ES-dRNN with Dynamic Attention for Short-Term Load Forecasting

Slawek Smyl
Meta
1 Hacker Way, Menlo Park, CA 94025, USA
slawek.smyl@gmail.com

Grzegorz Dudek
Department of Electrical Engineering
Czestochowa University of Technology
Czestochowa, Poland
grzegorz.dudek@pcz.pl

Paweł Pełka
Department of Electrical Engineering
Czestochowa University of Technology
Czestochowa, Poland
pawel.pelka@pcz.pl

***Abstract*—Short-term load forecasting (STLF) is a challenging problem due to the complex nature of the time series expressing multiple seasonality and varying variance. This paper proposes an extension of a hybrid forecasting model combining exponential smoothing and dilated recurrent neural network (ES-dRNN) with a mechanism for dynamic attention. We propose a new gated recurrent cell – attentive dilated recurrent cell, which implements an attention mechanism for dynamic weighting of input vector components. The most relevant components are assigned greater weights, which are subsequently dynamically fine-tuned. This attention mechanism helps the model to select input information and, along with other mechanisms implemented in ES-dRNN, such as adaptive time series processing, cross-learning, and multiple dilation, leads to a significant improvement in accuracy when compared to well-established statistical and state-of-the-art machine learning forecasting models. This was confirmed in the extensive experimental study concerning STLF for 35 European countries.**

***Index Terms*—exponential smoothing, hybrid forecasting models, multiple seasonality, recurrent neural networks, short-term load forecasting, time series forecasting**

I. Introduction

STLF is a challenging problem due to the complex nature of the time series. Electricity load or demand time series exhibit three seasonal components (annual, weekly and daily), a nonlinear trend, varying variance and random fluctuations. The daily load pattern differs in shape depending on the day of the week and season of the year. The load time series is strongly influenced by climatic, weather and economic conditions, all of which have a stochastic nature. All these properties place high demands on forecasting models. STLF is considered to be a very challenging problem, which is why it is often used for testing new models.

STLF is also very important in practice as an integral part of power system control and scheduling. It is needed for the efficient and safe operation of power systems and supporting transactions of participants in deregulated electricity markets. Due to the importance and complexity of the STLF problem, a large number of different STLF models have been reported in the literature. They employ conventional statistical methods, computational intelligence and machine learning methods as well as hybrid solutions. The most popular new approaches for STLF are based on neural networks (NNs) [1]. They offer more possibilities than statistical models and exceed many of their limitations. These limitations include their linear nature, limited ability to model complicated seasonal patterns, limited adaptability, a shortage of expressive power, and problems with capturing long-term dependencies and introducing exogenous variables.

Due to their flexibility and universal approximation property, NNs can model any nonlinear relationship and reflect process variability in an uncertain dynamic environment. Thus, they are often used for complex forecasting problems such as STLF. The classical NN architectures such as multilayer perceptron (MLP), radial basis function (RBF) NN, generalized regression neural network (GRNN), fuzzy counterpropagation NN, and self-organizing maps were compared as STLF models in [2]. Multiple seasonality, which is a real problem for the forecasting models that usually requires decomposition or deseasonalisation, was solved in this study by appropriate representation of the time series defining patterns of the daily profiles. Thanks to this, the forecasting problem was simplified and it was possible to apply less complex neural models with a smaller number of parameters, which were more resistant to overfitting. Recently, classical MLP with pattern representation was replaced by randomized NN [3]. Due to randomized learning, the training becomes much faster and easier and the numbers of parameters and hyperparameters to adjust is significantly reduced. At the same time, the accuracy of STLF improves in relation to MLP. Many STLF approaches combine neural models with effective optimization procedure and time series decomposition or a feature engineering method. For example, in [4], the time series is decomposed into orthonormal series generated by a wavelet transform, and MLP is learned using a particle swarm optimization algorithm. In [5], to control MLP complexity and to select input variables, a Bayesian approach was used. The Bayesian framework offered ways to avoid overfitting by regularisation, to decide on the number of neurons and to deal with the inputs by soft-pruning.

The rapid development of deep and recurrent NNs in recent years has led to the development of effective STLF methods. The new possibilities they offer, which are very attractive for forecasting models, include learning of representation, cross-

Partially supported by grant 020/RID/2018/19 from the Polish Minister of Science and Higher Education titled "Regional Initiative of Excellence", 2019-22 (PP, GD).

learning on massive datasets and modeling temporary relationships in sequential data. Some examples of STLF models based on deep learning are: [6], where deep residual NNs were proposed and applied to probabilistic load forecasting using Monte Carlo dropout; [7], where a multivariate fuzzy time series was converted into multi-channel images and processed by CNN to produce load forecasts; and [8], where an improved deep belief network for STLF with demand-side management was proposed.

Recurrent NNs, which were designed for sequential data such as time series and text data are extremely useful for forecasting problems. Modern RNNs such as long-short term memory (LSTM) and gated recurrent unit (GRU) are distinguished by their ability to model both short and long-term dependencies in time series [9]. Examples of RNN for STLF can be found in: [10], where a model based on bi-directional LSTM and attention mechanism was proposed; [11], where a STLF problem for individual residential households was addressed using LSTM; and [12], where the load is decomposed into different frequency components using the empirical-mode decomposition algorithm and then low-frequency components are predicted using linear regression while the high-frequency components are predicted using LSTM.

