ES-dRNN: A Hybrid Exponential Smoothing and Dilated Recurrent Neural Network Model for Short-Term Load Forecasting

Slawek Smyl, Grzegorz Dudek, and Paweł Pełka

***Abstract*— Short-term load forecasting (STLF) is challenging due to complex time series (TS) which express three seasonal patterns and a nonlinear trend. This article proposes a novel hybrid hierarchical deep-learning (DL) model that deals with multiple seasonality and produces both point forecasts and predictive intervals (PIs). It combines exponential smoothing (ES) and a recurrent neural network (RNN). ES extracts dynamically the main components of each individual TS and enables on-the-fly deseasonalization, which is particularly useful when operating on a relatively small dataset. A multilayer RNN is equipped with a new type of dilated recurrent cell designed to efficiently model both short and long-term dependencies in TS. To improve the internal TS representation and thus the model's performance, RNN learns simultaneously both the ES parameters and the main mapping function transforming inputs into forecasts. We compare our approach against several baseline methods, including classical statistical methods and machine learning (ML) approaches, on STLF problems for 35 European countries. The empirical study clearly shows that the proposed model has high expressive power to solve nonlinear stochastic forecasting problems with TS including multiple seasonality and significant random fluctuations. In fact, it outperforms both statistical and state-of-the-art ML models in terms of accuracy.**

***Index Terms*— Deep learning (DL), exponential smoothing (ES), hybrid forecasting models, recurrent neural networks (RNNs), short-term load forecasting (STLF), time series (TS) forecasting.**

I. Introduction

ELECTRICITY demand forecasting for different horizons and granularity is an integral part of power system control, scheduling, and planning. Thus, it is extremely important for energy suppliers, system operators, financial institutions, and other participants in electric energy generation, transmission, distribution, and markets. At a short-term level, i.e., with a horizon from 1 h to seven days ahead and hourly granularity or less, electricity demand forecasting is the basis of power system operation including unit commitment, generation dispatch, hydro scheduling, hydrothermal coordination, spinning reserve allocation, interchange and low flow evaluation, security assessment, and network diagnosis [1]. Modern power systems pose new challenges for the forecasting models due to issues connected with volatile distributed energy resources, integration of intermittent renewable energy resources, and deployment of demand-side management. As electricity demand is the primary driver of electricity prices, short-term load forecasting (STLF) plays a key role in competitive energy markets. The accuracy of forecasts translates directly into the financial performance of energy market participants. A related study revealed that a 1% reduction in forecasting error for a 10 GW utility can save up to $1.6 million annually [2].

Manuscript received 29 November 2021; revised 24 November 2022; accepted 13 March 2023. *(Corresponding author: Grzegorz Dudek.)*

Slawek Smyl is with Facebook, Menlo Park, CA 94025 USA (e-mail: slawek.smyl@gmail.com).

Grzegorz Dudek and Paweł Pełka are with the Department of Electrical Engineering, Częstochowa University of Technology, 42-200 Częstochowa, Poland (e-mail: grzegorz.dudek@pcz.pl; pawel.pelka@pcz.pl).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3259149.

Digital Object Identifier 10.1109/TNNLS.2023.3259149

A. Related Work

The importance of STLF for the safe, reliable, and efficient operation of power systems as well as the complexity of the problem translates into great interest from researchers in this field. Nonlinear trends, multiple seasonality, variable variance and daily profile, and random fluctuations, make STLF challenging and place high demands on forecasting models. STLF approaches can be divided into three categories: statistical or econometric models, ML models, and hybrid ones. The first category includes auto-regressive integrated moving average (ARIMA) [3], exponential smoothing (ES) [4], linear regression [5], and Kalman filtering [6].

The main problem with many statistical STLF methods is their linear nature, which limits the implementation of nonlinear system dynamics. To extend the model's capabilities to approximate nonlinear relationships, local modeling is used. For example, in [7], the linear state-space model is learned progressively from the data using a Kalman filter. The method works by assuming temporally local linearity, which can be seen as an approximation of an underlying nonlinearity, generalizing the standard linear-Gaussian model with static parameters. In [8], the target nonlinear function was modeled locally in the neighborhood around the query pattern using linear regression. Due to initial data normalization, which simplified relationships between input and output data, linear models, such as partial least-squares regression, were able to compete with more sophisticated ML models.

Another problem with statistical methods is their limited ability to model complicated seasonal patterns. Standard ARIMA and Holt-Winters models can be extended to multiple seasonality [9] but they assume that the cycle shapes are all the same. In practice, the seasonal patterns can greatly differ from each other (in STLF, daily cycles for workdays are usually significantly different from those for weekends). Considering the changing seasonal pattern requires a significant extension of the model. For example, in [10], an ES state space model was combined with Fourier terms, a Box-Cox transformation, and ARMA error correction. Extending the regression model with Fourier terms (harmonic regression) is a popular method of introducing seasonal components into statistical models [11]. Another approach to deal with seasonality is time series (TS) decomposition. Products of decomposition are less complex than the original TS and can be modeled using simpler models [8], [12], [13].

Other drawbacks statistical methods suffer from are limited adaptability, a shortage of expressive power, problems with capturing long-term dependencies, and introducing exogenous variables into the model. ML methods offer many more possibilities than statistical ones. They provide forecasting models with the ability to learn historical patterns and anomalies and successively improve prediction accuracy. The most researched ML models in the field of forecasting are neural networks (NNs) [14]. They can flexibly model complex nonlinear relationships between variables and reflect process variability in uncertain dynamic environments due to their universal approximation property. At the same time, NNs have their limitations such as disruptive and unstable training, the need for careful feature engineering, local optimality, weak interpretability, difficulty in matching the network architecture to the problem solved, tendency to overfitting, weak extrapolation ability and many parameters to estimate. These issues as well as problems with modeling complex seasonal patterns are addressed in STLF literature in various ways. For example, in [15], TS with multiple seasonality were represented by patterns of the daily profiles, which simplified greatly the forecasting problem. As a result, it was possible to use simpler, resistant-to-overfitting neural models with a small number of parameters. Among the NN architectures compared in [15] are multilayer perceptron (MLP), radial basis function (RBF) NN, generalized regression NN (GRNN), fuzzy counterpropagation NN, and self-organizing maps. Within the group, GRNN and MLP turned out to be the most accurate. A Bayesian approach was used in [16] to control MLP complexity and to select input variables. The Bayesian framework offered ways to avoid overfitting by regularization, to decide on the number of neurons by comparing the model evidence, and to deal with the inputs by soft-pruning. Many NN solutions for STLF combine the neural model, optimization method for hyperparameter selection and learning, and TS decomposition or a feature engineering method. An example can be found in [17] where TS is decomposed using wavelet transform to extract relevant information from the load curve, and MLP weights are adjusted using a particle swarm optimization algorithm. The most popular and universal NN, MLP, was recently replaced by randomized NN in STLF [18]. When a pattern-based representation is used, randomized NN can produce more accurate forecasts than MLP while having many advantages over MLP. These include extremely fast and easy training, simple architecture, a small number of hyperparameters and parameters to estimate, and ease of implementation.

NN architectures proposed for STLF in recent years are dominated by deep learning (DL) and recurrent NNs (RNN). The success of DL can be largely attributed to increased model complexity and the ability to cross-learn on massive datasets. This strengthens expressive power and the ability to extract patterns across multiple examples. DL architectures are composed of combinations of basic structures, such as MLPs, convolutional NNs (CNNs), and RNNs. New ideas in the field of DL have been successfully applied to STLF. Some examples are: [19], where deep residual NNs were proposed and applied to probabilistic load forecasting using Monte Carlo dropout; [20], where a multivariate fuzzy TS was converted into multichannel images and processed by CNN to produce load forecasts; [21], where an improved deep belief network for STLF considering demand-side management was proposed; and [22], where an STLF problem for individual residential households was addressed using long–short term memory (LSTM) RNN. Among NN forecasting models, modern RNNs such as LSTM and gated recurrent unit (GRU) are distinguished by their ability to model both short and long-term dependencies in TS. Therefore, they are readily used for STLF [23].

To improve further forecasting model performance, ensemble, and hybrid solutions have been developed. Ensembling is a reliable approach to increasing the forecast accuracy and robustness of both statistical and ML models. It combines, in some way, multiple models to produce a common response, controlling a bias-variance-covariance tradeoff [24]. Ensemble strategies for STLF take many forms. For example, in [25], an ensemble of NNs is proposed, which are trained on the products of wavelet decomposition; in [26], an empirical mode decomposition is applied to decompose the original interval-valued STLF data, and an ensemble of LSTMs is utilized to synchronously forecast and combine the components; and in [27], a data-driven multiobjective evolutionary ensemble learning is proposed with random vector functional link NNs as base learners. Hybrid approaches combine two or more methods in a common model, taking advantage of their strengths and avoiding their drawbacks. For example, statistical methods can help in data preprocessing and reduce the overfitting of ML models. Examples of model hybridization for STLF can be found in [28] where a temporal CNN is utilized to extract hidden information and long-term temporal relationships in the input data and a boosted tree model (LightGBM) is used to predict future loads based on the extracted features, and in [29], where both LSTM and wavelet decomposition extract TS features, on which an ensemble of RBF NNs is trained. The produced forecasts are aggregated by a localized generalization error model, which optimizes the ensemble member weights.

B. Motivation and Contribution

The motivation behind this work is threefold. First, STLF is extremely important for power system operation and energy market functioning. Forecast accuracy directly translates into the safe, reliable, and efficient operation of power systems as well as improved financial performance of energy market participants. Second, STLF is a challenging problem due to complex three-component seasonality, nonlinear trend, significant stochastic component, and changing seasonal patterns. It requires a flexible forecasting model capable of capturing long-term and short-term dependencies in TS. Third, new advances in ML and DL, especially in sequential data processing, encourage their application in complex forecasting problems such as STLF. Modern RNNs can deal with multiple seasonality and long-term dependencies in TS. Hybrid solutions utilizing both statistical and DL methods improve representation learning and the exploration of hidden patterns. In this study, we extend our recent works [30] and [31], where we used a combined ES and LSTM model for forecasting TS with single and double seasonality. It is worth noting that the hybrid model proposed in [30] won the renowned M4 forecasting competition in 2018, outperforming a wide variety of state-of-the-art models. The winning model produced both the most accurate forecasts and the most precise PIs for 100 000 real-life TS. It was close to 10% more accurate than the benchmark ensemble model, which is a huge improvement [32]. This means the model has been reliably and rigorously verified on a wide range of forecasting problems. In this study, based on the main concept of the winning solution, we develop a new model specifically for an STLF problem with three seasonal patterns.

