in the high variance of single NN based models and in generalization errors. The other focal point for researchers is improvements in data selection, pre-processing and feature extraction techniques, which are essential to "clean" the time-series from noisy data, including outliers or seasonal events, or to precisely uncover some "hidden" features (e.g., high frequency behaviors). As such, CA and different Box-Jenkins models were used to facilitate the modeling process, i.e., the extraction of hard (non-linear) relationships from the set of explanatory input variables.

The effectiveness of the proposed method was evaluated in the New England case study by assessing the error in four different months (from different seasons of the year) of 2016. The obtained error metrics revealed the ensemble methodology's ability to improve forecasting accuracy in comparison with the same error metrics for the input prediction models, achieving error improvements on the order of 10% to 25% (mean terms) in comparison with the base learners' prediction accuracy. The RMSE underscores well the effectiveness of the proposed methodology, since all the monthly RMSEs were lower than the lowest monthly standard deviation in the time-series (which occurred in April 2016).

The comparative analysis validated the ensemble methodology by confirming the improvements when using a DNN versus shallow NN, and it also allowed backing up the decision to consider relevant exogenous variables, i.e., there are individual contributions made to the ensemble approach by the different individual ARIMA forecasters that coupled with a correlated input dataset for the DFNN, which has produced meaningful error improvements. Moreover, the relatively small levels of variance between simulations of the DFNN show the consistency of the architecture independently of some random parameters, such as weight initialization and data division. These results attest the suitability of the designed approach to create the intended input diversity in the base learners with the different Box–Jenkins models. These models were assembled and evaluated from a pool of options based on data and information-criteria, thus successfully tackling the common model identification (hyperparameter decision) problem when using these statistical models, thus corroborating the design approach of the proposed stacking ensemble methodology.

In terms of future works, the authors will be looking to extend the diversity of the input forecasters (with different types of base learners), address the error bias and study different combinations of dissimilar methods, as well as increasing the number of training windows and its range, in order to gauge longer term trends with the forecasting models and its effect in the overall combined load forecast.

Author Contributions: Conceptualization, P.M.R.B. and J.A.N.P.; methodology, P.M.R.B. and J.A.N.P.; software, P.M.R.B.; validation, J.A.N.P., M.R.A.C. and S.J.P.S.M.; investigation, S.J.P.S.M., M.R.A.C., J.A.N.P. and P.M.R.B.; formal analysis, M.R.A.C. and S.J.P.S.M.; writing—original draft preparation, P.M.R.B. and J.A.N.P.; writing—review and editing, S.J.P.S.M. and M.R.A.C.; visualization, P.M.R.B. and M.R.A.C.; supervision, J.A.N.P., S.J.P.S.M. and M.R.A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by FCT/MCTES through national funds and, where applicable, co-funded EU funds under the project UIDB/EEA/50008/2020.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://www.iso-ne.com/isoexpress/web/reports/load-and-demand/accessionnumber (accessed on 4 November 2021).

Acknowledgments: P.M.R. Bento gives his special thanks to the Fundação para a Ciência e a Tecnologia (FCT), Portugal, for the Ph.D. Grant (SFRH/BD/140371/2018).

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

The following abbreviations, acronyms and variables are used in this manuscript:

a_j	weighted sum in neuron j;
ACF	Autocorrelation function;
ADF	Augmented Dickey–Fuller test;
ANFIS	Adaptive neuro-fuzzy inference system;
ARIMA	Autoregressive integrated moving average;
ARMA	Autoregressive moving average;
b_j	Bias connection in neuron j;
B	Backshift operator;
BIC	Bayesian information criterion;
CNN	Convolutional neural network;
d	Degree of nonseasonal integration;
D	Degree of seasonal integration;
DBN	Deep belief network;
DFNN	Deep feedforward neural network;
DL	Deep learning;
DNN	Deep neural network;
$exog_t^{variable}$	Exogenous (DNN) input variable at time-step t;
ELM	Extreme learning machine;
f_j	Activation cost in neuron j;
$g(x)$	Composite function illustrating the DNN cascaded nature;
GARCH	Generalized autoregressive conditional heteroskedasticity;
GRU	Gated recurrent unit (neural network);
$h_x^{(l)}$	Output of an arbitrary hidden layer l receiving an input x;
ISO-NE	New England indepedent system operator (regional transmission);
k	Number of (ARIMA) model parameters;
l	Arbitrary DNN hidden layer;
$load_i$	Actual (real) load at time-step i;
LSTM	Long short-term memory (neural network);
$\hat{load}_i$	(Final) forecasted load at time-step i;
L	Number of DNN hidden layers;
m	Number of neurons in layer l;
MAPE	Mean absolute percentage error;
ML	Machine Learning;
MLP	Multilayer perceptron (neural network);
ModPerfF	Modified DNN performance error;
MSE	Mean squared error;
MSW	Mean squared weights;
n	Number of neurons/inputs in layer $l-1$;
n_{length}	Number of time-series samples/observations;
N_S	Number of training samples;
N_{test}	Number of testing samples;
N_W	Number of DNN weights (total);
N-BEATS	Neural basis expansion analysis for interpretable time series forecasting;
$out(x)$	DNN output layer (linear) transfer function;
p	Nonseasonal autoregressive polynomial degree;
P	Seasonal autoregressive polynomial degree;
PACF	Partial autocorrelation function;
q	Nonseasonal moving average polynomial degree;
Q	Seasonal moving average polynomial degree;
RMSE	Root mean squared error;
RNN	Recurrent neural network;
RSS	Residual sum square error;
SARIMA	Seasonal autoregressive integrated moving average;
SCG	Scaled conjugate gradient algorithm;
STLF	Short-term load forecast;

SVM	Support-vector machine;
VMD	Variational mode decomposition;
x_i	Input signal from neuron i (to neuron j);
x_{max}	Maximum input time-series value;
x_{min}	Minimum input time-series value;
x_{norm}	Pre-processed normalized input value;
X	Generic time-series;
XGB	Extreme gradient boosting;
y_j	Output response from an arbitrary neuron j;
$\hat{y}_t$	Output forecast of the different forecasters j;
α	Slope of the linear output transfer function $out(x)$;
δ	Set of training samples (in months) to model the forecasters;
ϵ_t	Lagged error term at time-step t;
η	Generalization ratio
θ_i	Nonseasonal moving average coefficient at lag i;
Θ_i	Seasonal moving average coefficient at lag i;
μ	Constant (ARIMA model) term;
$\sigma(x)$	Sigmoid transfer function;
ϕ_i	Nonseasonal autoregressive coefficient at lag i;
Φ_i	Seasonal autoregressive coefficient at lag i;
χ_t	Generic time-series sample at time-step t;
ω_{ij}	Connection weight between neurons/input $i$and j.

Appendix A

The pseudo-code of the overall proposed methodology detailed in Section 3 is provided below:

(1) INIT;

(2) GET and format the electric ISO-NE load data and relevant calendar and exogenous variables;

(3) COMPUTE the load time-series analysis using ACF and PACF;

(4) LIST a series of suitable ARIMA and SARIMA models based on the correlation analysis (with different thresholds) and known seasonalities;

(5) SET load datasets with different windowing (number of past observations), as expressed in variable δ;

(6) CALCULATE the ADF test to check for stationarity and decide upon the required degree of time-series integration, d;

(7) FIT the different pool of Box-Jenkins models (base learners/forecasters) to the different training windows of (endogenous) load samples;

(8) COMPUTE BIC, i.e., Equation (7), to select the 3 best Box-Jenkins models for each training window;

(9) DETERMINE and store the 24 h-ahead STLF with the Box-Jenkins models selected in the previous step (totaling 15 base learners) in a rolling (ARIMA or SARIMA) scheme;

(10) DEFINE the input dataset format, concatenating the ARIMA forecasts with the ruled relevant exogenous and calendar variables, and its correspondent target value. Forming a pair of training sample to desired output (24 h ahead);

(11) NORMALIZE the input data using Equation (8), the training, validation and testing datasets (sets of input and target samples is prepared);

(12) DEFINE the different set of DFNN hyperparameters for a regression task:

(i) Architecture: 3 hidden layers with sizes (number of neurons) [20,10,5], respectively;
(ii) Optimizer (Learning Algorithm): SCG;
(iii) The training and validation performance is evaluated using modPerfF (Equation (9));
(iv) Train to Validation Ratio: 70% to 30%;
(v) Sigmoid transfer function in the hidden layers;
(vi) Linear transfer function in the (last) output layer;

(13) SET number of runs = 50;

(14) FOR each run out of number of runs;