In our recent work [13], motivated by new achievements in deep learning and RNNs [14], we proposed a hybrid hierarchical STLF model combining exponential smoothing (ES) and dilated RNN (ES-dRNN). Its hybrid architecture improved ability to learn representation and explore hidden patterns. To deal with multiple seasonalities and temporal dependencies in time series, we proposed a new dilated recurrent cell (dRNNCell) and multiple dilated stacked RNN architecture. The simultaneous learning of both ES and dRNN components enables the entire model to be optimized while learning on multiple time series (cross-learning) enables ES-dRNN to capture the shared features of individual series. The model proposed in this work inherits ES-dRNN properties and extends it with a dynamic attention mechanism [15]. This mechanism allows the model to focus on relevant input information while the target functions for both point forecasts and predictive intervals are being modeled.

The contribution of this study includes the following points:

1) We propose a new attentive dilated recurrent cell, adRNNCell, which implements an attention mechanism for weighting the input information. It produces an internal attention vector which dynamically weights input vector components. This attention mechanism permits the recurrent cell to utilize the most relevant components of the input patterns in a flexible manner to improve the forecasting performance of the model.
2) We develop a hybrid exponential smoothing and dilated recurrent NN model with attention based on adRNNCell. The model produces vectors of point forecasts and also predictive intervals. Due to its internal mechanisms such as dynamic attention, adaptive time series processing, cross-learning, and multiple dilations, the model can deal with complex time series expressing multiple seasonality and varying variance. Our model is available as open-source code in the github repository [16].
3) We empirically demonstrate on real data sets of 35 European countries that our proposed model significantly outperforms in terms of STLF accuracy its predecessor [13] and 13 baseline forecasting models, including well-established statistical approaches and state-of-the-art machine learning approaches.

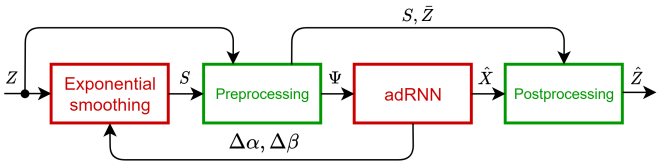

Fig. 1. Block diagram of the proposed ES-adRNN forecasting model.

The rest of the work is organized as follows. In Section II, we present the architecture of the hybrid exponential smoothing and dilated recurrent NN model with dynamic attention (ES-adRNN). Details of the proposed adRNNCell are described in Section III. The experimental framework used to evaluate the performance of the proposed approach is described in Section IV. Finally, Section V concludes the work.

II. ES-adRNN Architecture

A hybrid exponential smoothing and dilated recurrent NN model with an attention mechanism, ES-adRNN, is composed of four main components as shown in Fig. 1. This architecture is similar to that of ES-dRNN proposed in [13], except for the RNN component. In this study, we introduce RNN with a new gated recurrent cell, which is equipped with an attention mechanism. This new RNN component, adRNN, is described in details in Section III.

The proposed model works in a recursive manner, generating daily pattern forecasts for the following days in the subsequent recursive steps t. The model input, Z, represents a set of L time series: $\{\{z_\tau^l\}_{\tau=1}^{M_l}\}_{l=1}^{L}$, where M_l is an l-th time series length. In our case, the series express hourly electricity demand for L countries. The model learns simultaneously on L time series, i.e. in a cross-learning mode [14], which enables it to capture the shared features of the individual time series.

ES decomposes each series into level and seasonal components. The seasonal components, S, are used by a preprocessing component to deseasonalize the original series. The series are also normalized and squashed to prepare the training set, Ψ, for adRNN. adRNN learns not only the main mapping function transforming inputs into electricity demands for the next day (preprocessed) but also their predictive intervals (PIs) and the corrections of ES smoothing parameters, $\Delta\alpha$ and $\Delta\beta$. A postprocessing component transforms forecasts of the preprocessed demand and PIs, $\hat{X}$, into real values, $\hat{Z}$. It uses for this seasonal components, S, and average values of input sequences, $\bar{Z}$, determined by the ES and preprocessing components.

A. Exponential Smoothing

We use a simplified Holt-Winters multiplicative ES model, which decomposes the time series into two components: level component and seasonal component. A unique feature of the proposed approach is that the smoothing coefficients of the Holt-Winters model are not fixed, as in the typical case, but they are adapted in each recursive step t as follows:

$$\begin{aligned} \alpha_{t+1} &= \sigma(I\alpha + \Delta\alpha_t) \\ \beta_{t+1} &= \sigma(I\beta + \Delta\beta_t) \end{aligned} \tag{1}$$

where $\alpha, \beta \in [0, 1]$ are smoothing coefficients, $I\alpha$ and $I\beta$ are initial values of the smoothing coefficients, $\Delta\alpha_t$ and $\Delta\beta_t$ are the corrections, and σ denotes a sigmoid function.

The corrections are learned by adRNN simultaneously with the main mapping function. Thus, they have a dynamic character. They depend on the adRNN input representing the current time series characteristic and calendar variables. The dynamic version of the Holt-Winters multiplicative model takes the form:

$$\begin{aligned} l_{t,\tau} &= \alpha_t \frac{z_\tau}{s_{t,\tau}} + (1 - \alpha_t) l_{t,\tau-1} \\ s_{t,\tau+168} &= \beta_t \frac{z_\tau}{l_{t,\tau}} + (1 - \beta_t) s_{t,\tau} \end{aligned} \tag{2}$$

where $\{z_\tau\}_{\tau=1}^{M}$ is a decomposed time series, $l_{t,\tau}$ denotes a level component and $s_{t,\tau}$ denotes a weekly seasonal component.

