Automated Spatio-Temporal Weather Modeling for Load Forecasting

Julie Keisler, Margaux Bregere

AUTOMATED SPATIO-TEMPORAL WEATHER MODELING FOR LOAD FORECASTING

Julie Keisler
EDF R&D, Lab Paris Saclay
INRIA
julie.keisler@edf.fr

Margaux Brégère
EDF R&D, Lab Paris Saclay
LPSM, Sorbonne Université
margaux.bregere@edf.fr

ABSTRACT

Electricity is difficult to store, except at prohibitive cost, and therefore the balance between generation and load must be maintained at all times. Electricity is traditionally managed by anticipating demand and intermittent production (wind, solar) and matching flexible production (hydro, nuclear, coal and gas). Accurate forecasting of electricity load and renewable production is therefore essential to ensure grid performance and stability. Both are highly dependent on meteorological variables (temperature, wind, sunshine). These dependencies are complex and difficult to model. On the one hand, spatial variations do not have a uniform impact because population, industry, and wind and solar farms are not evenly distributed across the territory. On the other hand, temporal variations can have delayed effects on load (due to the thermal inertia of buildings). With access to observations from different weather stations and simulated data from meteorological models, we believe that both phenomena can be modeled together. In today's state-of-the-art load forecasting models, the spatio-temporal modeling of the weather is fixed. In this work, we aim to take advantage of the automated representation and spatio-temporal feature extraction capabilities of deep neural networks to improve spatio-temporal weather modeling for load forecasting. We compare our deep learning-based methodology with the state-of-the-art on French national load. This methodology could also be fully adapted to forecasting renewable energy production.

1 Introduction

The cost of large-scale electricity storage remains high, and the current systems in use remain inefficient. Furthermore, the secure and smooth operation of the power grid depends on maintaining a constant and precise balance between electricity production and demand. The aforementioned equilibrium is achieved via the adaptability of programmable power plants, which modify their production in accordance with load forecasts. It is therefore essential to have accurate forecasts of both electricity demand and the output of renewable energy sources in order to schedule power plants and maintain grid stability. The two signals depend on meteorological variables, specifically temperature, wind speed, and solar radiation, which vary in both space and time. As consumer demand and renewable energy generation facilities are not evenly distributed across a given area, variations in meteorological conditions at a particular location will affect these signals. In addition, temporal weather variations can have a delayed effect, particularly with regard to the load, due to the thermal inertia of buildings and the reactivity of consumers. It can be assumed that the manner in which temporal and spatial variations in weather patterns are modelled has a significant impact on the efficacy of the forecasting models.

This article concentrates on short-term load forecasting with a forecast time horizon of one day. Such forecasts enable power system operators to make adjustments to production and spot market prices. This signal is challenging to forecast due to its dependence on a multitude of variables, including meteorological factors (temperature, wind, etc.) and calendar-related elements (holidays, weekdays, etc.). Consequently, the models employed in the industry and those that have been successful in load forecasting competitions (see Farrokhabadi et al. [2022], for a recent example) are regression-based models, such as Generalized Additive Models (GAMs) or tree-based models. In general, lagged load is not employed. While this variable offers valuable insight, it can also limit the model's interpretability by reducing the importance of other variables. Furthermore, the model would become unusable in case of data stream issues. To address this challenge while leveraging the insights offered by lagged load, a promising approach is to construct a static model

only based on the explanatory variables and then recalibrate this model by adjusting certain parameters a posteriori using lagged load. For instance, Ba et al. [2012] proposes adaptive learning algorithms that combine additive models with a recursive least squares filter, while Vilmarest [2022] employs a Kalman filter to perform an online adaptation of the weights of their models.

Despite their impressive performance in various fields, such as computer vision and natural language processing, deep neural networks (DNNs)are still not widely used in the load forecasting community. However, recent work presents promising results for load forecasting using DNNs. In particular, Keisler et al. [2024a] proposed EnergyDragon, a deep neural network optimization framework designed for load forecasting. EnergyDragon automatically finds high-performance neural networks for the static part of load forecasting models and is able to outperform state-of-the-art regression models. Since neural networks have demonstrated their ability to extract relevant features from data in a variety of formats, we thought it would be interesting to try them on raw spatio-temporal weather data to see if they could automatically find more relevant spatio-temporal representations than the fixed ones used in the state of the art. In summary, our contributions are as follows:

- A DNN-based spatio-temporal weather modeling for load forecasting, which improves on the static modeling currently in use while remaining interpretable.
- The integration of this weather modeling approach into the framework EnergyDragon.
- An application of our results to a concrete use case: the day-ahead French load forecasting over a turbulent period: sobriety during the year 2023.

We start this paper by presenting in section 2 the state of the art in short-term load forecasting: regression-based models, EnergyDragon and recalibration methods. In Section 3, we show how to learn the actual spatio-temporal modeling approach with DNN. Section 4 introduce how to incorporate the spatio-temporal weather modeling into EnergyDragon. Finally, Section 5 details our experimental results obtained on a real-world use case: the French national load forecasting. Section 6 concludes the paper and presents further research opportunities.

2 Related Work

The load signal can be explained almost entirely by a set of explanatory variables that do not include the past target data. Consequently, the majority of performing models are based on regression rather than time series techniques. Multiple Linear Regressions (MLRs) can be used to calculate the relationships between multiple variables. However, the relationships between load and some exogenous variables are not linear, and thus, these models require the specification of functional forms for these variables. For instance, Generalized Additive Models (GAMs) employ a spline basis to model the nonlinear effects, as detailed in [Pierrot and Goude, 2011]. These models, which are highly accurate for load forecasting, are used in industry and have been the winners of several competitions (see, for example, Nedellec et al. [2014]).

DNNs have dominated the fields of computer vision and natural language processing for some years now. They offer the ability to process data in a variety of formats - e.g., text, images, graphs - make them particularly interesting for load forecasting, which depends on a large number of explanatory variables that may come from data sets in a variety of formats. Recently, they have also revolutionized the field of weather forecasting, proving more effective than Numerical Weather Prediction (see for example [Pathak et al., 2022] and [Lam et al., 2022]).While the initial work was based on gridded reanalysis data, McNally et al. [2024] have shown that DNNs could also be effective in extracting spatio-temporal features directly from raw weather data. In the field of load forecasting, they are currently less widely used than multilinear regression models. However, Keisler et al. [2024a] have demonstrated that by optimizing the structure and hyperparameters of DNNs, it is possible to develop models that surpass the current state of the art. In their article, the authors optimize DNNs using the DRAGON package[1] (see Keisler et al. [2024b]). The models are represented using Directed Acyclic Graphs (DAGs). The search space is defined as $\Omega = (\mathcal{A} \times \{\Lambda(\alpha), \alpha \in \mathcal{A}\})$, where $\mathcal{A}$ is the set of all considered architectures and $\Lambda(\alpha)$ is the set of all considered hyperparameters induced by the architecture α. Each architecture $\alpha \in \mathcal{A}$ is represented by a DAG Γ, where the nodes are the DNN layers and the edges are the connections between them. See Keisler et al. [2024b] for more information about this search space.