Our research contributions can be summarized as follows.

1) We propose a new dilated recurrent cell, dRNNCell, as a building block of dilated RNNs designed especially for STLF to deal with both short and long-term dependencies in TS.
2) We develop a new hybrid forecasting model for STLF combining ES and RNN. The model produces point and probabilistic forecasts in the form of PIs. It does not require initial TS decomposition and, due to its internal mechanisms such as adaptive TS preprocessing, cross-learning, and multiple dilations, can deal with complex TS expressing nonlinear trends, varying variance, and multiple seasonality. Our model is available as open-source code in the GitHub repository [33].
3) We propose a new mechanism for dynamically adjusting the smoothing coefficients used by ES. These coefficients are learned by RNN simultaneously with the main mapping function transforming inputs into forecasts to ensure optimal internal representation of TS and finally maximize the accuracy of the model.
4) We introduce a new three-component loss function based on pinball loss to optimize both the point forecasts and PIs.
5) We empirically demonstrate on real-world data for 35 European countries that the proposed hybrid model outperforms in STLF well-established statistical and state-of-the-art ML approaches.

The rest of the work is organized as follows. Section II describes the STLF data and defines the STLF problem. Section III presents the proposed forecasting model: its architecture, components, and features. The experimental framework used to evaluate the proposed model is described in Section IV. Finally, Section V concludes the work.

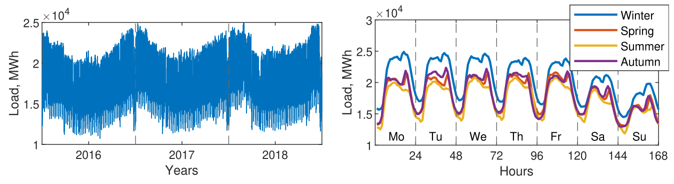

Fig. 1. Hourly electricity demand TS for Poland.

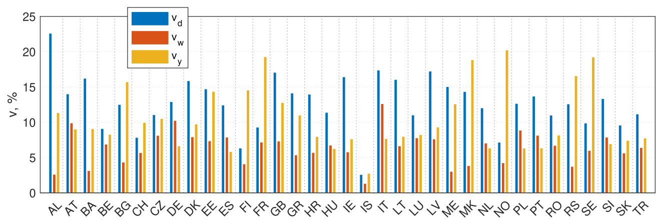

Fig. 2. Coefficients of daily, weekly, and yearly variations of electricity demand.

II. STLF Problem and Data

In this study, we consider a univariate STLF problem where the task is to forecast future values of the hourly electricity demand TS for the next day, $\{z_\tau\}_{\tau=M+1}^{M+24}$, given a sequence of past observations, $\{z_\tau\}_{\tau=1}^{M}$. The problem is challenging because the electricity demand TS exhibits a trend, three types of variability: the daily, weekly, and yearly ones, and random fluctuations—see Fig. 1, where hourly electricity demand for the Polish power system is shown. The level of electricity demand and its long-term trend depend on a country's economic development and growth rate. One of the most important factors that can upset electricity demand in the short-term perspective are extreme weather conditions.

A very important issue from the point of view of power system control and planning is electricity demand variability observed in daily, weekly, and yearly periods. Fig. 2 shows variation coefficients for 35 European countries defined as $v = 100s/\bar{z}$. For daily electricity demand variations, v_d, $\bar{z}$, and s express the daily mean and standard deviation, for weekly variations, v_w, express the weekly mean and standard deviation of daily means, and for yearly variations, v_y, express the yearly mean and standard deviation of weekly means. Greater demand variability requires greater flexibility in generating units and the entire power system. As can be seen from Fig. 2, the lowest demand variations are for Iceland. The strongest daily variations ($v_d > 17\%$) are for Albania, Italy, Latvia, and Great Britain, while the strongest yearly variations ($v_y > 18\%$) are for Norway, France, Sweden, and Macedonia. The weekly variations are usually weaker than the daily and yearly ones, $v_w < 13\%$. Countries with the strongest weekly variations are Italy, Germany, Austria, and Poland. It is worth noting that the electricity demand variation changes over time, which is an additional challenge for forecasting models.

The electricity demand seasonalities are related to local climate, weather variability, and the types of consumers. Intensities of the seasonal fluctuations can be identified using harmonic analysis. Based on Parseval's theorem, the variance of a TS can be expressed by the sum of squares of its harmonic amplitudes. The contribution of the ith harmonic to the vari-

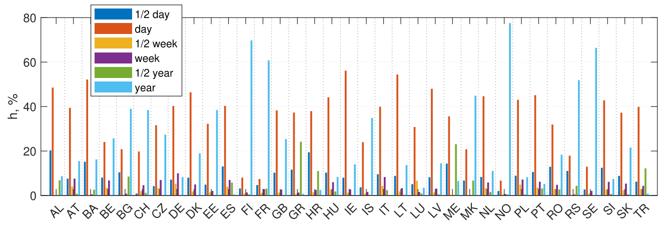

Fig. 3. Contribution of the most important harmonics in the TS variance.

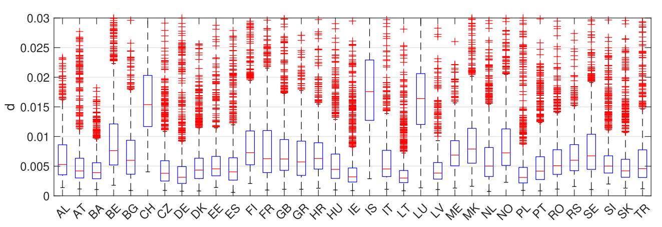

Fig. 4. Boxplots of distances between daily patterns of electricity demand.

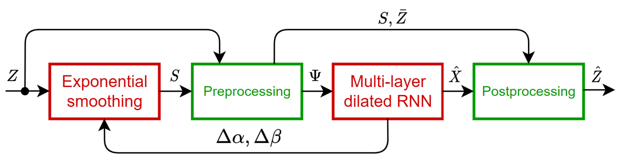

Fig. 5. Block diagram of the proposed forecasting system.

ance can be expressed by the ratio $h_i = 100A_i^2/(2\text{Var}(z_t))$, where A_i is the ith harmonic amplitude and $\text{Var}(z_t)$ is the TS variance. Fig. 3 shows ratio h_i for the most important harmonics for 35 European countries. Note that for some countries the yearly seasonality strongly dominates compared to others ($h > 60\%$). These countries include Finland, France, Norway, and Sweden. Another extremely important seasonality is the daily one. The countries with the strongest daily seasonality ($h > 50$) are Bosnia and Herzegovina, Ireland, and Lithuania. In contrast, the weekly seasonality is less distinct with h below 10% for all countries. The highest weekly fluctuations ($h > 7\%$) are shown by Germany, Italy, Austria, and Poland. Some countries demonstrate stronger half-yearly seasonality than yearly ones. They include southern European countries such as Spain, Greece, Croatia, Italy, Montenegro, and Turkey. The half-yearly seasonality is related to the tourism industry, which increases energy demand in the summer season.

As can be seen from Fig. 1, the daily patterns for Tuesday through Friday from the same period of the year are similar, while those for Monday, Saturday, and Sunday are distinct. The daily shapes are dependent on the period of the year and can vary over the years. High similarity in daily shapes makes forecasting easier. Fig. 4 shows boxplots for distances between daily patterns representing the same days of the neighboring weeks, $d_t = \|\hat{\mathbf{z}}_t - \hat{\mathbf{z}}_{t+7}\|_2$. The daily pattern is defined as the centered and normalized daily vector: $\hat{\mathbf{z}}_t = (\mathbf{z}_t - \bar{z}_t)/\|(\mathbf{z}_t - \bar{z}_t)\|_2$, where $\mathbf{z}_t = [z_{t,1}, \ldots, z_{t,24}]$ is the vector of hourly demands for the tth day and $\bar{z}_t$ is the mean demand for that day. Vectors $\hat{\mathbf{z}}_t$ have mean zero, the same variance and unity length. The daily profiles expressed by $\hat{\mathbf{z}}_t$ differ only in shape. From Fig. 4, we can observe that the most similar profiles are for Lithuania, Poland, Germany, and Ireland ($0.0030 < d < 0.0033$), while the most dissimilar ones are for Iceland, Luxembourg, and Switzerland ($0.015 < d < 0.018$).

III. Forecasting Model

A block diagram of the proposed forecasting model is shown in Fig. 5. The model is trained in a cross-learning mode [30], i.e., simultaneously on L hourly electricity load TS. Input Z represents a set of L TS: $\{\{z_\tau^l\}_{\tau=1}^{M_l}\}_{l=1}^{L}$, where M_l is an lth TS length. Output $\hat{Z}$ is a set of L forecasts of the daily sequences $\{\{\hat{z}_\tau^l\}_{\tau=M_l+1}^{M_l+24}\}_{l=1}^{L}$. An ES component expresses each TS from Z by two smoothing equations, i.e., for level and seasonality. The seasonal components, S, are used by the preprocessing component to deseasonalize the TS. This component also normalizes and squashes the TS and prepares training sets Ψ for RNN learning. It feeds the processing parameters, i.e., seasonal components S and average values of TS sequences $\bar{Z}$, to the postprocessing component. RNN produces forecasts for the deseasonalized, normalized, and squashed daily sequences and their PIs for each TS ($\hat{X}$). These forecasts are postprocessed to obtain forecasts in real values, $\hat{Z}$. RNN also produces corrections of the ES smoothing parameters, $\Delta\alpha$ and $\Delta\beta$, to tune them properly. In the ensemble version, the model is trained E times and the forecasts are averaged. The diversity of ensemble learners, which decides about ensemble learning success [24], is achieved by random initial parameters.