B. Preprocessing and Postprocessing

The preprocessing component prepares training patterns for adRNN. An input pattern represents a weekly period covered by moving window Δ^{in} of size 168 hours. An output pattern represents the forecasted daily sequence covered by moving window Δ^{out} of size 24 hours, which follows Δ^{in}. The windows are shifted by 24 hours to obtain subsequent input and output patterns:

$$\begin{aligned} &\mathbf{x}_1^{in} = [x_1, ..., x_{168}], \quad \mathbf{x}_1^{out} = [x_{169}, ..., x_{192}], \\ &\mathbf{x}_2^{in} = [x_{25}, ..., x_{192}], \quad \mathbf{x}_2^{out} = [x_{193}, ..., x_{216}], \\ &... \end{aligned} \tag{3}$$

The components of the t-th pair of patterns express deseasonalized, normalized and squashed time series sequences covered by the t-th pair of windows. They are determined as follows:

$$x_\tau = \log \frac{z_\tau}{\bar{z}_t \hat{s}_{t,\tau}} \tag{4}$$

where $\tau \in \Delta_t^{in} \cup \Delta_t^{out}$, $\bar{z}_t$ is the average value in Δ_t^{in} and $\hat{s}_{t,\tau}$ is the seasonal component predicted by ES (2) for recursive step t.

The $\log$ function in (4) squashes the data to prevent outliers from adversely affecting model performance. Note that the seasonal component in (4) is adapted in each recursive step t. This makes the training data dynamic and enables the model to learn data representation.

Input patterns $\mathbf{x}_t^{in}$ are the main component of the input vectors for adRNN learning. Additionally, to introduce more information related to the forecasted period, the input vectors include: (i) a seasonal vector predicted by ES for the t-th output period reduced by 1, i.e. $\hat{\mathbf{s}}_t = [\hat{s}_{t,\tau} - 1]_{\tau=24(t-1)+169}^{24(t-1)+192}$, (ii) a current level of the time series, $\log_{10}(\bar{z}_t)$, and (iii) calendar variables, $\mathbf{d}_t^w \in \{0,1\}^7, \mathbf{d}_t^m \in \{0,1\}^{31}$ and $\mathbf{d}_t^y \in \{0,1\}^{52}$, as binary one-hot vectors encoding day of the week, day of the month and week of the year for the forecasted day, respectively. The input vector takes the form:

$$\mathbf{x}_t^{in'} = [\mathbf{x}_t^{in}, \hat{\mathbf{s}}_t, \log_{10}(\bar{z}_t), \mathbf{d}_t^w, \mathbf{d}_t^m, \mathbf{d}_t^y] \tag{5}$$

The daily patterns forecasted by adRNN, $\hat{\mathbf{x}}_t^{out} = [\hat{x}_\tau]_{\tau \in \Delta_t^{out}}$, are transformed by the postprocessing component to obtain real forecasts of the hourly electricity demand. For this, transformed equation (4) is used:

$$\hat{z}_\tau = \exp(\hat{x}_\tau) \bar{z}_t \hat{s}_{t,\tau} \tag{6}$$

where $\tau \in \Delta_t^{out}$.

C. Loss Function

The proposed model predicts hourly electricity demands, z_τ, and their PIs in the form of two quantiles of orders $\underline{q}$ (lower) and $\overline{q}$ (upper). To optimize both point forecasts and PIs, we use a loss function in the form [13]:

$$L_\tau = \rho(z'_\tau, \hat{z}'_{q^*,\tau}) + \gamma[\rho(z'_\tau, \hat{z}'_{\underline{q},\tau}) + \rho(z'_\tau, \hat{z}'_{\overline{q},\tau})] \tag{7}$$

where ρ denotes a pinball loss function:

$$\rho(z, \hat{z}_q) = \begin{cases} (z - \hat{z}_q)q & \text{if } z \geq \hat{z}_q \\ (z - \hat{z}_q)(q - 1) & \text{if } z < \hat{z}_q \end{cases} \tag{8}$$

z is an actual value; $\hat{z}_q$ is a forecasted value of q-th quantile; $q \in (0, 1)$ is a quantile order; $q^* = 0.5$ corresponds to the median; $z'_\tau = z_\tau / \bar{z}_t$ is a normalized actual time series value from the output window Δ_t^{out}; $\hat{z}'_{q^*,\tau} = \exp(\hat{x}_\tau)\hat{s}_{t,\tau}$ is a forecasted value of z'_τ; $\underline{q}$ and $\overline{q}$ are the quantile orders for the lower and upper bounds of PI, respectively; $\hat{z}'_{\underline{q},\tau} = \exp(\underline{\hat{x}}_\tau)\hat{s}_{t,\tau}$ is a forecasted value of $\underline{q}$-quantile of z'_τ; $\hat{z}'_{\overline{q},\tau} = \exp(\overline{\hat{x}}_\tau)\hat{s}_{t,\tau}$ is a forecasted value of $\overline{q}$-quantile of z'_τ; and $\gamma \geq 0$ is a controlling parameter.

Loss function (7) uses normalized values of the time series, z'_τ, to bring the errors calculated for different time series to the same level, as this is important in cross-learning.

