Recurrent Neural Networks for Forecasting Time Series with Multiple Seasonality: A Comparative Study

Grzegorz Dudek[1)], Slawek Smyl[2)], Paweł Pełka[1)]

grzegorz.dudek@pcz.pl, slawek.smyl@gmail.com, pawel.pelka@pcz.pl

1) POLITECHNIKA Czestochowa University of Technology [-e> Faculty of Electrical Engineering

2) Meta

Outline

1. Motivation and contribution
2. Forecasting problem
3. Data preprocessing
4. Recurrent cells
5. RNN architecture
6. Loss function
7. Experimental study
8. Conclusion

Motivation

- Time series (TS) forecasting is a challenging problem – TS expressing different phenomena and processes may include a nonlinear trend, multiple seasonal patterns and random fluctuations
- Forecasting problems need sophisticated specialized models, which are able to capture temporal relationships
- Modern recurrent neural networks with gating mechanism are particularly suitable for modeling sequential data

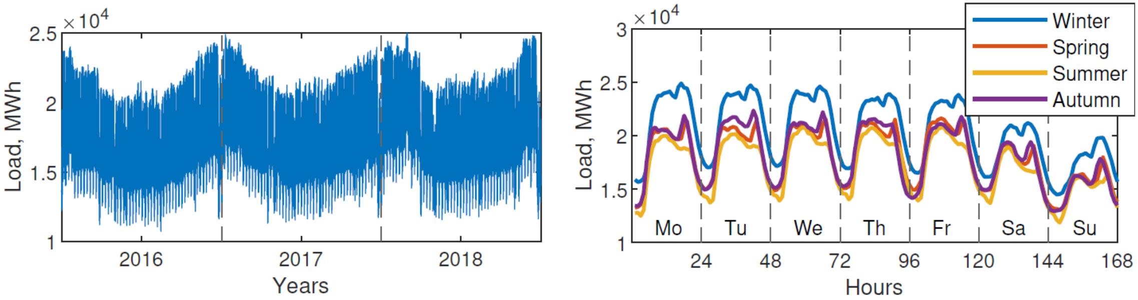
Contribution

1. We propose a new RNN architecture for forecasting TS with multiple seasonality. It is composed of three dilated recurrent layers stacked with hierarchical dilations to deal with multiple seasonality. It uses a combined asymmetrical loss function which enables the model to produce both point forecasts and predictive intervals and also to reduce the forecast bias.

2. We compare five types of gated recurrent cells: classical LSTM and GRU, modified LSTM with dilation, and two new cells we proposed recently, which are equipped with dilation and attention mechanisms.

3. We empirically demonstrate on real data for the electricity demand for 35 European countries that our proposed model copes successfully with complex seasonality. The new attentive dilated recurrent cell significantly outperforms its competitors in terms of accuracy.

- We formulate the forecasting problem given a length *n* forecast horizon and a length *M* observed time series, $\{z_\tau\}_{\tau=1}^{M}$.
- We focus on the problem of short-term electrical load forecasting (STLF) expressing triple seasonality: yearly, weekly and daily.
- Our goal is to forecast the daily profile (24 hours) for the next day based on historical loads.



Hourly electrical load time series for Poland

Input vector $\mathbf{x}'_t$ is composed of five elements:

- standardized weekly profile, which precedes the forecasted day:

$$\mathbf{x}_t = \frac{\mathbf{z}_t^w - \bar{z}_t^w}{\mathrm{std}(z_t^w)}$$

  where $\mathbf{z}_t^w \in \mathbb{R}^{168}$ is the original sequence of the t-th week, $\mathbf{x}_t \in \mathbb{R}^{168}$ is its standardized pattern,

- $\log_{10}(\bar{z}_t^w)$, which informs about the level of the time series,

- $\mathbf{d}_t^w \in \{0,1\}^7$, $\mathbf{d}_t^m \in \{0,1\}^{31}$ and $\mathbf{d}_t^y \in \{0,1\}^{52}$, which are binary one-hot vectors encoding day of the week, day of the month and week of the year.