Details of the model are described below.

A. ES Component

As shown in Section II, the hourly electricity load TS, $\{z_\tau\}_{\tau=1}^{M}$, exhibits complex phenomena with three seasonalities. To deal with the challenging STLF problem, the TS is deseasonalized, normalized, and squashed. Then, it is predicted by RNN. Deseasonalization is performed using a seasonal component produced by a simplified Holt-Winters multiplicative seasonal model in the form

$$\begin{aligned} l_\tau &= \alpha\frac{z_\tau}{s_\tau} + (1-\alpha)l_{\tau-1} \\ s_{\tau+168} &= \beta\frac{z_\tau}{l_\tau} + (1-\beta)s_\tau \end{aligned} \tag{1}$$

where l_τ is a level component, s_τ is a weekly seasonal component, and $\alpha, \beta \in [0, 1]$ are smoothing coefficients.

The series are hourly and exhibit daily, weekly, and yearly seasonalities. The (1) includes only weekly seasonality. However, the daily seasonality (24 h values) is part of the weekly seasonality (168 h values). In addition, the series are processed in 24-h steps, so the daily seasonality is to some extend "escaped"—the RNN always learns to forecast a whole day starting from midnight. The yearly seasonality impact is dealt with partly by normalization and partly by using date-related regressors, month, and week of the year.

A unique feature of the model is that the smoothing coefficients are learned by RNN. In addition to predicting

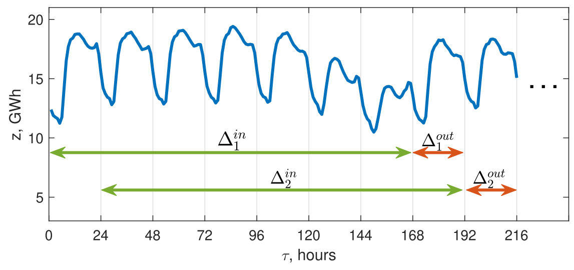

Fig. 6. Moving windows used for preprocessing TS z_τ.

the TS sequence and its PI, RNN also predicts corrections for smoothing coefficients, $\Delta\alpha_t$ and $\Delta\beta_t$. The smoothing coefficients are adapted in each recursive step t using the following corrections:

$$\alpha_{t+1} = \sigma(I\alpha + \Delta\alpha_t)$$
$$\beta_{t+1} = \sigma(I\beta + \Delta\beta_t) \quad (2)$$

where $I\alpha$ and $I\beta$ are initial values of the smoothing coefficients (hyperparameters), and σ is a sigmoid function, which maintains the coefficients within a range from 0 to 1.

The smoothing coefficients have a dynamic character. This is because the corrections produced by RNN in each recursive step depend on current and past TS characteristics (shape, level, and seasonal pattern) and time variables that indicate in what phase of the weekly, monthly, and yearly cycles the predicted daily sequence is (see RNN input pattern (6)). The dynamic Holt-Winters equations take the form

$$l_{t,\tau} = \alpha_t \frac{z_\tau}{s_{t,\tau}} + (1-\alpha_t) l_{t,\tau-1}$$
$$s_{t,\tau+168} = \beta_t \frac{z_\tau}{l_{t,\tau}} + (1-\beta_t) s_{t,\tau}. \quad (3)$$

B. Preprocessing and Postprocessing Components

To prepare input and output data for RNN we use two adjacent moving windows: input window Δ^{in} of size 168 h and output window Δ^{out} of size 24 h. The input window covers a weekly period to expose the RNN to the specific features of the series in this period directly. The output window covers the forecast daily sequence.

The windows are shifted by 24 h to obtain subsequent input and output patterns (see Fig. 6), which are defined as

$$\mathbf{x}_1^{\text{in}} = [x_1, \ldots, x_{168}], \quad \mathbf{x}_1^{\text{out}} = [x_{169}, \ldots, x_{192}]$$
$$\mathbf{x}_2^{\text{in}} = [x_{25}, \ldots, x_{192}], \quad \mathbf{x}_2^{\text{out}} = [x_{193}, \ldots, x_{216}]$$
$$\cdots \quad (4)$$

where the tth pair of patterns represent deseasonalized, normalized, and squashed TS sequences covered by the tth pair of windows

$$x_\tau = \log \frac{z_\tau}{\bar{z}_t \hat{s}_{t,\tau}} \quad (5)$$

where $\tau \in \Delta_t^{\text{in}} \cup \Delta_t^{\text{out}}$, i.e., $\tau \in [24(t-1)+1, 24(t-1)+192]$, $\bar{z}_t$ is the average TS value in the tth input window, i.e., $\bar{z}_t = 1/168 \sum_{24(t-1)+1}^{24(t-1)+168} z_\tau$, and $\hat{s}_{t,\tau}$ is the seasonal component determined using (3) for recursive step t (in this step tth pair of patterns (4) are used for RNN training).

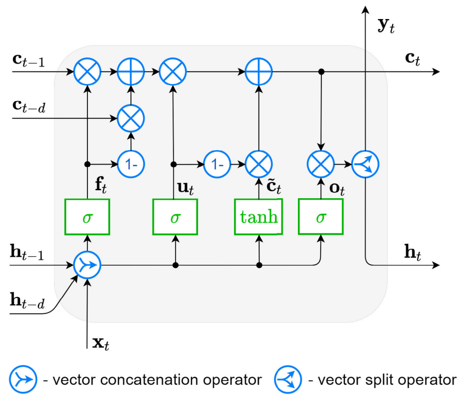

Fig. 7. Proposed dRNNCell.

TS sequences are squashed using a log function to prevent outliers from upsetting the learning process. Note that in (5), seasonal component $s_{t,\tau}$ is adapted for each tth pattern in each training epoch. Thus, the training set has a dynamic character. It is updated on-the-fly during learning. This process can be seen as the search for the optimal representation for RNN.

To introduce more input information related to the forecast sequence, the input patterns are extended as follows:

$$\mathbf{x}_t^{\text{in}'} = \left[\mathbf{x}_t^{\text{in}}, \hat{\mathbf{s}}_t, \log_{10}(\bar{z}_t), \mathbf{d}_t^w, \mathbf{d}_t^m, \mathbf{d}_t^y\right] \quad (6)$$

where $\hat{\mathbf{s}}_t$ is a vector of 24 seasonal components predicted by ES for the output period t reduced by 1, i.e., $\hat{\mathbf{s}}_t = [\hat{s}_{t,\tau} - 1]_{\tau=24(t-1)+169}^{24(t-1)+192}$, $\mathbf{d}_t^w \in \{0,1\}^7$, $\mathbf{d}_t^m \in \{0,1\}^{31}$, and $\mathbf{d}_t^y \in \{0,1\}^{52}$ are binary one-hot vectors encoding day of the week, day of the month, and week of the year for the forecast day, respectively.

Vectors $\mathbf{d}_t^w$ and $\mathbf{d}_t^y$ inform about the location of the forecast sequence in the weekly and yearly cycles, $\mathbf{d}_t^m$ helps to deal with fixed-date public holidays, $\log_{10}(\bar{z}_t)$ informs about the level of the TS (squashing function matches the level range with the range of the other components of $\mathbf{x}_t^{\text{in}'}$), and $\hat{\mathbf{s}}_t$ introduces additional information about the daily variability.

RNN is trained on training samples $(\mathbf{x}_t^{\text{in}'}, \mathbf{x}_t^{\text{out}})$ (updated in each recursive step t), and produces forecasts of the output patterns $\hat{\mathbf{x}}_t^{\text{out}} = [\hat{x}_\tau]_{\tau\in\Delta_t^{\text{out}}}$ and their quantiles defining PIs. The postprocessing component converts these forecasts to real value forecasts using transformed (5)

$$\hat{z}_\tau = \exp(\hat{x}_\tau)\bar{z}_t \hat{s}_{t,\tau}. \quad (7)$$

C. RNN Component

RNN employs a new type of gated recurrent cell, dilated RNN cell (dRNNCell), which is shown in Fig. 7. It is derived from the LSTM [34] and GRU [35] cells. It is designed to operate as part of a multilayer dilated RNN [36] and as in [37] its output is split into "real output" $\mathbf{y}_t$, which goes to the next layer, and a controlling output $\mathbf{h}_t$, which is an input to the gating mechanism in the following time steps.

The cell uses two states, c-state (also called a cell state), which is close to the standard LSTM or GRU state, and h-state, which is the controlling state (also called a hidden state). At each time step t, the whole dRNNCell input is a concatenation of $\mathbf{x}_t$, $\mathbf{h}_{t-1}$, and $\mathbf{h}_{t-d}$, where $\mathbf{x}_t$ is a standard input at time t (either from a previous layer or an input to the RNN), $\mathbf{h}_{t-1}$ is the most recent h-state, and $\mathbf{h}_{t-d}$ is the delayed state ($d > 1$). Both c- and h-states are saved in a list, to be used as delayed states. The size of the c-state is equal to the summed sizes of h-state and y-output, i.e., $s_c = s_h + s_y$.

The dRNNCell uses the following gates: fusion (f), update (u), and output (o) gates. All the gates transform nonlinearly input vectors $\mathbf{x}_t$, $\mathbf{h}_{t-1}$, and $\mathbf{h}_{t-d}$ using sigmoid function (σ). A candidate c-state, $\tilde{\mathbf{c}}_t$, is produced by transforming input vectors using tanh nonlinearity. All nonlinear transformations of the input vectors are as follows:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{V}_f \mathbf{h}_{t-1} + \mathbf{U}_f \mathbf{h}_{t-d} + \mathbf{b}_f) \tag{8}$$

$$\mathbf{u}_t = \sigma(\mathbf{W}_u \mathbf{x}_t + \mathbf{V}_u \mathbf{h}_{t-1} + \mathbf{U}_u \mathbf{h}_{t-d} + \mathbf{b}_u) \tag{9}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{V}_o \mathbf{h}_{t-1} + \mathbf{U}_o \mathbf{h}_{t-d} + \mathbf{b}_o) \tag{10}$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{V}_c \mathbf{h}_{t-1} + \mathbf{U}_c \mathbf{h}_{t-d} + \mathbf{b}_c) \tag{11}$$

where $\mathbf{W}$, $\mathbf{V}$, and $\mathbf{U}$ are weight matrices, and $\mathbf{b}$ are bias vectors.