The first component in (7), $\rho(z'_\tau, \hat{z}'_{q^*,\tau})$, is almost (see below) a symmetrical loss for the point forecast (normalized) while the second and third components, $\rho(z'_\tau, \hat{z}'_{\underline{q},\tau})$ and $\rho(z'_\tau, \hat{z}'_{\overline{q},\tau})$, are asymmetrical losses for the quantiles. The asymmetry level, which determines PI, results from the quantile orders. Parameter γ enables us to control the impact of PI-related components on the loss value. Its typical value ranges from 0.1 to 0.5.

It is worth noting that the pinball loss gives us the opportunity to reduce the forecast bias by penalizing positive and

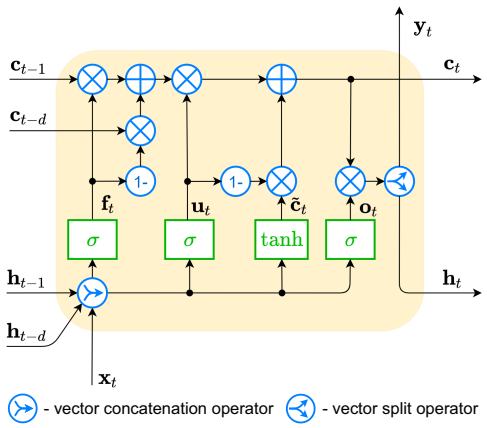

Fig. 2. Dilated recurrent cell dRNNCell.

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 [14], [17]).

III. Dilated RNN with Dynamic Attention

The RNN component of ES-adRNN employs a dilated RNN cell (dRNNCell) introduced in [13]. This cell is shown in Fig. 2. It was designed for STLF to deal with a complex seasonal pattern expressing three seasonalities. The distinguishing features of dRNNCell are:

- dRNNCell is fed by two cell states (c-states) and two controlling states (h-states). They represent recent states, $\mathbf{c}_{t-1}$, $\mathbf{h}_{t-1}$, and delayed states, $\mathbf{c}_{t-d}$, $\mathbf{h}_{t-d}$, $d > 1$.
- The output of dRNNCell 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.

In this study, we combine two dRNNCells to obtain a more efficient gated recurrent cell, which is able to preprocess dynamically the input data. A new cell, attentive dilated RNN cell (adRNNCell), introduces an attention mechanism for weighting the input information. It is shown in Fig. 3. The first cell (at the bottom of the figure) produces attention vector $\mathbf{m}_t$ of the same length as input vector $\mathbf{x}_t$. The components of $\mathbf{m}_t$, after processing by exp function, are treated as weights for the inputs collected in vector $\mathbf{x}_t$. Thus, they strengthen or weaken the particular inputs to the second cell (at the top of the figure). Note that $\mathbf{m}_t$ has a dynamical character - the weights are adjusted to the current inputs at time t. Both cells learn simultaneously. Based on preprocessed input vector, $\mathbf{x}_t^2$, the second cell produces vector $\mathbf{y}_t$, which feeds the next layer.

dRNNCell as a separate cell and as a component cell of adRNNCell, uses three gates. Namely, a fusion gate f, update gate u, and output gate o. The gates transform the input vectors, $\mathbf{x}_t$, $\mathbf{h}_{t-1}$ and $\mathbf{h}_{t-d}$, using sigmoid nonlinearity σ. A candidate c-state, $\tilde{\mathbf{c}}_t$, is produced by transforming input vectors using tanh nonlinearity. The dRNNCell formulation is as follows:

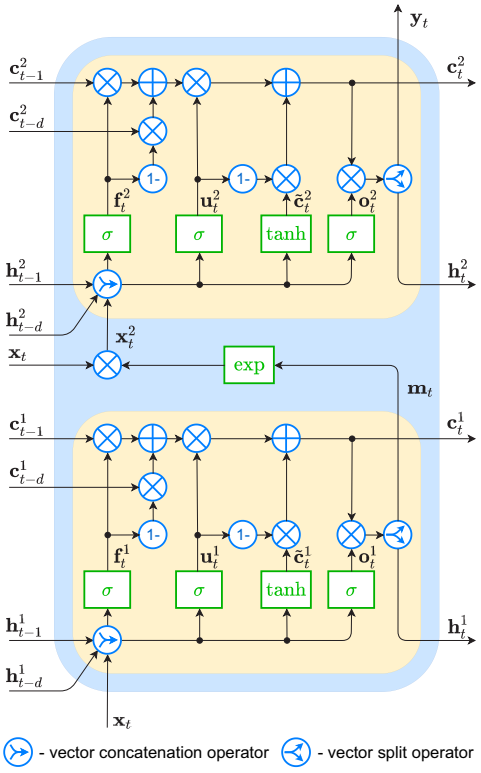

Fig. 3. Attentive dilated recurrent cell adRNNCell.