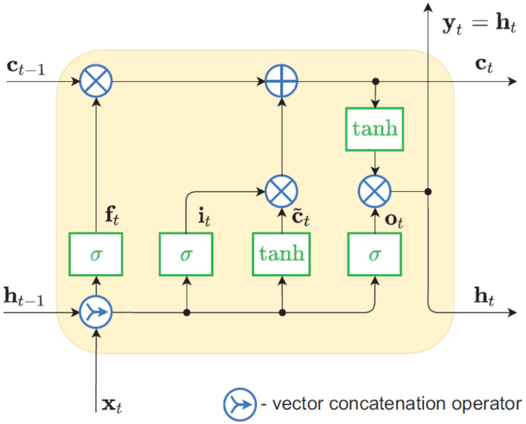
An output pattern encodes the forecasted daily sequence:

$$\mathbf{y}_t = \frac{\mathbf{z}_t^d - \bar{z}_t^w}{\mathrm{std}(z_t^w)}$$

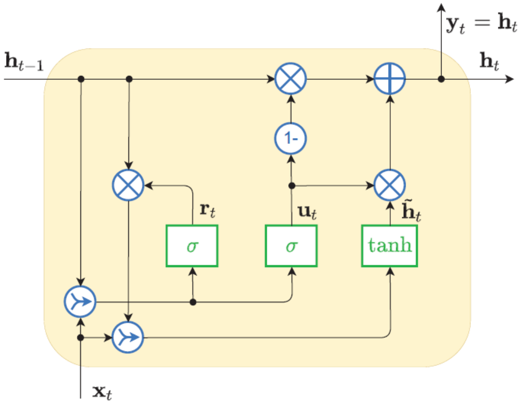
where $\mathbf{y}_t \in \mathbb{R}^{24}$ is the t-th daily pattern and $\mathbf{z}_t^d \in \mathbb{R}^{24}$ is the forecasted sequence.

Decoding:

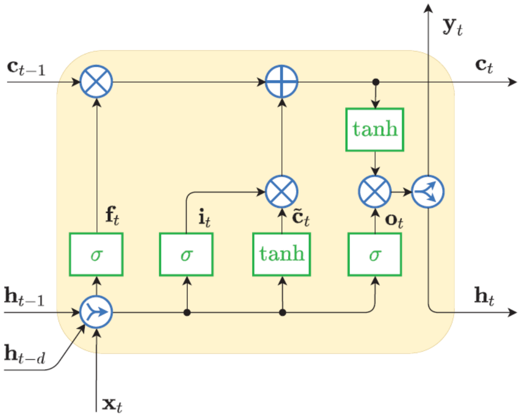
$$\mathbf{z}_t^d = \mathbf{y}_t \mathrm{std}(z_t^w) + \bar{z}_t^w$$



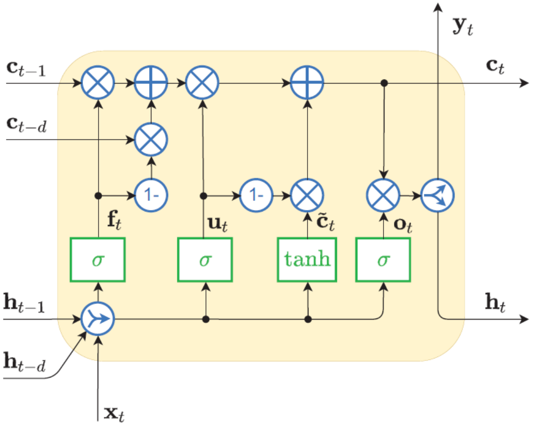
$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{V}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{V}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{V}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{V}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \otimes \tanh(\mathbf{c}_t) \\
\mathbf{y}_t &= \mathbf{h}_t
\end{aligned}
$$



$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{V}_r \mathbf{h}_{t-1} + \mathbf{b}_r)$$
$$\mathbf{u}_t = \sigma(\mathbf{W}_u \mathbf{x}_t + \mathbf{V}_u \mathbf{h}_{t-1} + \mathbf{b}_u)$$
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{V}_h (\mathbf{r}_t \otimes \mathbf{h}_{t-1}) + \mathbf{b}_h)$$
$$\mathbf{h}_t = (1 - \mathbf{u}_t) \otimes \mathbf{h}_{t-1} + \mathbf{u}_t \otimes \tilde{\mathbf{h}}_t$$
$$\mathbf{y}_t = \mathbf{h}_t$$



