Next-day Load Forecasting with Smart Meter Data Using Global Recurrent Neural Networks

Evgenii Genov[a], Stefanos Petridis[b], Petros Iliadis[b], Luis Ramirez Camargo[a,c], Thierry Coosemans[a], Nikolaos Nikolopoulos[b], Maarten Messagie[a]

[a]*Electric Vehicle and Energy Research Group (EVERGI), Mobility, Logistics and Automotive Technology Research Centre (MOBI), Department of Electrical Engineering and Energy Technology, Vrije Universiteit Brussel, Brussels, Belgium*
[b]*Centre for Research and Technology Hellas - CERTH, Athens, Greece*
[c]*Copernicus Institute of Sustainable Development - Utrecht Universit, Utrecht, Netherlands*

Abstract

In a smart grid, consistent and accurate load forecasting is critical to successful operation and energy management. With the growing availability of smart-meter data, machine learning models have the ability to train one model globally across entire datasets, rather than training a separate model individually for each time series. A global training set-up enables the learning of patterns of electricity consumption in households and brings potential computational advantages. We present an experiment that evaluates a Long-Short-Term Memory (LSTM) Global Forecasting Model (GFM) in comparison to benchmark methods: Feed-Forward Artificial Neural Network, Seasonal Autoregressive Integrated Moving Average and standard load profile models. The selected methods and the evaluation framework are derived from an extensive literature review. We use the Irish smart-meter dataset, collected by the Commission for Energy Regulation. Assessment includes the scalability and computational performance of the algorithms. Comparisons show that the average error in terms of several metrics is at least 3% smaller than the benchmark performance, indicating that the LSTM-GFM model obtains predictions with a superior accuracy. A complementary 'weak learner' model, used to generate features from a seasonal decomposition, further decreases the error. The study shows that LightGBM models are faster and more suitable for quick model iterations and LSTM-based models are more appropriate for accuracy-focused load forecasting applications.

Keywords: Smart meter, load forecasting, global model, long- and short-term memory, machine learning.

1. Introduction

Researchers and industry practitioners have been working towards accurate forecasting of electricity load for decades. The forecasting field has been evolving, with better metrics and more advanced models proposed. The assessment and direct comparison of such methods is often complicated by the varying setup of experiment across papers. To address this problem, forecasting competitions have been set up among academics and industry practitioners. These give an indication of a rising dominance of globally trained machine learning models. Firstly, in 2018, a method that combined a recurring neural network (RNN) and exponential smoothing (ES) won the M4 competition [1]. The competition data set included 100,000 time series from various fields. In 2020, the M5 competition was set up to accurately predict 42,840 time series that represent hierarchical retail sales at Walmart. The results confirmed the added value of training models globally. The top participants used cross-learning from multiple series simultaneously. These findings are to be confirmed in applications concerning smart meter time series data.

The deployment of smart meters increases the potential for the application of data-driven models to forecast consumption in individual households. Bandara et al. [2] show that global RNN models are well suited to forecasting groups of related time series with multiple seasonal cycles. Montero-Manso and Hyndman present a theoretical framework that states that any local approach applied to a dataset of numerous series can be reproduced by a global model of sufficient complexity, regardless of the relatedness of the underlying series. Smart meter data comprises of many series with complex properties. High stochasticity, multiple seasonalities, and nonstationary statistical properties, present in time-series, make it a tedious task to forecast. Therefore, we here evaluate the performance of global RNN models in application to smart-meter data from individual households.

To determine a benchmark for global RNN models, a bibliometric study is conducted and a review of state-of-the-art electric load forecasting is performed. Consequently, an evaluation framework is proposed that allows the best reproducibility and most correlates with existing research. Section 2 describes the method used for building the model, as well as the necessary pre-processing and feature engineering. We address the design of the experiment in Section 3. The evaluation of the model in terms of accuracy and computation performance is presented in Section 4. And the conclusions follow in Section 5.

1.1. Bibliometric analysis and state-of-the-art

Bibliometric analysis is a systematic process that can be used to quantitatively analyze forecast models and results reported in the literature. The rationale behind this step is to gain insight into the state-of-the-art in electric load forecasting. In addition, the analysis helps identify an experimental framework most suitable for benchmarking the results against existing and future research. The procedure reveals important information about aspects of model development, such as temporal granularity, forecasting horizon, use of exogenous variables, and evaluation metrics.

The bibliometric analysis was carried out on June 14, 2021. A query was made in the SCOPUS database of scientific literature for studies that include the keywords 'electric load forecasting' and 'smart meter' of the last decade. Of the 82 papers returned, 35 papers were considered out of scope, resulting in a final set of 47 papers. The main criterion of this filtering is the requirement for papers to have a clear focus on forecasting performance. Many of the reviewed articles were mainly aimed at other topics such as, for example, pricing, telecommunications, and security, and there was little or no information on the actual forecasting procedure.

Figure 1 presents an account of machine learning and statistical models implemented in this set of studies. For machine learning models, regular multilayer perceptron (MLP), support vector regression (SVR), long-short-term memory (LSTM) networks. For statistical models, the most popular methods are ARIMA models and linear regression. Finally, a special reference is made to the persistence naive method, a popular benchmark, which has been utilized frequently in this set.