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{V}_f \mathbf{h}_{t-1} + \mathbf{U}_f \mathbf{h}_{t-d} + \mathbf{b}_f) \quad (9)$$

$$\mathbf{u}_t = \sigma(\mathbf{W}_u \mathbf{x}_t + \mathbf{V}_u \mathbf{h}_{t-1} + \mathbf{U}_u \mathbf{h}_{t-d} + \mathbf{b}_u) \quad (10)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{V}_o \mathbf{h}_{t-1} + \mathbf{U}_o \mathbf{h}_{t-d} + \mathbf{b}_o) \quad (11)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{V}_c \mathbf{h}_{t-1} + \mathbf{U}_c \mathbf{h}_{t-d} + \mathbf{b}_c) \quad (12)$$

where subscript t denotes a time step; σ denotes a logistic sigmoid function; $\mathbf{x}_t$ is an input vector; $\mathbf{h}_{t-1}$ and $\mathbf{h}_{t-d}$ are recent and delayed controlling states; $d > 1$ is a dilation; $\mathbf{W}$, $\mathbf{V}$ and $\mathbf{U}$ are weight matrices; and $\mathbf{b}$ are bias vectors.

The c-state is calculated from the recent ($\mathbf{c}_{t-1}$), delayed ($\mathbf{c}_{t-d}$) and candidate ($\tilde{\mathbf{c}}_t$) states as follows:

$$\mathbf{c}_t = \mathbf{u}_t \otimes (\mathbf{f}_t \otimes \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \otimes \mathbf{c}_{t-d}) + (1 - \mathbf{u}_t) \otimes \tilde{\mathbf{c}}_t \quad (13)$$

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

Note that the c-state is a weighted combination of past c-states and a new candidate state. Fusion vector $\mathbf{f}_t$ decides in what proportion the recent and delayed c-states are mixed, while update vector $\mathbf{u}_t$ decides about the share of old and new information in the resulting c-state.

Based on state $\mathbf{c}_t$, the output vectors $\mathbf{h}_t$ and $\mathbf{y}_t$ (or $\mathbf{m}_t$) of the cells shown in Fig. 3 are determined as follows:

$$\mathbf{h}_t^{1\prime} = \mathbf{o}_t^1 \otimes \mathbf{c}_t^1, \mathbf{m}_t = [h_{t,1}^{1\prime}, ..., h_{t,s_m}^{1\prime}], \mathbf{h}_t^1 = [h_{t,s_m+1}^{1\prime}, ..., h_{t,s_m+s_h}^{1\prime}] \quad (14)$$

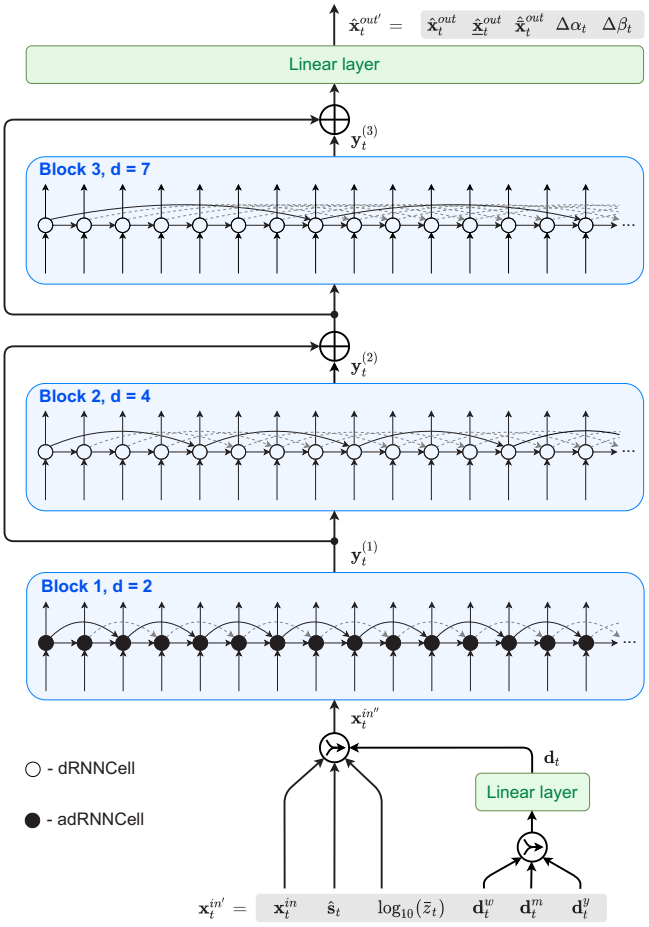

Fig. 4. adRNN architecture.

$$\mathbf{h}_t^{2\prime} = \mathbf{o}_t^2 \otimes \mathbf{c}_t^2, \mathbf{y}_t = [h_{t,1}^{2\prime}, ..., h_{t,s_y}^{2\prime}], \mathbf{h}_t^2 = [h_{t,s_y+1}^{2\prime}, ..., h_{t,s_y+s_q}^{2\prime}] \quad (15)$$

where s_y, s_h, s_m and s_q denote the lengths of vectors $\mathbf{y}_t, \mathbf{h}_t^1, \mathbf{m}_t$ and $\mathbf{h}_t^2$, respectively (in our implementation s_h and s_q are the same).

Fig. 4 shows the adRNN architecture. It is composed of three single-layer blocks. Block 1 consists of adRNNCell dilated 2, while blocks 2 and 3 consist of dRNNCells dilated 4 and 7, respectively. Note that the cells are fed by delayed states of different dilations in different layers. Delayed connections enable the direct input to the cell of information from a few time steps ago. This can be useful in modeling seasonal dependencies. In our architecture, we use multiple dilated recurrent layers stacked with hierarchical dilations to model the temporal dependencies of different scales. To facilitate the learning process, we use ResNet-style shortcuts between blocks [18].