$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{V}_f\mathbf{h}_{t-1} + \mathbf{U}_f\mathbf{h}_{t-d} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{V}_i\mathbf{h}_{t-1} + \mathbf{U}_i\mathbf{h}_{t-d} + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{V}_o\mathbf{h}_{t-1} + \mathbf{U}_o\mathbf{h}_{t-d} + \mathbf{b}_o) \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{V}_c\mathbf{h}_{t-1} + \mathbf{U}_c\mathbf{h}_{t-d} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \tilde{\mathbf{c}}_t \\
\mathbf{h}'_t &= \mathbf{o}_t \otimes \tanh(\mathbf{c}_t) \\
\mathbf{h}_t &= [h'_{t,1}, \ldots, h'_{t,s_h}] \\
\mathbf{y}_t &= [h'_{t,s_h+1}, \ldots, h'_{t,s_h+s_y}]
\end{aligned}
$$

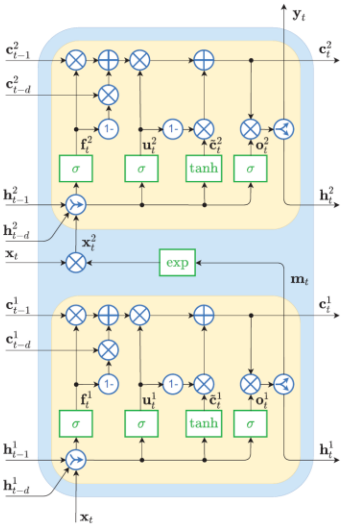


$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{V}_f\mathbf{h}_{t-1} + \mathbf{U}_f\mathbf{h}_{t-d} + \mathbf{b}_f) \\
\mathbf{u}_t &= \sigma(\mathbf{W}_u\mathbf{x}_t + \mathbf{V}_u\mathbf{h}_{t-1} + \mathbf{U}_u\mathbf{h}_{t-d} + \mathbf{b}_u) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{V}_o\mathbf{h}_{t-1} + \mathbf{U}_o\mathbf{h}_{t-d} + \mathbf{b}_o) \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{V}_c\mathbf{h}_{t-1} + \mathbf{U}_c\mathbf{h}_{t-d} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{u}_t \otimes (\mathbf{f}_t \otimes \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \otimes \mathbf{c}_{t-d}) \\
&\quad + (1 - \mathbf{u}_t) \otimes \tilde{\mathbf{c}}_t \\
\mathbf{h}'_t &= \mathbf{o}_t \otimes \mathbf{c}_t \\
\mathbf{h}_t &= [h'_{t,1}, \ldots, h'_{t,s_h}] \\
\mathbf{y}_t &= [h'_{t,s_h+1}, \ldots, h'_{t,s_h+s_y}]
\end{aligned}
$$

- adRNNCell combines two dRNNCells to obtain a more efficient cell, which is able to preprocess dynamically the input data.

- The bottom dRNNCell produces attention vector $\mathbf{m}_t$, whose components weight the inputs to the upper dRNNCell.

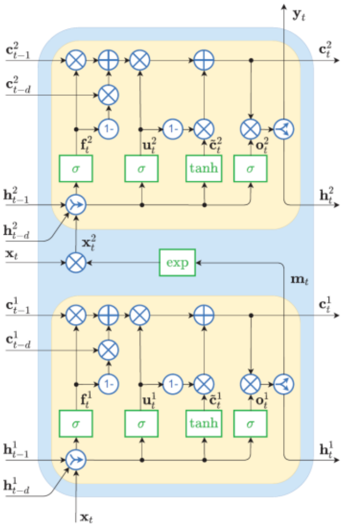
- Vector $\mathbf{m}_t$ has a dynamical character. It is adjusted to the current inputs at time t.