The c-state is a weighted combination of past c-states and new candidate state $\tilde{\mathbf{c}}_t$ computed in the current step

$$\mathbf{c}_t = \mathbf{u}_t \otimes (\mathbf{f}_t \otimes \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \otimes \mathbf{c}_{t-d}) + (1 - \mathbf{u}_t) \otimes \tilde{\mathbf{c}}_t \tag{12}$$

where $\otimes$ denotes the Hadamard product (element-wise product).

Update vector $\mathbf{u}_t$ decides in what proportion the old and new information are mixed in the c-state, while fusion vector $\mathbf{f}_t$ decides about the contribution of recent and delayed c-states in the new state.

The controlling state and the output of the cell are calculated based on the new c-state and output gate as follows:

$$\mathbf{h}'_t = \mathbf{o}_t \otimes \mathbf{c}_t \tag{13}$$

$$\mathbf{y}_t = \left[h'_{t,1}, \ldots, h'_{t,s_y}\right] \tag{14}$$

$$\mathbf{h}_t = \left[h'_{t,s_y+1}, \ldots, h'_{t,s_y+s_h}\right]. \tag{15}$$

The new features of dRNNCell can be summarized as follows.

1) dRNNCells are fed by both recent states, $\mathbf{c}_{t-1}, \mathbf{h}_{t-1}$, and delayed states, $\mathbf{c}_{t-d}, \mathbf{h}_{t-d}, d > 1$. Delayed states introduce the lagged information from d steps back to facilitate seasonal and long-term modeling.
2) The cell includes two internal weighting mechanisms for c-states, which are controlled by f- and u-gates, respectively. The first weights recent and delayed states, while the second weights the past and candidate states. The purpose of this mechanism is to improve the introduction of new information into the c-state and eliminate old information from the c-state.
3) Instead of the typical one output, the cell has two outputs: $\mathbf{y}_t$, which passes the forecast information to the next layer, and $\mathbf{h}_t$, which contains information for controlling the gates in the next step. This split allows these outputs to be better adapted to their distinct functions.

The dRNNCell is part of a multilayer dilated RNN, which is composed of a number of blocks, each composed of one or more cells. In Fig. 8, there are two blocks, the first with two layers dilated 2 and 7, respectively, and the second, just with a single layer dilated 4. Dilated RNN architecture

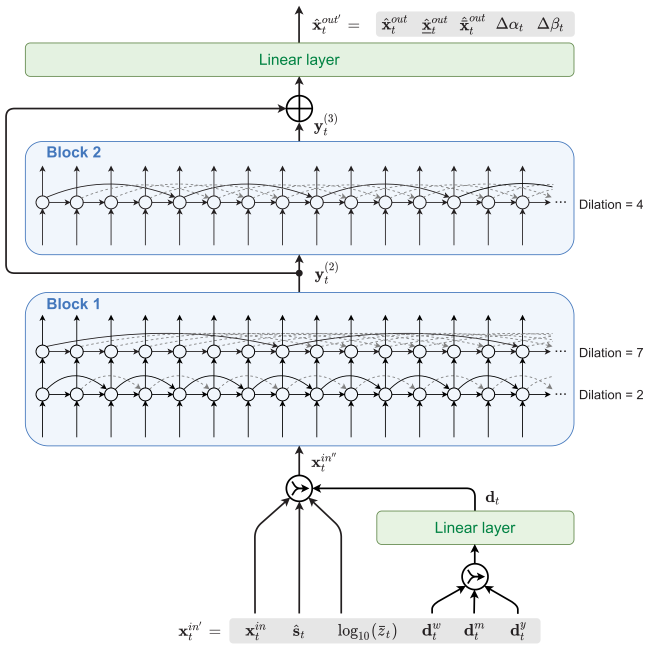
$\hat{\mathbf{x}}_t^{out'} = \hat{\mathbf{x}}_t^{out}\ \underline{\hat{\mathbf{x}}}_t^{out}\ \hat{\bar{\mathbf{x}}}_t^{out}\ \Delta\alpha_t\ \Delta\beta_t$
Linear layer
$\mathbf{y}_t^{(3)}$
Block 2
Dilation = 4
$\mathbf{y}_t^{(2)}$
Block 1
Dilation = 7
Dilation = 2
$\mathbf{x}_t^{in''}$
$\mathbf{d}_t$
Linear layer
$\mathbf{x}_t^{in'} = \mathbf{x}_t^{in}\ \hat{\mathbf{s}}_t\ \log_{10}(\bar{z}_t)\ \mathbf{d}_t^w\ \mathbf{d}_t^m\ \mathbf{d}_t^y$

Fig. 8. Proposed RNN architecture. The circles represent dRNNCells.

was introduced in [36] to tackle the three major challenges presented by RNN when learning on long sequences, i.e., complex dependencies, vanishing and exploding gradients, and efficient parallelization. Note that our new dRNNCell is fed by both recent ($t - 1$) and delayed ($t - d$) states. Thanks to this, the cell uses direct information from both the previous step and a step distant in time. This can be useful for seasonal TS, where the relationships between the series elements have a cyclical character. These relationships can be modeled more accurately using dilated connections related to seasonality. To enable RNNs to learn the temporal dependencies of different scales, we use multiple dilated recurrent layers stacked with hierarchical dilations. The proposed RNN uses ResNet-style shortcuts between blocks [38] to improve the learning process by preventing vanishing or exploding gradients.

As can be seen from Fig. 8, binary vectors encoding calendar data, $\mathbf{d}_t^w$, $\mathbf{d}_t^m$, and $\mathbf{d}_t^y$, are embedded using a linear layer into d-dimensional continuous vectors $\mathbf{d}_t$. This reduces input dimensionality and meaningfully represents sparse binary vectors in the embedding space. The embedding is learned along with the model itself.

The output layer in Fig. 8 is a linear one. It produces vector $\hat{\mathbf{x}}_t^{\text{out}'}$ which is a concatenation of the forecast output pattern, $\hat{\mathbf{x}}_t^{\text{out}} = [\hat{x}_\tau]_{\tau\in\Delta_t^{\text{out}}}$, lower bounds of PI, $\underline{\hat{\mathbf{x}}}_t^{\text{out}} = [\underline{\hat{x}}_\tau]_{\tau\in\Delta_t^{\text{out}}}$, upper bounds of PI, $\hat{\bar{\mathbf{x}}}_t^{\text{out}} = [\hat{\bar{x}}_\tau]_{\tau\in\Delta_t^{\text{out}}}$, and corrections for smoothing coefficients, $\Delta\alpha_t$ and $\Delta\beta_t$

$$\hat{\mathbf{x}}_t^{\text{out}'} = \left[\hat{\mathbf{x}}_t^{\text{out}},\ \underline{\hat{\mathbf{x}}}_t^{\text{out}},\ \hat{\bar{\mathbf{x}}}_t^{\text{out}},\ \Delta\alpha_t,\ \Delta\beta_t\right]. \tag{16}$$

D. Loss Function

To define the loss function we employ a pinball loss

$$\rho(z, \hat{z}_q) = \begin{cases} (z - \hat{z}_q)q, & \text{if } z \geq \hat{z}_q \\ (z - \hat{z}_q)(q - 1), & \text{if } z < \hat{z}_q \end{cases} \tag{17}$$

where z is an actual value, $\hat{z}_q$ is a forecast value of qth quantile, and $q \in (0, 1)$ is a quantile order.

The pinball loss is commonly used in quantile regression and probabilistic forecasting [39]. It helps us to determine the point forecasts and PIs, whose lower and upper bounds are expressed by quantiles of orders $\underline{q}$ and $\overline{q}$, respectively (e.g., $\underline{q} = 0.05$ and $\overline{q} = 0.95$).

Our loss function has the following three components:

$$L_\tau = \rho\left(z'_\tau, \hat{z}'_{q*,\tau}\right) + \gamma\left(\rho\left(z'_\tau, \hat{z}'_{\underline{q},\tau}\right) + \rho\left(z'_\tau, \hat{z}'_{\overline{q},\tau}\right)\right) \quad (18)$$

where $q^* = 0.5$ corresponds to the median, $z'_\tau = z_\tau/\bar{z}_t$ is a normalized actual TS value from the output window Δ_t^{out}, $\hat{z}'_{q*,\tau} = \exp(\hat{x}_\tau)\hat{s}_{t,\tau}$ is a forecast value of z'_τ, $\underline{q}, \overline{q}$ are the quantile orders for the lower and upper bounds of PI, respectively, $\hat{z}'_{\underline{q},\tau} = \exp(\hat{\underline{x}}_\tau)\hat{s}_{t,\tau}$ is a forecast value of $\underline{q}$-quantile of z'_τ, $\hat{z}'_{\overline{q},\tau} = \exp(\hat{\overline{x}}_\tau)\hat{s}_{t,\tau}$ is a forecast value of $\overline{q}$-quantile of z'_τ, and $\gamma \geq 0$ is a parameter controlling the impact of the components related to PI on the loss function, typically between 0.1 and 0.5.

Note that loss function (18) operates on the normalized TS values z'_τ. This is because different TS can have different levels and normalization allows us to bring their errors expressed by (17) to the same level, which is crucial in cross-learning. The forecasts of z'_τ in (18) are calculated from (7) excluding $\bar{z}_t$ to obtain normalized forecasts. These forecasts are based on the RNN outputs, $\hat{\mathbf{x}}_t^{\text{out}}, \hat{\underline{\mathbf{x}}}_t^{\text{out}}$ and $\hat{\overline{\mathbf{x}}}_t^{\text{out}}$, and ES outputs $\{\hat{s}_{t,\tau}\}_{\tau\in\Delta_t^{\text{out}}}$.