The inputs representing the calendar variables, $\mathbf{d}_t^w$, $\mathbf{d}_t^m$ and $\mathbf{d}_t^y$, are embedded using a linear layer into d-dimensional continuous vector $\mathbf{d}_t$. 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 projects linearly vector $\mathbf{y}_t^{(3)}$ produced by the third block to output vector $\hat{\mathbf{x}}_t^{out'}$ composed as follows:

$$\hat{\mathbf{x}}_t^{out'} = [\hat{\mathbf{x}}_t^{out}, \underline{\hat{\mathbf{x}}}_t^{out}, \hat{\bar{\mathbf{x}}}_t^{out}, \Delta\alpha_t, \Delta\beta_t] \quad (16)$$

where $\hat{\mathbf{x}}_t^{out} = [\hat{x}_\tau]_{\tau \in \Delta_t^{out}}$ is a forecasted output pattern; $\underline{\hat{\mathbf{x}}}_t^{out} = [\underline{\hat{x}}_\tau]_{\tau \in \Delta_t^{out}}$ is a vector of lower bounds of PI; $\hat{\bar{\mathbf{x}}}_t^{out} = [\hat{\bar{x}}_\tau]_{\tau \in \Delta_t^{out}}$ is a vector of upper bounds of PI; $\Delta\alpha_t$ and $\Delta\beta_t$ are corrections for smoothing coefficients.

IV. Experimental Study

In this section, to evaluate ES-adRNN forecast accuracy, we consider STLF for 35 European countries. The data set includes real-world hourly electrical load time series from the period 2016-2018. The data was collected from ENTSO-E repository, www.entsoe.eu/data/power-stats/ (we share this data with the ES-adRNN code in our github repository [16]). The data provides a variety of time series with different properties such as different levels, trends, variance and daily shapes (see Section II in [13] where these time series are analysed). This great variety of time series makes the results more reliable.

We consider a one day-ahead forecasting problem: prediction of the load profile (24 hourly values) for each day of 2018 based on historical data. This was performed for each country with the exception of three countries for which data for the last month (Estonia and Italy) or the last two months (Latvia) of 2018 was unavailable. For these three countries, the test periods were shorter. The ES-adRNN model was optimized on data from the period 2016-17. We perform STLF in two variants: using ES-adRNN as an individual model and using an ensemble of five ES-adRNN base models. We denote the latter variant by ES-adRNNe. As performance metrics we use: 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).

A. Training and Optimization Setup

Our learning process, schedule of hyper-parameter changes is organized around a notion of an epoch. This 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. Our aim is to define the epoch as the number of updates which brings in a meaningful change in the learning process, (we use 2500), and because the data set contains a small number of series, a single epoch is actually composed with n_o number "sub-epochs", 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 sub-epoch diminishes, so the number of the sub-epochs needs to grow. However, in our experience the linear growth is too fast, and we use a sub-linear formula.

During each epoch a number of updates is 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 recurrent cells operate as described above only after d steps, because only after d steps are the delayed states available. Additionally, 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.

We use a schedule of increasing batch sizes and decreasing learning rates proposed in [19]. 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. We use another schedule of decreasing learning rates, which has a similar, but not exactly 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), 10^{-4} (epochs 7-9).

The sizes of c-state and h-state were $s_c = 100$, $s_h = s_q = 40$. These values were obtained by experimentation starting with $s_c = 50$, $s_h = 20$ and doubling it 3 times.

Finally, as described in Section II C, 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: 0.485, 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.

Other hyperparameters were selected as described in [13].

B. Baseline Models

As baseline models we employ:

- Naive – naive model in the form: the forecasted demand profile for day i is the same as the profile for day $i-7$
- ARIMA – autoregressive integrated moving average model [20],
- ES – exponential smoothing model [20],
- Prophet – modular additive regression model with nonlinear trend and seasonal components [21],
- FNM – fuzzy neighborhood model [20]
- GRNN – general regression NN [2],
- MLP – perceptron with a single hidden layer and sigmoid nonlinearities [2],
- SVM – linear epsilon insensitive support vector machine (ϵ-SVM) [22],
- LSTM – long short-term memory [23],
- ANFIS – adaptive neuro-fuzzy inference system [24],
- MTGNN – graph NN for multivariate time series forecasting [25],
- ES-dRNN – hybrid exponential smoothing and dilated recurrent NN model [13],
- ES-dRNNe – ensemble of five ES-dRNN base models [13].

The baseline models include classical statistical models (ARIMA, ES), new statistical models (Prophet), nonparametric pattern-based machine learning models (FNM, N-WE), classical machine learning models (GRNN, MLP, SVM, ANFIS) and new recurrent and deep NN architectures (LSTM, MTGNN). They also include the predecessor of the proposed model, i.e. ES-dRNN and its ensemble variant ES-dRNNe.

TABLE I
FORECASTING QUALITY METRICS.