Finally, this paper deals with short-term forecasting of electricity consumption. The COVID crisis and recent european energy crises have highlighted the importance of models that can rapidly adapt to new contexts. This is why research in the field has focused on different techniques for online model adaptation. These include the Kalman filter adaptation of Generalized Additive Models (GAMs), which won the post-covid electricity load forecasting competition (see Farrokhabadi et al. [2022] and De Vilmarest and Goude [2022]). The adaptation is done by multiplying the GAM

[1] `https://dragon-tutorial.readthedocs.io/en/latest/`

effects vector by a linear correction. Let's have $x_t \in \mathbb{R}^D$ our features vector, with $D \in \mathbb{N}^\star$, $y_t \in \mathbb{R}$ the target and $\hat{y}_t \in \mathbb{R}$ the forecasted target. The static GAM model can be defined as:

$$\hat{y}_t = \sum_{i=1}^{D} f_i(x_t).$$

Let's have $f(x_t) = [f_i(x_t)]_{i=1}^D$ the GAM effects vector, the adaptation is done by fitting a vector $\theta_t \in \mathbb{R}^D$ called state, such that:

$$\hat{y}_t = \sum_{i=1}^{D} \theta_{i,t} f_i(x_t) + \epsilon_t = \theta_t^T f(x_t) + \epsilon_t \tag{1}$$

$$\theta_{t+1} = \theta_t + \eta_t, \tag{2}$$

where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ and $\eta_t \sim \mathcal{N}(0, Q)$, with σ^2 and Q are time-invariant and assumed to be known. The algorithm achieves the estimation of the state θ_t by computing its state posterior distribution as a Gaussian distribution: $\theta_t|(x_s, y_s)_{s<t} \sim \mathcal{N}(\hat{\theta}_t, P_t)$. The algorithm relies on the choice of σ and Q. Vilmarest [2022] suggests an iterative grid search. In this work, we propose to apply a state vector θ_t on the last layer of a DNN and to optimize σ and Q directly through the EnergyDragon framework, as any other hyperparameter.

3 Weather Modeling with Deep Neural Networks for load forecasting

In this work, we aim to forecast at each time step $t \in [1, \dots, T]$, with $T \in \mathbb{N}^\star$ a daily load variable $y_t \in \mathbb{R}^H$, using a features vector $x_t = (w_t, c_t) \in (\mathbb{R}^{H \times V \times I} \times \mathbb{R}^{H \times F})$, where T represents the number of days in the data set and H the number of time steps within a day. The features vector x_t is made of two elements: w_t gathering the spatio-temporal weather data and c_t containing the other $F \in \mathbb{N}$ explanatory variables such as calendar data (e.g., months, years holidays). The vector $w_t \in \mathbb{R}^{H \times V \times I}$ contains the forecasts at time t from different weather stations, or to a weather forecast grid, produced by, for example, a NWP model. The dimension $I \in \mathbb{N}^\star$ corresponds to the number of spatial points (i.e., the number of weather stations in the first case and the number of grid points in the second) and $V \in \mathbb{N}^\star$ to the number of weather variables present in w_t (e.g., temperature, wind speed, solar radiation).

3.1 Spatio-temporal weather modeling

In order to be integrated into load forecasting models, spatio-temporal weather is deterministically transformed into "electrical" weather. Several functions are applied in order to reduce the information and extract what will be most useful for load forecasting. These functions have been defined with industry expertise, but are not adapted to a particular dataset or period. An example of such functions for the french load signal are given by the French Transmission System Operator called RTE[2].

Ponderation The first step is to switch back from the multi-variate, spatial signal to an aggregated univariate signal. The I spatial locations are not necessarily evenly distributed throughout the considered region and do not contribute equally to the electrical weather. For instance locations in densely-populated parts have more weights than others located in isolated areas. Let's denote $w_h^{v,i} \in \mathbb{R}$ the forecast of the weather variable v (e.g. temperature) at time step h of the location i, and $a^i \in [0, 1]$ the weight of the location i. The weights are shared accross the variables v. The aggregated signal at time h can then be written as:

$$w_h^v = \sum_{i=1}^{I} a^i w_h^{v,i}, \text{ with: } \sum_{i=1}^{I} a^i = 1. \tag{3}$$

This behavior can easily be reproduced with a Multi-Layer Perceptron (MLP):

$$w_h^v = A\mathbf{w}_h^v + b, \text{ with } \mathbf{w}_h^v = [w_h^{v,i}]_{i=1}^I, A = [a^i]_{i=1}^I \text{ and } b = 0 \tag{4}$$

However, a Deep Neural Networks requires the scaling of the input data. Each v variable is scaled independently, so that variables with large amplitudes (e.g. temperature) don't override the others. We scale each location i independently and denote $\tilde{w}_h^{v,i} = (w_h^{v,i} - \min^{v,i})/(\max^{v,i} - \min^{v,i})$ the min-max scaled version of $w_h^{v,i}$, with $\min^{v,i} = \min_{h \in [1,\dots,H]} w_h^{v,i} \in \mathbb{R}$

[2] https://www.services-rte.com/files/live/sites/services-rte/files/documentsLibrary/2022-04-01_REGLES_MA-RE_SECTION_2_F_3590_en

and $\max^{v,i} = \max_{h\in[1,\dots,H]} w_h^{v,i} \in \mathbb{R}$. If we consider the aggregated target to also be scaled, with $\min^v = \min_{h\in[1,\dots,H]} w_h^v \in \mathbb{R}$ and $\max^v = \max_{h\in[1,\dots,H]} w_h^v \in \mathbb{R}$, we have:

$$\begin{aligned}
\tilde{w}_h^v = \frac{w_h^v - \min^v}{\max^v - \min^v} &= (\sum_{i=1}^{I} a^i w_h^{v,i} - \min^v)/(\max^v - \min^v) \\
&= \big(\sum_{i=1}^{I} \tilde{w}_h^{v,i}(\max^{v,i} - \min^{v,i}) + \min^{v,i} - \min^v\big)/(\max^v - \min^v) \\
&= \sum_{i=1}^{I} a^i \frac{\max^{v,i} - \min^{v,i}}{\max^v - \min^v} \tilde{w}_h^{v,i} + \sum_{i=1}^{I} \frac{\min^{v,i} - \min^v}{\max^v - \min^v} \\
&= A_v \tilde{\mathbf{w}}_h^v + b_v
\end{aligned}$$

,

with $\tilde{\mathbf{w}}_h^v = [\tilde{w}_h^{v,i}]_{i=1}^I$, $A_v = [a^i(\max^{v,i} - \min^{v,i})/(\max^v - \min^v)]_{i=1}^I$ and $b_v = \sum_{i=1}^I (\min^{v,i} - \min^v)/(\max^v - \min^v)$. Therefore, we need V MLP layers to aggregate the data variable by variable.