(i) TRAIN the DFNN in an offline process with updated weights after each test week is predicted;
(a) WHILE (iterations $<$ 2500);
(I) IF (validation error stops decreasing and counter $<$ 50);
(A) Counter += 1;
(II) ELSE IF (counter $>=$ 50);
(A) BREAK loop;
(III) ELSE;
(A) Counter = 0;
(ii) COMPUTE the STLF using the trained DFNN in the testing dataset;
(iii) STORE the predicted loads and the respective forecasting;
(iv) UPDATE the training and validation input dataset to include the most recently predicted data and return to (i);
(v) IF (no more test weeks to predict);
(a) BREAK loop;

(15) END

References

1. Mishra, M.; Nayak, J.; Naik, B.; Abraham, A. Deep learning in electrical utility industry: A comprehensive review of a decade of research. *Eng. Appl. Artif. Intell.* **2020**, *96*, 104000. [CrossRef]
2. Xu, F.Y.; Cun, X.; Yan, M.; Yuan, H.; Wang, Y.; Lai, L.L. Power Market Load Forecasting on Neural Network with Beneficial Correlated Regularization. *IEEE Trans. Ind. Inform.* **2018**, *3203*, 1–10. [CrossRef]
3. Xie, Y.; Ueda, Y.; Sugiyama, M. A Two-Stage Short-Term Load Forecasting Method Using Long Short-Term Memory and Multilayer Perceptron. *Energies* **2021**, *14*, 5873. [CrossRef]
4. El-Hendawi, M.; Wang, Z. An ensemble method of full wavelet packet transform and neural network for short term electrical load forecasting. *Electr. Power Syst. Res.* **2020**, *182*, 106265. [CrossRef]
5. Ghelardoni, L.; Member, S.; Ghio, A.; Anguita, D.; Member, S. Energy Load Forecasting Using Empirical Mode Decomposition and Support Vector Regression. *IEEE Trans. Smart Grid* **2013**, *4*, 549–556. [CrossRef]
6. Yuansheng, H.; Shenhai, H.; Jiayin, S. A Novel Hybrid Method for Short-Term Power Load Forecasting. *J. Electr. Comput. Eng.* **2016**, *2016*, 2165324. [CrossRef]
7. Feinberg, E.a.; Genethliou, D. Applied Mathematics for Restructured Electric Power Systems. *IEEE Trans. Autom. Control* **2005**, *50*, 269–285. [CrossRef]
8. Rueda, F.D.; Suárez, J.D.; Torres, A.d.R. Short-Term Load Forecasting Using Encoder-Decoder WaveNet: Application to the French Grid. *Energies* **2021**, *14*, 2524. [CrossRef]
9. Acakpovi, A.; Ternor, A.T.; Asabere, N.Y.; Adjei, P.; Iddrisu, A.S. Time Series Prediction of Electricity Demand Using Adaptive Neuro-Fuzzy Inference Systems. *Math. Probl. Eng.* **2020**, *2020*, 4181045. [CrossRef]
10. Bento, P.; Pombo, J.; Calado, M.; Mariano, S. Optimization of neural network with wavelet transform and improved data selection using bat algorithm for short-term load forecasting. *Neurocomputing* **2019**, *358*, 53–71. [CrossRef]
11. Moon, J.; Kim, Y.; Son, M.; Hwang, E. Hybrid Short-Term Load Forecasting Scheme Using Random Forest and Multilayer Perceptron. *Energies* **2018**, *11*, 3283. [CrossRef]