	MAPE	MdAPE	IqrAPE	RMSE	MPE	StdPE
Naive	5.08	4.84	3.32	704.34	-0.26	7.91
ARIMA	3.30	3.01	3.00	475.09	**-0.01**	5.31
ES	3.11	2.88	2.73	439.26	**0.01**	5.13
Prophet	4.53	4.32	3.03	619.39	-0.13	6.82
FNM	2.50	2.30	2.29	334.08	-0.11	4.27
GRNN	2.48	2.28	2.27	332.91	-0.11	4.25
MLP	3.05	2.78	2.94	419.01	-0.04	5.07
SVM	2.55	2.29	2.52	357.24	-0.13	4.37
LSTM	2.76	2.57	2.52	381.76	0.02	4.47
ANFIS	3.65	3.17	3.66	507.08	-0.10	6.43
MTGNN	2.99	2.74	2.69	405.18	-0.47	4.85
ES-dRNN	2.33	2.13	2.23	319.04	-0.20	3.90
ES-dRNNe	2.25	2.05	2.17	309.88	-0.20	3.79
ES-adRNN	2.28	2.08	2.19	315.44	-0.16	3.82
ES-adRNNe	**2.20**	**2.01**	**2.13**	**303.70**	-0.13	**3.71**

C. Results

Table I summarizes the forecasting quality metrics averaged over the 35 countries. It is clear from this table that ES-adRNNe outperforms all other models in terms of accuracy. It shows the lowest MAPE, MdAPE and RMSE. To confirm the best performance of ES-adRNNe, we perform a pairwise one-sided Giacomini-White test (GM test) for conditional predictive ability [26] (we used the multivariate variant of the GW test implemented in https://github.com/jeslago/epftoolbox [27]). Fig. 5 demonstrates results of this test as a heat map representing the obtained p-values. The closer the p-values 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 color 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. Fig. 5 clearly show the state-of-the-art performance of ES-adRNNe and ES-adRNN. ES-adRNNe performed significantly better in terms of accuracy than all the other comparative models. ES-adRNN is second only to the ES-dRNNe.

IqrAPE and StdPE shown in Table I measure the dispersion of APE and PE, respectively. Note that ES-adRNNe produced the least dispersed forecasts. The individual version, ES-adRNN, is close behind it. MPE is a measure of the forecast bias. Note that most of the models including ES-adRNN and ES-adRNNe produced negatively biased forecasts, which means overprediction. The proposed model is equipped with a mechanism for controlling the bias included in the loss function (7), so the bias can be reduced. But reduction in bias

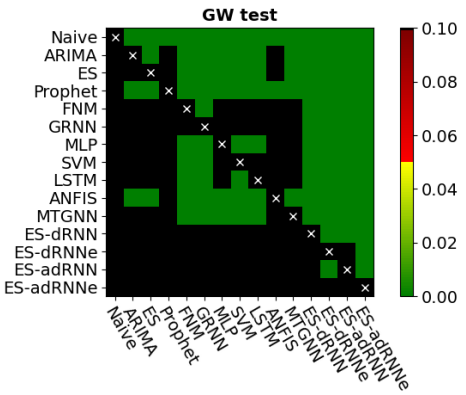

Fig. 5. Results of the Giacomini-White test.

can lead to a decrease in forecast accuracy due to overfitting, so we purposely avoided it.

Fig. 6 shows several examples of forecasted daily patterns. PIs are also shown for ES-adRNNe. To evaluate the accuracy of the PIs predicted by this model, we calculate the number of forecasts lying inside, above and below of their PIs. We achieved: $90.18\% \pm 2.86\%$ forecasts are inside PIs, $4.74\% \pm 1.47\%$ are below PIs and $5.08\% \pm 1.61\%$ are above PIs. These values correspond to the assumed PIs of 90% with lower and upper bounds $\underline{q} = 0.05$ and $\overline{q} = 0.95$, respectively.

V. Conclusion

STLF is challenging due to multiple seasonality, nonlinear trend and variable variance. To deal with this problem our model combines ES for time series preprocessing and RNN for capturing both short and long-term dependencies in time series. It is equipped with many useful mechanisms and procedures such as cross-learning on many time series, common learning procedure for ES and RNN, adjusting of ES parameters by RNN, on-the-fly deseasonalization, dilated recurrent cells, ResNet-style shortcuts, and extended input vectors.

In this work, we extend further the model by adding dynamic attention. We introduce a new type of the gated recurrent cell, adRNNCell, which implements an attention mechanism for weighting the input information. This mechanism permits the cell to utilize the most relevant components of the input patterns in a flexible manner to improve the forecasting performance of the model.

As the experimental study showed, dynamic attention significantly improves the accuracy of forecasting. The proposed ES-adRNNe outperformed statistical and machine learning models including its predecessor ES-dRNNe. Note that ensembling in our approach does not require additional effort related to the selection of additional hyperparameters such as a parameter for controlling learners' diversity. The diversity is provided by the random initialization of the models. The great advantages of ES-adRNNe are its ability to deal with raw time series, without requiring their decomposition, and its ability to produce both point forecasts and PIs. PIs with a