3.2 Temperature smoothing

Load does not respond instantaneously to changes in the weather. In particular, temperature effects are more gradual due to the thermal inertia of buildings. This is why the concept of smoothed temperature is useful for understanding the factors that influence electricity consumption. Exponential smoothing is typically employed in this context. We denote, for a day t, $T_t = [T_{t,1}, \dots, T_{t,H}] \in \mathbb{R}^H$ the aggregated temperature and $\overline{T}_t = [\overline{T}_{t,1}, \dots, \overline{T}_{t,H}] \in \mathbb{R}^H$ the smoothed version. We define:

$$\overline{T}_{t,1} = (1-\alpha)T_{t,1} + \alpha\overline{T}_{t-1,H}, \text{ and, } \forall i \in [2,H] : \overline{T}_{t,i} = (1-\alpha)T_{t,i} + \alpha\overline{T}_{t,i-1}, \tag{5}$$

where $\alpha \in [0,1]$ is the smooth coefficient, which can be optimized.

Recurrent Neural Networks Smoothed temperature requires at each time step t to pass $\overline{T}_{t,H} \in \mathbb{R}$ to the next time step $t+1$. Such information passing can be reproduced by Recurrent Neural Networks (RNNs), which are designed with a memory vector. The equations of a recurrent neural network with input $T_t \in \mathbb{R}^H$ and output $\overline{T}_t \in \mathbb{R}^H$ can be written as:

$$\overline{T}_t = \phi(T_t W_1^T + b_1 + \overline{T}_{t-1} W_2^T + b_2)\,,$$

where ϕ is an activation function (typically non-linear), and $W_1 \in \mathbb{R}^{H\times H}$, $W_2 \in \mathbb{R}^{H\times H}$, $b_1 \in \mathbb{R}^H$ and $b_2 \in \mathbb{R}^H$ are some parameters which can be learned through gradient descent. For writing simplicity, we now index our temperature by $t^\star$, such that, if $t^\star = \{t, i\}$, we have, if $i < H$:

$$\begin{cases} t^\star = \{t, i+1\} \\ \text{else, } t^\star + 1 = \{t+1, 1\}. \end{cases}$$

Let $\tau > 0$. To compute $\overline{T}_{t^\star}$ based on the smoothed temperature at instant $t^\star - \tau$, $\overline{T}_{t^\star-\tau}$, and the sequence of temperatures $T_{t^\star-\tau+1}, \dots, T_{t^\star}$, we have:

$$\begin{aligned}
\overline{T}_{t^\star-\tau+1} &= \overline{T}_{t^\star-\tau} \\
\overline{T}_{t^\star-\tau+1} &= (1-\alpha)T_{t^\star-\tau+1} + \alpha\overline{T}_{t^\star-\tau} \\
\vdots \quad &= \quad \vdots \\
\overline{T}_{t^\star} &= \sum_{s=0}^{\tau-1} \alpha^s (1-\alpha) T_{t^\star-s} + \alpha^\tau \overline{T}_{t^\star-\tau}\,.
\end{aligned} \tag{6}$$

Based on Equation 6, by setting:

$$W_1 = \begin{bmatrix} (1-\alpha) & 0 & 0 & \dots & 0 \\ \alpha(1-\alpha) & (1-\alpha) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha^{H-1}(1-\alpha) & \alpha^{H-2}(1-\alpha) & \alpha^{H-3}(1-\alpha) & \dots & (1-\alpha) \end{bmatrix},$$

$$W_2 = \begin{bmatrix} 0 & \dots & 0 & \alpha \\ 0 & \dots & 0 & \alpha^2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \vdots & 0 & \alpha^H \end{bmatrix},$$

and $b_1 = b_2 = 0$,

It is possible to induce a recurrent neural network (RNN) to learn the behaviour defined by the exponential smoothing model, as set out in Equation 5. Nevertheless, this configuration results in the network optimizing a total of $\mathcal{O}(H^2)$ parameters. In the context of seeking novel approaches to temperature smoothing, the utilization of a recurrent neural network (RNN) is a logical choice. On the other hand, if the objective is to restrict the network to exponential smoothing, with optimization limited to the α smoothing coefficient, the necessity for optimizing a vast number of parameters renders the process complex and may ultimately prove inefficient. For this reason, we propose a new DNN layer that enables the efficient computation of one or more exponential smoothings over several batches, with only the smoothing coefficients as parameters to optimize.

Exponential Smoothing Layer Let's consider a batched input of size $B \in \mathbb{N}^\star$, containing the temperature from the days $t - B$ to t: $\mathbf{T}_{t-B:t} = [T_{t-B}, \dots, T_t] \in \mathbb{R}^{B \times H}$. The exponential smoothing layer first reshape this data into a size BH, to treat all sequences at once. We then use Equation 6, with $\tau = H \times B - 1$ to compute $\overline{\mathbf{T}}_{t-B:t}$:

$$\begin{bmatrix} \overline{T}_{t^\star-\tau} \\ \overline{T}_{t^\star-\tau+1} \\ \vdots \\ \overline{T}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ \alpha & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha^\tau & \alpha^{\tau-1} & \alpha^{\tau-2} & \dots & 1 \end{bmatrix} \begin{bmatrix} \overline{T}_{t^\star-\tau} \\ (1-\alpha)T_{t^\star-\tau+1} \\ \vdots \\ (1-\alpha)T_{t^\star} \end{bmatrix},$$

$$\overline{\mathbf{T}}_{t^\star-\tau:t^\star} = M \times \left[\overline{T}_{t^\star-\tau} \,|\, (1-\alpha)\mathbf{T}_{t^\star-\tau+1:t^\star}\right], \text{with}, \forall i \geq j : M_{i,j} = \alpha^{i-j}\,. \tag{7}$$

Finally, we reshape $\overline{\mathbf{T}}_{t^\star-\tau:t^\star} \in \mathbb{R}^{HB}$ back to the original shape $\overline{\mathbf{T}}_{t-B:t} \in \mathbb{R}^{H \times B}$. The matrix M is constructed during the forward pass as its shape and formula depend on the batch size B. Given that the coefficient α belongs to $[0, 1]$, it is encoded through a sigmoid: $\alpha = \text{Sigmoid}(\overline{\alpha}) = 1/(1 + \exp(-\overline{\alpha})) \in [0, 1]$ for $\overline{\alpha} \in \mathbb{R}$, where $\overline{\alpha}$ would be the weight optimized through back-propagation.

In our search space we let the optimization framework choose between the Exponential Smoothing Layer, a RNN layer, a Long-Short Term Memory (LSTM) layer and a Gated Recurrent Unit (GRU) layer, to perform the smoothing operation (see Section 4.2).