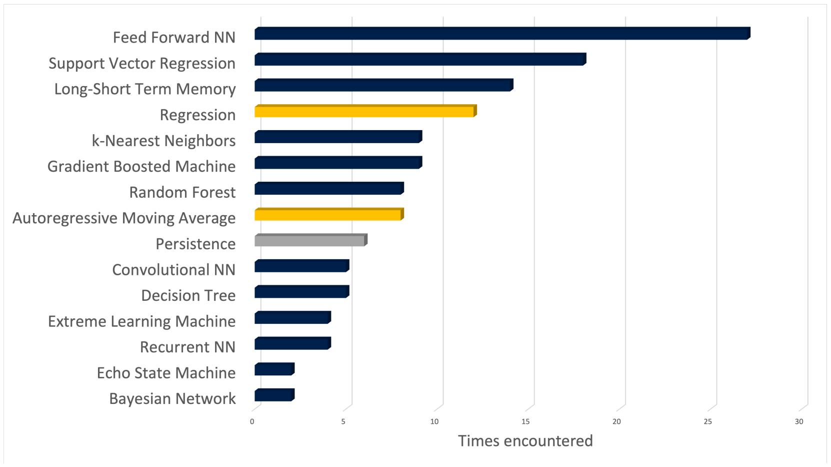


Figure 1: Models applied in the reviewed studies. Machine learning (blue), statistical (yellow), and naive persistence (gray).

With respect to temporal granularity and forecast horizon, a high variation is observed between the studied. The common intervals used include half and one hour, with 38% and 40% of the studies, respectively. Other resolutions encountered are 1 minute [4], 15 minutes [5, 6], and 1 day [7]. Regarding the forecast horizon, the values range from 30 minutes [8] to 200 hours ahead [9]. Some variation is also encountered in the usage of evaluation metrics. However, the most popular metrics include mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE), which are used by 83% of the studies. For exogenous parameters, two main categories have been identified. The first includes weather information, usually obtained by reanalysis of data sets, and is used in 38%. The second category includes various forms of calendar information such as weekdays/weekends and holidays identifiers. The calendar features are used in 14% of the studies.

Regarding the use of data sets, only 56% of the publications use an open-access data set for evaluation. Private datasets exhibit disadvantages, such as the lack of the ability to replicate the results, compare the performance of the model with the literature, or build on the development of the previously proposed model. The most used open-access data set was the Irish data set of the Commission for Energy Regulation (CER) [10]. According to [11], the CER dataset is also the largest in the number of series. Therefore, the CER dataset is considered to be most suitable for the comparison of different forecasting models, across studies, and in experimentation.

A literature survey is conducted on a selection of 15 publications that make predictions for the CER dataset. Studies and their main characteristics are presented in Figure 2, where each vertical axis represents a specific attribute. Although this study focuses on individual households' consumption, most (9 out of 15) of the studies perform forecasts at a higher level of aggregation. The most common specifications, as shown in Figure 2, guide the choices for the experimental design in Section 3. The evaluation is done for models using the original consumption records, as well as exogenous weather and calendar features. The final evaluation is done to make predictions within a 24 hours ahead horizon. Table 1 presents a review of the accuracy of the forecast at the individual build-

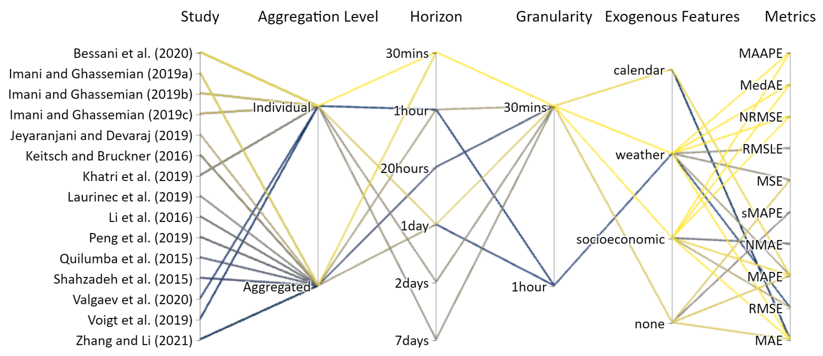


Figure 2: Studies utilizing the open-access CER dataset and its main characteristics. A different color has been assigned to each publication.

ing level for a horizon of 24 hours. The selection of accuracy metrics used in forecast evaluation varies between studies. There is one common metric, the root mean square error (RMSE). However, Valgaev et al. [12] evaluate RMSE for normalized data, rescaled by a maximum value of each series, which leads to the loss of an original scale. Furthermore, the comparison is not strictly valid due to discrepancies in the selected test set. RMSE scores are scale-dependent. This means that the results are not comparable between different data sets. Khatri et al. [13] conduct the study over the time-series meter data of 3,637 residential users. Valgaev et al. [12] consider a subset from a control group and narrows down to a group of 444 buildings with few missing values and no outlier series. The control group is selected since the other buildings in the dataset are subject to interventions, e.g., demand response schemes, part of the behavior trials. Voigt et al. [14] use a representative set of 38 households to evaluate the forecast. Therefore, the results reported in Table 1, should be considered indicative but are not completely suitable to serve as a benchmark between publications. The results reported in the literature show that the models running LSTM cells are the most promising. Furthermore, the research by Valgaev et al. [12] highlights the importance of building a standardized load profile (SLP) benchmark. The further experiment broadens the state-of-the-art work with the construction and testing of a global LSTM model using the CER Irish dataset for a 24-hour ahead horizon.

	Network	MAE	MSE	RMSE	RMSE (scaled)
[13]	LSTM	0.3	0.24	0.49	
	GRU	0.44	0.28	0.53	
	SimpleRNN	0.34	0.32	0.57	
[14]	ANN			0.7	
[12]	SLP				0.086
	MISO-MLP				0.104
	MIMO-MLP				0.095
	NAR				0.097
	NARX				0.188

Table 1: CER dataset accuracy results for the 24-hour forecast reported in the literature. The models under evaluation include Long-Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), a simple Recurrent Neural Network (SimpleRNN), Artifical Neural Network (ANN), Standardized Load Profile (SLP), Multi-Input Single-Output Multi-Layer Perceptron (MISO-MLP), Multi-Input Multi-Output Multi-Layer Perceptron (MIMO-MLP), Nonlinear Autoregressive model (NAR) and Nonlinear Autoregressive Model with exogenous inputs (NARX).