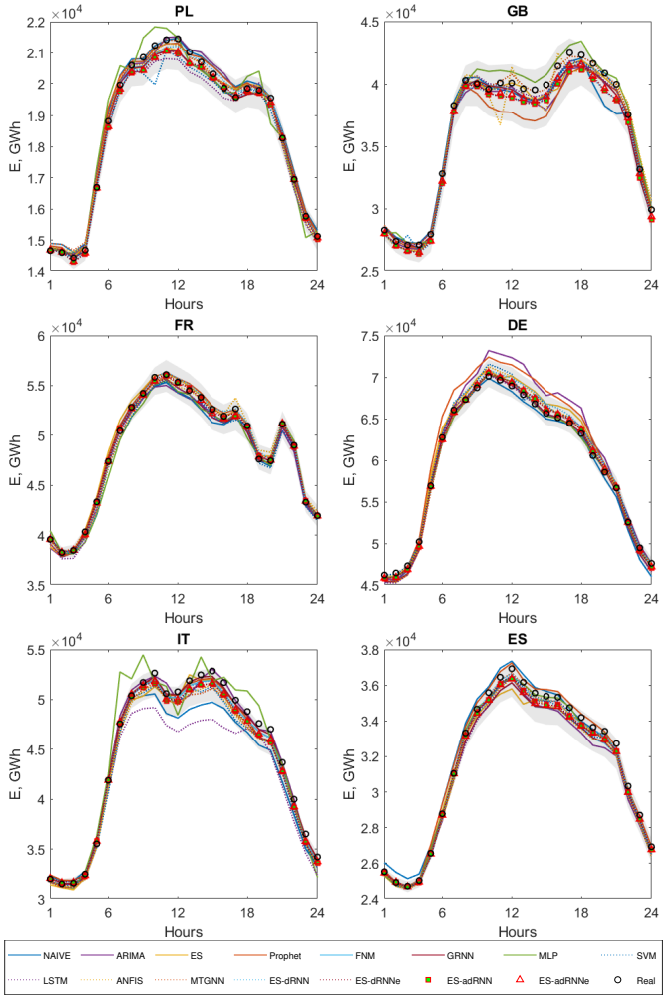

Fig. 6. Examples of the forecasted daily profiles. 90% PIs for ES-adRNNe are shown as gray-shaded areas.

specified probability coverage give very valuable information about the uncertainty of the prediction.

In further research, we plan to enrich the input information with a learned context vector. This represents information extracted from other time series, which can help predict a given time series.

References

[1] K. Benidis, S.S. Rangapuram, V. Flunkert, B. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, L. Callot, T. Januschowski, "Neural forecasting: Introduction and literature overview," *arXiv:2004.10240*, 2020.

[2] G. Dudek, "Neural Networks for Pattern-based short-term load forecasting: A comparative study," Neurocomputing, vol. 2015, pp. 64-74, 2016.

[3] G. Dudek, "Randomized neural networks for forecasting time series with multiple seasonality," in proc. *16th International Work-Conference on Artificial Neural Networks, IWANN 2021*, Springer LNCS 12862, pp. 196-207, 2021.

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

[5] H.S. Hippert, 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 Networks*, vol. 23, pp. 386-395, 2010.

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

[7] H.J. Sadaei, P.C. de Lima e Silva, F.G. Guimarães, 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, 2019.

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

[9] H. Hewamalage, C. Bergmeir and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1 pp. 388-427, 2021.

[10] 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," *International Journal of Electrical Power & Energy Systems*, vol. 109, pp. 470-479, 2019.

[11] 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 Transactions on Smart Grid*, vol. 10, no. 1, pp. 841-851, 2019.

[12] J. Li et al., "A Novel Hybrid Short-Term Load Forecasting Method of Smart Grid Using MLR and LSTM Neural Network," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2443-2452, 2021.

[13] S. Smyl, G. Dudek, P. Pełka, "ES-dRNN: A Hybrid Exponential Smoothing and Dilated Recurrent Neural Network Model for Short-Term Load Forecasting," *arXiv:2112.02663*, 2021.

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

[15] A. Vaswani, N Shazeer, N Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems, NIPS*, 2017.

[16] ES-adRNN code and data: https://github.com/slaweks17/ES-adRNN.

[17] G. Dudek, P. Pełka and S. Smyl, "A hybrid residual dilated LSTM and exponential smoothing model for midterm electric load forecasting," *IEEE Transactions on Neural Networks and Learning Systems*, doi:10.1109/TNNLS.2020.3046629.

[18] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in proc. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 770-778, 2016.

[19] S.L. Smith, P.J. Kindermans, C. Ying, Q.V. Le, "Don't decay the learning rate, increase the batch size," in *Proc. ICLR*, 2018. Available at https://openreview.net/pdf?id=B1Yy1BxCZ

[20] G. Dudek, "Pattern similarity-based methods for short-term load forecasting – part 2: Models," *Applied Soft Computing*, vol. 36, pp. 422-441, 2015.

[21] S.J. Taylor, B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37-45, 2018.

[22] P. Pełka, "Pattern-based forecasting of monthly electricity demand using support vector machine," in proc. *International Joint Conference on Neural Networks, IJCNN 2021*, pp. 1-8, 2021.

[23] P. Pełka, G. Dudek, "Pattern-based long short-term memory for mid-term electrical load forecasting," in proc. *International Joint Conference on Neural Networks, IJCNN 2020*, pp. 1-8, 2020.

[24] P. Pełka, G. Dudek, "Neuro-fuzzy system for medium-term electric energy demand forecasting," in proc. *38th International Conference on Information Systems Architecture and Technology, ISAT 2018*, Springer AISC 655, pp. 38-47, 2018.

[25] 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 International Conference on Knowledge Discovery & Data Mining, ACM SIGKDD*, 2020.

[26] R. Giacomini, H. White, "Tests of conditional predictive ability," *Econometrica*, vol. 74(6), pp. 1545-1578, 2006.

[27] J. Lago, G. Marcjasz, B. De Schutter, R. Weron, "Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark," *Applied Energy*, vol. 293, pp. 116983, 2021.