3.3 Online learning

The last layer of the search space from Keisler et al. [2024a] is a linear layer, transforming an input $h_t \in \mathbb{R}^{H \times D}$ into an output $y_t \in \mathbb{R}^H$, where y_t is the load consumption for the day t, H the number of instants during the day and $D \in \mathbb{N}^\star$ is the hidden dimension within the network before the last layer. Let's name $A_F \in \mathbb{R}^D$ and $b_F \in \mathbb{R}$ respectively the weights and bias matrices of this last MLP layer, we have:

$$y_t = A_F h_t + b_F = \sum_{i=1}^{D} a_F^i h_t^i + b_F. \tag{8}$$

To adapt our DNN, we use a daily Kalman state vector $\theta_t \in \mathbb{R}^D$ to adapt the coefficients of equation 8:

$$\tilde{y}_t = \theta_t^T (A_F h_t + b_F) = \sum_{i=1}^{D} \theta_t^i (a_F^i h_t^i + b_F) + \epsilon_t \tag{9}$$

$$\theta_{t+1} = \theta_t + \eta_t, \tag{10}$$

where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ and $\eta_t \sim \mathcal{N}(0, Q)$. Vilmarest [2022] suggests to use iterative grid search for $\sigma \in \mathbb{R}$ the diagonal coefficients of $Q \in \mathbb{R}^{D\times D}$. This search can be quite expensive, with a complexity of $\mathcal{O}(LD^2)$, where L is number of values that the coefficients of Q may take. We experimented empirically that the number of coefficients of the last MLP layer is usually larger than the number of coefficient of the GAMs. The iterative grid search was not usuable in this case, therefore we included the optimization of σ and the coefficients of Q within EnergyDragon search space (see Section 4.2.

4 Automated weather modeling

This Section presents the integration of the various elements presented in Section 3, namely the weather modeling and Kalman adaptation modules, into EnergyDragon with the objective of optimizing them.

4.1 Objective function

The objective is to identify the optimal DNN $\hat{f} \in \Omega$ with the lowest forecast error on a given load signal with a short forecast horizon (e.g., 24 hours). The load dataset, denoted by $\mathcal{D}$, contains the spatio-temporal W data, the explanatory variables C and the target (the load signal) Y. For any subset $\mathcal{D}_0 = ((W_0, C_0), Y_0)$, the forecast error ℓ_{MSE} is defined as:

$$\ell_{\text{MSE}} \colon \Omega \times \mathcal{D} \to \mathbb{R}$$
$$f \times \mathcal{D}_0 \mapsto \ell_{\text{MSE}}\big(f(\mathcal{D}_0)\big) = \ell_{\text{MSE}}\big(Y_0, f(W_0, C_0)\big) = \text{MSE}\big(Y_0, f(W_0, C_0)\big) .$$

Where MSE is the Mean Squared Error. Each DNN $f \in \Omega$ is parameterized by:

- $\alpha \in \mathcal{A}$, its architecture, optimized by the framework.
- $\lambda \in \Lambda(\alpha)$, its hyperparameters, optimized by the framework, where $\Lambda(\alpha)$ is induced by α. The hyperparameters include Q and σ from the Kalman adaptation. It should be noted that the shape of Q depends on the architecture and hyperparameters of the networks.
- $\delta \in \Delta(\alpha, \lambda)$, the DNN weights, where $\Delta(\alpha, \lambda)$ is generated by α and λ and optimized by gradient descent when training the model.

The optimization process is done in several steps. First, the optimal DNN weights $\hat{\delta} \in \Delta(\alpha, \lambda)$ for a given architecture $\alpha \in \mathcal{A}$ and set of hyperparameters $\lambda \in \Lambda(\alpha)$ are found using gradient descent over the training set $\mathcal{D}_{\text{train}} = \big((W_{\text{train}}, C_{\text{train}}), Y_{\text{train}})\big) = (X_{\text{train}}, Y_{\text{train}})$:

$$\hat{\delta} \in \underset{\delta\in\Delta(\alpha,\lambda)}{\operatorname{argmin}} \Big(\ell_{\text{MSE}}\big(f_\delta^{\alpha,\lambda}(X_{\text{train}}, Y_{\text{train}})\big)\Big) .$$

Once the DNN is trained, the performance of the selected α and λ are evaluated on $\mathcal{D}_{\text{valid}}$. As Q and σ are part of λ, the evaluation is done using the- Kalman recalibration of the model First, the state vector $\theta \in \mathbb{R}^{T\times D}$ is estimated on the last MLP layer of the trained DNN $f_{\hat{\delta}}^{\alpha,\lambda}$ using Equations 9 and 10. Let's have $\Theta_\lambda\big(f_{\hat{\delta}}^{\alpha,\lambda}(X_{\text{valid}})\big)$ the recalibration of $f_{\hat{\delta}}^{\alpha,\lambda}(X_{\text{valid}})$ as defined Equation 9. The architecture α and hyperparameters λ are optimized as:

$$(\hat{\alpha}, \hat{\lambda}) \in \underset{(\alpha,\lambda)\in(\mathcal{A}\times\Lambda(\alpha))}{\operatorname{argmin}} \left(\ell_{\text{MSE}}\Big(\Theta_\lambda\big(f_{\hat{\delta}}^{\alpha,\lambda}(X_{\text{valid}})\big), Y_{\text{valid}}\Big)\right) .$$

The framework output will be ℓ_{MAPE}, the Mean Absolute Percentage Error. Given a load series $Y = (\mathbf{y}_1 \dots \mathbf{y}_n)$ and the predictions $\hat{Y} = (\hat{\mathbf{y}}_1, \dots \hat{y}_n)$, $\text{MAPE}(Y, \hat{Y}) = 1/n \sum_{i=1}^{n} \left|(\mathbf{y}_i - \hat{\mathbf{y}}_i)/\mathbf{y}_i\right|$. The MAPE is computed using the DNN with the best architecture, hyperparameters, weights and calibration using Kalman on the test dataset:

$$\ell_{MAPE}\Big(\Theta_{\hat{\lambda}}\big(f_{\hat{\delta}}^{\hat{\alpha},\hat{\lambda}}(X_{\text{test}})\big), Y_{\text{test}}\Big) .$$

In the following section (4.2), we explicit our search space, defined by $\mathcal{A}$ and $\Lambda(\alpha)$.

4.2 Search space

The search space used in this work extends the one from Keisler et al. [2024a] by adding a weather modeling and a Kalman module, and is depicted Figure 1. Each DNN $f \in \Omega$ maps two batched inputs: $w_b \in \mathbb{R}^{B\times H\times V\times I}$ containing the spatio-temporal weather and $c_b \in \mathbb{R}^{B\times H\times F}$ containing the other explanatory variables into a target $y_b \in \mathbb{R}^{B\times H}$, where $B \in \mathbb{N}^\star$ represents the size of the batch.

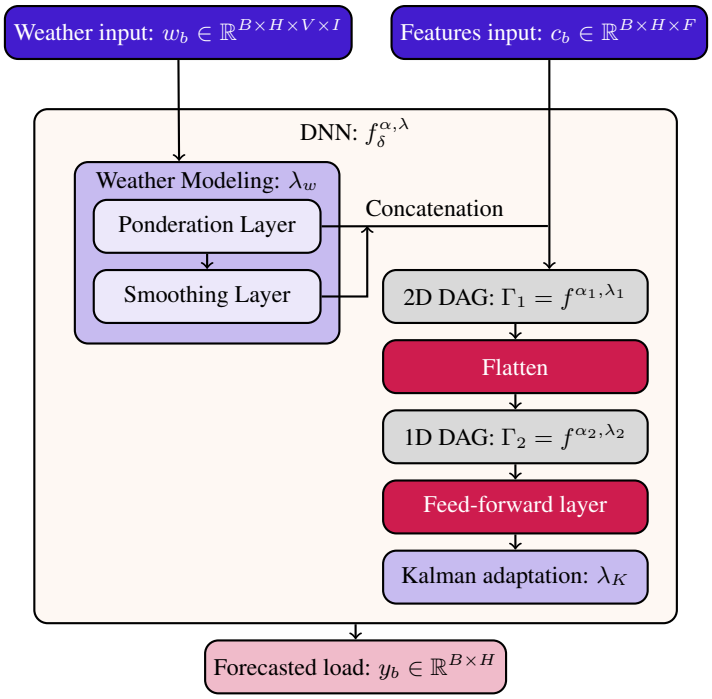


Figure 1: Daily meta-model for load datasets from Keisler et al. [2024a], with the integration of the weather modeling and the Kalman adaptation modules.