2. Methodology

In this section, the methodology used to generate the forecast is outlined. We choose the 24-hour horizon, as commonly found in literature, and it is well suited for energy management tasks such as battery scheduling. First, we formally define the problem and then describe the core concepts within the forecasting workflow. This section discusses the approach to feature engineering, scaling, data preparation, and model architecture.

2.1. Problem statement

The problem scope is defined as a multi-step direct univariate out-of-sample forecasting. For a time series in a database $X_i \in \{X_0, X_1, ...X_n\}$, the forecast model on an h horizon is formulated as follows:

$$X_{i,t} = m_h\left(X_{i,t-h}, \ldots, X_{i,t-h-p_h}; \boldsymbol{\theta}_h\right) + e_{i,t,h} \tag{1}$$

Having a generalized set of parameters $\boldsymbol{\theta}_h$ for all series X_i defines a global model approach. The term $e_{i,t,h}$ is the error term. In addition to historical past observations $X_{i,t-h}, \ldots, X_{i,t-h-p_h}$, the model m_h also incorporates variables that model seasonality $S \in \{S_1, S_2 ... S_{t-h-p_h}\}$ and exogenous variables $T \in \{T_1, T_2 ... T_{t-h-p_h}\}$. Finally, the model for each horizon has the form

$$\hat{m}^{(h)}(X_{i,t}) = m_h \left(X, S, T; \hat{\boldsymbol{\theta}}_h \right) \tag{2}$$

2.2. Preprocessing

Neural networks show a weak ability to learn from raw data. Time series often contain major drifts and non-stationarities. An adequate preprocessing step is required. The adapted workflow is illustrated in Figure 3. To a large extent, we adapt the steps performed in [15]. In this manner, the CER smart meter data is cleaned, additional features are generated, and transformations in normalization and variance stabilization are applied. A supplementary seasonal decomposition step is tested. Data cleaning includes removing samples with duplicate timestamps and interpolating the missing values linearly. Finally, the data are mapped according to a moving-window scheme and solved as a supervised learning problem. The input of the model contains labels with

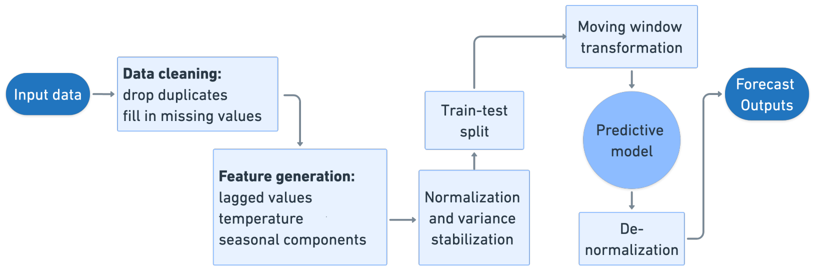


Figure 3: Processes and the data flow

historical electrical consumption, as well as predictor features, such as seasonality, weather data, and lagged values. The next Section 2.4 goes into detail on modeling seasonality features. Lagged values include the measurement of consumption at the same hour on a previous day and a previous week. Historical weather data series are obtained from the ERA5-Land hourly dataset [16]. The original location of the households in the smart meter data cannot be disclosed, for the data being anonymized. Therefore, weather features are taken average over three major cities in Ireland (Dublin, Cork, and Limerick).

In load forecasting publications dealing with the CER data set, an exact methodology on normalization is often unspecified, making it difficult for an experiment to be replicable. When training a global model, a common scale is imperative to avoid varying scales between series. Initially, two methods for scaling are evaluated: Min-Max normalization and division by mean. Models using Min-Max Normalization show lower accuracy. We chose a method that involves mean normalization for our final evaluation. However, even after scaling the time series in the dataset to a common level, the statistical properties of the series, such as mean, variance and autocorrelation, still change over time.

This indicates that the time series still exhibit non-stationary behavior. The transformation $log(x+1)$ is used to make a time series stationary because it can handle non-positive values, unlike the natural logarithm transformation

2.3. Recurrent neural networks

Recurrent neural networks (RNNs) have become a widely used method to solve problems that incorporate variable sequence length data. RNNs enable sequential processing of the input. Every time the RNN reads a new input, the internal hidden state is updated and fed back to the model when reading the subsequent input. The fundamental idea for the design of RNN architecture is so-called truncated backpropagation through time. The network steps forward through time instances of the internal state and computes loss. During the backward pass, the network updates the gradients in reverse over time. The truncated approach implies the use of batches when calculating the gradients during backpropagation. Approximate gradients are calculated in short sequences instead of the full learning sequence. This approach speeds up the performance of the network without affecting its ability to learn long-range dependencies, as shown in [17] for RNN language modeling applications.

During backpropagation, the calculation of gradients includes many multiplications by the transposed weight matrix and a *tanh* activation function. Therefore, vanishing and exploding gradient problems are common in the training of recurrent neural networks, particularly in the 'Vanilla' RNN cells [18, 19]. To address the problem of vanishing gradients, a more complex RNN architecture needs to be employed. LSTM topology of the RNN cells was introduced by [20]. The final version of the architecture included a forget gate proposed by [21]. The LSTM architecture is designed to have better gradient flow properties. An LSTM cell keeps two states, the main hidden state h_t and an internal cell state c_t. Current data inputs are stacked with a previous hidden state. The resulting vector is used to compute four gates: input gate i, forget gate f, output gate o, and cell input gate g.

