Decomposition-Residuals Neural Networks: Hybrid system identification applied to electricity demand forecasting

Konstantinos Theodorakos, Oscar Mauricio Agudelo, Marcelo Espinoza, and Bart De Moor, *Fellow, IEEE & SIAM*

Day-ahead energy forecasting systems struggle to provide accurate demand predictions due to pandemic mitigation measures. Decomposition-Residuals Deep Neural Networks (DR-DNN) are hybrid point-forecasting models that can provide more accurate electricity demand predictions than single models within the COVID-19 era. DR-DNN is a novel two-layer hybrid architecture with: a decomposition and a nonlinear layer. Based on statistical tests, decomposition applies robust signal extraction and filtering of input data into: trend, seasonal and residuals signals. Utilizing calendar information, temporal signals are added: sinusoidal day/night cycles, weekend/weekday, etc. The nonlinear layer learns unknown complex patterns from all those signals, with the usage of well-established deep neural networks. DR-DNN outperformed baselines and state-of-the-art deep neural networks on next-day electricity forecasts within the COVID-19 era (from September 2020 to February 2021), both with fixed and Bayesian optimized hyperparameters. Additionally, model interpretability is improved, by indicating which endogenous or exogenous inputs contribute the most to specific hour-ahead forecasts. Residual signals are very important on the first hour ahead, whereas seasonal patterns on the 24th. Some calendar features also ranked high: whether it is day or night, weekend or weekday and the hour of the day. Temperature was the most important exogenous factor.

***Index Terms*—electricity demand forecasting, signal decomposition, deep neural networks, system identification, COVID-19**

I. Introduction

DAY-ahead forecasts are key elements used in the power markets for the planning and pricing of the day-ahead operations, and they are critical for the optimal balance of real-time operations. Our objective is to improve 24 hours-ahead city-wide electricity forecasting accuracy within the COVID-19 era.

AutoRegressive Integrated Moving Average (ARIMA) [1] is a class of forecasting models that contain three main components: (1) AutoRegression (AR) for the relationship between current and p lagged (previous) observations, (2) Integration (I) that applies differencing (subtraction) between an observation and d previous ones and (3) Moving Average (MA) that is the linear combination of q previous predictions. Combinations of statistical and Machine Learning (ML) models seem to produce more accurate results than individual methods on forecasting [2]. In a Short-Term Load Forecasting (STLF) problem, AR-Nonlinear AutoRegressive eXogenous (AR-NARX) models combined with Fixed-Size Least-Squares Support Vector Machines (FS-LSSVM) [3] gave the best forecasting accuracy. Neural networks in the past have been characterized as "not well suited" for time-series forecasting [4]. In the M4 forecasting competition of 2018 [5], 61 methods were tested against 100,000 time series. Hybrid modelling approaches were the most performant, with the best approach being: mixed Exponential Smoothing (ES) with Long Short-Term Memory (LSTM) networks [6]. LSTM networks have been proven effective in electricity price forecasting [7] and residential load forecasting [8] after tuning their hyperparameters. In a more recent STLF publication, hybrid machine learning methods combining support vector regression, random forest modelling, grey catastrophe (avoiding residuals greater than a threshold) and statistical tests, showed superior forecasting performance against single models [9]. In a literature review on energy forecasting of the past 10 years, it was suggested that deep learning and machine learning models

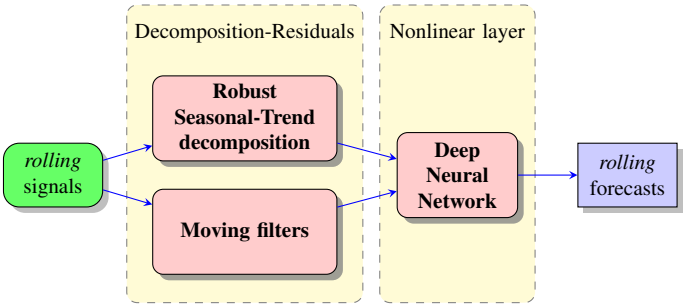

Figure 1. Decomposition-Residuals Deep Neural Networks (DR-DNN) is a two-layer hybrid architecture with: (1) a decomposition and (2) a nonlinear layer of a Deep Neural Network (DNN) model. Given rolling input signals, robust seasonal-trend decomposition and moving filters extract new highly contributing signals. The trained DNN produces rolling forecasts.

Preprint submitted on 8 October 2021; revised on January 7, 2022. Corresponding author: Konstantinos Theodorakos. The authors are with KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Kasteelpark Arenberg 10, box 2446, 3001 Leuven, Belgium (e-mails: {konstantinos.theodorakos, mauricio.agudelo, bart.demoor}@esat.kuleuven.be).

Acknowledgments: ● KU Leuven: Research Fund (projects C16/15/059, C3/19/053, C24/18/022, C3/20/117, C3I-21-00316), Industrial Research Fund (Fellowships 13-0260, IOFm/16/004, IOFm/20/002) and several Leuven Research and Development bilateral industrial projects; ● Flemish Government Agencies: ○ FWO: EOS Project no G0F6718N (SeLMA), SBO project S005319N, Infrastructure project I013218N, TBM Project T001919N; PhD Grants (SB/1SA1319N, SB/1S93918, SB/1S1319N), ○ EWI: the Flanders AI Research Program ○ VLAIO: CSBO (HBC.2021.0076) Baekeland PhD (HBC.20192204) and Innovation mandate (HBC.2019.2209) ● European Commission: European Research Council under the European Union's Horizon 2020 research and innovation programme (ERC Adv. Grant grant agreement No 885682); ● Other funding: Foundation 'Kom op tegen Kanker', CM (Christelijke Mutualiteit).

should not blindly include raw and unprocessed data [10]. There is also evidence that common neural networks are not efficient in learning nonlinear or dynamic patterns that contain moving average terms [11]. For this reason, in our proposed architecture we include distinct signals generated from moving filters. In addition, it was recently found that Recurrent Neural Networks (RNN) are capable of modelling seasonality, but only if the data possess homogeneous seasonal patterns [12]. Our architecture overcomes this issue by automatically applying Seasonal-Trend decomposition based on LOWESS (STL) [13]. Attention-based models, which were initially used in neural machine translation tasks [14], also achieve high performance in multi-horizon time series forecasting on real-world load, traffic, retail and stock market datasets [15], [16]. Temporal Convolutional Networks (TCN), originally designed as generative models of raw audio sequences [17], recently managed to successfully predict months-ahead earth-scale climate phenomena [18]. Nonlinear models, even though they can provide good forecasting accuracy, are usually difficult to interpret. Model interpretability with shapley additive explanations for time series Deep Neural Network (DNN) models was successfully applied in the past, in the context of hours-ahead air-pollution forecasting (atmospheric NO_2 concentrations) [19]. Neural networks have been employed in aggregated power demand forecasting, for regions with severe COVID-19 pandemic impact in northern Italy [20]. However, to our knowledge, very few publications exist that directly tackle the forecasting difficulties introduced by the pandemic. Finally, many publications seem to use data from 5 to 10 years ago and very small time periods for their out-of-sample model testing (only a few days or weeks at best).

In this paper, we propose Decomposition-Residuals Deep Neural Networks (DR-DNN), a novel two-layer hybrid architecture with a decomposition and a nonlinear layer (see Fig. 1). The decomposition layer extracts trends, seasonal, lagged, differenced, filtered and temporal signals, based on statistical tests on the raw time series. The nonlinear layer is a DNN with large learning capacity and is explainable via the game-theoretic SHapley Additive exPlanations (SHAP) [21]. We used five years of real-world data from the IEEE DataPort contest "Day-Ahead Electricity Demand Forecasting: Post-COVID Paradigm" [22], which contains city-wide hourly time series from March 2017 to February 2021. The main contribution of this paper is the DR-DNN architecture and its application on a city-wide load forecasting problem. The primary characteristics and advantages of DR-DNN are:

- DR-DNN is a two-layered architecture that can be readily applied on existing or future DNN architectures.
- DR layers reduced the power load forecasting error on all DNN architectures that we tried, by extracting new highly contributing signals on the fly.
- Minimal manual preprocessing is required with DR. Instead of discarding the extremes (residuals/noise), external disturbances are fed to the models, along with statistically-robust signals. In addition, statistical tests employ majority voting for the fitting of the DR layer parameters. Finally, DNN with their high learning capacity, are able to learn complex nonlinear patterns.
- SHAP aids interpretability, by indicating which endogenous (i.e., power load) or exogenous (e.g., weather data, calendar information, ...) inputs contribute the most to specific hour-ahead forecasts.
- DR-DNN can adapt to the COVID-19 era of electricity forecasting better than the single DNN models (using 6 months of out-of-sample test data, from September 2020 to February 2021).

This document is divided into 4 sections. Section II provides a top-level view, a step-by-step flowchart for model training, as well as a detailed formalism of the DR-DNN model prototype. It also describes the concepts of moving filters, seasonal-trend decomposition, neural networks and feature importance. Section III shows and discusses the performance of DR-DNN against single DNN models. Section IV, contains insights and conclusions. Appendix A provides details on the development environment, source code and data repositories.