Weather Modeling Tdesignated as w_b, is initially processed by a Weather Modeling module containing V ponderation layers and a smoothing layer as defined Section 3. Each of the V weighting layers, designated as v is associated with an MLP layer, enabling the I signals to be weighted into F_v signals, with $I >> F_v$. This enables the network to identify multiple potential weightings. The $F_W = \sum_{v=1}^{V} F_v$ weighted signals are then concatenated into a vector of size $B \times H \times F_W$. The F_T signals corresponding to aggregated temperatures are smoothed by a smoothing layer being either a recurrent network (RNN, LSTM or GRU), or an exponential smoothing layer as defined in Section 3. They are then concatenated to the vector, now of size $B \times H \times (F_w + F_T)$. The set of hyperparameters for the Weather Modeling module is called λ_w and include the V output dimensions of the weighting layers for each of the V variables, as well as the type of layer used for smoothing along, with the hyperparameters associated with that layer.

Load Forecasting Network The vector generated by the Weather Modeling module is merged with the other vector of features, designated as c_b, resulting in a vector x_b of size $B \times H \times (F_W + F_T + F)$, which is then fed to the load forecasting model. This model is identical to the one used by Keisler et al. [2024a]. The load forecasting network should map $x_b \in \mathbb{R}^{B\times H\times(F_W+F_T+F)}$ into the target $y_b \in \mathbb{R}^{B\times H}$. For this, two DAGs Γ_1 and Γ_2 are used. The graph Γ_1 is made of 2-dimensional layers operations to treat the 2-dimensional input and is parameterized by α_1 and λ_1. A flattened layer follows Γ_1 to transform the 2-dimensional latent representation into a 1-dimensional one. The graph Γ_2 is then made of 1-dimensional layers operations and is parameterized by α_2 and λ_2. We have $\alpha = [\alpha_1, \alpha_2]$ and $\lambda = [\lambda_1, \lambda_2]$. A final output layer maps the output shape of Γ_2 to H.

Finally, a Kalman adaptation is made using two last hyperparameters Q and σ. We call λ_K this hyperparameter set. To summarize, our search space can be written as: $(\alpha, \lambda) = (\{\alpha_1, \alpha_2\}, \{\lambda_w, \lambda_1, \lambda_2, \lambda_K\}) \in \big(\mathcal{A} \times \Lambda(\alpha)\big)$.

5 Experiments

In this section, we evaluate the efficacy of our weather modeling and Kalman adaptation techniques on the French load dataset from January 2023 to May 2024. In contrast with the paper by Keisler et al. [2024a], which compares data from a relatively stable and distant period, our analysis focuses on a more dynamic and operational context. The training period spans from 2018 to 2022 and encompasses both the global pandemic caused by the SARS-CoV-2 virus and the subsequent energy crisis at the end of 2022. The test period encompasses the winter of 2023, during which consumers were encouraged to voluntarily reduce their consumption, a period commonly referred to as the "sobriety period".

5.1 Data

The load dataset was obtained from the website of the French Transmission System Operator (RTE)[3] and contains the French national load data at half-hourly intervals. Therefore, each day contains $H = 48$ time steps. The models were trained from January 2018 to December 2022 and subsequently evaluated from January 2023 to May 2024 using the MAPE. The weather data set comprises three-hourly weather forecasts for 32 weather stations across France (see Figure 2). These forecasts are provided by Meteo France [4]. Prior to employing this data in our forecasting models, we performed a temporal linear interpolation. The other explanatory features used to explain the load data are calendar features including the day of the week, the month, the year, and whether the day in question fell on a public holiday or a surrounding day.

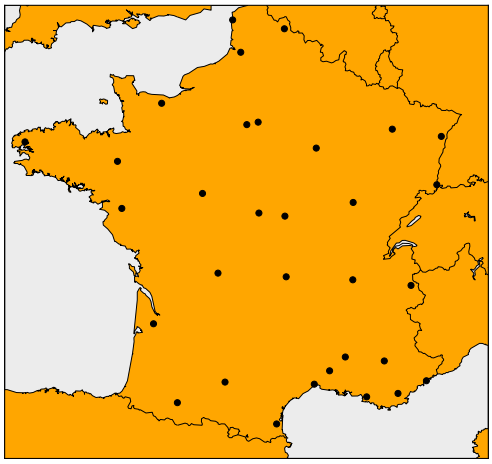


Figure 2: Location of the 32 french weather stations from our spatio-temporal weather dataset.

5.2 Baseline

We compare our results to models at the state-of-the-art in load forecasting: the day-ahead load forecast provided on RTE website, a Generalized Additive Model (GAM) used in the industry and EnergyDragon as proposed by Keisler et al. [2024a]. In the case of the GAM model, a single model is calibrated for each instant, resulting in a total of 48 models. The training set was modified by removing periods corresponding to lockdowns that were implemented during the pandemic caused by the novel coronavirus. EnergyDragon produces daily forecasts of $H = 48$ values, necessitating the use of a single model for all instants. A "Covid" feature has been incorporated into the model to indicate which periods corresponding to lockdowns are retained in the training set. With the exception of the "Covid" variable, the features are identical between the GAM and EnergyDragon models. The weather variables utilized in this study correspond to the weather at the 32 stations, with the data weighted and smoothed in accordance with the recommendations outlined in the RTE report on incorporating climatic contingencies into consumption forecasts[5]. Both models are recalibrated in an identical manner, utilising a Kalman filter that is updated on a daily basis with data from two days ago. To optimize Q and σ for the GAM model, an Iterative Grid Search was employed with the years 2020 to 2022 as validation set. For EnergyDragon, the years 2018 to 2020 were used as training dataset ($\mathcal{D}_{\text{train}}$) and the years 2021 and 2022 as validation set ($\mathcal{D}_{\text{valid}}$). Concerning the RTE model, no information is given on the structure of the model or its recalibration. For our model, later called ED Weather Modeling (for EnergyDragon Weather Modeling), which includes space-time weather modeling, we remove all weather-related features from EnergyDragon (all weather variables and their smoothed versions).