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \tag{3}$$

$$c_t = f \odot c_{t-1} + i \odot g \tag{4}$$

$$h_t = o \odot \tanh(c_t) \tag{5}$$

The LSTM cell uses the forget gate f to regulate how much of the previous cell state should be erased in a current state computation. While the loss function sets the objective to find the most likely next state in a sequence, RNNs are capable of extracting both short- and long-term dependencies in time series. The LSTM cell can process data sequentially and keep its hidden state throughout time. LSTM networks are selected to build a global forecasting model (GFM).

A benefit of LSTM-GFM models is their ability to be applied to forecasting out-of-sample series. In this case, no additional training is required. An example of this can be found in [22], demonstrating superior results with a global RNN model for households outside the training sample for the London smart-meter dataset. In context of electric load forecasting, this can be beneficial to produce forecast in newly-connected households – scarce in recorded time series data.

2.4. Modelling seasonality

Prior knowledge about the seasonality of the series can be incorporated into the model. A common practice is to include features that encode seasonality. These features denote periodicity on multiple-seasonal cycles: daily, weekly, and yearly. Two strategies are employed to encode the seasonal patterns, namely calendar features or a seasonal decomposition model. These strategies establish multiple configurations of a global Recurrent Neural Network. The basic configuration, LSTM-GFM, is inputted with calendar features. Seasonal decomposition is tested in two variants. LSTM-GFM-XS employs seasonal components as exogenous features. LSTM-GFM-DS is fitted to predict only the residual component, while the seasonal and trend components are modeled separately.

The basic configuration model, LSTM-GFM, uses explanatory variables of the calendar effect. A useful method applied to calendar variables is to encode cyclical features using a first-order Fourier approximation, a combination of sine and cosine variables [23]. The two variables model the relative position of a timestamp within a seasonal cycle.

LSTM-GFM-DS and LSTM-GFM-XS methods employ a time series decomposition, also known as deseasonalization. Decomposing a time series into seasonality, trend, and residual components potentially reduces the complexity of a time series, which improves the accuracy of the forecast. For the deseasonalized (DS) configuration, the components are modeled independently and combined afterward. Alternatively, seasonal components are used as exogenous inputs (XS). [2] showed that the use of decomposition as an explanatory variable is more beneficial in the prediction of real-world data sets with homogeneous series.

In this study, an unobserved univariate components model (UCM), developed by [24], is used. UCM performs a time series decomposition to extract seasonal, trend, and irregular components.

$$y_t = \underbrace{\mu_t}_{\text{trend}} + \underbrace{\gamma_t}_{\text{seasonal}} + \underbrace{\varepsilon_t}_{\text{residual}} \tag{6}$$

UCM is implemented in a Python module for statistical modeling – statsmodels [25]. The application of an additive seasonal decomposition using the CER data is shown in Fig. 6. The particular implementation has the deterministic seasonality and a level component, which is a regression model that includes an intercept with a fixed slope. The deterministic seasonal term, γ_t, is estimated with trigonometric functions:

$$\gamma_t = \sum_{j=1}^{[s/2]} (\alpha_j \cos \lambda_j t + \beta_j \sin \lambda_j t) \tag{7}$$

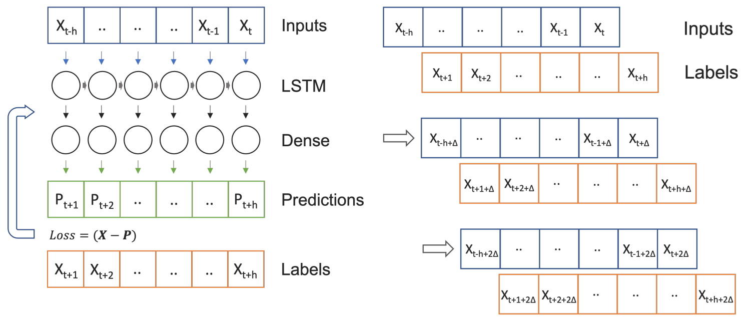


Figure 4: Left: Multiple input multiple output architecture of LSTM network in unrolled representation. Right: Moving window scheme. Time series are split into samples shifted by δ steps. Each sample is composed of sets of inputs and labels, predicted values

2.5. Input formatting

The input format maps a forecasting problem to a supervised learning framework. The time series sequence is converted to pairs of input and output sequences. On the left, Figure 4 shows the architecture used for the forecasting model with an LSTM layer. The architecture is referred to in literature as stacked architecture. The model configuration includes an input layer, an LSTM layer that is stacked with a dense layer, and an output layer. The dense layer maps the LSTM output to the output sequence.

The models are set up to perform multiple input, multiple output forecasting. The input and output sequences are formed using a moving-window scheme. A diagram of the moving window scheme is presented on the right of Figure 4. The model accepts a sample which contains a set of input features and labels limited to a time window of a limited number of time steps. The sizes of both the input and output windows are selected equal to the forecast horizon, 24 hours. Training is performed on a sample window that rolls through the original data with a step shift. When testing the models, we choose to evaluate the predictions at every timestamp; therefore, the step shift is 1. However, in training, a step size of 1 considerably increases the computation time. It may also lead to overly correlated validation scores and subsequent model bias. Model training configurations with a different step shift set to a half or full prediction horizon $(24, 48)$ are tested.

3. Experimental setup

3.1. Dataset

The data set used in this study is provided by CER. Data are collected as part of the Smart Metering Electricity Customer Behavior Trials. The regulator initiated the trials to assess the impact of smart meter infrastructure on electricity consumption in Ireland.