The first component in (18), $\rho(z'_\tau, \hat{z}'_{q*,\tau})$, represents a symmetrical loss for the forecast value (normalized) while the second and third components, $\rho(z'_\tau, \hat{z}'_{\underline{q},\tau})$ and $\rho(z'_\tau, \hat{z}'_{\overline{q},\tau})$, represent asymmetrical losses for the quantiles. The asymmetry level, which determines PI, results from the quantile orders. Hyperparameter γ determines the share of the three components in the loss function. For $\gamma = 1$, all the components have the same impact on the loss function. To increase the importance of the first component over the other two, we decrease the γ value. Note that due to the three-component parametrized loss function, we have the ability to optimize both the point forecasts and their PI. Moreover, the pinball loss allows us to reduce the forecast bias by penalizing positive and negative deviations differently. When the model tends to have a positive or negative bias, we can reduce the bias by introducing q^* smaller or larger than 0.5, respectively (see [30], [31]).

E. Mechanisms and Solutions for Performance Improvement

The proposed model has the following mechanisms and solutions for improved performance.

1) *dRNNCell With Expanded States (Recent and Dilated Ones):* dRNNCell is able to model both short-term and long-term dependencies in TS. This feature is useful, especially for STLF where TS expresses multiple seasonality. Temporal dependencies in this case have a cyclical character and can be modeled hierarchically using different dilations in different RNN layers.
2) *Hybrid Architecture Combining ES and RNN:* ES extracts dynamically the main components of each individual TS and enables appropriate TS representation for RNN. A multiple dilated stacked RNN architecture is able to deal with complex TS expressing multiple seasonality. The two components, ES and RNN, are optimized simultaneously by the same optimization algorithm. This fine-tunes RNN weights as well as ES smoothing coefficients. So the resulting forecasting model, including dynamic data preprocessing, is optimized as a whole.
3) *Cross-Learning:* The model is global. Learning across many TS enables it to capture the shared features and components of the TS. Cross-learning is a type of multitask learning [40] which is known to be an effective method of improving generalization by using the domain information contained in the training samples of related tasks as an inductive bias. Moreover, cross-learning greatly speeds up the learning of deep architectures.
4) *A Dynamic Training Set for RNN:* The training samples are updated on-the-fly during learning. The optimal representation of TS is searched for to ensure the best predictive performance of the model.
5) *Delaying the Moment of Over-training:* To delay the onset of the over-training, the starting point of training is sampled, so the same TS is likely to look slightly different each time a batch is formed.
6) *Three-Component, Parametrized Pinball Loss Function:* This enables the model to optimize both the point forecasts and their PIs. Moreover, it enables the forecast bias to be reduced.
7) *Ensembling:* Which is a powerful regularization technique. This exploits the beneficial effects of combining forecasts [41], improves accuracy and stability compared to a single learner, and enhances the robustness, thus mitigating model and parameter uncertainty [42].

IV. Experimental Study

In this section, we apply the proposed ES-dRNN model to STLF and compare its performance with that of other models including statistical and ML ones. We test the models on real-world data comprising hourly electricity demand TS for 35 European countries from the period 2016–2018 (source—ENTSO-E repository www.entsoe.eu/data/power-stats/; we share this data with the ES-dRNN code in our GitHub repository [33]). The TS was described and analyzed in Section II. They differ substantially in levels, trends, dispersion, and daily shapes. Thus, the data provides a variety of TS with different properties, which translates into a more reliable test for the forecasting models.

A one-day-ahead forecasting problem is considered. We optimize ES-dRNN using the data from 2016 and 2017 (data for 2016 for Albania is unavailable). The model forecasts the daily load profile for each day of 2018 for each of the 35 countries with the exception of Estonia and Italy for which data for the last month of 2018 is unavailable, and Latvia for which data for the last two months of 2018 is unavailable.

The model was implemented in Python using PyTorch. It was run on an eight-core CPU (AMD Ryzen 7 1700, 3.0 GHz, 32 GB RAM).

A. Optimization and Training Procedures

During each epoch a number of updates are executed, guided by the average error accumulated by executing l_o (e.g., 50) forward steps, moving by one day, on a batch. The starting point is chosen randomly; the batches include random b series. The model is trained using Adam optimizer.

The d-dilated dRNNCell operates as described above only after d steps, because only after d steps are the delayed states available. In addition, the Holt-Winters formulas require at least twice the seasonality steps to stabilize, so the system uses several weeks (w_o) at the beginning of each batch as a warm-up period, during which all the ES and RNN calculations take place, with the exception of the training errors, which are not calculated. Similarly, an even longer warm-up period w_s is applied when producing the test results.

An epoch is usually defined as using all the training data once. Our definition here is based on the number of updates or processed batches, as during training we step l_o times on a batch (with random assignment of series) and for each batch execute a single update based on the average error. We aim to define the epoch as the number of updates that bring in a meaningful change in the learning process, and because the dataset contains a small number of series, a single epoch is actually composed of n_o number "subepochs," defined in the traditional fashion as one scan of all available data. An additional factor is the batch size: when it grows, the number of updates per subepoch diminishes, so the number of subepochs needs to grow. However, in our experience, the linear growth is too fast, and risks overfitting within a single epoch, so finally we use the following formula:

$$n_o = \min\left(1, \left(\frac{Nb}{L}\right)^p\right) \tag{19}$$

where N is the maximum number of updates per epoch, b is the current batch size, L is the number of TS in the dataset, and p is a hyperparameter, between 0 and 1, which by experimentation is set to 0.7.

The pseudo-code for the ES-dRNN training algorithm is shown in Algorithm 1.

Algorithm 1 Pseudo-Code for the Training Algorithm

```
for $iEpoch$ in range($Num\_of\_Epochs$) do
  if Scheduled then
    Update learning rate and/or batch size
  end if
  Calculate number of sub-epochs
  for $iSubEpoch$ in range($Num\_of\_SubEpochs$) do
    Shuffle series and create $batches$
    for $batch$ in $batches$ do
      – Reset state of the RNN
      – Step through first $Input\_Window$ number of steps and update levels and seasonality, using the per-series $\alpha_t$ and $\beta_t$ (RNN not used)
      – Step through the remaining warm-up steps and training steps and update levels and seasonality, using the per-series $\alpha_t$ and $\beta_t$ adjusted by the RNN; if warm-up finished calculate the loss, average across all batch members, add the loss to per-batch list
      – Calculate average loss, update
    end for
  end for
end for
```

The model hyperparameters were selected as follows.

1) *Number of Epochs:* In early testing we established that 9 epochs is usually sufficient to reach a plateau of accuracy.
2) *Number of TS in the Batch:* We use the schedule of increasing batch sizes and decreasing learning rates proposed in [43]. We start with a small batch size of 2, and increase it, although only once, due to the small number of series, to 5 at epoch 4.
3) *Learning Rates:* Decreasing learning rates has a similar, if not the same, effect as increasing the batch size: it allows the validation error to be further reduced. We use the following schedule: $3 \cdot 10^{-3}$ (epochs 1–4), 10^{-3} (epoch 5), $3 \cdot 10^{-4}$ (epoch 6), and 10^{-4} (epochs 7–9).
4) *Size of the c-State, h-State, and y-Output:* $s_c = 100$, $s_h = 40$, $s_y = 60$. Increasing the size of cells causes a quadratic increase in the number of parameters. Larger models may be beneficial for larger datasets. The values above were obtained by experimentation starting with $s_c = 50$, $s_h = 20$, and doubling it three times.
5) *RNN Architecture and Dilations:* We use three layers: two dilated by 2 and 7 in the first block, and a single layer dilated by 4 in the second block (see Fig. 8). We arrived at this layout partly by heuristics and partly by experimentation. We started with two layers, the first with a minimum dilation of 2 (because the previous values, dilated by 1, are always used by the cells), and the second layer dilated by 7, to match them with the main weekly seasonality. Then, we tried to add a third layer, and to avoid the vanishing gradient problem it had to be in a new block. We chose dilation 4 because together with 2 and 7, this makes an almost perfect geometric series, as advocated by [36]. But of course, we also experimented with larger dilation 11 and 14 in the third layer, but the results were worse, which is as expected because our horizon is just 1 day. Among the alternative arrangements for the three layers, three blocks of a single layer each dilated 2, 4, and 7 works equally well.
6) *Loss Function Parameters:* As described in Section III-D, the pinball loss function was utilized, with three different quantile values q, to achieve quantile regression for 0.5, 0.05, and 0.95. The actual values for q^*, $\underline{q}$, and $\overline{q}$ were slightly different: 0.49, 0.035, 0.96. These values were arrived at by experimentation, reducing the bias of the center value, and fine-tuning the percentage of exceedance for PIs. The γ parameter was 0.3, set, without experimentation, following a rule of thumb that states that the sum of PIs losses should be around 10%–20% of the center value loss, reflecting the usual higher importance applied to the center loss. However, it is not a sensitive parameter, and leaving it at 1 would not make much difference.

