
AutoML Algorithms for Online Generalized Additive Model Selection: Application to Electricity Demand Forecasting

Keshav Das¹ Julie Keisler^{1,2} Margaux Brégère^{1,3} Amaury Durand¹

¹EDF R&D, Lab Paris-Saclay

²Inria Paris

³LPSM, Sorbonne Université

Abstract Electricity demand forecasting is key to ensuring that supply meets demand lest the grid would blackout. Reliable short-term forecasts may be obtained by combining a Generalized Additive Models (GAM) with a State-Space model, leading to an adaptive (or online) model. A GAM is an over-parameterized linear model defined by a formula and a state-space model involves hyperparameters. Both the formula and adaptation parameters have to be fixed before model training and have a huge impact on the model's predictive performance. We propose to optimize them using automated Machine Learning. For this purpose, we define an efficient modeling of the search space (namely, the space of the GAM formulae and adaptation parameters) as well as mutation and crossover operators on this space and apply an evolutionary algorithm to select the best parameters. We evaluate our method on short-term French electricity demand forecasting which demonstrates the relevance of the approach.

1 Introduction

Given the challenges of storing electricity on a large scale, it is essential to forecast electricity demand as accurately as possible to ensure an efficient balance between production and demand, thus maintaining the continuous operation of the electricity system. Hence, power stations can calibrate their output by knowing the required electricity demand at any given moment. Generalized Additive Models (GAM) are highly effective electricity demand forecasting models that have been widely used over the last two decades to forecast short- and mid-term demand (ranging from a day to a couple of years) - see, among others, Pierrot and Goude (2011) and Wood et al. (2015). They model the expected value of the random target variable Y with a sum of potentially nonlinear covariate effects, which we call a formula. The forecasts' accuracy depends entirely on the choice of terms in this sum, namely, for each term (or effect), the covariate or covariates and their relationship (linear, polynomial, etc.) with the response variable. Further, in Section 3, we will detail how such a formula f characterizes a GAM.

GAMs are widely used in the electricity field because they combine the flexibility of fully non-parametric models with the simplicity of multiple regression models. Indeed, they are trained on large historical data sets (several years) and make the underlying assumption that the distribution of the target variable is stationary conditionally to covariates. However, they fail to capture recent changes in electricity demand behavior (sobriety, new electrical appliances, etc.). To overcome this issue, models can be made adaptive in a second phase. Specifically, forecasters use the latest data to dynamically reweight the terms in the GAM sum; see Ba et al. (2012) and Obst et al. (2021) for further details. This procedure is detailed in Section 3. We will see that it depends on some hyperparameter Q which needs to be fixed before the adaptation process. Thus, to create a reliable GAM for electricity demand forecasting, forecasters must set a GAM formula and the hyperparameter for

The code is available here: <https://anonymous.4open.science/r/dragam-88D0>

its adaptive version. Then, the model is estimated with the Penalized Iterative Re-Weighted Least Squares (P-IRLS) method from Wood et al. (2015), implemented in the `mgcv` R-package.

Choosing the formula and the adaptive hyperparameter is therefore crucial to obtain a good electricity demand model (i.e. one that is efficient or simple to interpret, etc.). Testing a large number of models (i.e., running the P-IRLS algorithm on many formulae) can be challenging, time-consuming, and computationally expensive. In this work, we propose to automate these choices using Automated Machine Learning (AutoML) algorithms. The latter estimate the best adaptive model (f^*, Q^*) in a pool of models Ω containing all the candidates. To choose among candidates, they need to evaluate the performance of any model $(f, Q) \in \Omega$: this is the costly part because it requires training the model and evaluating its performance on a validation data set using a loss function $\ell : \Omega \rightarrow \mathbb{R}$. A search algorithm is used to select the most promising models to be tested. A summary of the AutoML framework as well as an outline of the paper are illustrated in Figure 1.

$$\begin{array}{c}
 \text{Search Algorithm (Section 5)} \\
 (f^*, Q^*) \in \underset{(f, Q) \in \Omega}{\text{argmin}} \ell(f, Q) . \quad \text{Performance Evaluation (Section 4)} \\
 \text{Search Space (Section 3)}
 \end{array}$$

Figure 1: Summary of the AutoML framework.

After a literature discussion in Section 2, we define the search space in which the AutoML algorithm will search for its solutions in Section 3. In Section 4, we focus on performance evaluation, while Section 5 discusses the search algorithms we consider. Finally, we demonstrate the relevance of our method on French electricity demand data in Section 6.

2 Literature discussion

AutoML. Automated Machine Learning is made of two main subproblems: model selection and hyperparameter optimization, which can be tackled jointly with the Combined Algorithm Selection and Hyperparameter (CASH) optimization problem (Feurer et al., 2015). The state of the art in AutoML offers both very general algorithms for selecting models and/or hyperparameters that perform well for any type of machine learning model desired (see, for example Akiba et al. 2019 or Tang et al. 2024), and more specialized algorithms for certain types of specific machine learning models, such as models from the scikit-learn library (Feurer et al., 2022) or neural networks (White et al., 2023). To our knowledge, GAMs have only been the subject of one specific approach, GAGAM (Cus, 2020), which is described below. While generic approaches can be applied to GAMs to a certain extent, they quickly become limited when it comes to capturing the hierarchical structure of formulae.