This work considers only a subset of households, labeled the "control group". An adjustment is made for daylight saving time: the extra hour measurements are removed, while an extra hour is interpolated by averaging near values. A final selection of 909 houses is made after discarding the series with ≥ 50 missing values. The summary statistics of the data set are given in Table 2. In Figure 5, we can observe a normalized day load profile for the CER data set. It shows that on average there is no significant discrepancy between weekday and weekend activities. These observations are also confirmed by the seasonal decomposition plot in Figure 6. It is also evident that seasonal patterns can vary between series, particularly in the weekly and yearly cycles.

The time series are divided into training, validation, and test sets. The test set is selected at the end of the time series. The duration of a test set is 1440 time steps, equivalent to 30 calendar days. The validation data are selected similarly, 30 days at the end of the remaining series.

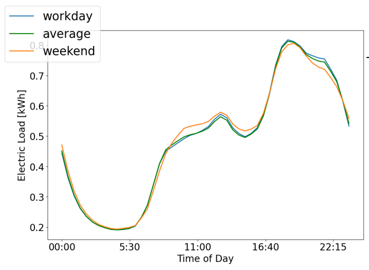


Figure 5: Average daily load profile

Country	Ireland
No. of time series	909
Frequency	30 min
Start Date	2009-07-15
End Date	2011-01-02
Max. Length	25728
Min. Length	25468
missing values [%]	< 0.001%

Table 2: CER Data Description

3.2. Error Metrics and computational performance

Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are commonly used to calculate the error of the model. These measures are scale-dependent, meaning that the results of such an evaluation are not comparable between time series of different magnitudes.

An accuracy metric, the mean absolute scaled error (MASE), first proposed by [26], scales the error with respect to a naive forecast.

$$MASE = \frac{MAE}{MAE_{\text{in-sample, naive}}} \tag{8}$$

where MAE is the mean absolute error, a commonly used accuracy metric.

$$\text{MAE} = \frac{1}{N}\sum_{i=0}^{N-1} |\hat{y}_i[t] - y_i[t]| \tag{9}$$

MAE is scale-dependent, meaning that the magnitude of the error varies for each dataset, depending on the scale of the measurement. The MASE metric is scale-invariant and symmetric and shows predictable behavior at near-zero values. The results also have a real-world interpretation. A MASE score less

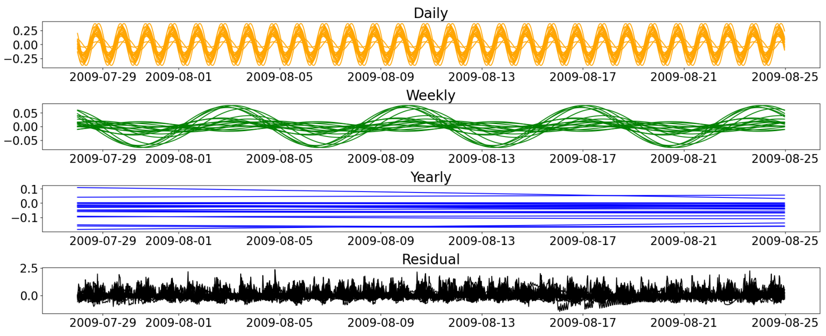


Figure 6: Additive seasonal-trend decomposition plot over a 16 week period. Obtained by applying a UCM model to a sample of 20 time series within CER dataset

than one indicates superior performance over the forecast produced by a naive method.

The methods under evaluation are measured in terms of pre-processing, training, and execution times. All experiments are run subsequently on the same virtual machine (VM) on a server. The virtual machine runs on an Intel® Xeon® Silver 4210R processor, with 2 sockets, 8 cores per socket, and 125GB of memory.

3.3. Benchmark models

Benchmark models are selected to match the state-of-the-art (SARIMAX, ANN, LightGBM), as well as reference methods, commonly used to assess the feasibility of additional modeling. Performance is evaluated in terms of the same metrics as described in Section 3.2.

One of the basic benchmark methods is a seasonal naive model, also known as a persistence model. The naive method makes the prediction using the values of the last observed period. In the context of this study, persistence outputs the observed values for the last 24 hours.

The next benchmark is standard load profiles (SLP), a fixed-rule method currently used for household forecasting. Residential SLPs (”24 hour urban domestic”) are taken from [27], a retail market administrator in Ireland. Standard load profiles are a statistical method that requires low computational effort. Standard profiles are scaled by the total annual consumption. This number is extracted from aggregating total demand over a year-long period in a training subset.

A feed-forward neural network method is implemented. The two-stage artificial neural network (ANN) model proposed by [28], is purposely designed for short-term load forecasting. The ANN is of a supervised type 'multi-layer perceptron'. It is trained locally in a series-specific fashion.

We test a seasonal autoregressive integrated moving average model with exogenous parameters (SARIMAX) to provide the benchmark results. It is also trained series-by-series. ARIMA models are a well-established statistical method that has been extensively used in various forecasting applications. Adding exogenous parameters and seasonal decomposition to the original ARIMA algorithm produces more accurate results, since the time series' behavior is highly dependent on external factors such as temperature and calendar features.

The paper uses LightGBM as a benchmark model by training it globally on the entire dataset (LightGBM-GFM) and individually on each time series. LightGBM has several advantages for time series forecasting, including its ability to handle large datasets with high dimensionality, its fast training speed, and its ability to handle non-linear and non-monotonic relationships between features and the target variable [29]. The LightGBM model is a prevalent technique scoring highly and winning in forecasting competitions [30] [31]. Non-model parameters, set in the design of the architecture, are referred to as hyperparameters. The explored range of hyperparameters is provided in the Table 3.