II. DR-DNN Model

A. DR-DNN top level view

Fig. 2 shows the DR-DNN architecture. *Decomposition* extracts highly contributing signals given endogenous (power load) and exogenous (weather data) time series. DR applies moving filters, differencing and lag shifting on the endogenous signal (electrical load). Locally Weighted Scatterplot Smoothing (LOWESS) [23] is a local regression smoother, that is robust to outliers and missing values. Robust Seasonal-Trend decomposition based on LOWESS (STL) [23], [13] decomposes time series into: (1) trend, (2) seasonal and (3) residuals signals. To utilize temporal information (in other words, for the models to learn patterns in conjunction with the natural passage of time cycles: exact season, day of week, time of day, ...), sinusoidal calendar signals are extracted (i.e., sin/cosines of day of week/month/year, weekend/weekday, day/night cycles etc). The *nonlinear* layer is a black-box model that learns unknown nonlinearities. Ranking the importance of input features with SHAP [21], assists in model interpretability, by indicating which inputs contribute the most to specific hour-ahead forecasts. Extracting too many signals slows model training [2] and DR avoids it by properly selecting signal extraction parameters via statistical tests. Models use rolling windows of 48 hours: 24 hours input and 24 hours-ahead model output as forecast. Fig. 3 illustrates three random 24 hour ahead forecasts of a DR-DNN during model training.

B. DR-DNN formalism

Fig. 4 presents a step-by-step flowchart for the DR-DNN model training. *DR-DNN* is a hybrid two-layered model architecture, defined as the tuple:

$$DR\text{-}DNN = < DR, DNN, w_i, w_o, \sigma > \quad (1)$$

where *DR* is the Decomposition-Residuals layer that extracts signals given time series, w_i is the input rolling window length (24 hours in our case), w_o is the output rolling window length (24 hours) and σ is the window stride (how much

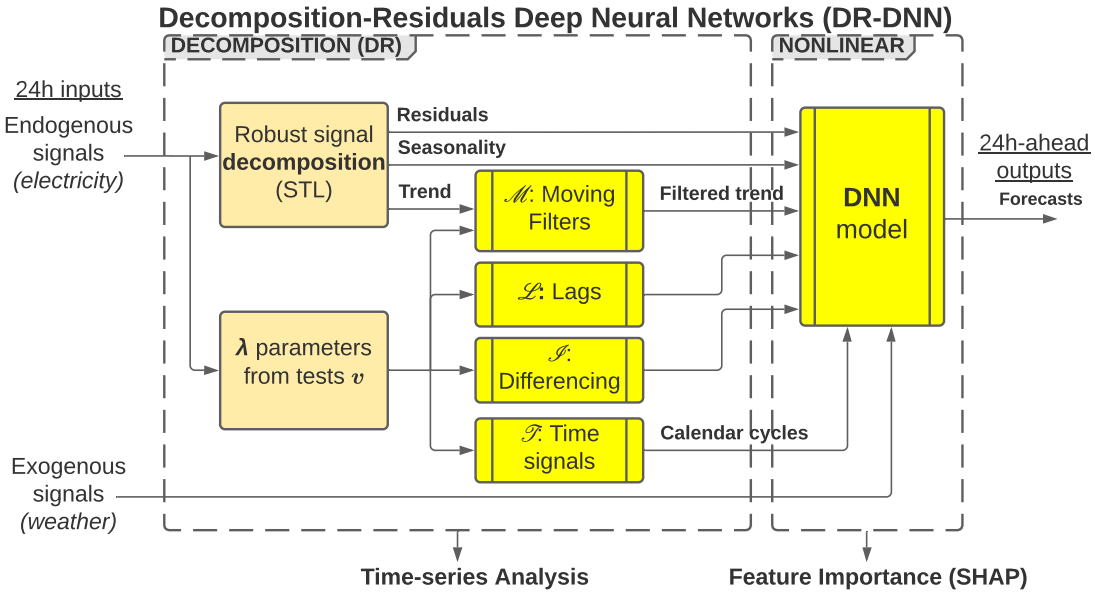

Figure 2. Decomposition-Residuals Deep Neural Networks (DR-DNN) is a two-layer hybrid architecture with: (1) a *decomposition* and (2) a *nonlinear* layer of a Deep Neural Network (DNN). Robust signal decomposition on electrical load extracts: residuals, seasonal and trend signals. The trend is fed to moving filters $\mathscr{M}$. Statistical tests v vote to decide the parameters λ of the extracted signals: lagged $\mathscr{L}$, differenced $\mathscr{I}$ and the cyclical sinusoidal date-time $\mathscr{T}$. All the extracted signals along with the exogenous (weather variables) are fed to a DNN model (right layer) to learn unknown nonlinearities. Time series analysis and SHapley Additive exPlanations (SHAP) [21] provide model explainability with ranked feature importance.

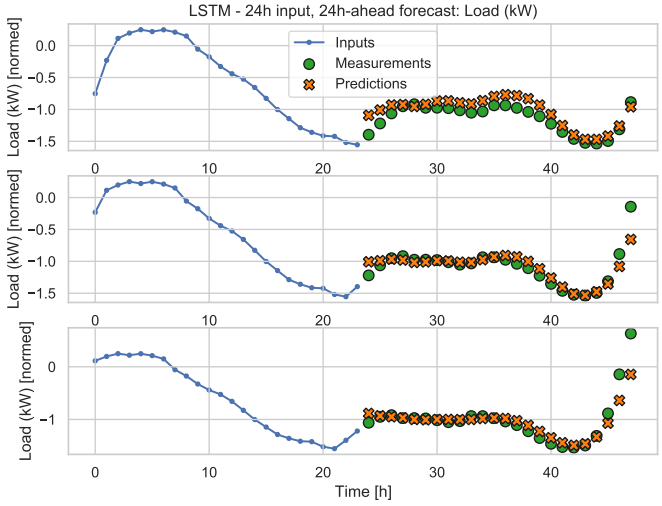

Figure 3. Illustration of three random 24-hour ahead forecasts of city-wide load during model training (x-axis: time in hours, y-axis: power in kW [normalized]). Model: Decomposition-Residuals (DR) Long Short-Term Memory (LSTM) [24]. Input is 24 hours of past data (decomposed signals from power/weather/calendar features: 38 variables from hour 0 up to hour 23). Output is a single-shot 24 hours-ahead forecast of electrical load (from hour 24 up to hour 47). DR-DNN models can also provide intra-day forecasts (24-hour ahead predictions at any time of a day).

we shift the rolling window: 1 hour in our case). To our knowledge, traditional 24-hour ahead models in the field electricity forecasting usually employ strides of 24 hours, which may not be able to forecast unexpected demand events. For example, unusually high or low intra-day wind and solar power generation moments, may lead to unusual intra-day demand events [10] that may not be forecasted by models that provide predictions only once per day. *DNN* can be a single or multiple input/output Deep Neural Network, acting as a nonlinear component with large learning capacity that is able to learn patterns in time from multiple endogenous and exogenous signals.

1) Decomposition Residuals layer

Nowadays, there exist software packages like tsfresh [25] that can extract thousands of signals from time series. However one should be eclectic in formulating new hybrid and combination methods [2]. Signal extraction should be sparse and selective. In our experiments, we realized that blindly decomposing all possible signal combinations is not ideal: too many signals result to very slow model training and accuracy improvements will not be significant after a point.

To set up a DR layer, a parameter set λ is needed:

$$\lambda = \{p, d, q\} \tag{2}$$

where p is the time series shift amount (number of lags), d is the degree of differencing (how many times past data have been subtracted) and q is the span of moving filters. v are statistical tests that decide the p, d, q parameters via majority voting. For the majority voting, we use the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) [26], the Augmented Dickey–Fuller (ADF) [27] tests, AutoCorrelation Function (ACF) and Partial AutoCorrelation Function (PACF). KPSS and ADF use null hypothesis testing to determine the degree of differencing d. For the ADF we use all the available

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OAJPE.2022.3145520, IEEE Open Access Journal of Power and Energy