Model	MAPE	Recalibration	MAPE
RTE	-	Not specified	2.316
GAM	7.429	Kalman	2.019
EnergyDragon	**2.988**	Kalman	1.947
ED Weather Modeling	3.501	Kalman	**1.848**

Table 1: Results over 2023 - May 2024

5.3 Results

We evaluated each algorithm from the baseline on the French load signal. Both versions from EnergyDragon were run using 20 GPUs V100. The search algorithm used for the optimisation is the steady-state evolutionary algorithm used in [Keisler et al., 2024a]. The initial population is of size 100. Each algorithm was run with a global seed of 0 to ensure reproducibility. The results can be found in Table 1 and support the findings of [Keisler et al., 2024a]. Indeed, even during an erratic period, EnergyDragon managed to beat the static version of GAM. As anticipated, GAM's Kalman recalibration is superior to EnergyDragon's static version, thereby validating the incorporation of a DNN recalibration brick. This addition, in combination with the optimization of σ and Q, have proven to be effective, as evidenced by the superior performance of the recalibrated EnergyDragon model in comparison to both RTE and GAM-Kalman. With regard to the incorporation of the weather modeling module, in the recalibrated version it enables us to achieve a slight improvement (5%) in mean absolute percentage error (MAPE) compared to EnergyDragon.

During the optimization phase, we noticed that the exponential smoothing layer exhibits superior performance in comparison to alternative modules. As Figure 3 shows, there are rapidly no RNNs left in the new DNNs created. We noticed they are no longer employed after the 200th neural network is created (the initial population is 100 individuals, and ultimately, more than 2.000 models are evaluated during optimization). An examination of the output of the weather modeling module reveals the manner in which the DNN has modeled the weather. This modeled weather can then be compared to the data supplied by RTE, which was used in the GAMs and EnergyDragon models. Figures 4 and 5 show a comparison between the data as modeled by the functions given by RTE, and the one found by two DNNs using the Weather Modeling modules and achieving good performance (respectively 2.1% and 1.85% of MAPE). We can see that DNN's modeling close from that proposed by RTE, without being identical. It's interesting to notice, for example, that wind is almost identically represented, while for temperature we have two different aggregations. One is very close to the signal proposed by RTE, the other is opposite and larger in amplitude. As for smoothing, while Figure 5 has a smoothing fairly close to that used in the GAM and EnergyDragon models, for Figure 4 the first smoothing coefficient is much lower.

Finally Figure 6 shows the models found by EnergyDragon with and without the weather modeling part. We focused on the core of the network without showing the weather modeling brick, to only compare the load forecasting network. The structure of EnergyDragon with the weather modeling network shown Figure 6b is a lot simpler than the one without shown Figure 6a. It can be hypothesized that the weather is represented in a more comprehensible way for the DNN thanks to the weather modeling module. As a result, fewer transformations would be required to output the load forecast.

6 Conclusion

In conclusion, this article builds upon the work initiated by Keisler et al. [2024a] on automated deep learning for load forecasting. In this initial article, a framework, designated as EnergyDragon, was proposed for the optimization of both the architecture and hyperparameters of Deep Neural Networks, specifically designed for load forecasting. This article improves upon the previous work by incorporating an automated spatio-temporal weather modeling approach based on DNN and a recalibration module based on Kalman filtering. The efficacy of our approach is evaluated in a more dynamic and operational context, namely the national French load during the sobriety period.

[3] https://www.rte-france.com/eco2mix

[4] https://www.data.gouv.fr/fr/organizations/meteo-france/

[5] https://www.services-rte.com/files/live/sites/services-rte/files/documentsLibrary/2022-04-01_REGLES_MA-RE_SECTION_2_F_3590_en

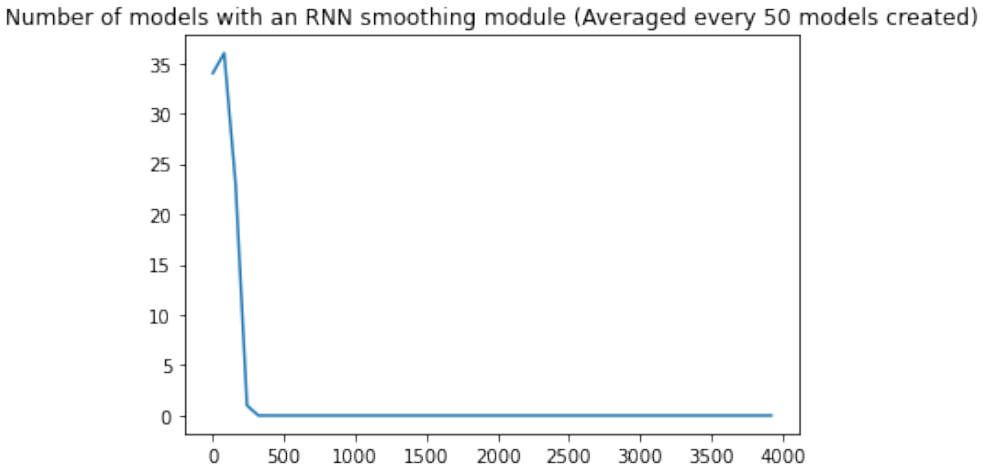
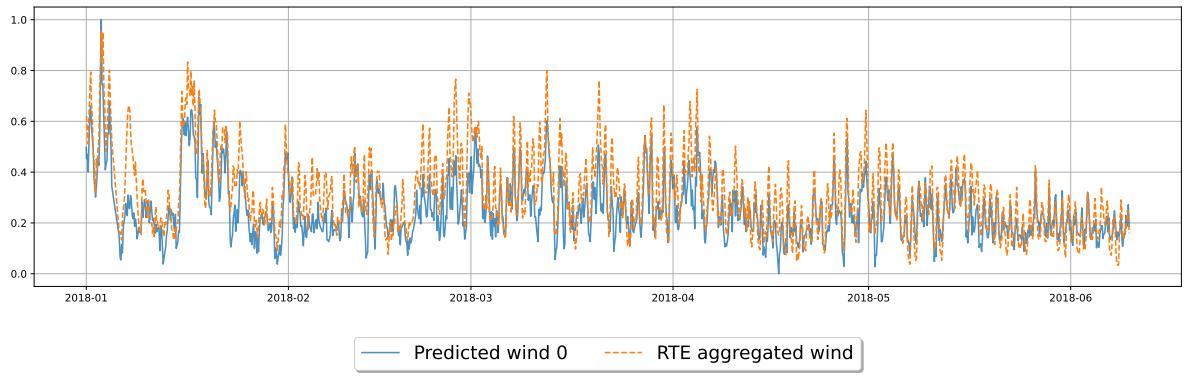
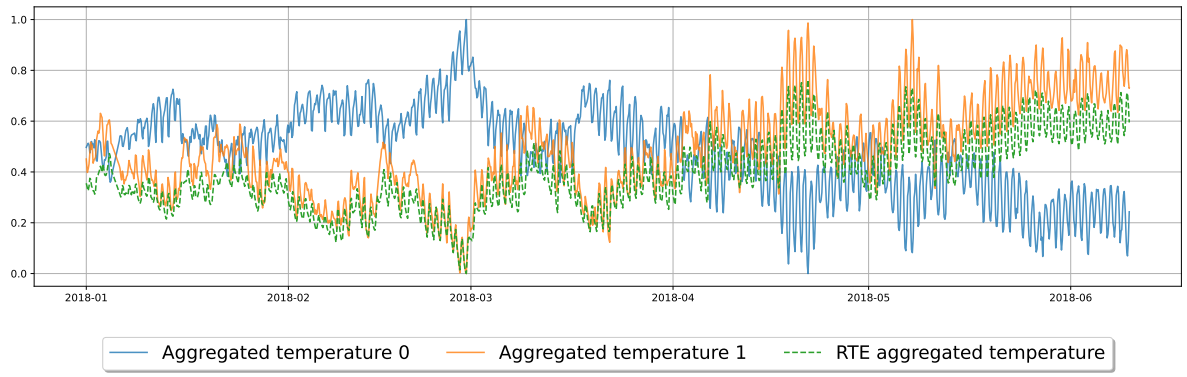


Figure 3: Number of DNNs created having a Recurrent Neural Network as smoothing layer through time.