$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{V}_f\mathbf{h}_{t-1} + \mathbf{U}_f\mathbf{h}_{t-d} + \mathbf{b}_f)$$
$$\mathbf{u}_t = \sigma(\mathbf{W}_u\mathbf{x}_t + \mathbf{V}_u\mathbf{h}_{t-1} + \mathbf{U}_u\mathbf{h}_{t-d} + \mathbf{b}_u)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{V}_o\mathbf{h}_{t-1} + \mathbf{U}_o\mathbf{h}_{t-d} + \mathbf{b}_o)$$
$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{V}_c\mathbf{h}_{t-1} + \mathbf{U}_c\mathbf{h}_{t-d} + \mathbf{b}_c)$$
$$\mathbf{c}_t = \mathbf{u}_t \otimes (\mathbf{f}_t \otimes \mathbf{c}_{t-1} + (1-\mathbf{f}_t) \otimes \mathbf{c}_{t-d}) + (1-\mathbf{u}_t) \otimes \tilde{\mathbf{c}}_t$$
$$\mathbf{x}^2_t = \mathbf{x}_t \otimes \mathbf{m}_t$$
$$\mathbf{h}^{1\prime}_t = \mathbf{o}^1_t \otimes \mathbf{c}^1_t$$
$$\mathbf{h}^1_t = [h^{1\prime}_{t,1}, \ldots, h^{1\prime}_{t,s_h}]$$
$$\mathbf{m}_t = [h^{1\prime}_{t,s_h+1}, \ldots, h^{1\prime}_{t,s_h+s_m}]$$
$$\mathbf{h}^{2\prime}_t = \mathbf{o}^2_t \otimes \mathbf{c}^2_t$$
$$\mathbf{h}^2_t = [h^{2\prime}_{t,1}, \ldots, h^{2\prime}_{t,s_q}]$$
$$\mathbf{y}_t = [h^{2\prime}_{t,s_q+1}, \ldots, h^{2\prime}_{t,s_q+s_y}]$$

- Both dRNNCells are fed by recent states, $\mathbf{c}_{t-1}, \mathbf{h}_{t-1}$, and delayed states, $\mathbf{c}_{t-d}, \mathbf{h}_{t-d}, d > 1$.

- The outputs of dRNNCells are split into "real output", $\mathbf{m}_t$ or $\mathbf{y}_t$, and a controlling output $\mathbf{h}_t$, which is an input for the gating mechanism in the following time steps.

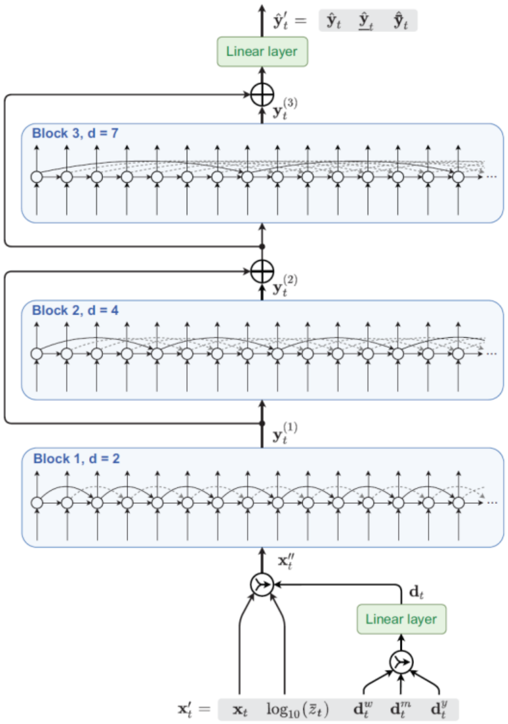


$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{V}_f\mathbf{h}_{t-1} + \mathbf{U}_f\mathbf{h}_{t-d} + \mathbf{b}_f) \\
\mathbf{u}_t &= \sigma(\mathbf{W}_u\mathbf{x}_t + \mathbf{V}_u\mathbf{h}_{t-1} + \mathbf{U}_u\mathbf{h}_{t-d} + \mathbf{b}_u) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{V}_o\mathbf{h}_{t-1} + \mathbf{U}_o\mathbf{h}_{t-d} + \mathbf{b}_o) \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{V}_c\mathbf{h}_{t-1} + \mathbf{U}_c\mathbf{h}_{t-d} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{u}_t \otimes (\mathbf{f}_t \otimes \mathbf{c}_{t-1} + (1-\mathbf{f}_t) \otimes \mathbf{c}_{t-d}) \\
&\quad + (1-\mathbf{u}_t) \otimes \tilde{\mathbf{c}}_t \\
\mathbf{x}^2_t &= \mathbf{x}_t \otimes \mathbf{m}_t \\
\mathbf{h}^{1\prime}_t &= \mathbf{o}^1_t \otimes \mathbf{c}^1_t \\
\mathbf{h}^1_t &= [h^{1\prime}_{t,1}, \ldots, h^{1\prime}_{t,s_h}] \\
\mathbf{m}_t &= [h^{1\prime}_{t,s_h+1}, \ldots, h^{1\prime}_{t,s_h+s_m}] \\
\mathbf{h}^{2\prime}_t &= \mathbf{o}^2_t \otimes \mathbf{c}^2_t \\
\mathbf{h}^2_t &= [h^{2\prime}_{t,1}, \ldots, h^{2\prime}_{t,s_q}] \\
\mathbf{y}_t &= [h^{2\prime}_{t,s_q+1}, \ldots, h^{2\prime}_{t,s_q+s_y}]
\end{aligned}
$$

- The input linear layer reduces the dimensionality of the calendar variables by embedding them in r-dimensional continuous vector $\mathbf{d}_t$.