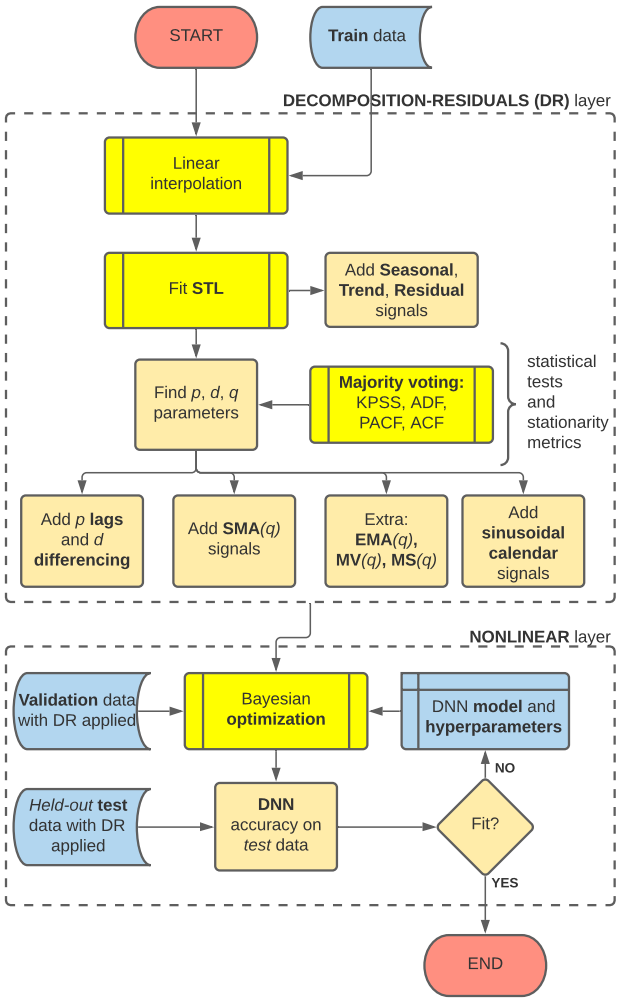

Figure 4. Flow chart for the training of a DR-DNN model. We have two main phases: the *DR layer* and the *nonlinear layer* fitting. For the DR phase we use the train data while we hold-out our validation and test data from the start. First we apply linear interpolation to fill in missing values. After fitting an STL model on the electricity load data, we decompose this endogenous variable into seasonal, trend and residual signals. Then we employ statistical tests to find the p, d, q parameters that define with majority voting: the amount lags p, degree of differencing d. q defines the window of the moving filters: Simple Moving Average (SMA), Exponential Moving Average (EMA), Moving Variance (MV) and Moving Sum (MS). The sinusoidal calendar signals extracted depend on the time-step of the dataset. For the nonlinear phase: using the newly extracted signals and a set of hyperparameters, we apply Bayesian optimization to train a specific DNN model type (minimizing the validation data error). The final model evaluation is performed on the held-out test data.

lag length calculation methods: t-statistic, Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC). The amount of lags that has the largest ACF is set as p and the lag with the largest PACF value is set as q.

DR is a tuple (see Fig. 2):

$$DR =< STL, v, \lambda, \mathscr{L}, \mathscr{M}, \mathscr{I}, \mathscr{T} > \quad (3)$$

where STL is the Seasonal-Trend decomposition using LOWESS, v are the statistical tests that decide the λ parameter set of the DR layer, $\mathscr{L}$ are the time series of maximum lag p, $\mathscr{I}$ is the differenced series of order d, $\mathscr{M}$ are the moving filters of span q, $\mathscr{T}$ are the sinusoidal calendar features (sin/cosines of day of week/month/year, weekend/weekday, day/night cycles etc). Due to STL constraints, for training, the DR layer requires input signals of at least 100 steps. However, it requires only a single pass to compute the extracted signals (it took 13.2 seconds for our training data of 24052 hours). The signals that DR extracts (from 7 endogenous/exogenous inputs to 38 output signals in our case) are then fed to Multiple-Input Multiple-Output (MIMO) DNN models for gradient descent-based training.

The STL tuple consists of the robustly decomposed signals, extracted from the endogenous time series (electrical load):

$$STL =< Trend, Seasonal, Residuals > \quad (4)$$

Seasonal-Trend decomposition based on LOWESS (STL) [23] is a filtering technique for decomposing time series into (1) trend, (2) seasonal and (3) residuals components:

$$Y_t = T_t + S_t + R_t \quad (5)$$

where t is the time-step $\in \{1, ..., s\}$ of span k, Y_t is the original time series, T_t is the trend component (low-frequency variations along with long-term level changes), S_t is the seasonal component (data variations at seasonal frequency), and R_t is the residuals component (remaining data variations). STL uses two recursive loops, an inner and outer one. Given an input time series and a seasonal period, the inner loop updates the seasonal and trend components with a series of: detrending, low-pass filtering, deseasonalizing and smoothing operations. Each outer loop pass computes robustness weights that improve the subsequent inner loop passes. STL can decompose time series with missing values. We applied robust STL [13] (that uses LOWESS) on the endogenous variable (electrical load).

We use the tuple of moving filters $\mathscr{M}$ of span q:

$$\mathscr{M} =< SMA, MS, MV, EMA > \quad (6)$$

where SMA is the Simple Moving Average, MS is the Moving Sum, MV is the Moving Variance and EMA is the Exponential Moving Average.

To detect local variations in time series, we utilize the Moving Sum (MS). Given a time series $x_t = \{x_0, x_1, x_2, ...\}$, MS provides sequences of partial sums of the last k steps:

$$MS_s = x_{s-k+1} + x_{s-k+2} + ... + x_s = \sum_{i=s-k+1}^{s} x_i \quad (7)$$

where MS_s is the moving sum at time s of the partial time-series $\{x_{s-k+1}, x_{s-k+2}, ..., x_s\}$ of k elements, with $k \geq 1$. Moving Average (MA) [28] is a Finite Impulse Response (FIR) filter, which can smooth out short-term fluctuations in time series datasets. Simple Moving Average (SMA) at time-step s is expressed formally as:

$$SMA_s = \frac{x_{s-k+1} + x_{s-k+2} + ... + x_s}{k} = \frac{1}{k} \sum_{i=s-k+1}^{s} x_i \quad (8)$$

EMA or Exponential Weighted Moving Average (EWMA) [29] imposes higher emphasis on most recent samples of a partial time series $\{x_0, ..., x_s\}$ of span k:

$$\begin{aligned} EWMA_s &= \frac{x_s + (1-\alpha)x_{s-1} + (1-\alpha)^2 x_{s-2} + ... + (1-\alpha)^s x_0}{1 + (1-\alpha) + (1-\alpha)^2 + ... + (1-\alpha)^s} = \\ &= \frac{\sum_{i=0}^{s} (1-\alpha)^i x_{s-i}}{\sum_{i=0}^{s} (1-\alpha)^i}, \\ \alpha &= \frac{2}{(k+1)}, k \geq 1 \end{aligned} \tag{9}$$

where $EWMA_s$ is the exponentially weighted moving average at time-step s, α is the smoothing factor $\in (0, 1]$ and $(1-\alpha)^i$ for $i = 0, 1, ..., s$ are the exponential weights.

To measure how far electricity readings are spread out from the moving average of a span k at time s, we use the unbiased sample Moving Variance (MV):

$$MV_s = \frac{\sum_{i=0}^{s} (x_i - SMA_s)^2}{k-1} \tag{10}$$

where SMA_s is the simple moving average at time s of the partial times-series $\{x_0, ..., x_s\}$.

2) Deep Neural Network layer

Artificial Neural Networks (ANN) are models that mimic biological neuron activity via mathematical abstractions. The most common ANN is the multilayer perceptron [30]:

$$\mathbf{y} = \mathbf{W}\sigma(\mathbf{V}\mathbf{x} + \mathbf{b}) \tag{11}$$

where $\mathbf{x} \in \mathbb{R}^m$ is the vector of m input signals, $\mathbf{y} \in \mathbb{R}^l$ is the output vector of l outputs, n_h is the number of hidden neurons (units), σ a sigmoid function, $\mathbf{b} \in \mathbb{R}^{n_h}$ is the bias vector of the hidden neurons and $\mathbf{W} \in \mathbb{R}^{l \times n_h}$, $\mathbf{V} \in \mathbb{R}^{n_h \times m}$ are the interconnection weight matrices. *Convolutional Neural Networks (CNN)* [24] are ANN that mask neuron inputs and outputs via convolution kernels. In other words, CNN apply the convolution operation instead of general matrix multiplication in their layers. *Recurrent Neural Networks (RNN)* [31] are commonly used as sequence and time series models. RNN retain sequential patterns as stable entities in their neurons. *Long Short-Term Memory (LSTM)* [32] cells can store temporal information of longer time spans than RNN. *Gated Recurrent Units (GRU)* [33] have less trainable parameters than the LSTM, because they omit output gates. The LSTM [32] cell equations for feed-forward passes are defined as:

$$\begin{aligned} \mathbf{i}_t &= \sigma_g(\mathbf{W_i}\mathbf{x}_t + \mathbf{U_i}\mathbf{h}_{t-1} + \mathbf{b_i}) \\ \mathbf{f}_t &= \sigma_g(\mathbf{W_f}\mathbf{x}_t + \mathbf{U_f}\mathbf{h}_{t-1} + \mathbf{b_f}) \\ \mathbf{o}_t &= \sigma_g(\mathbf{W_o}\mathbf{x}_t + \mathbf{U_o}\mathbf{h}_{t-1} + \mathbf{b_o}) \\ \mathbf{h}_t &= \mathbf{o}_t \circ \sigma_h(\mathbf{c}_t) \\ \mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \sigma_c(\mathbf{W_c}\mathbf{x}_t + \mathbf{U_c}\mathbf{h}_{t-1} + \mathbf{b_c}) \end{aligned} \tag{12}$$

where d is the number of input features, h is the number of hidden units, $\circ$ the Hadamard product. $\mathbf{x}_t \in \mathbb{R}^d$ is the input vector to the LSTM unit at time t, $\mathbf{i}_t \in \mathbb{R}^h$ is the input gate activation vector, $\mathbf{f}_t \in \mathbb{R}^h$ is the forget gate's activation vector, $\mathbf{o}_t \in \mathbb{R}^h$ is the output gate activation vector, $\mathbf{h}_t \in \mathbb{R}^h$ is the hidden state vector and $\mathbf{c}_t \in \mathbb{R}^h$ is the cell state vector. The $\mathbf{W} \in \mathbb{R}^{h \times d}$, $\mathbf{U} \in \mathbb{R}^{h \times h}$ weight matrices and the bias vectors $\mathbf{b} \in \mathbb{R}^h$, are learned during training. σ_g is a sigmoid function and σ_c, σ_h are hyperbolic tangent functions.