7) *Initial Smoothing Coefficients:* $I\alpha = -3.5$, $I\beta = 0.3$. These were arrived at by observing, during early runs of the training, the direction and size of average adjustments to both smoothing coefficients, by the NN. As expected, the level is quite stable, so the smoothing coefficient tends to be close to zero, and therefore the $I\alpha$ is a relatively large negative number. Seasonality, on the other hand, is likely to change more, and this is confirmed by larger, typically above 0.5, smoothing coefficients, starting from $I\beta = 0.3$.
8) *Lengths of TS Sequences in the Optimization Mode:* $l_o = 50$. The longer the sequence, the more smooth the gradient should be, but at the same time the system may "see" the same parts of a series too often and can overtrain quickly. In addition, experimentation suggested that smaller values of 20 and 30 brought worse results.
9) *Lengths of Training w_o and Testing w_s Warm-Up Periods:* Three and 13 weeks, respectively. The training warm-up period w_o needs to be just slightly longer than two weeks, twice the seasonality size, for ES to stabilize. The number of warming-up steps should also be larger than the smallest dilation, but this is just two steps (days), so this second condition is not important here. The length of the testing warm-up period was chosen mostly due to prior experience that stepping through two to three months of typical business TS is enough for the system to fully "zero-in" on a particular series.
10) *Embedding Size of the Calendar Variables:* This value was arrived at first by the expectation that the one-hot encoded input of size $7 + 31 + 52$ should be able to be converted to an order of magnitude smaller floating point vector, and then by experimentation.
11) *Ensemble size*: $E = 100$, although a size as small as 5 is often sufficient.

B. Baseline Models

We compare our ES-dRNN in terms of accuracy with the baseline models outlined below:

1) Naive—Naive model in the form: the forecast demand profile for day i is the same as the profile for day $i-7$;
2) ARIMA—Autoregressive integrated moving average model [44];
3) ES—Exponential smoothing model [44];
4) Prophet—Modular additive regression model with nonlinear trend and seasonal components [11];
5) N-WE—Nadaraya–Watson estimator [44];
6) GRNN—General regression NN [15];
7) MLP—Perceptron with a single hidden layer and sigmoid nonlinearities [15];
8) SVM—Linear epsilon insensitive support vector machine (ϵ-SVM) [45];
9) LSTM—Long–short-term memory [46];
10) ANFIS—Adaptive neuro-fuzzy inference system [47];
11) MTGNN—Graph NN for multivariate TS forecasting [48];
12) DeepAR—Autoregressive RNN model for probabilistic forecasting [49];

TABLE I
FORECAST RESULTS

	MAPE	MdAPE	IqrAPE	RMSE	MPE	StdPE	SuccRate
Naive	5.08	4.84	3.32	704.34	-0.26	7.91	0
ARIMA	3.30	3.01	3.00	475.09	**-0.01**	5.31	0
ES	3.11	2.88	2.73	439.26	**0.01**	5.13	0
Prophet	4.53	4.32	3.03	619.39	-0.13	6.82	0
N-WE	2.49	2.28	2.30	332.49	-0.13	4.26	0
GRNN	2.48	2.28	2.27	332.91	-0.11	4.25	2.86
MLP	3.05	2.78	2.94	419.01	-0.04	5.07	0
SVM	2.55	2.29	2.52	357.24	-0.13	4.37	2.86
LSTM	2.76	2.57	2.52	381.76	0.02	4.47	0
ANFIS	3.65	3.17	3.66	507.08	-0.10	6.43	0
MTGNN	2.99	2.74	2.69	405.18	-0.47	4.85	0
DeepAR	3.42	3.25	2.95	487.14	-0.51	5.16	0
WaveNet	3.03	2.84	2.69	417.49	-0.83	4.68	0
N-BEATS	2.56	2.36	2.39	356.83	-0.04	4.29	5.71
LGBM	3.03	2.74	2.65	414.19	-0.25	4.68	0
XGB	2.96	2.69	2.62	405.91	**0.01**	4.61	0
ES-RNNe	2.38	2.18	2.26	324.11	-0.25	4.07	2.86
ES-dRNN	2.33	2.13	2.23	320.98	-0.20	3.89	0
ES-dRNNe	**2.23**	**2.02**	**2.15**	**306.94**	-0.20	**3.75**	**85.71**

13) WaveNet—Autoregressive deep-NN model combining causal filters with dilated convolutions [50];
14) N-BEATS—Deep NN with hierarchical doubly residual topology [51];
15) LGBM—Light gradient-boosting machine based on decision trees [52];
16) XGB—eXtreme gradient boosting algorithm based on decision trees [53]; and
17) ES-RNNe—Hybrid residual dilated LSTM and ES model [31].

The baseline models include classical statistical models (ARIMA and ES), new statistical models (Prophet), nonparametric pattern-based ML models (N-WE), classical ML models (MLP, GRNN, SVM, and ANFIS), new recurrent and deep-NN architectures (LSTM, MTGNN, DeepAR, WaveNet, and N-BEATS), boosted trees-based models (XGB and LGBM) and a predecessor of the proposed model (ES-RNNe).

C. Results

In this section, we report the results for our proposed model in two variants: as an individual model, denoted by ES-dRNN, and as an ensemble of E ES-dRNNs, denoted by ES-dRNNe.

Table I shows the results of forecasting averaged over all 35 countries, i.e., mean absolute percentage error (MAPE), median of APE (MdAPE), interquartile range of APE (IqrAPE), root mean square error (RMSE), mean PE (MPE), and standard deviation of PE (StdPE). MdAPE measures the average error without the influence of outliers, while RMSE is especially sensitive to outliers as a square error. MPE measures the forecast bias. Note the lowest values for MAPE, MdAPE, and RMSE for ES-dRNNe and the second lowest for ES-dRNN. Our models also produce the least dispersed predictions compared to the baseline models (IqrAPE ≤ 2.25). The models that produce the least biased forecasts are XGB, ES, ARIMA, and LSTM. Our model is equipped with a bias reduction mechanism. However, excessive MPE reduction leads to an increase in the remaining error measures.

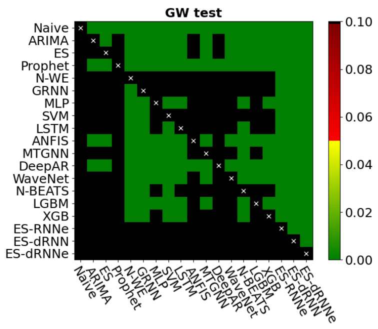

Fig. 9. Results of the GW tests.

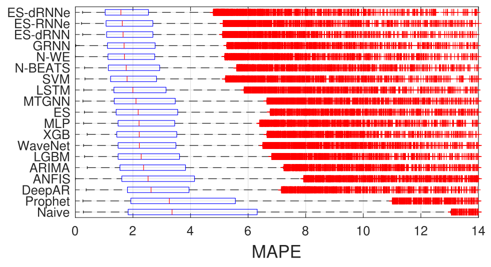

Fig. 10. Boxplots for daily MAPE.

The winning performance of ES-dRNNe and ES-dRNN was confirmed using a pairwise one-sided Giacomini-White test (GW test) for conditional predictive ability [54]. We used an implementation of the GW test in the multivariate variant from https://github.com/jeslago/epftoolbox [55]. Fig. 9 shows the results of the GW test, i.e., a heat map representing the obtained p-values. The closer they are to zero the significantly more accurate the forecasts produced by the model on the x-axis are than the forecasts produced by the model on the y-axis. The black is for p-values larger than 0.10 indicating rejection of the hypothesis that the model on the x-axis is more accurate than the model on the y-axis. Note that both ES-dRNNe and ES-dRNN performed significantly better in terms of accuracy than all the other comparative models.

More detailed results are shown in Figs. 10–12. From Fig. 10, we can assess the distribution of the daily MAPE. Note the smallest medians and the most compact distributions for our model, which is ahead of the group of nonparametric ML models designed specifically for STLF. Fig. 11 shows MAPE for individual countries. Our model was the most accurate for most of the countries apart from Estonia and Iceland, where it was beaten by N-BEATS, France, where it was beaten by SVM, Montenegro, where it was beaten by GRNN, and Czechia, where it was beaten by ES-RNNe. The model success rates based on the mean errors for the individual countries are shown in Table I (SuccRate).

Fig. 12 shows the average errors for each hour of the day, each day of the week, and each month of the test period (2018). It gives an overview of the accuracy of modeling individual seasonal components. Note that in each case errors for ES-dRNNe and ES-dRNN are among the lowest. The annual component is modeled most accurately in September and August, while the greatest errors occur in the spring months of March, April, and May. Note that our model is not very

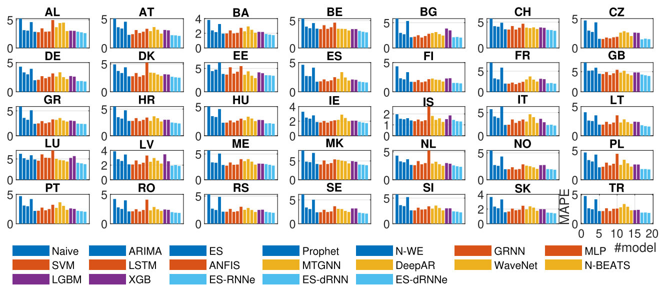

Fig. 11. MAPE of the models for each country.

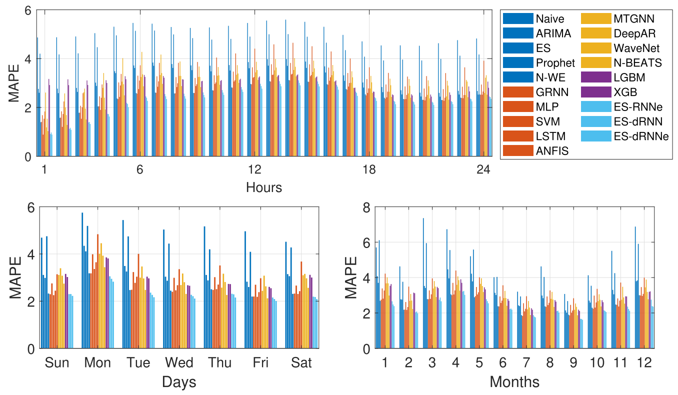

Fig. 12. MAPE for each hour of the day, each day of the week, and each month.