3.4. Implementation details

Every LSTM model is built using version 2.4.0 of Tensorflow, an open-source deep-learning platform. To handle large amounts of data, the data input pipeline is set up using a 'tensorflow.data' API. The Adam learning algorithm is used to iteratively update the network weights based on the gradients of the loss function with respect to the weights. All necessary scripts are programmed in Python.

To limit the overfitting of the model, we use an early stopping technique. The training of the model stops when the maximum number of training epochs is reached or a validation error grows. The number of epochs with no improvement in error is a patience parameter. An epoch is a training iteration that takes the learning algorithm to work through the entire data set.

The hyperparameter configuration includes the following parameters: number of layers, number of cells, learning rate, batch size, and number of epochs. The tuning method and the ranges of values are defined, following the general recommendations outlined in [32]. A Bayesian hyperparameter tuning technique is applied to find an optimal combination of hyperparameters that produces the model with the best performance on the validation set. In contrast to a grid search, which goes over all possible combinations, the Bayesian method models the distribution of the objective function with a Gaussian prior; therefore, reducing the number of total evaluations. Each hyperparameter configuration runs for 1 epoch. The loss function used to fit the model to the training data optimizes the mean squared error (MSE) metric. The initial ranges for the hyperparameters used for RNN are shown in Table 4. The number of epochs and patience used in the final configuration are set to fixed values, 20 and 5 respectively, in the interest of limiting the computation time.

4. Results and Discussion

LSTM-GFM models show superiority over the available benchmarks in terms of all error metrics in the evaluation (Table 5). Best LSTM model makes pre-

Table 3: LightGBM hyperparameters

Hyperparameter	Type	Range	Step
num_iterations	categorical	[100]	-
learning_rate	float	[0.01, 0.3]	-
num_leaves	integer	[20, 3000]	20
max_depth	categorical	[-1, 5, 10, 15, 20, 25]	-
min_data_in_leaf	integer	[20, 100]	10
lambda_l1	integer	[0, 100]	5
lambda_l2	integer	[0, 100]	5

Table 4: LSTM hyperparameters

layers	step shift	cell size	learning rate	batch size
$1-2$	$12, 24$	$32-128$	$0.0005-0.002$	$1024-4096$

dictions with an absolute mean error 3% lower than the closest benchmark. The average root mean squared error is also 3% lower. The ANN model shows competitive accuracy, being runner-up in terms of RMSE and MASE. The distributions of forecast error are demonstrated in the violin plots in Figure 7. It should be noted that the SARIMAX methods show the largest variance in terms of MAE and RMSE. Methods that employ gradient boosting, LightGBM-GFM and LightGBM, show the highest spread in terms of MASE. Additionally, the violin plot for MASE shows some discrepancy in the shape of an error distribution. The RNN- and GBM-based configurations show flatter distributions of MASE, while ANN scores more consistently around the mean.

The experiments highlight the importance of a consistent benchmarking process. A comparison with benchmark models indicates a relatively poor performance of a SARIMAX and LightGBM-based methods. Notably, the MASE metric exceeds 1 for the mentioned benchmark methods. The values above 1 indicate that the model outputs predictions with a larger error than a naive method fit to the training data. Consequently, MASE over 1 indicates that the naive method applied to the test data performs worse than the same method applied to the training data.

Evidently, a naive forecast shows better accuracy than the standard load profile. Therefore, the availability of recent data increases the accuracy of the forecast without additional modeling. At the individual household level, an addition of high-granularity smart-meter data customizes the forecast to each building's specification. This highlights the added value of smart meter data, as well as the importance of benchmarking using a naive persistence model.

Furthermore, it can be seen that a global LSTM model benefits significantly from a previous seasonal decomposition step. A baseline global LSTM model shows an inferior ability to learn seasonality directly from the series. This confirms the findings reported in [2], where a baseline GFM outperforms the GFM-DS variant, but obtains worse results than the seasonal exogenous approach

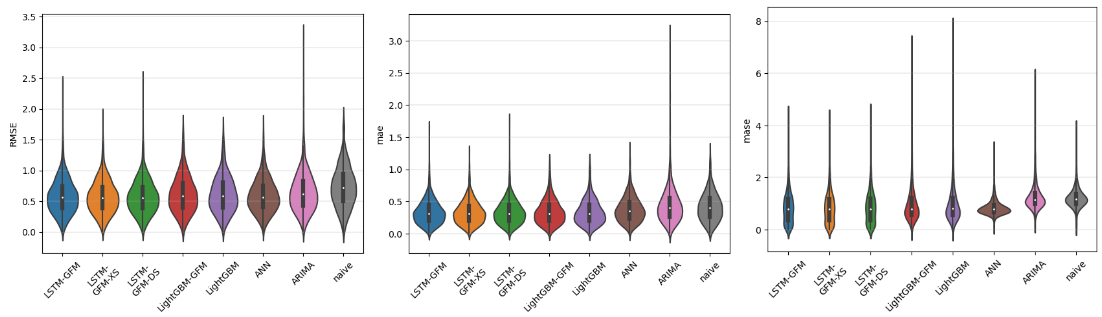
	Name	MAE	RMSE	MASE
Global	LSTM-GFM	0.335	0.578	0.864
	LSTM-GFM-XS	**0.331**	**0.571**	**0.856**
	LSTM-GFM-DS	0.337	0.574	0.872
	LightGBM-GFM	0.342	0.611	1.148
Local	SARIMAX	0.442	0.653	1.313
	ANN	0.380	0.595	0.865
	LightGBM	0.340	0.613	1.147
	naive	0.419	0.732	1.229
	SLP	0.435	0.731	1.323

Table 5: The accuracy of the forecast of CER data as obtained in the experiment.