Attention-based [14] models, take inputs from hidden states of LSTM and selectively focus (via weight scoring) on specific time segments. Using the Luong's additive style of scoring (with $score(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \mathbf{h}_t^\top \mathbf{W} \bar{\mathbf{h}}_s$), the forecast y_t is produced by the attention vector α_t:

$$\begin{aligned} \alpha_{ts} &= \frac{exp(score(\mathbf{h}_t, \bar{\mathbf{h}}_s))}{\sum_{s'=1}^{S} exp(score(\mathbf{h}_t, \bar{\mathbf{h}}_{s'}))} \\ \mathbf{c}_t &= \sum_s a_{ts} \bar{\mathbf{h}}_\mathbf{s} \\ \alpha_t &= f(\mathbf{c}_t, \mathbf{h}_t) = tanh(\mathbf{W_c}[\mathbf{c}_t; \mathbf{h}_t]) \end{aligned} \tag{13}$$

where $\bar{\mathbf{h}}_s$ is the source hidden state of total states S, $\mathbf{h}_t$ is the target hidden state, $\mathbf{W}$ are the trainable attention coefficients, α_{ts} are the attention weights, $\mathbf{c}_t$ is the context vector and $\mathbf{W_c}$ are the context weights.

Temporal Convolutional Networks (TCN) [34] are extensions of the CNN. By stacking several convolutional layers, residual blocks and using the concept of receptive fields, TCN can exhibit even longer memory than recurrent architectures of similar unit sizes. Formally the TCN receptive field R_{field} (the amount of steps that can be remembered) is defined as:

$$R_{field} = 1 + 2 \times (K_{size} - 1) \times N_{stack} \times \sum_i d_i \tag{14}$$

where N_{stack} is the number of stacks, d_i is the dilated convolution factor of a residual block of stack i, and K_{size} is the kernel size (length of the convolution filter in steps). We used TCN with causal convolutions, which is a variant more suitable for sequence modelling [34] (like with our time-series forecasting case).

3) Feature importance

SHapley Additive exPlanations (SHAP) values [21] is a game theoretic, model agnostic representation of feature importance, that provides interpretability to complex models. In other words, this representation provides a ranking of the most important inputs in a model, as well as the scale of their positive or negative impact to the model outputs. Given a model prediction $f(\mathbf{x})$, the impact of feature i is the weighted sum:

$$\phi_i(f, \mathbf{x}) = \sum_{S \subseteq S_{all/\{i\}}} \frac{1}{(M \text{ choose } |S|)(M - |S|)} [f_x(S \cup \{i\}) - f_x(S)] \tag{15}$$

where f is a DR-DNN model, $\mathbf{x}$ is the input vector (electricity/weather values), i is a feature, M is the number of interpretable inputs, S is the feature subset (coalition), $|S|$ is the feature cardinality, "M choose $|S|$" is the number of subsets S that we can sample from M without order and without replacement and $f_x(S)$ is the model prediction given S. ϕ_i (with $\sum_{i=0}^{M} \phi_i = 1$) is a single SHAP numerical value, impact of feature i of model f, given input $\mathbf{x}$ and output $f(\mathbf{x})$. In practice, there are too many summation terms and models may not support missing values, so the sum and $f_x(S)$ are approximated using k-medians and random sampling.

III. Results and Discussion

A. Description of the data and the experiments

The goal of the experiments is to assess if DR-DNN are more accurate than typical DNN in 24-hours ahead electricity forecasting during the COVID-19 era. In this paper we refer to hourly electricity load as measured in kW, while technically it should be understood as kWh/h. We used five years of real-world data from the IEEE DataPort contest "Day-Ahead Electricity Demand Forecasting: Post-COVID Paradigm" [22]. This dataset contains seven hourly time series, from unknown city/country, spanning from 18 March 2017 to 16 February 2021 (34360 timesteps): city-wide power demand (kW), air pressure (kPa), cloud coverage (%), humidity (%), temperature (°C), wind direction (degrees) and wind speed (km/h). To avoid overfitting, data were split into three distinct subsets: train (70%: from March 2017 to December 2019), validation (20%: from December 2019 to September 2020), test (10%: COVID-19 timespan from September 2020 to February 2021). To avoid post-sample data leakage, data normalization and DR-layers were fitted on each subset separately. In terms of data preprocessing, we converted the wind direction/velocity into x and y vector components because: (1) vectors use polar coordinates that smooth wind angle transitions from 360° to 0° and (2) with weak winds, wind direction becomes insignificant (most samples had very low-speed winds). To solve data scaling issues, we applied Z-score normalization (e.g., cloud cover ranges $\in [1\%, 100\%]$ whereas electrical load (kW) $\in [0.9, 1.7] \times 10^6$):

$$z = \frac{(x - \mu)}{\sigma} \tag{16}$$

where x is a random variable, μ is the sample mean, σ is the standard deviation. We used nine naive model baselines: (1) "repeat24h" repeats all the 24 hour values from the previous day, (2) "repeat6d24h", (3) "repeat7d24h" and (4) "repeat8d24h" repeat all the 24 hour values from 6, 7 and 8 days ago. (5) "SMA2h" and (6) "EMA2h" are the simple and the exponential moving average from 2 hours ago. Finally, (7) "ARIMA24h", (8) "ARIMA30d" and (9) "ARIMA90d" are 24 hour-ahead ARIMA forecasts from models trained with the last 24 hours, 30 days or 90 days of rolling data (3437 models for 1 hour stride on the test data, with manually tuned parameters: $p = 2$, $d = 0$, $q = 2$). We also compare against seven fixed Artificial Neural Networks (ANN) [24] model baselines: linear (single-layered), dense (two-layered ANN of 512 units and the relu (rectified linear unit) activation function), 1-dimensional Convolutional Neural Network (CNN) (256 units, relu activation function and kernel size of 6 steps), Gated Recurrent Unit (GRU) (64 units, 0.2 dropout rate), Long Short-Term Memory (LSTM) (64 units, 0.2 dropout rate), Long Short-Term Memory (LSTM) with Attention (LSTM: 32 units, Attention: 32 units, 0.2 dropout rate) and Temporal Convolutional Networks (TCN) (64 units, 0.2 dropout rate, kernel size of 6 steps). By number of units, we mean the number of hidden neurons of each DNN and by dropout, we mean the fraction of units to randomly ignore during model training (helps prevent neuron overfitting). We chose these ARIMA and ANN parameters via manual tuning, fitted only on the plain model versions (without using any DR extracted signals, only using the original 8 features). For a fair comparison, all fixed models (in both the DR and No DR case) use the same default Keras framework [35] settings and training setup: 150 max epochs, the adam algorithm [36] and the Mean Squared Error (MSE) loss function. For overfitting avoidance we used learning rate reduction after 5 iterations (with 0.2 reduction factor) and early stopping after 8 iterations. For the optimized models, we used 20 iteration Bayesian optimization [37], with 300 maximum epochs of model training. Table I presents the hyperparameter bounds used in the model search.

Table I
Hyperparameter bounds used in the Bayesian optimization experiments.

Parameter	Range	Description
Common hyperpameters for LSTM Attention, GRU, TCN		
Units	Discrete value in: {8, 16, 24, 32, 40, 48, 56, 64}.	Neuron count.
Optimizer	Choice from: { adam [36], nadam [38], amsgrad [39], adagrad [40], adadelta [41] }.	Gradient descent neuron training algorithm.
Learning rate	Discrete value in: {0.0, 0.01, 0.1, 0.2, 0.25, 0.35, 0.45, 0.5}.	Gradient coefficient used in neuron training.
Dropout rate [42]	Discrete value in: {0.0, 0.01, 0.1, 0.2, 0.25, 0.35, 0.45, 0.5}.	Unit fraction to leave-out during model training.
Explicit TCN hyperparameters		
Kernel size	Integer value $\in [1, 24]$.	Length of the convolution window (in hourly steps).

B. Signal analysis

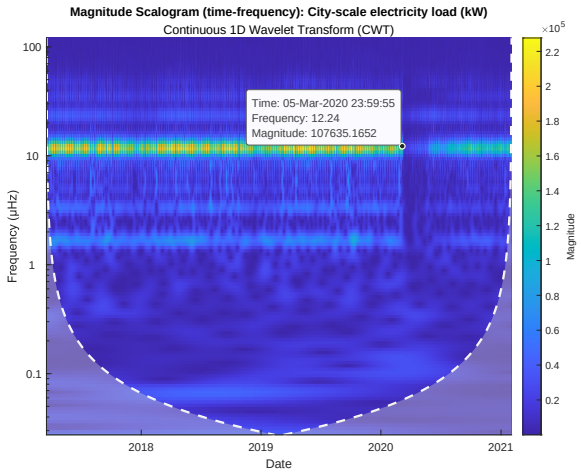

Figure 5. Time-frequency analysis with a magnitude scalogram (of the continuous 1 dimensional Morse wavelet transform). The x-axis indicates time in hours (spanning from March 2017 to February 2021), the y-axis indicates the frequency, and the colour indicates the magnitude intensity. There exists a frequency alteration band from March to May 2020, indicating the impact of COVID-19 on city-wide electrical load (kW).