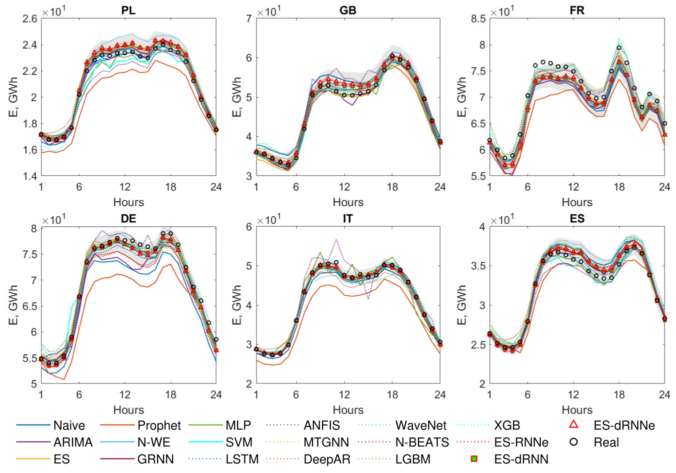

Fig. 13. Examples of the forecast daily profiles. 90% PIs for ES-dRNNe are shown as gray-shaded areas.

sensitive to the day of the week. The errors for individual days of the week are at a similar level, MAPE $\approx 2\% - 2.2\%$, except for Mondays where the errors reach 2.8%. Daily seasonality is modeled most accurately in the first hours of the day (MAPE $\approx 0.9\% - 1.3\%$), and least accurately in the period from 1 to 3 P.M., in which MAPE reaches 3%.

Fig. 13 shows some examples of the daily profile forecasts produced by our models and baseline models. From Fig. 13, we can assess the fitting of the models to the real data. In Fig. 13, the PIs predicted by ES-dRNNe are also shown. To assess the PIs, we calculate for each country the number of observed values in PIs, below PIs and above PIs. We achieved: $90.14\% \pm 2.43\%$, $4.88\% \pm 1.29\%$, and $4.98\% \pm 1.41\%$,

TABLE II
TRAINING TIME OF THE FORECASTING MODELS

Model	ARIMA	ES	Prophet	N-WE	GRNN	MLP	SVM	LSTM	ANFIS	MTGNN	DeepAR	WaveNet	N-BEATS	LGBM	XGB	ES-RNN	ES-dRNN
Time	180h	50h	7d	15s	8s	217m	27m	28h	18h	3h	42m	7h	32m	10m	40m	70m	46m

respectively. These values correspond to our assumed 90% PIs with lower and upper bounds $\underline{q} = 0.05$ and $\overline{q} = 0.95$, respectively.

Table II compares the computation time of the forecasting models for producing forecasts for each day of the test period for 35 countries. This is the total training time and query response time for each individual model. It does not include ensembling (training of many models) and optimization of the models except for ARIMA, ES, and Prophet whose implementations include automatic optimization. As can be seen from Table II, N-WE and GRNN models, which are lazy learners and do not require training, are the fastest. They are able to produce forecasts in seconds. The next fastest models are LGBM, SVM, and N-BEATS. The computation time for them was from 10 to 32 min. Our proposed model with a computation time of 46 min belongs to the third fastest group, which includes XGB and DeepAR. The computation time for the remaining models was over an hour.

D. Ablation Study

The proposed ES-dRNN has several components and mechanisms to increase its predictive power for STLF. In the ablation study, we test the performance of the reduced model. We reduce the model as follows.

Ab1 ES component is removed. dRNN learns on the normalized, but not deseasonalized TS. Seasonality vector $\hat{\mathbf{s}}_t$ is excluded from input pattern (6).

Ab2 ResNet-style shortcut between blocks 1 and 2 is removed. ES-dRNN learns without residual connection. The linear output layer is fed directly with the output vector of block 2, $\mathbf{y}_t^{(3)}$ [see Fig. (8)].

Ab3 dRNNCell without fusion gate is used. The delayed c-state is used, if available, otherwise recent, $t-1$, the state is used.

Ab4 dRNNCell without dilated states is used. It is fed with only $t-1$ states.

Ab5 dRNNCell without recent states is used. It is fed with only $t-d$ states.

Ab6 LSTM cell is used instead of dRNNCell. The cell is simplified, without delayed connections.

Ab7 Level input $\log_{10}(\bar{z}_t)$ is excluded from input pattern (6).

Ab8 Inputs $\log_{10}(\bar{z}_t)$, $\mathbf{d}_t^w$, $\mathbf{d}_t^m$, and $\mathbf{d}_t^y$ are excluded from input pattern (6). No input information about the TS level and current location in the weekly, monthly, and yearly cycles is introduced to dRNN.

Ab9 Inputs $\log_{10}(\bar{z}_t)$, $\mathbf{d}_t^w$, $\mathbf{d}_t^m$, $\mathbf{d}_t^y$, and $\hat{\mathbf{s}}_t$ are excluded from input pattern (6). dRNN is fed with only input pattern $\mathbf{x}_t^{\text{in}}$. Additional input information such as TS level, current seasonality, and calendar data is removed.

Ab10 Embedding is excluded. Extended input vector (6) is directly introduced on block 1. Input linear layer transforming calendar one-hot vectors into a continuous embedding vector is removed.

TABLE III
ERRORS FOR FULL AND REDUCED MODEL

	Full	Ab1	Ab2	Ab3	Ab4	Ab5	Ab6	Ab7	Ab8	Ab9	Ab10
MAPE	**2.227**	2.267	2.247	2.240	2.231	2.264	2.245	2.232	2.286	2.345	2.291
RMSE	**306.9**	310.2	310.5	309.1	307.0	310.9	307.2	307.9	313.1	321.8	315.4

As can be seen from Table III, the lowest errors were achieved by the full model. Any reduction in the model leads to an increase in the forecast error. The most beneficial way of improving the accuracy of the model was extending the input pattern with daily variability, TS level, and calendar variables (Ab9). A model without this extension produced 5.3% worse forecasts in terms of MAPE. The second most beneficial procedure was embedding the calendar variables using a linear layer (Ab10). This resulted in a reduction in MAPE of 2.8%. Excluding the ES component from the model (Ab1) resulted in an increase in MAPE of 1.8% while excluding the recent states from dRNNCell (Ab5) resulted in an increase in MAPE of 1.7%. Ab2, Ab3, Ab4, and Ab6 resulted in a MAPE reduction of less than 1%. Among them, excluding dilated states from dRNNCell (Ab4) reduced the error the least, by 0.18%.

E. Discussion

The experimental study proves that both variants of ES-dRNN clearly outperform all other models in terms of accuracy. The wide range of STLF problems on which we have tested the algorithms increases our confidence in this conclusion. The distinguishing feature of our model from other ML baseline models is that it produces both point forecasts and PIs with a specified probability coverage. Thus, the user gains additional information about the uncertainty of the prediction.

The proposed model is equipped with several mechanisms and solutions for performance improvement (see Section III-E). Many of them were tested and proved their effectiveness in forecasting problems from diverse domains (see the winning submission to the M4 competition [30] and the model for monthly electricity demand forecasting [31]). Other components and mechanisms were designed in this study especially for STLF to deal with complex seasonality and short and long-term dependencies. As the ablation study has shown, the most important of them turned out to be: extended input information including daily variability, TS level and the calendar variables (Ab9), embedding of the calendar data (Ab10), and ES component (Ab1). Removing these components and mechanisms worsens the results the most.

We confirmed the beneficial effect of ensembling on increasing the accuracy of the forecasting model. The errors for the ensemble version were lower than those for the individual version by 5.51% for MAPE, 6.05% for MdAPE, and 5.86%

for RMSE. In our approach, ensembling does not require additional effort related to the selection of additional hyperparameters, e.g., controlling the diversity of individual learners. The diversity of learners is provided by the random initialization of the model parameters. However, controlling diversity could be an additional way of improving performance. The individual ensemble members produced forecasts with MAPE ranging from 2.30% to 2.36%. MAPE standard deviation was 0.0145% and its ratio to mean error was 0.62%.

The proposed ES-dRNN is more complex than the baseline statistical and ML models. It has a larger number of parameters (around 229 K) and hyperparameters to tune. However, the development of the model did not require long processing times nor any special hardware—it was done on a desktop-class computer, without GPU. A single training takes less than 1 h and can be done in parallel using a number of workers, allowing the results of the ensemble to be calculated immediately. Our experience with similar models allowed us to limit the number of hyperparameter combinations and code modifications. It is also worth noting that, once this kind of model is trained, the NN weights can be saved, and a serving program that uses them can be built. Such a serving program can forecast automatically and rapidly (in a matter of seconds) all the TS when fed with new data. We worked with a relatively small dataset covering two to three years. Considering strong yearly seasonality, extensive tuning of the model, e.g., in order to completely remove bias, would likely have led to overfitting, so we purposefully avoided it.

V. Conclusion

In this article, we proposed and empirically validated a new hybrid hierarchical architecture for STLF—ES-dRNN. The empirical study of STLF for 35 European countries showed that our ES-dRNN had a significantly better performance than statistical and ML methods. It clearly outperformed its competitors in terms of accuracy. Its success is due to its unique hybrid architecture which combines ES and RNN. To deal with multiple seasonalities and short and long-term dependencies in TS we designed a new dilated recurrent cell and multiple dilated stacked RNN architecture. ES extracts dynamically the main components of each individual TS and enables appropriate TS representation for RNN. Due to the simultaneous learning of both ES and dRNN components, the model is optimized as a whole. Cross-learning, i.e., learning on multiple TS, enables ES-dRNN to capture the shared features and components of each individual TS.

The model produces point forecasts and PIs to express the forecast uncertainty. To optimize both, we introduced a new three-component, parametrized loss function. This also allows the forecast bias to be controlled. A major advantage of ES-dRNN is its ability to deal with raw TS without any kind of preprocessing such as decomposition or rationalization. All necessary data processing takes place inside the model.

The high expressive power of the proposed model to solve nonlinear stochastic forecasting problems with complex seasonalities and significant random fluctuations has encouraged us to apply it to solve other complicated forecasting problems. This will be the focus of our future work.

References

[1] G. Dudek, "Multilayer perceptron for short-term load forecasting: From global to local approach," *Neural Comput. Appl.*, vol. 32, no. 8, pp. 3695–3707, Apr. 2020.