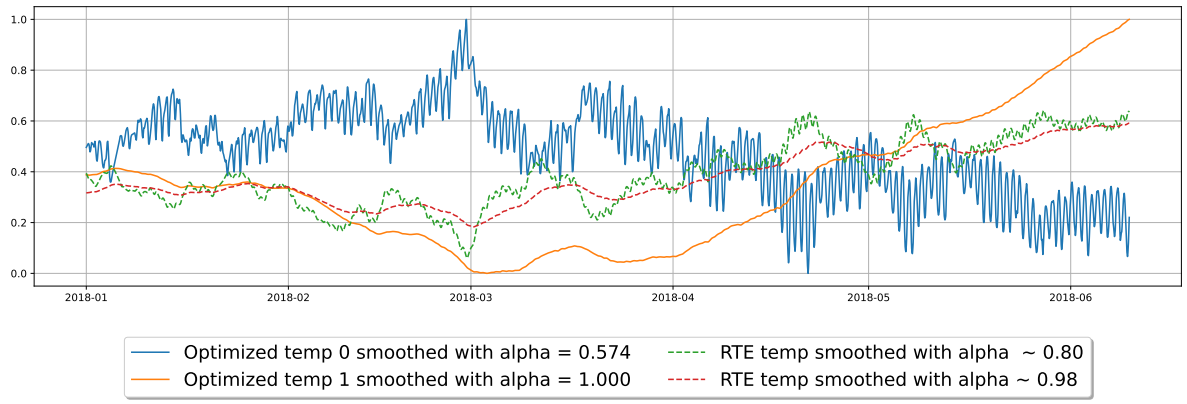
To automate the spatio-temporal representation of weather, we have maintained a close alignment with the functions employed in the state of the art for load forecasting. In the Section 5, we demonstrate that the representations identified by our DNNs are close to those used in the other models from our baseling. This approach offers the advantage of remaining interpretable, enabling a comparison between the DNN-generated model and the insights derived from domain expertise. However, these preliminary results could probably be further enhanced by incorporating more sophisticated DNNs layers and employing over-parameterization.



(a) Dotted line: the wind signal aggregated using weights from RTE, used in the GAM and EnergyDragon models. Solid line: the wind signal aggregated by the weather modeling module.

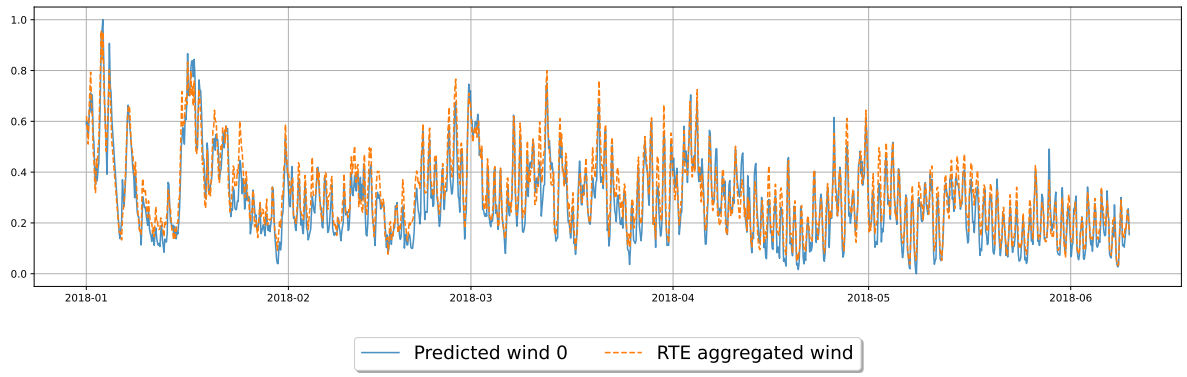


(b) Dotted line: the temperature signal aggregated using weights from RTE, used in the GAM and EnergyDragon models. Solid lines: two aggregated temperature signals found by the weather modeling module.

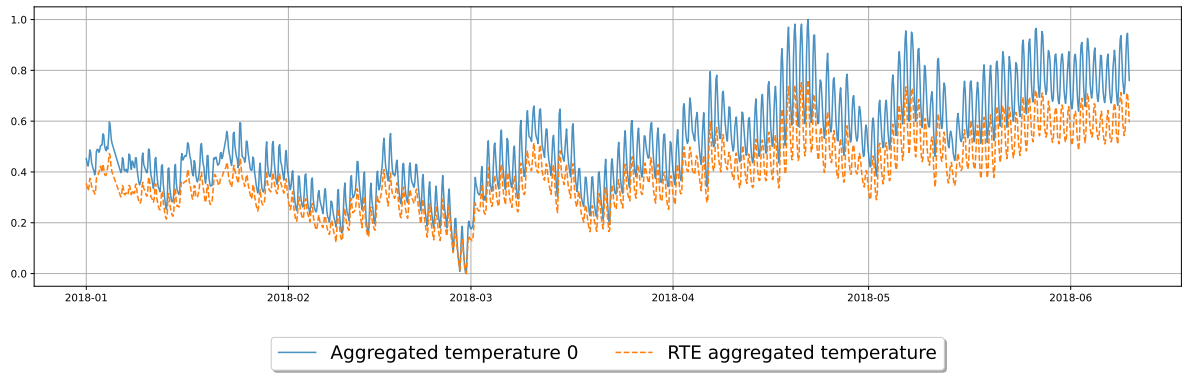


(c) Dotted line: the smoothed temperature signals used in the GAM and EnergyDragon models. Solid lines: the two aggregated temperature signals smoothed by the weather modeling modeling module.