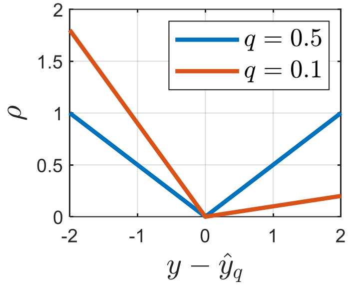
- The output linear layer produces the point forecasts, $\hat{\mathbf{y}}_t$, and two vectors of quantiles, a lower one, $\hat{\mathbf{y}}_t^l$, and an upper one, $\hat{\mathbf{y}}_t^u$. These quantiles of assumed orders, q^l and q^u, define the PI.

- RNN uses ResNet-style shortcuts between blocks to improve the learning process.



To enable RNN to learn both point forecasts and PI quantiles, we employ the following loss function:

$$L = \rho\left(y, \hat{y}_{q^*}\right) + \gamma\left(\rho\left(y, \hat{y}_{q^l}\right) + \rho\left(y, \hat{y}_{q^u}\right)\right)$$



where

$$\rho\left(y, \hat{y}_q\right) = (y - \hat{y}_q)(q - \mathbf{1}_{(y<\hat{y}_q)})$$

is a pinball loss, $q \in (0,1)$ is a quantile order, y is an actual value (standardized), $\hat{y}_q$ is a forecasted value of q-th quantile of y, $q^* = 0.5$ corresponds to the median, $q^l \in (0, q^*)$ and $q^u \in (q^*, 1)$ correspond to the lower and upper bound of PI, respectively, and $\gamma \geq 0$ is a control parameter.

We compare the performance of the proposed RNN with different recurrent cells on the short-term load forecasting problem for 35 European countries.

The data period is from 2006 to 2018 but a large amount of data is missing in this period (about 60% of the countries have complete data). The RNN was optimized on data from the period 2006-17 and tested on data from 2018.

The key hyperparameters used in the simulation study:
$s_c = 250$, $s_h = s_q = s_y = 125$, $q^* = 0.5$, $q^l = 0.05$, $q^u = 0.95$, $\gamma = 0.3$,
number of epochs: 10, learning rates: $3 \cdot 10^{-3}$ (epochs 1-5), 10^{-3} (epoch 6), $3 \cdot 10^{-4}$ (epoch 7), $3 \cdot 10^{-4}$ (epochs 8-10), batch size: 2 (epochs 1-3), 5 (epochs 4-10)

Table 1. Forecasting quality metrics.

Cell type	MAPE	MdAPE	IqrAPE	RMSE	MPE	StdPE
GRU1	2.31	2.10	2.23	318.69	**-0.06**	3.86
GRU2	2.26	2.04	2.19	308.92	-0.15	3.78
LSTM1	2.25	2.03	2.18	307.09	-0.19	3.78
LSTM2	2.16	1.94	2.10	293.00	-0.10	3.60
dLSTM	2.19	1.97	2.12	297.58	-0.19	3.66
dRNNCell	2.15	1.93	2.09	292.60	-0.15	3.57
adRNNCell	**2.12**	**1.91**	**2.07**	**289.32**	-0.14	**3.52**

LSTM1, GRU1 – the delayed connections are removed and cells are fed with only recent inputs $t-1$

LSTM2, GRU2 – the recent connections are removed and cells are fed with only delayed inputs, $t-2$, $t-4$ or $t-7$, depending in the layer.