Due to COVID-19, electricity consumption patterns have changed. The magnitude scalogram on Fig. 5, shows a consumption alteration band of electrical load after the 3rd year

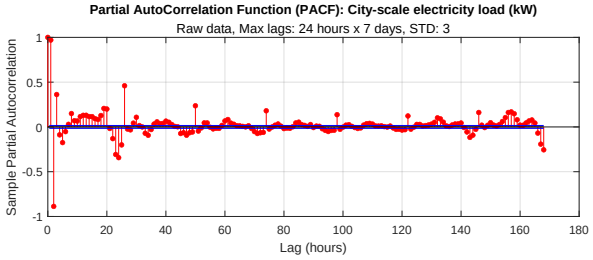

Figure 6. Partial AutoCorrelation Function (PACF) weekly plot (x-axis: lag in hours, y-axis: partial autocorrelation): city-wide electrical load (kW) exhibits hourly cyclical patterns. Lag correlations are at their highest every 24 hours. The first 2-hour lags are also highly significant ($|PACF| > 0.85$).

(from March 5 to May 14 of 2020). Load (kW) exhibits hourly autocorrelation cyclical patterns (Fig. 6). Lag correlations are at their highest every 24 hours. The first 2-hour lags are also highly significant ($|PACF| > 0.85$). According to the Pearson Correlation Coefficient (PCC), the six weather variables are strongly correlated between them, with cloud coverage against humidity being the highest (PCC=0.79). Out of all the exogenous variables, electrical load had the highest correlation with temperature (PCC=0.31) and with wind direction (PCC=0.17). It is worth noting that, on the DR layer it is possible to set up more than one variable as endogenous, along with the electrical load. In our experiments, we tried to combine temperature (the exogenous variable with the highest correlation) along with load as endogenous variables, but we decided to avoid this combination: we did not empirically notice any forecasting accuracy improvements, while the total signal count increased significantly.

C. Decomposition-Residuals model performance

Forecasting accuracy metrics

To compare the performance of DNN against DR-DNN on a 24-hour forecasting horizon, with a forecasting window stride σ of 1-hour, we used the Mean Absolute Error (MAE), the Mean Absolute Percentage Error (MAPE), the symmetric MAPE (sMAPE) and the Mean Absolute Scaled Error (MASE) (that we adapted from the formulation in [5]) on the $T = 34360$ hourly load values:

$$MAE = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h}\sum_{t=n+1}^{n+h}|y(t) - \hat{y}_n(t)| \quad (17)$$

$$MAPE = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h}\sum_{t=n+1}^{n+h}\frac{|y(t) - \hat{y}_n(t)|}{y(t)} * 100\% \quad (18)$$

$$sMAPE = \frac{1}{N}\sum_{n=1}^{N}\frac{2}{h}\sum_{t=n+1}^{n+h}\frac{|y(t) - \hat{y}_n(t)|}{|y(t)| + |\hat{y}_n(t)|} * 100\% \quad (19)$$

$$MASE = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h}\frac{\sum_{t=n+1}^{n+h}|y(t) - \hat{y}_n(t)|}{\sum_{t=n+1}^{n+h}|y(t) - y(t-m)|} \quad (20)$$

where n is the forecasting window index of the total N forecasting windows (with a rolling window stride $\sigma = 1$ hour), h is the forecasting horizon ($h = 24$ hours in our case), $y(t)$ is the electrical load measurement at time $t \in T$, $y(t-m)$ is the naive-m seasonal forecast at time t (with season $m = 24$ hours: the last known observation from 24 hours ago) and $\hat{y}_n(t)$ is the estimated forecast of the window n at time t.

Ablation study

To examine the performance of individual components of DR-DNN, we performed an ablation study (Table II). In other words, we examine how model performance is affected by selectively removing components from the full DR-DNN formulation. We examined the following fixed model versions:

- "No DR": Plain models in their original form.
- "DR: Temporal": DR models including only a moving average filter (applied on the raw load data) and the cyclical temporal signals (time of day, day of week, ...).
- "DR: STL": DR models including only the seasonal-trend decomposition with STL and a moving average filter (applied on the trend load data).
- "DR: STL + Temporal": "DR: STL" models including the cyclical temporal signals.
- "DR: STL + Temporal + Extra filters": "DR: STL + Temporal" models, including all the filter types (exponential moving average, moving variance and moving sum) applied on the trend load data.

We can see that most of the performance improvements come from the STL decomposition (Table II). On average, the DR-LSTM using the STL decomposition had the lowest error. Some model types benefit the most from the full DR-DNN formulation (CNN and Linear models), others by omitting the extra filters (TCN, Dense) or by omitting the temporal signals too (GRU, LSTM, LSTM Attention). We can see that the temporal signals are detrimental for DNN model accuracy (first versus second row for: TCN, GRU, LSTM with Attention and CNN) if they are not accompanied by STL decomposition (first versus fourth row). The best model overall was a DR-TCN (17352 kW MAE), followed by a DR-LSTM with Attention (17372 kW MAE) and a DR-GRU (18016 kW MAE). For our further experiments, we decided to omit the extra filters and to focus the Bayesian Optimization experiments on the TCN, LSTM with Attention and GRU models.

Modelling results

For all the data subsets (train, validation and test), the statistical tests voted that the DR layer should have one lag ($p = 1$), no differencing ($d = 0$) and the moving filters should have a span of two time steps ($q = 2$).

In regards to the fixed models we tested (experiments repeated 100 times), DR-DNN consistently outperform the plain DNN (Fig. 7 and Table III) on the load (kW) Mean Absolute Error (MAE). The DR layer reduces the average error on all model types by $\approx 30.99\%$, as well as the variance: all the models without DR layers had an average MAE of 45288 kW $\pm$ 1482 kW, whereas all the DR-DNN had average MAE 27074 kW $\pm$ 994 kW. DR-TCN had the smallest forecasting error (20749 kW $\pm$ 1684 kW). Only the DR-LSTM with Attention, DR-GRU and DR-TCN consistently surpass the "Repeat7d24h" baseline on average. The best fixed model

Table II
ABLATION STUDY: 24HOUR-AHEAD FORECASTING TEST ERROR (KW MAE, FIXED MODELS, 100 RUNS, MIN AND MEAN ± ONE STANDARD DEVIATION).

	TCN	**GRU**	**LSTM**	**LSTM Attention**	**CNN**	**Linear**	**Dense**
				Mean ± STD			
No DR	26700 ± 2010	29367 ± 1088	33449 ± 4226	28969 ± 1853	36815 ± 751	82305 ± 0	79413 ± 447
DR: Temporal	48979 ± 5601	34360 ± 1458	32510 ± 2140	36837 ± 2148	56789 ± 1317	55652 ± 0	44916 ± 849
DR: STL	23109 ± 1771	**20372 ± 1507**	**19810 ± 983**	**21021 ± 1469**	38368 ± 147	57731 ± 0	48521 ± 307
DR: STL + Temporal	**20749 ± 1684**	24315 ± 547	26916 ± 1057	25829 ± 1942	26823 ± 1341	37567 ± 0	**27321 ± 386**
DR: STL + Temporal + Extra filters	24145 ± 3257	23454 ± 570	25180 ± 1527	25702 ± 1558	**26518 ± 551**	**37106 ± 0**	27593 ± 424
				Best model			
No DR	21382	26973	26793	25840	34980	82305	78293
DR: Temporal	38507	30970	27915	32538	54523	55652	43112
DR: STL	19933	**18016**	**18048**	**17372**	38010	57731	47683
DR: STL + Temporal	**17352**	23304	23956	22781	**22502**	37567	**26174**
DR: STL + Temporal + Extra filters	17651	22064	21761	22510	23153	**37106**	26533

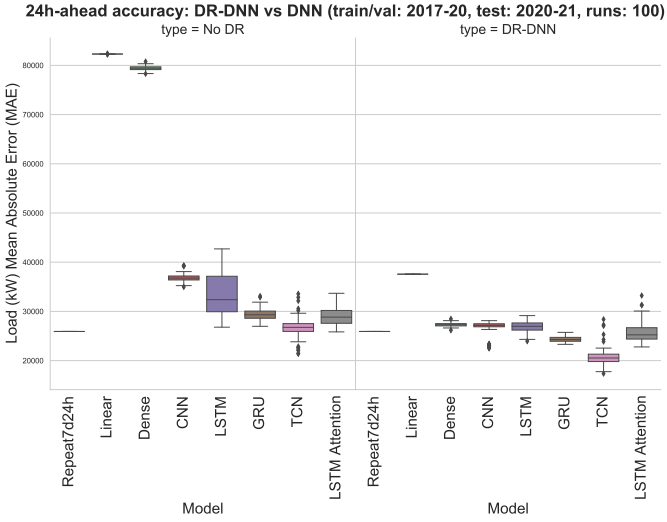

Figure 7. Day-ahead forecasting test error of the fixed size models (x-axis: model, y-axis: load error in kW). All initial models (left boxplot: No DR), did improve using the Decomposition Residuals (DR) layer (right boxplot: DR-DNN). With the DR layer, model error is reduced by ≈ 30.99%. Data: 70% train (from March 2017 to December 2019), 20% validation (from December 2019 to September 2020), 10% test (COVID-19 timespan from September 2020 to February 2021).