(XS), when tested for a dataset of 300 Australian households. Removing the seasonal component prior to training, thus making the model uninformed of seasonality, has a negative effect on the model's performance. Instead, seasonal components have been added as additional features to the training. It may lead to the implication that seasonality properties within the Irish CER dataset are less homogeneous. The seasonal component varies by series, therefore, learning seasonality explicitly and exclusively for each series achieves better results.

When conducting the literature review, we assemble the accuracy results reported in existing publications on short-term forecasting in individual households, shown in Table 1. A direct comparison has implications associated with the accuracy metric used and the discrepancies between experimental setups. However, if we attempt this analysis, it can be seen that the results produced show a higher accuracy than those reported in [14] and [12]. The latter uses a different accuracy metric; however, a comparison can be drawn since the same standard load profile is used as a benchmark. The LSTM model presented in [13] gives better accuracy in terms of RMSE and is comparable in terms of MAE. This can be attributed to building a separate model for each time interval, which we expect to be sub-optimal in terms of computation time.

Table 6 gives an overview of computational costs in terms of processing time. The computation time is scaled per time series. The division is drawn between pre-processing the data, fitting a model and an optimal set of hyperparameters, and finally, making the prediction. The training time also includes the time spent on the hyperparameter search. The network architecture is an influential factor for the speed of training. It is notable that modes trained with LightGBM are significantly faster in training and execution. The varying performance in time and accuracy between models indicates the existing trade-offs when choosing between gradient boosted tree-based and LSTM-based models. LightGBM training takes less time, making it possible for forecast practitioners to conduct more tests with different model configurations. This enables a more comprehensive approach at feature engineering and feature selection. Therefore, LSTM models are worthwhile and bring the superior accuracy in the applications with an established and pre-defined input feature set. This highlights the importance



(a) RMSE (b) MAE (c) MASE

Figure 7: Violin plots of the relative performance of different forecasting methods. Each method is represented in a different color. Thickness of the plot indicates the frequency of errors at a given value

		Duration per series [seconds]		
	Name	pre-processing	training	execution
Global	LSTM-GFM	**1.15**	79.29	1.00
	LSTM-GFM-XS	8.52	84.00	1.00
	LSTM-GFM-DS	8.52	72.97	1.52
	LightGBM-GFM	**1.15**	**22.93**	0.04
Local	LightGBM	**1.15**	39.14	**0.01**
	SARIMAX	**1.15**	142.57	0.08
	ANN	**1.15**	312.87	0.42

Table 6: Processing, training, and execution times of forecasting algorithms

of carefully selecting the appropriate modeling approach for the task at hand, taking into consideration factors such as the size and complexity of the dataset, the desired level of accuracy, and the available computing resources.

The results indicate that the models trained globally take a shorter training time. This applies both to the GBM and RNN models. Training local models takes longer out of necessity to fit a set of parameters and find an optimal set of hyperparameters for each of multiple series. It is evident that a seasonal decomposition step increases the preprocessing time of a GFM model. However, the impact on training time is more moderate and is mainly attributed to an increase in the number of features used in the input tensor.

5. Conclusions

We investigate the utilization of global RNN models in the context of short-term (24 hours ahead) load forecasting on a household level. The proposed LSTM global forecasting model (LSTM-GFM) methodology allows learning cross-series information and can be efficient in terms of computational performance. The global method has the capacity to leverage massive time series databases, now available with a wide rollout of digital meters. The ability of such a configuration is evaluated in the experiment using the CER Smart Metering dataset.

In conclusion, the LSTM-GFM models exhibit superior accuracy compared to benchmark models in short-term forecasting for individual households, as demonstrated by all error metrics evaluated. While the ANN model is competitive, the SARIMAX and LightGBM-based methods show relatively poor performance, with MASE values exceeding 1. The results emphasize the importance of consistent benchmarking using a naive persistence model, highlighting the added value of recent and high-granularity smart-meter data for improving forecast accuracy. Additionally, the experiments confirm the benefits of using a previous seasonal decomposition step for global LSTM models to explicitly learn seasonality properties for each series.

Finally, the choice of algorithm can significantly impact the processing time, with LightGBM models being significantly faster than LSTM-based models. However, there are trade-offs between speed and accuracy, and it is essential to consider factors such as the size and complexity of the dataset, the desired level of accuracy, and the available computing resources when selecting an appropriate approach. The results of this study suggest that LightGBM models are more suitable for cases where quick model iterations are needed, while LSTM-based models are more appropriate for situations where accuracy is of utmost importance. Therefore, practitioners should carefully consider the trade-offs and select the modeling approach that best fits their specific needs.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 824342 as well as from VLAIO in the ICON project MAMUET grant number HBC.2018.0529.

References

[1] S. Makridakis, E. Spiliotis, V. Assimakopoulos, The m4 competition: Results, findings, conclusion and way forward, International Journal of Forecasting 34 (2018) 802–808.

[2] K. Bandara, C. Bergmeir, H. Hewamalage, LSTM-MSNet: Leveraging forecasts on sets of related time series with multiple seasonal patterns, arXiv (2019). doi:10.1109/tnnls.2020.2985720.

[3] P. Montero-Manso, R. J. Hyndman, Principles and algorithms for forecasting groups of time series: Locality and globality, International Journal of Forecasting 37 (2021) 1632–1653.

[4] Q. Peng, Z.-W. Liu, Short-term residential load forecasting based on smart meter data using temporal convolutional networks, in: 2020 39th Chinese Control Conference (CCC), IEEE, 2020, pp. 5423–5428.