[2] B. F. Hobbs, S. Jitprapaikulsarn, S. Konda, V. Chankong, K. A. Loparo, and D. J. Maratukulam, "Analysis of the value for unit commitment of improved load forecasts," *IEEE Trans. Power Syst.*, vol. 14, no. 4, pp. 1342–1348, Mar. 1999.

[3] S. Arora and J. W. Taylor, "Rule-based autoregressive moving average models for forecasting load on special days: A case study for France," *Eur. J. Oper. Res.*, vol. 266, pp. 259–268, Jan. 2018.

[4] J. W. Taylor, "Short-term load forecasting with exponentially weighted methods," *IEEE Trans. Power Syst.*, vol. 27, no. 1, pp. 458–464, Feb. 2012.

[5] N. Charlton and C. Singleton, "A refined parametric model for short term load forecasting," *Int. J. Forecasting*, vol. 30, no. 2, pp. 364–368, Apr. 2014.

[6] H. Takeda, Y. Tamura, and S. Sato, "Using the ensemble Kalman filter for electricity load forecasting and analysis," *Energy*, vol. 104, pp. 184–198, May 2016.

[7] S. Sharma, A. Majumdar, V. Elvira, and É. Chouzenoux, "Blind Kalman filtering for short-term load forecasting," *IEEE Trans. Power Syst.*, vol. 35, no. 6, pp. 4916–4919, Nov. 2020.

[8] G. Dudek, "Pattern-based local linear regression models for short-term load forecasting," *Electr. Power Syst. Res.*, vol. 130, pp. 139–147, Jan. 2016.

[9] J. W. Taylor, "Triple seasonal methods for short-term load forecasting," *Eur. J. Oper. Res.*, vol. 204, pp. 139–152, Jan. 2010.

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

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

[12] S. Fan and R. J. Hyndman, "Short-term load forecasting based on a semi-parametric additive model," *IEEE Trans. Power Syst.*, vol. 27, no. 1, pp. 134–141, Feb. 2012.

[13] B. A. Høverstad, A. Tidemann, H. Langseth, and P. Öztürk, "Short-term load forecasting with seasonal decomposition using evolution for parameter tuning," *IEEE Trans. Smart Grid*, vol. 6, no. 4, pp. 1904–1913, May 2015.

[14] K. Benidis et al., "Deep learning for time series forecasting: Tutorial and literature survey," 2020, *arXiv:2004.10240*.

[15] G. Dudek, "Neural networks for pattern-based short-term load forecasting: A comparative study," *Neurocomputing*, vol. 205, pp. 64–74, Sep. 2016.

[16] H. S. Hippert and J. W. Taylor, "An evaluation of Bayesian techniques for controlling model complexity and selecting inputs in a neural network for short-term load forecasting," *Neural Netw.*, vol. 23, no. 3, pp. 386–395, Apr. 2010.

[17] Z. A. Bashir and M. E. El-Hawary, "Applying wavelets to short-term load forecasting using PSO-based neural networks," *IEEE Trans. Power Syst.*, vol. 24, no. 1, pp. 20–27, Feb. 2009.

[18] G. Dudek, "Randomized neural networks for forecasting time series with multiple seasonality," in *Proc. 16th Int. Work-Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 2021, pp. 196–207.

[19] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu, and J. He, "Short-term load forecasting with deep residual networks," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3943–3952, Jul. 2019.

[20] H. J. Sadaei, P. C. de Lima e Silva, F. G. Guimarães, and M. H. Lee, "Short-term load forecasting by using a combined method of convolutional neural networks and fuzzy time series," *Energy*, vol. 175, pp. 365–377, May 2019.

[21] X. Kong, C. Li, F. Zheng, and C. Wang, "Improved deep belief network for short-term load forecasting considering demand-side management," *IEEE Trans. Power Syst.*, vol. 35, no. 2, pp. 1531–1538, Mar. 2020.

[22] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019.

[23] S. Wang, X. Wang, S. Wang, and D. Wang, "Bi-directional long short-term memory method based on attention mechanism and rolling update for short-term load forecasting," *Int. J. Electr. Power Energy Syst.*, vol. 109, pp. 470–479, Jul. 2019.

[24] G. Brown, J. L. Wyatt, and P. Tiňo, "Managing diversity in regression ensembles," *J. Mach. Learn. Res.*, vol. 6, pp. 1621–1650, Dec. 2005.

[25] M. El-Hendawi and Z. Wang, "An ensemble method of full wavelet packet transform and neural network for short term electrical load forecasting," *Electr. Power Syst. Res.*, vol. 182, May 2020, Art. no. 106265.

[26] D. Yang, J.-E. Guo, S. Sun, J. Han, and S. Wang, "An interval decomposition-ensemble approach with data-characteristic-driven reconstruction for short-term load forecasting," *Appl. Energy*, vol. 306, Jan. 2022, Art. no. 117992.

[27] Y. Hu et al., "Short-term load forecasting using multimodal evolutionary algorithm and random vector functional link network based ensemble learning," *Appl. Energy*, vol. 285, Mar. 2021, Art. no. 116415.

[28] Y. Wang et al., "Short-term load forecasting for industrial customers based on TCN-LightGBM," *IEEE Trans. Power Syst.*, vol. 36, no. 3, pp. 1984–1997, May 2021.

[29] C. S. Lai et al., "Multi-view neural network ensemble for short and mid-term load forecasting," *IEEE Trans. Power Syst.*, vol. 36, no. 4, pp. 2992–3003, Jul. 2021.

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

[31] G. Dudek, P. Pelka, and S. Smyl, "A hybrid residual dilated LSTM and exponential smoothing model for midterm electric load forecasting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 7, pp. 2879–2891, Jul. 2022.

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

[33] *ES-dRNN Code and Data*. [Online]. Available: https://github.com/slaweks17/ES-dRNN

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[35] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.

[36] S. Chang et al., "Dilated recurrent neural networks," in *Proc. NIPS*, 2017, pp. 1–15.

[37] I. Ben-Ari and R. Shwartz-Ziv, "Sequence modeling using a memory controller extension for LSTM," in *Proc. NIPS*, 2017, pp. 1–12.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[39] I. Takeuchi, Q. V. Le, T. D. Sears, and A. J. Smola, "Nonparametric quantile estimation," *J. Mach. Learn. Res.*, vol. 7, pp. 1231–1264, Jul. 2006.

[40] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, Jul. 1997.

[41] F. Chan and L. L. Pauwels, "Some theoretical results on forecast combinations," *Int. J. Forecasting*, vol. 34, no. 1, pp. 64–74, Jan. 2018.

[42] F. Petropoulos, R. J. Hyndman, and C. Bergmeir, "Exploring the sources of uncertainty: Why does bagging for time series forecasting work?" *Eur. J. Oper. Res.*, vol. 268, no. 2, pp. 545–554, Jul. 2018.

[43] S. L. Smith, P. J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," in *Proc. ICLR*, 2018, pp. 1–10.

[44] G. Dudek, "Pattern similarity-based methods for short-term load forecasting—Part 2: Models," *Appl. Soft Comput.*, vol. 36, pp. 422–441, Nov. 2015.

[45] P. Pelka, "Pattern-based forecasting of monthly electricity demand using support vector machine," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–8, doi: 10.1109/IJCNN52387.2021.9534134.

[46] P. Pelka and G. Dudek, "Pattern-based long short-term memory for mid-term electrical load forecasting," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[47] P. Pełka and G. Dudek, "Neuro-fuzzy system for medium-term electric energy demand forecasting," in *Proc. 38th Int. Conf. Inf. Syst. Archit. Technol. (ISAT)*. Cham, Switzerland: Springer, 2018, pp. 38–47.

[48] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 1–6.

[49] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *Int. J. Forecasting*, vol. 36, no. 3, pp. 1181–1191, Jul. 2020.

[50] A. van den Oord et al., "WaveNet: A generative model for raw audio," 2016, *arXiv:1609.03499*.

[51] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–17.

[52] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–5.

[53] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1–10.

[54] R. Giacomini and H. White, "Tests of conditional predictive ability," *Econometrica*, vol. 74, no. 6, pp. 1545–1578, 2006.

[55] J. Lago, G. Marcjasz, B. De Schutter, and R. Weron, "Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark," *Appl. Energy*, vol. 293, Jul. 2021, Art. no. 116983.

Slawek Smyl received the M.Sc. degree in physics from Jagiellonian University, Kraków, Poland, in 1988, the M.Eng. degree in information technology from RMIT, Melbourne, VIC, Australia, in 1997, and the Grad.D. degree in legal studies from UNSW, Sydney, NSW, Australia, in 2004.

He is currently a Quantitative Engineer at Facebook, Menlo Park, CA, USA, working in the area of time series forecasting.

Mr. Smyl has ranked highly in forecasting competitions: he won the Computational Intelligence in Forecasting International Time Series Competition in 2016, got third place in the Global Energy Forecasting Competition in 2017, and won the M4 Forecasting Competition in 2018.

Grzegorz Dudek received the Ph.D. degree in electrical engineering from the Częstochowa University of Technology (CUT), Częstochowa, Poland, in 2003, and the Habilitation degree in computer science from the Lodz University of Technology, Lodz, Poland, in 2013.

Currently, he is an Associate Professor at the Department of Electrical Engineering, CUT. He is the author of two books concerning machine learning methods for load forecasting and evolutionary algorithms for unit commitment and over 100 scientific articles. His research interests include pattern recognition, machine learning, artificial intelligence, and their application to practical classification, regression, forecasting, and optimization problems.

Dr. Dudek came third at the Global Energy Forecasting Competition in 2014 (price forecasting track).

Paweł Pełka received the Ph.D. degree in informatics from the Częstochowa University of Technology, Częstochowa, Poland, in 2021. His Ph.D. thesis was focused on pattern-based mid-term load forecasting issues.

He is currently an Assistant Professor at the Department of Electrical Engineering, Częstochowa University of Technology. He is the author of several scientific articles. His research interests cover data analysis, machine learning, pattern recognition, artificial intelligence, and their usage in time series forecasting problems.