Table III
OVERALL RESULTS: 24HOUR-AHEAD FORECASTING TEST ERROR (KW MAE, MEAN ± ONE STANDARD DEVIATION).

Model type	**Without DR**	**DR-DNN** (ours)	Error reduction
Naive model baselines			
Repeat24h	35486	-	-
Repeat6d24h	43447	-	-
Repeat7d24h	25925	-	-
Repeat8d24h	45002	-	-
SMA2h	160085	-	-
EMA2h	160102	-	-
ARIMA24h	74239	-	-
ARIMA30d	69594	-	-
ARIMA90d	69423	-	-
Fixed models (150 epochs, 100 runs)			
Linear	82305 ± 0.0	37567 ± 0.0	54.36%
Dense	79413 ± 447	27321 ± 386	**65.6%**
LSTM	33449 ± 4226	26916 ± 1057	19.53%
CNN	36815 ± 751	26823.0 ± 1341	27.14%
LSTM Attention	28969 ± 1853.0	25829 ± 1942.0	10.84%
GRU	29367 ± 1088	24315 ± 547	17.2%
TCN	26700 ± 2010	20749 ± 1684	22.29%
Best model	21382 (TCN)	**17352** (TCN)	18.85%
Bayesian optimized [37] models (20 iterations, 300 epochs, 10 runs)			
GRU	27299 ± 1386	23977 ± 2188	12.17%
LSTM Attention	47791 ± 26092	28300 ± 2946	**40.78%**
TCN	24187 ± 3293	19248 ± 1762	20.42%
Best model	19483 (TCN)	**16232** (TCN)	9.95%

overall was a DR-TCN with 17352 kW MAE. Models perform similarly on all the additional metrics that we used (MAPE, sMAPE, MASE) (see Table IV). Since the DR layer introduces several new signals (≈ 4x times more, from 7 features to 38), the model training times are significantly increased (for LSTM, LSTM Attention, GRU and TCN). Especially for the case of the TCN, they become the slowest to train (from 129 seconds up to 766 seconds per model). For the Linear, Dense and CNN however, the training times decrease because these models can converge to a good global minima faster with the addition of the DR signals. Finally, the Welch's *t*-test of unequal variances [43] shows that on average, "No DR" models have consistently higher error against the "DR" variants (high statistical significance with p-value less than 0.001).

For the optimized models (experiments repeated 10 times), we focused on TCN, LSTM with Attention and GRU models, with Bayesian optimization (BO) of 20 iterations [37] (Table III). Table I contains the hyperparameters that we used. The greatest improvement can be seen on the DR-LSTM with Attention: mean MAE reduction was 40.78% and the variance was decreased from 26090 kW down to 2946 kW. Even though LSTM with Attention and GRU belong to the same model family, Attention-based models had very high model error and variance without DR on BO. With DR it is greatly reduced, but they still fall behind DR-GRU and DR-TCN. A possible reason is that the LSTM with Attention models have half of their neurons allocated to the LSTM model part and the other half on their Attention model part. In contrast, GRU

Table IV
ALL METRICS ON FIXED MODELS (100 RUNS EACH, MEAN ± ONE STANDARD DEVIATION). THE WELCH'S t-TEST OF UNEQUAL VARIANCES STATISTIC [43] IS POSITIVE IF THE "DR" MODELS HAVE LOWER ERROR THAN THE "NO DR" (STATISTICAL SIGNIFICANCE: *** FOR P-VALUE < .001, ** FOR P-VALUE < .01 AND * FOR P-VALUE < .05).

Model		**MAE** (kW)	**MAPE** (%)	**sMAPE** (%)	**MASE**	**Training time** (seconds)
Repeat24h		35486	3.16	3.2	-	-
Repeat6d24h		43447	3.96	3.92	0.45	-
Repeat7d24h		25925	2.35	2.35	0.34	-
Repeat8d24h		45002	4.02	4.05	0.5	-
SMA2h		160085	14.02	15.48	2.45	-
EMA2h		160102	14.02	15.48	2.45	-
ARIMA24h		74239	6.75	6.75	1.87	428
ARIMA30d		69594	6.33	6.34	1.72	1695
ARIMA90d		69423	6.28	6.32	1.83	4217
Linear	No DR	82305 ± 0	7.83 ± 0.0	7.438 ± 0.0	1.234 ± 0.0	19.6 ± 1.4
	DR	37567 ± 0	3.49 ± 0.0	3.53 ± 0.0	0.519 ± 0.0	**10.6 ± 0.5**
	t-Statistic	$1.03e+09^{***}$	$5.39e+08^{***}$	$7.71e+08^{***}$	$5.27e+08^{***}$	-
Dense	No DR	79413 ± 447	7.532 ± 0.042	7.121 ± 0.038	1.216 ± 0.009	23.1 ± 1.5
	DR	27321 ± 386	2.492 ± 0.034	2.474 ± 0.033	0.35 ± 0.009	13.1 ± 0.4
	t-Statistic	$8.81e+02^{***}$	$9.27e+02^{***}$	$9.31e+02^{***}$	$6.63e+02^{***}$	-
LSTM	No DR	33449 ± 4226	3.07 ± 0.407	3.028 ± 0.386	0.457 ± 0.057	56.0 ± 19.9
	DR	26916 ± 1057	2.479 ± 0.1	2.469 ± 0.103	0.335 ± 0.029	259.7 ± 57.8
	t-Statistic	$1.50e+01^{***}$	$1.41e+01^{***}$	$1.40e+01^{***}$	$1.91e+01^{***}$	-
CNN	No DR	36815 ± 751	3.412 ± 0.071	3.298 ± 0.066	0.57 ± 0.018	225.5 ± 37.0
	DR	26823 ± 1341	2.454 ± 0.123	2.43 ± 0.123	0.385 ± 0.018	16.8 ± 10.2
	t-Statistic	$6.50e+01^{***}$	$6.76e+01^{***}$	$6.20e+01^{***}$	$7.28e+01^{***}$	-
LSTM Attention	No DR	28969 ± 1853	2.648 ± 0.163	2.621 ± 0.157	0.443 ± 0.044	80.8 ± 15.6
	DR	25829 ± 1942	2.393 ± 0.179	2.395 ± 0.185	0.352 ± 0.035	269.3 ± 51.2
	t-Statistic	$1.17e+01^{***}$	$1.06e+01^{***}$	$9.30e+00^{***}$	$1.62e+01^{***}$	-
GRU	No DR	29367 ± 1088	2.681 ± 0.093	2.649 ± 0.09	0.363 ± 0.032	56.8 ± 23.8
	DR	24315 ± 547	2.24 ± 0.048	2.234 ± 0.048	0.289 ± 0.015	229.6 ± 36.9
	t-Statistic	$4.15e+01^{***}$	$4.20e+01^{***}$	$4.07e+01^{***}$	$2.09e+01^{***}$	-
TCN	No DR	26700 ± 2010	2.435 ± 0.184	2.423 ± 0.185	0.382 ± 0.055	129.3 ± 88.8
	DR	**20749 ± 1684**	**1.894 ± 0.156**	**1.879 ± 0.153**	**0.264 ± 0.044**	766.3 ± 244.1
	t-Statistic	$2.27e+01^{***}$	$2.24e+01^{***}$	$2.26e+01^{***}$	$1.69e+01^{***}$	-

(and simple LSTM) of similar total neuron count do not have to split up their neurons, so they allocate all the trainable parameters within their hidden state part. The best models on all experiments were the DR-TCN, with an average MAE of 19248 kW ± 1762 kW. The best single model was a DR-TCN with a MAE of 16232 kW and the following hyperparameters: 48 units, 0.1 learning rate, 0.25 dropout rate, kernel size of 6 steps (hours) and the adadelta optimizer. Worth noting is that, both the fixed and the optimized DR-TCN had similarly good performance, whereas on average, the plain TCN could barely surpass the baseline models. One of the disadvantages of TCN is that they required the longest training times with the addition of the DR layer (Table IV). As also noted in the literature [34], depending on the problem domain, TCN may need tuning of their receptive field. TCN models with the default kernel size of 2 (hours) performed poorly, that is why we increased it to 6 steps for the fixed models, and added the kernel size as a hyperparameter in the Bayesian optimized TCN models.

Fig. 8 shows the ranked input feature importance using SHAP values [21]. The DR-layer, using the original 7 signals and statistical tests, extracted 38 signals in total for 24 input steps. Newly extracted signals from the DR layer (e.g., electric power load: residual, lagged and seasonal, sine of the hour of day and whether it is daytime or nighttime) have higher contribution (≈ from 0.2 to 0.55) than the original raw electric power load variable (4th feature: "Load (kW)-7h" at ≈ 0.22). Fig. 9 examines the feature impact on the 1st hour that we try to forecast: the residuals and raw load values are very important for predictions in the first few hours. When the residuals values are high (1st row, red dots), they increase the forecast output (positive output impact, with SHAP value $\approx [0.03, 0.12]$). The most important calendar features were: whether it is a day or night (5th feature: "nighttime"), the hour of the day (6th: "cos_hour") and whether it is a weekend or weekday (8th: "weekday"). Now for forecasts at the 24th hour ahead (Fig. 10): the seasonal signal becomes the most important. Worth noting is that temperature from the 9th position in 1st-hour ahead, moves to the 4th position for the 24th-hour ahead. Day/night cycles, have the opposite effect: when it is night (fifth row, 'nighttime-23h", red dots), the forecast output increases (negative SHAP values $\approx [-0.05, 0.0]$). No exogenous variable other than temperature reached the top 20 features.