12. Semero, Y.K.; Zhang, J.; Zheng, D. EMD–PSO–ANFIS-based hybrid approach for short-term load forecasting in microgrids. *IET Gener. Transm. Distrib.* **2020**, *14*, 470–475. [CrossRef]
13. Ceperic, E.; Ceperic, V.; Baric, A. A strategy for short-term load forecasting by support vector regression machines. *IEEE Trans. Power Syst.* **2013**, *28*, 4356–4364. [CrossRef]
14. Yuan, T.L.; Jiang, D.S.; Huang, S.Y.; Hsu, Y.Y.; Yeh, H.C.; Huang, M.N.L.; Lu, C.N. Recurrent Neural Network Based Short-Term Load Forecast with Spline Bases and Real-Time Adaptation. *Appl. Sci.* **2021**, *11*, 5930. [CrossRef]
15. Bento, P.; Pombo, J.; Mariano, S.; Calado, M.d.R. Short-Term Load Forecasting using optimized LSTM Networks via Improved Bat Algorithm. In Proceedings of the 2018 International Conference on Intelligent Systems (IS), Funchal, Portugal, 25–27 September 2018; pp. 351–357. [CrossRef]
16. Ciechulski, T.; Osowski, S. High Precision LSTM Model for Short-Time Load Forecasting in Power Systems. *Energies* **2021**, *14*, 2983. [CrossRef]
17. Wu, W.; Liao, W.; Miao, J.; Du, G. Using Gated Recurrent Unit Network to Forecast Short-Term Load Considering Impact of Electricity Price. *Energy Procedia* **2019**, *158*, 3369–3374. [CrossRef]
18. Yeom, C.U.; Kwak, K.C. Short-Term Electricity-Load Forecasting Using a TSK-Based Extreme Learning Machine with Knowledge Representation. *Energies* **2017**, *10*, 1613. [CrossRef]
19. Yu, Y.; Ji, T.Y.; Li, M.S.; Wu, Q.H. Short-term Load Forecasting Using Deep Belief Network with Empirical Mode Decomposition and Local Predictor. In Proceedings of the IEEE Power and Energy Society General Meeting, Portland, OR, USA, 5–10 August 2018. [CrossRef]
20. Mansoor, M.; Grimaccia, F.; Leva, S.; Mussetta, M. Comparison of echo state network and feed-forward neural networks in electrical load forecasting for demand response programs. *Math. Comput. Simul.* **2021**, *184*, 282–293. [CrossRef]
21. Acharya, S.K.; Wi, Y.M.; Lee, J. Short-Term Load Forecasting for a Single Household Based on Convolution Neural Networks Using Data Augmentation. *Energies* **2019**, *12*, 3560. [CrossRef]
22. Oreshkin, B.N.; Dudek, G.; Pełka, P.; Turkina, E. N-BEATS neural network for mid-term electricity load forecasting. *Appl. Energy* **2021**, *293*, 116918. [CrossRef]
23. Sowinski, J. The Impact of the Selection of Exogenous Variables in the ANFIS Model on the Results of the Daily Load Forecast in the Power Company. *Energies* **2021**, *14*, 345. [CrossRef]
24. Jin, Y.; Guo, H.; Wang, J.; Song, A. A Hybrid System Based on LSTM for Short-Term Power Load Forecasting. *Energies* **2020**, *13*, 6241. [CrossRef]
25. Khashei, M.; Hajirahimi, Z. A comparative study of series arima/mlp hybrid models for stock price forecasting. *Commun. Stat.-Simul. Comput.* **2018**, *48*, 2625–2640. [CrossRef]
26. Nazar, M.S.; Fard, A.E.; Heidari, A.; Shafie-khah, M.; Catalão, J.P. Hybrid model using three-stage algorithm for simultaneous load and price forecasting. *Electr. Power Syst. Res.* **2018**, *165*, 214–228. [CrossRef]
27. Nie, H.; Liu, G.; Liu, X.; Wang, Y. Hybrid of ARIMA and SVMs for short-term load forecasting. *Energy Procedia* **2011**, *16*, 1455–1460. [CrossRef]
28. Büyükşahin, Ü.Ç.; Ertekin, Ş. Improving forecasting accuracy of time series data using a new ARIMA-ANN hybrid method and empirical mode decomposition. *Neurocomputing* **2019**, *361*, 151–163. [CrossRef]
29. Guo, W.; Che, L.; Shahidehpour, M.; Wan, X. Machine-Learning based methods in short-term load forecasting. *Electr. J.* **2021**, *34*, 106884. [CrossRef]
30. Shen, Y.; Ma, Y.; Deng, S.; Huang, C.J.; Kuo, P.H. An Ensemble Model based on Deep Learning and Data Preprocessing for Short-Term Electrical Load Forecasting. *Sustainability* **2021**, *13*, 1694. [CrossRef]
31. Massaoudi, M.; Refaat, S.S.; Chihi, I.; Trabelsi, M.; Oueslati, F.S.; Abu-Rub, H. A novel stacked generalization ensemble-based hybrid LGBM-XGB-MLP model for Short-Term Load Forecasting. *Energy* **2021**, *214*, 118874. [CrossRef]
32. Salmi, T.; Kiljander, J.; Pakkala, D. Stacked Boosters Network Architecture for Short-Term Load Forecasting in Buildings. *Energies* **2020**, *13*, 2370. [CrossRef]
33. Lai, C.S.; Mo, Z.; Wang, T.; Yuan, H.; Ng, W.W.; Lai, L.L. Load forecasting based on deep neural network and historical data augmentation. *IET Gener. Transm. Distrib.* **2020**, *14*, 5927–5934. [CrossRef]
34. Sharma, R.R.; Kumar, M.; Maheshwari, S.; Ray, K.P. EVDHM-ARIMA-Based Time Series Forecasting Model and Its Application for COVID-19 Cases. *IEEE Trans. Instrum. Meas.* **2021**, *70*. [CrossRef]
35. Zhao, Z.; Wang, C.; Nokleby, M.; Miller, C.J. Improving short-term electricity price forecasting using day-ahead LMP with ARIMA models. In Proceedings of the IEEE Power and Energy Society General Meeting, Chicago, IL, USA, 16–20 July 2018; pp. 1–5. [CrossRef]
36. Chen, W.; Xu, H.; Chen, Z.; Jiang, M. A novel method for time series prediction based on error decomposition and nonlinear combination of forecasters. *Neurocomputing* **2021**, *426*, 85–103. [CrossRef]
37. Rajagukguk, R.A.; Ramadhan, R.A.A.; Lee, H.J. A Review on Deep Learning Models for Forecasting Time Series Data of Solar Irradiance and Photovoltaic Power. *Energies* **2020**, *13*, 6623. [CrossRef]
38. Shrestha, A.; Mahmood, A. Review of deep learning algorithms and architectures. *IEEE Access* **2019**, *7*, 53040–53065. [CrossRef]
39. Shinozaki, T.; Watanabe, S. Structure discovery of deep neural network based on evolutionary algorithms. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, Australia, 19–24 April 2015; pp. 4979–4983. [CrossRef]