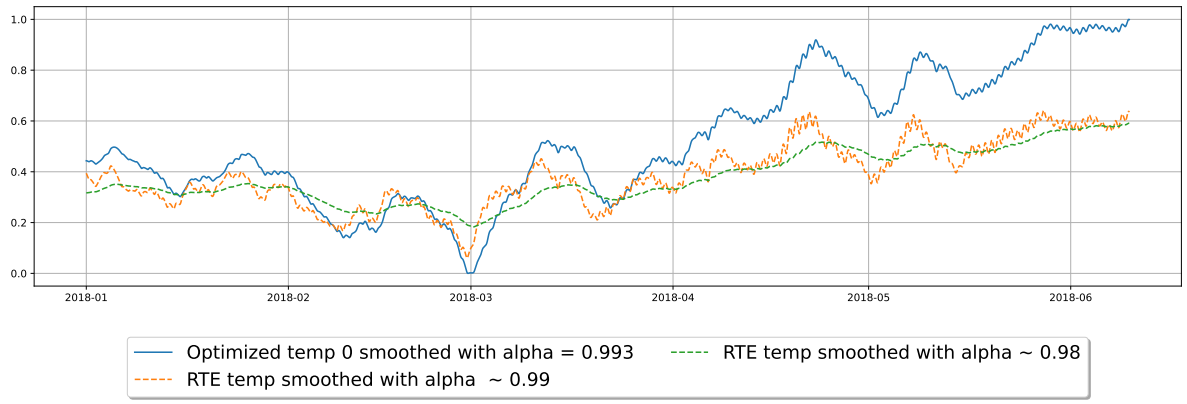
Figure 4: Comparison between the weather as modeled within the GAM and EnergyDragon models, versus the weather modeled by the DNN based Weather Modeling module, for a model having a MAPE of 2.1%



(a) Dotted line: the wind signal aggregated using weights from RTE, used in the GAM and EnergyDragon models. Solid line: the wind signal aggregated by the weather modeling module.

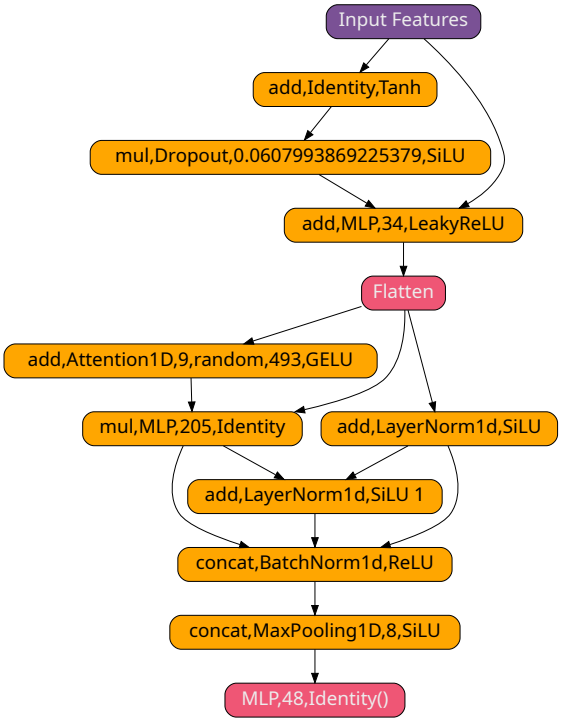


(b) Dotted line: the temperature signal aggregated using weights from RTE, used in the GAM and EnergyDragon models. Solid line: the aggregated temperature signal found by the weather modeling module.

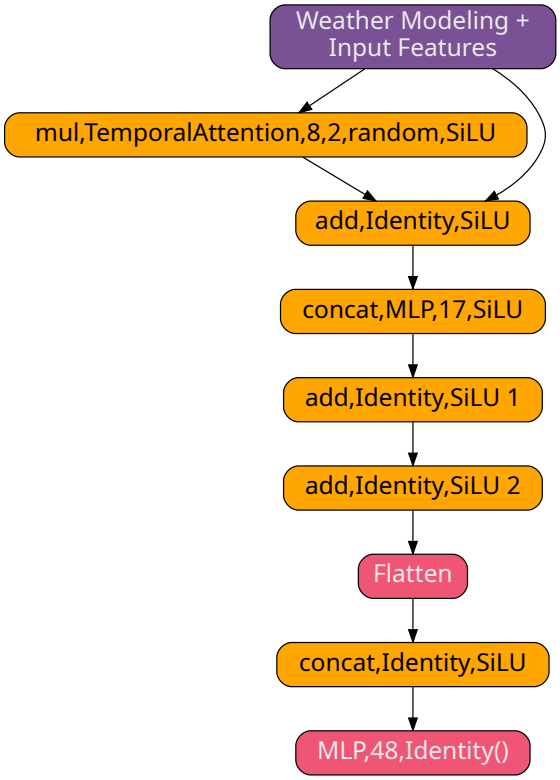


(c) Dotted line: the smoothed temperature signals used in the GAM and EnergyDragon models. Solid line: the aggregated temperature signal smoothed by the weather modeling modeling module.

Figure 5: Comparison between the weather as modeled within the GAM and EnergyDragon models, versus the weather modeled by the DNN based Weather Modeling module, for a model having a MAPE of 1.85%



(a) Best model found by EnergyDragon without the automated weather modeling part.



(b) Best model found by EnergyDragon with the automated weather modeling part.

Figure 6: Best models found by EnergyDragon without and with the automated weather modeling module.

References

A. Ba, M. Sinn, Y. Goude, and P. Pompey. Adaptive learning of smoothing functions: Application to electricity load forecasting. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper_files/paper/2012/file/b571ecea16a9824023ee1af16897a582-Paper.pdf`.

J. De Vilmarest and Y. Goude. State-space models for online post-covid electricity load forecasting competition. *IEEE Open Access Journal of Power and Energy*, 9:192–201, 2022.

M. Farrokhabadi, J. Browell, Y. Wang, S. Makonin, W. Su, and H. Zareipour. Day-ahead electricity demand forecasting competition: Post-covid paradigm. *IEEE Open Access Journal of Power and Energy*, 9:185–191, 2022.

J. Keisler, S. Claudel, G. Cabriel, and M. Brégère. Automated deep learning for load forecasting. *arXiv preprint arXiv:2405.08842*, 2024a.

J. Keisler, E.-G. Talbi, S. Claudel, and G. Cabriel. An algorithmic framework for the optimization of deep neural networks architectures and hyperparameters. *Journal of Machine Learning Research*, 25(201):1–33, 2024b. URL `http://jmlr.org/papers/v25/23-0166.html`.

R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, et al. Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*, 2022.

A. McNally, C. Lessig, P. Lean, E. Boucher, M. Alexe, E. Pinnington, M. Chantry, S. Lang, C. Burrows, M. Chrust, et al. Data driven weather forecasts trained and initialised directly from observations. *arXiv preprint arXiv:2407.15586*, 2024.

R. Nedellec, J. Cugliari, and Y. Goude. Gefcom2012: Electric load forecasting and backcasting with semi-parametric models. *International Journal of forecasting*, 30(2):375–381, 2014.

J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

A. Pierrot and Y. Goude. Short-term electricity load forecasting with generalized additive models. *Proceedings of ISAP power*, 2011, 2011.

J. D. Vilmarest. *State-Space Models for Time Series Forecasting. Application to the Electricity Markets. (Modèles espace-état pour la prévision de séries temporelles. Application aux marchés électriques)*. PhD thesis, Sorbonne University, Paris, France, 2022. URL `https://tel.archives-ouvertes.fr/tel-03783480`.