IV. CONCLUSION

In this paper we have applied DR-DNN to the problem of short-term load forecasting. Our findings indicate that DR-DNN outperformed baselines and typical DNN types on 24 hour-ahead electricity forecasts during the COVID-19 era (test data: last 143 days, from September 2020 to February 2021). By decomposing electrical load and using calendar

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OAJPE.2022.3145520, IEEE Open Access Journal of Power and Energy

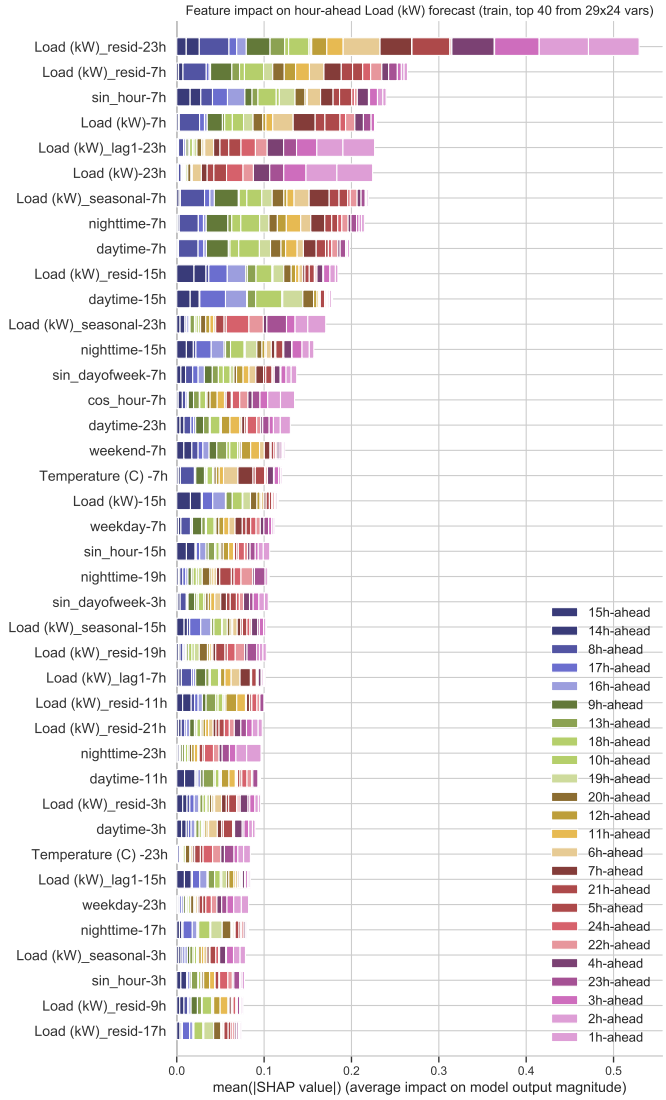

Figure 8. Ranked feature importance using SHapley Additive exPlanations (SHAP) values [21] from a trained DR-TCN model (y-axis: extracted signal at input timestep $t \in [0h, 23h]$, x-axis: absolute feature impact to hour-ahead outputs, color: specific hour-ahead output contribution). This figure shows the average absolute contribution of each input feature of a specific hour, against a specific hour-ahead forecast, as well as the total contribution on all 24 hours ahead. The extracted signals from DR (within top 10: residual, lagged and seasonal load, sine of the hour of day and whether it is daytime or nighttime) have higher contribution ($\approx 0.2 - 0.55$), than the original raw electric power load variable (4th feature: "Load (kW)-7h" at ≈ 0.22). "Load (kW)-23h" contributes significantly on 1, 3, 4, 7 hours-ahead but only slightly on 9-15 hours-ahead forecasts. For forecasts 9-15 hours ahead, the extracted "Load (kW)-residual" signals (first and second), have the highest contribution. "Temperature(C)" is the most important exogenous variable (ranked 18th and 33rd).

information, new, highly contributing signals are extracted on the fly. DNN on the nonlinear layer utilize unknown disturbances, outliers, noise and cyclical pattern signals. Performance increased because instead of discarding extremes, unknown nonlinearities can be learned. SHAP increases model explainability by providing a ranking of the most influential input signals at specific hour-ahead forecasts, while showcasing the significance of the extracted signal contribution. The

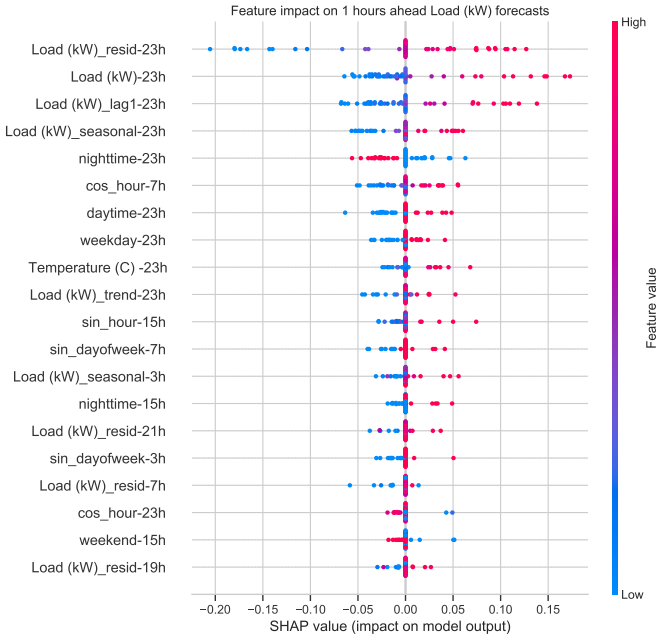

Figure 9. Ranked feature importance for one hour ahead forecasts, from a trained DR-TCN model (y-axis: extracted signal at input timestep $t \in [0h, 23h]$, x-axis: feature impact to the 24th hour-ahead forecast, color: denotes feature value against the full feature range). This figure shows the positive or negative contribution of each data sample of the top 20 features, specifically for the 1st hour-ahead forecast. Positive SHAP values increase the value of the forecast output whereas negative ones decrease it. Residual values (remainder of the signal that can not be explained by seasonal or trend) become the most important factor in the forecasts.

DR-TCN were the most accurate, but also the slowest to train.

Residual signals are very important on the first hours of predictions, whereas seasonal patterns help the most on the last forecasting hours. Some calendar features also ranked high in importance: whether it is day or night, weekend or weekday and the hour of the day. However, temporal signals can be detrimental for DNN model accuracy if they are not accompanied by STL decomposition. Temperature ranked as the most important exogenous factor.

For future work, we would like to assess if there is a suggested cut-off SHAP value, below which a feature can be omitted during the training of a model. We would like to also examine the performance of DR-DNN in similar domains like wind/solar energy forecasting, or in vastly different domains like air-quality and atmospheric pollutant forecasting. Finally, we plan to release an open public code repository with DR-DNN as an Application Programmable Interface (API), for the research community to experiment with.

APPENDIX A
DEVELOPMENT AND REPOSITORIES

Hardware:

- CPU: Intel i9 10850K @ 3.6-5.2GHz with 10 cores.
- System memory: 64GB DDR4 3600MHz.
- Storage: Samsung 980 Pro 1TB M.2 drive.
- GPU: Gigabyte RTX 3090 24GB (Ampere architecture).
- GPU driver: version 497.29.

Figure 10. Ranked feature importance for 24 hours ahead forecasts, from a trained DR-TCN model (y-axis: extracted signal at input timestep $t \in [0h, 23h]$, x-axis: feature impact to the 24th hour-ahead forecast, color: denotes feature value against the full feature range). The seasonal signal is the most variable factor in the forecasts, with temperature moving to the fourth position.

- Operating system: Windows 11 version 21H2 (build 22000.376).

Software:

- Languages: Python 3.8.8, Matlab 2020b.
- Libraries and frameworks:
  - Pandas 1.3.2 [44], SciPy 1.7.1, Numpy 1.19.5.
  - TensorFlow 2.6.0.
  - Cuda: 11.2 v11.2.152, cuDNN library: v6.5.0.

Source code:

- Base source code adapted from the TensorFlow tutorial "Time series forecasting" [45].
- DR-DNN source code Github repository (work in progress): https://github.com/temp3rr0r/DR-DNN/.

Dataset:

- Hourly time series of electrical load and weather variables were accessed from the IEEE DataPort contest "Day-Ahead Electricity Demand Forecasting: Post-COVID Paradigm" [22].

REFERENCES

[1] J. Brownlee, *Introduction to time series Forecasting with Python*. Machine Learning Mastery, 2013.

[2] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and Machine Learning forecasting methods: Concerns and ways forward," *PLOS ONE*, vol. 13, no. 3, pp. 1–26, 3 2018.