[5] S. Park, S. Jung, S. Jung, S. Rho, E. Hwang, Sliding window-based lightgbm model for electric load forecasting using anomaly repair, The Journal of Supercomputing 77 (2021) 12857–12878.

[6] T. Zufferey, A. Lepouze, G. Hug, Inadequacy of standard algorithms and metrics for short-term load forecasts in low-voltage grids, in: 2019 IEEE Milan PowerTech, IEEE, 2019, pp. 1–6.

[7] Y. Ding, M. A. Neumann, P. G. Da Silva, M. Beigl, A framework for short-term activity-aware load forecasting, in: Joint Proceedings of the Workshop on AI Problems and Approaches for Intelligent Environments and Workshop on Semantic Cities, 2013, pp. 23–28.

[8] M. Bessani, J. A. Massignan, T. M. Santos, J. B. London, C. D. Maciel, Multiple households very short-term load forecasting using bayesian networks, Electric Power Systems Research 189 (2020) 106733. URL: https://doi.org/10.1016/j.epsr.2020.106733. doi:10.1016/j.epsr.2020.106733.

[9] M. N. Fekri, H. Patel, K. Grolinger, V. Sharma, Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network, Applied Energy 282 (2021) 116177.

[10] C. Mannion, Smart metering project commission for energy regulation (CER) Ireland, in: IET Seminar on Smart Metering 2010: Delivering a Smart UK, 2010, pp. 1–12. doi:10.1049/ic.2010.0051.

[11] Y. Wang, Q. Chen, T. Hong, C. Kang, Review of Smart Meter Data Analytics: Applications, Methodologies, and Challenges, IEEE Transactions on Smart Grid 10 (2019) 3125–3148. doi:10.1109/TSG.2018.2818167.

[12] O. Valgaev, F. Kupzog, H. Schmeck, Adequacy of neural networks for wide-scale day-ahead load forecasts on buildings and distribution systems using smart meter data, Energy Informatics 3 (2020). doi:10.1186/s42162-020-00132-6.

[13] I. Khatri, X. Dong, J. Attia, L. Qian, Short-term Load Forecasting on Smart Meter via Deep Learning, 51st North American Power Symposium, NAPS 2019 (2019) 1–6. doi:10.1109/NAPS46351.2019.9000185.

[14] N. Voigt, T. Wawer, T. Albert, Optimizing prosumers' battery energy storage management using machine learning, International Conference on the European Energy Market, EEM 2019-Septe (2019) 1–6. doi:10.1109/EEM.2019.8916289.

[15] E. Genov, S. Petridis, P. Iliadis, N. Nikopoulos, T. Coosemans, M. Messagie, L. R. Camargo, Short-term load forecasting in a microgrid environment: Investigating the series-specific and cross-learning forecasting methods, in: Journal of Physics: Conference Series, volume 2042, IOP Publishing, 2021, p. 012035.

[16] J. M. Sabater, et al., Era5-land: a new state-of-the-art global land surface reanalysis dataset, Earth Syst. Sci. Data 13 (2017) 4349–83.

[17] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, S. Khudanpur, Recurrent neural network based language model., in: Interspeech, volume 2, Makuhari, 2010, pp. 1045–1048.

[18] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE transactions on neural networks 5 (1994) 157–166.

[19] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International conference on machine learning, PMLR, 2013, pp. 1310–1318.

[20] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[21] F. A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with LSTM, Neural Computation 12 (2000). doi:10.1162/089976600300015015.

[22] A. M. Alonso, F. J. Nogales, C. Ruiz, A single scalable lstm model for short-term forecasting of massive electricity time series, Energies 13 (2020) 5328.

[23] R. J. Hyndman, G. Athanasopoulos, Forecasting: principles and practice, OTexts, 2018.

[24] A. C. Harvey, Forecasting, Structural Time Series Models and the Kalman Filter, Cambridge University Press, Cambridge, 1990. doi:DOI:10.1017/CBO9781107049994.

[25] S. Seabold, J. Perktold, Statsmodels: Econometric and statistical modeling with python, in: Proceedings of the 9th Python in Science Conference, volume 57, Austin, TX, 2010, pp. 10–25080.

[26] R. J. Hyndman, A. B. Koehler, Another look at measures of forecast accuracy, International Journal of Forecasting 22 (2006) 679–688. doi:10.1016/j.ijforecast.2006.03.001.

[27] RMDS, Standard Load Profiles, ???? URL: https://rmdservice.com/standard-load-profiles/.

[28] L. Hernández, C. Baladrón, J. M. Aguiar, L. Calavia, B. Carro, A. Sánchez-Esguevillas, J. Sanjuán, Á. González, J. Lloret, Improved short-term load forecasting based on two-stage predictions with artificial neural networks in a microgrid environment, Energies 6 (2013) 4489–4507.

[29] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, Advances in neural information processing systems 30 (2017).

[30] S. Makridakis, E. Spiliotis, V. Assimakopoulos, M5 accuracy competition: Results, findings, and conclusions, International Journal of Forecasting (2022).

[31] T. Hong, J. Xie, J. Black, Global energy forecasting competition 2017: Hierarchical probabilistic load forecasting, International Journal of Forecasting 35 (2019) 1389–1399.

[32] H. Hewamalage, C. Bergmeir, K. Bandara, Recurrent Neural Networks for Time Series Forecasting: Current status and future directions, International Journal of Forecasting 37 (2021) 388–427. URL: `https://doi.org/10.1016/j.ijforecast.2020.06.008`. doi:`10.1016/j.ijforecast.2020.06.008`.