40. Witten, I.H.; Frank, E.; Hall, M.A.; Pal, C.J. Chapter 10—Deep learning. In *Data Mining: Practical Machine Learning Tools and Techniques*; Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.B.T.D.M.F.E., Eds.; chapter Deep learn; Morgan Kaufmann: Burlington, MA, USA, 2017; pp. 417–466. [CrossRef]
41. Jagait, R.K.; Fekri, M.N.; Grolinger, K.; Mir, S. Load Forecasting Under Concept Drift: Online Ensemble Learning with Recurrent Neural Network and ARIMA. *IEEE Access* **2021**, 98992–99008. [CrossRef]
42. Barak, S.; Sadegh, S.S. Forecasting energy consumption using ensemble ARIMA–ANFIS hybrid algorithm. *Int. J. Electr. Power Energy Syst.* **2016**, *82*, 92–104. [CrossRef]
43. Ganaie, M.A.; Hu, M.; Tanveer, M.; Suganthan, P.N. Ensemble Deep Learning: A Review. *arXiv* **2021**, arXiv:2104.02395
44. Prado, F.; Minutolo, M.C.; Kristjanpoller, W. Forecasting based on an ensemble Autoregressive Moving Average–Adaptive neuro–Fuzzy inference system–Neural network–Genetic Algorithm Framework. *Energy* **2020**, *197*, 117159. [CrossRef]
45. Hyndman, R.J.; Kostenko, A.V.; Hyndman, R.; Kostenko, A.V. Minimum Sample Size requirements for Seasonal Forecasting Models. *Foresight Int. J. Appl. Forecast.* **2007**, *6*, 12–15.
46. Shetty, J.; Shobha, G. An ensemble of automatic algorithms for forecasting resource utilization in cloud. In *FTC 2016—Proceedings of Future Technologies Conference*; Institute of Electrical and Electronics Engineers Inc.: San Francisco, CA, USA, 2017; pp. 301–306. [CrossRef]
47. Van Giang, T.; Debusschere, V.; Bacha, S. One week hourly electricity load forecasting using Neuro-Fuzzy and Seasonal ARIMA models. *IFAC Proc. Vol.* **2012**, *45*, 97–102. [CrossRef]
48. Møller, M.F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw.* **1993**, *6*, 525–533. [CrossRef]
49. Delavar, M.R. Hybrid machine learning approaches for classification and detection of fractures in carbonate reservoir. *J. Pet. Sci. Eng.* **2022**, *208*, 109327. [CrossRef]
50. England, I.N. ISO New England—Energy, Load, and Demand Reports. 2021. Available online: https://www.iso-ne.com/isoexpress/web/reports/load-and-demand (accessed on 26 July 2021).