[3] M. Espinoza, J. A. K. Suykens, R. Belmans, and B. De Moor, "Electric Load Forecasting," *IEEE Control Systems*, vol. 27, no. 5, pp. 43–57, 2007.

[4] R. J. Hyndman, "A brief history of time series forecasting competitions," 2018. [Online]. Available: https://robjhyndman.com/hyndsight/forecasting-competitions/

[5] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 Competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020.

[6] 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.

[7] L. Peng, S. Liu, R. Liu, and L. Wang, "Effective long short-term memory with differential evolution algorithm for electricity price prediction," *Energy*, vol. 162, pp. 1301–1314, 11 2018.

[8] 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.

[9] G.-F. Fan, M. Yu, S.-Q. Dong, Y.-H. Yeh, and W.-C. Hong, "Forecasting short-term electricity load using hybrid support vector regression with grey catastrophe and random forest modeling," *Utilities Policy*, vol. 73, p. 101294, 2021.

[10] T. Hong, P. Pinson, Y. Wang, R. Weron, D. Yang, and H. Zareipour, "Energy Forecasting: A Review and Outlook," *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 376–388, 2020.

[11] A. Tealab, H. Hefny, and A. Badr, "Forecasting of nonlinear time series using ANN," *Future Computing and Informatics Journal*, vol. 2, no. 1, pp. 39–47, 2017.

[12] 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, 1 2021.

[13] S. Seabold and J. Perktold, "Statsmodels: Econometric and Statistical Modeling with Python," in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 92–96.

[14] M. T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," in *EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics (ACL), 8 2015, pp. 1412–1421.

[15] B. Lim, S. Arık, N. Loeff, and T. Pfister, "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 10 2021.

[16] Y. Chen, Y. Kang, Y. Chen, and Z. Wang, "Probabilistic forecasting with temporal convolutional neural network," *Neurocomputing*, vol. 399, pp. 491–501, 7 2020.

[17] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," 9 2016. [Online]. Available: https://arxiv.org/abs/1609.03499

[18] J. Yan, L. Mu, L. Wang, R. Ranjan, and A. Y. Zomaya, "Temporal Convolutional Networks for the Advance Prediction of ENSO," *Scientific Reports 2020 10:1*, vol. 10, no. 1, pp. 1–15, 5 2020.

[19] M. Vega García and J. L. Aznarte, "Shapley additive explanations for NO2 forecasting," *Ecological Informatics*, vol. 56, p. 101039, 3 2020.

[20] P. Scarabaggio, M. La Scala, R. Carli, and M. Dotoli, "Analyzing the Effects of COVID-19 Pandemic on the Energy Demand: the Case of Northern Italy," in *2020 AEIT International Annual Conference (AEIT)*, 2020, pp. 1–6.

[21] S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K. W. Low, S. F. Newman, J. Kim, and S. I. Lee, "Explainable machine-learning predictions for the prevention of hypoxaemia during surgery," *Nature Biomedical Engineering*, vol. 2, no. 10, pp. 749–760, 10 2018.

[22] M. Farrokhabadi, "Day-Ahead Electricity Demand Forecasting: Post-COVID Paradigm," 2020. [Online]. Available: https://dx.doi.org/10.21227/67vy-bs34

[23] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A Seasonal-Trend Decomposition Procedure Based on Loess (with Discussion)," *Journal of Official Statistics*, vol. 6, no. 1, pp. 3–33, 1990.

[24] Ian Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[25] M. Christ, A. W. Kempa-Liehr, and M. Feindt, "Distributed and parallel time series feature extraction for industrial big data applications," 10 2016. [Online]. Available: http://arxiv.org/abs/1610.07717

[26] D. Kwiatkowski, P. C. B. Phillips, P. Schmidt, and Y. Shin, "Testing the null hypothesis of stationarity against the alternative of a unit root. How sure are we that economic time series have a unit root?" *Journal of Econometrics*, vol. 54, no. 1-3, pp. 159–178, 1992.

[27] S. E. Said and D. A. Dickey, "Testing for Unit Roots in Autoregressive-Moving Average Models of Unknown Order," *Biometrika*, vol. 71, no. 3, pp. 599–607, 1984.

[28] R. J. Hyndman, "Moving Averages," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Springer, 2011, pp. 866–869.
[29] M. B. Perry, "The Exponentially Weighted Moving Average," in *Wiley Encyclopedia of Operations Research and Management Science*. American Cancer Society, 2 2011.
[30] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.
[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[32] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, 1997.
[33] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.
[34] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *CoRR*, 3 2018.
[35] F. Chollet and others, "Keras," 2015. [Online]. Available: https://keras.io
[36] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, San Diego, 2015.
[37] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, and others, "Keras Tuner," 2019. [Online]. Available: https://github.com/keras-team/keras-tuner
[38] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *30th International Conference on Machine Learning, ICML 2013*, no. PART 3, Atlanta, Georgia, USA, 2013, pp. 1139–1147.
[39] S. J. Reddi, S. Kale, and S. Kumar, "On the Convergence of Adam and Beyond," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
[40] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," in *COLT 2010 - The 23rd Conference on Learning Theory*, vol. 12, 2010, pp. 257–269.
[41] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *CoRR*, 12 2012.
[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.
[43] B. L. Welch, "THE GENERALIZATION OF 'STUDENT'S' PROBLEM WHEN SEVERAL DIFFERENT POPULATION VARLANCES ARE INVOLVED," *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.
[44] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 56–61.
[45] "Time series forecasting | TensorFlow Core," 2020. [Online]. Available: https://www.tensorflow.org/tutorials/structured_data/time_series

Konstantinos Theodorakos received the B.Sc. in informatics engineering in 2013, from the Technological Educational Institute (TEI) of Crete (Technical College), Greece and the Master of Computer Science in software engineering in 2017, from the Universiteit Antwerpen in Belgium.

He worked as an Electronic Hardware Technician from 2008 to 2011 and as a Numerics Software Developer for Spatial and Spatio-Temporal Machine Learning on Geographic Information Systems from 2013 to 2018. He is currently a Ph.D. student in Engineering Science, at the Department of Electrical Engineering (ESAT) and the STADIUS Center for Dynamical Systems, Signal Processing, and Data Analytics of the KU Leuven in Belgium. He is working on time series analysis and modelling in data science applications.

Mr. Theodorakos received the "Microsoft Azure credits for Research" award in 2017 and the third place on the "Amazon Web Services IoT APP Challenge" contest in 2017.

Oscar Mauricio Agudelo received the B.S. degree in electronics engineering from the Universidad Autónoma de Occidente, Cali, Colombia, in 1997, the M.S. degree in industrial control engineering from the Universidad de Ibagué (in cooperation with KU Leuven and Universiteit Gent), Ibagué, Colombia, in 2004, and the Ph.D. degree in electrical engineering from KU Leuven, Leuven, Belgium, in 2009.

He worked at the Universidad Autónoma de Occidente, from 1997 to 2004, as a full time professor of control and automation. After his Ph.D., he held a postdoctoral position and later on a research manager position in the research group STADIUS at the Department of Electrical Engineering of KU Leuven. He is currently a project coordinator on systems and control in the same research group. His research interests are in model reduction techniques, systems and control theory, machine learning, model predictive control, data assimilation, deep learning, polynomial optimization, system identification, and analysis and design of intelligent control systems.

Marcelo Espinoza obtained his Ph.D. degree in 2006 from the Katholieke Universiteit Leuven, Belgium, working in the field of nonlinear system identification. Prior to that, he received the degrees of Master in Artificial Intelligence (KU Leuven, Belgium, 2002), M.Sc. in Applied Economics (Universidad de Chile, 1998) and Civil Engineer (Universidad de Chile, 1998).

He was a postdoctoral researcher with the Department of Electrical Engineering of the KU Leuven, between 2006 and 2008. Since 2008 he has held several analysis positions in global energy companies.

Bart De Moor (Fellow, IEEE and SIAM) received the doctoral degree in applied sciences, in 1988, from the Katholieke Universiteit, Leuven, Belgium.

He was a research associate at Stanford University, is now full Professor at the Department of Electrical Engineering of KU Leuven and a guest professor at the Universita di Siena, Italy. He served several times as head of cabinet of ministers of Science and Socio-Economic Policy in Belgium/Flanders and acted as a vice-rector of International Policy of KU Leuven. His fields of research are numerical linear algebra, system theory and control, machine learning and data science, in which he has been guiding more than 80 PhD students and co-authored more than 400 scientific papers and 11 books.

Dr. De Moor has been serving in numerous international scientific and science policy assessment committees, is member/head of the board of several (inter-)national scientific institutes (including the Flanders Biotech Institute) and funding agencies. He co-founded 8 spin-off companies (4 in Health 2.0, 4 in Industry 4.0). Since 2019, he is one of the architects and coordinators of the large Flanders AI program. He is the chairman of the Capricorn Digital Growth Fund (venture capital), of Health-House (a high-tech science outreach centre) and the Alamire Foundation (digital humanities, polyphonic music) and co-founded Technopolis (children's science center). A member of the Royal Academy, he was awarded with numerous scientific prizes and recognitions (a.o. an ERC Advanced Grant) and honours (Fellowships of IEEE, SIAM). In 2010 he received the Science Excellence Award from King Albert II of Belgium and in 2020 was nominated a Commander in the Order of Leopold, by King Filip I.