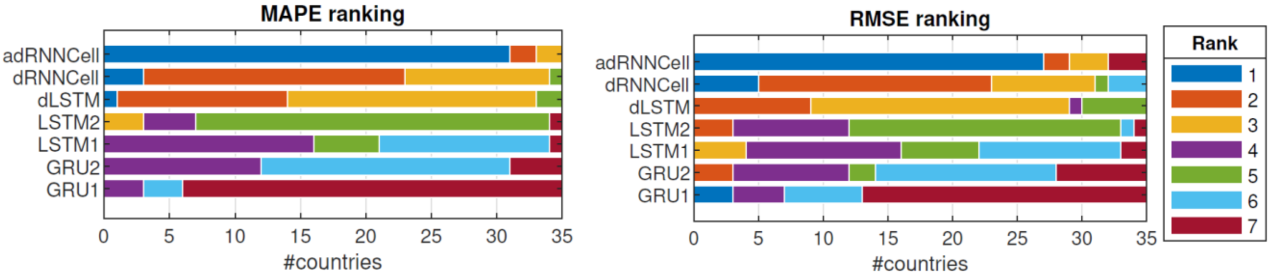


Fig. 8. Results of MAPE and RMSE rankings.

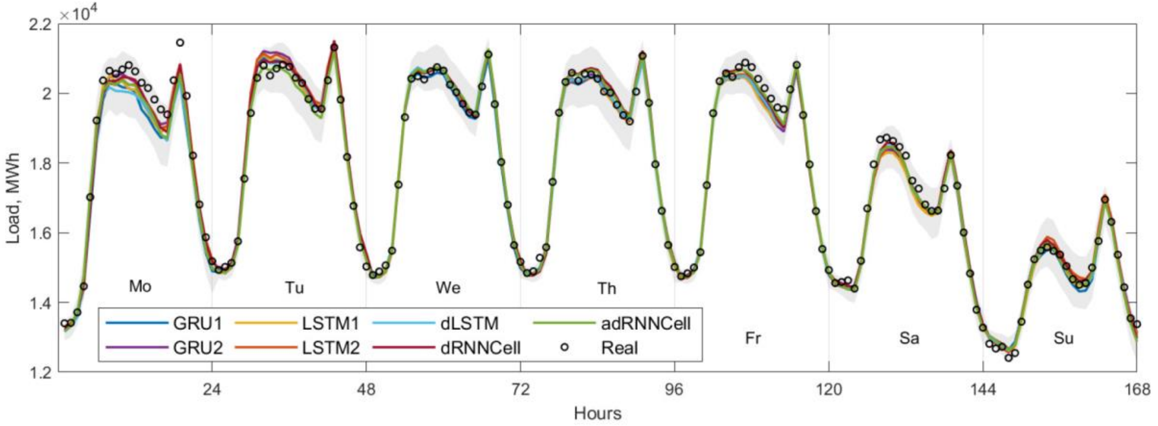


Fig. 9. Examples of the forecasts. 90% PIs for adRNNCell are shown as gray-shaded areas.

Table 2. Forecasting quality metrics.

Cell type	% in PI	% below PI	% above PI	Winkler score
GRU1	92.78±2.80	3.19±1.33	4.03±1.68	0.1524±0.2579
GRU2	**90.40±3.40**	**4.79±1.74**	**4.81±1.80**	0.1448±0.2710
LSTM1	89.16±4.07	5.30±2.05	5.53±2.21	0.1428±0.2760
LSTM2	88.52±3.88	5.53±1.96	5.96±2.04	0.1368±0.2741
dLSTM	88.84±3.74	5.59±1.96	5.57±1.94	0.1393±0.2737
dRNNCell	87.51±3.54	6.16±1.92	6.33±1.77	0.1363±0.2771
adRNNCell	88.41±3.14	5.68±1.67	5.91±1.62	**0.1332±0.2683**

Comparison with other forecasting models [22]:

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**
adRNNCell	**2.12**	**1.91**	**2.07**	**289.32**	-0.14	**3.52**

[22] Smyl S., Dudek G., Pełka P.: ES-dRNN with Dynamic Attention for Short-Term Load Forecasting. IJCNN 2022.

- The experimental study performed on a challenging STLF problem with multiple seasonality showed that adRNNCell outperformed in terms of accuracy its competitors.
- The adRNNCell model is equipped with many mechanisms and procedures designed to increase forecasting efficiency. They include:
  - hierarchical architecture, which learns temporal dependencies of different scales at different layers,
  - new recurrent cell with dilation and attention mechanisms, which help in modeling seasonal dependencies as well as selecting input information,
  - cross-learning, which enables RNN to capture the shared features of the individual series and prevents over-fitting,
  - a composed asymmetrical loss function based on quantiles, which enables RNN to produce both point forecasts and PI, and also to reduce the forecast bias, and
  - encoding the output sequences using variables determined from recent history, which helps to capture the current dynamics of the process.

Thank you for your attention