Model selection for GAMs. Given a set of features, along with some training data, the Genetic (Evolutionary) Algorithm for Generalized Additive Models (GAGAM) developed in the R language by Cus (2020) applies a sequence of crossovers and mutations to subsets of a randomly initialized population. After each sequence of evolutions, the models are ranked according to their loss, which is calculated using the input training data. Eventually, the aim is that the algorithm will retain the models with the best formula, constructed using some of the input features and by deciding whether they should enter linearly or not. However, the same type of spline and the same degrees of freedom are applied to all nonlinear features, which restrains the search space. Moreover, bi-variate effects are not supported simultaneously with linear features.

Dragon. DRAGON, or DiRected Acyclic Graphs optimization, is an open-source Python package offering tools to conceive Automated Machine Learning frameworks for diverse tasks. The package is based on three main elements: building bricks for search space design, search operators to modify those bricks and search algorithms. Originally developed for Deep Neural Networks optimization, the package is based on the encoding of various computer objects (integers, floats, lists, etc.) and can be easily extended to other optimization tasks. Search operators can be used to create neighbors or mutants of a solution, allowing local search or evolutionary approaches to converge on optimal solutions. The DRAGON package implements four search algorithms based on the search operators: a random search, an asynchronous evolutionary algorithm (Keisler et al., 2024b), Hyperband (Li et al., 2018), and Mutant-UCB (Br  g  re and Keisler, 2024). In this work, we used DRAGON’s various building blocks to encode GAMs. We then adapted the search operators to our representation, allowing us to directly use the search algorithms in the package.

3 Search space

Introduced by Hastie and Tibshirani (1986) in 1986, Generalized Additive Models (GAM) are semiparametric models widely used for electricity demand forecasting. Before training a GAM and using it for electricity demand forecasting, the GAM formula - i.e. the equation which links the expectation of the target variable Y with the covariates X - and the hyperparameters for its adaptive version must be defined. We detail these two steps below.

3.1 Fixed setting

With g a link function (e.g., identity or logarithmic function), a GAM models the value $g(\mathbb{E}[Y|X])$ as a sum of smooth functions of the covariate X . With K the number of terms (also called effects) in the sum, let, for any $k = 1, \dots, K$, J_k be the set of indices of the covariates associated with the effect k and f_k the function that models the relationship between the target variable Y and the subset X_{J_k} of the covariates X ; a GAM has a structure like:

$$g(\mathbb{E}[Y|X]) = \sum_{k=1}^K f_k(X_{J_k}).$$

As it is classically done for electricity demand forecasting, in all that follows, and without loss of generality, we will consider additive models, namely g is equal to the identity. In our context, the target variable is a time series, and for a time t , Y_t is the amount of electricity demanded between t and $t + 1$ (in the experiments of Section 6, the time step is half an hour). The latter depends on various variables Z_t , including weather (temperature, wind, etc.) and calendar-related (public holidays, weekdays, etc.) factors. Electricity consumers do not respond instantaneously to changes in the weather, among other things, because of the thermal inertia of buildings. This is why (recent) past weather may influence electricity demand. To be as general as possible, the set of covariates X_t includes present and past covariates, so $X_t = (Z_s)_{s=1, \dots, t}$. Therefore, we obtain the equation $\mathbb{E}[Y_t|X_t] = \sum_{k=1}^K f_k(X_{J_k,t})$. It remains to specify the smooth functions f_k to get a proper formula. This task is divided into two: feature engineering (to extract useful information from $X_{J_k,t}$) and relationship specification (to link this information with the target variable).

Feature engineering. For some $k = 1, \dots, K$, feature engineering for covariates X_{J_k} can provide summarized information and prove extremely valuable. For example, taking into account an exponentially smoothed temperature or selecting a subset of categories from a categorical variable can be achieved by designing parameterized functions denoted by ψ_k . These two possibilities are fully detailed in Appendix A. Only the parameters of the feature engineering functions have to be set-up in the formula and these will be the ones optimized by the AutoML algorithm.

<https://dragon-tutorial.readthedocs.io/en/latest/>

Relationship specification. For every $k = 1, \dots, K$, we need a basis of functions in which f_k may be represented. To define the formula, this basis, completely specified by its functions $(b_k^i)_{i=1, \dots, q_k}$, where q_k is the size of the basis, has to be picked. In our set-up, this means choosing q_k and the type of functions in the basis (linear, cubic spline, tensor product, etc.) - we refer to Appendix A for further details.

Overall, a GAM can be seen as an over-parameterized linear model. The number of effects K , and for each $k = 1, \dots, K$, the subset J_k , the function for potential feature engineering ψ_k , the size q_k and the spline basis $(b_k^i)_{i=1, \dots, q_k}$ have to be chosen by the forecaster. It defines the formula f and implicitly assumes that:

$$f(X_t) = \mathbb{E}[Y_t | X_t] = \sum_{k=1}^K \sum_{i=1}^{q_k} \beta_{k,i} b_{k,i}(\psi_k(X_{J_k,t})). \quad (1)$$

The coefficients $\beta_{i,k}$ are estimated using the Penalized Iterative Re-Weighted Least Squares (P-IRLS) method from Wood et al. (2015), implemented in the `mgcv` R-package. This last step can be considered as *model training*, while the choice of the formula corresponds to *model optimization*.

3.2 Adaptive setting

GAMs have a good ability to generalize relations between inputs and an output, but they lag behind in terms of adaptability to new data. Hence, adaptive GAMs have been introduced by Ba et al. (2012) to overcome this issue. They are well suited for short-term demand forecasting as they take into account recent data and catch changes in the target variable distribution. With $\mathbf{f}(X_t)$ the K -dimensional vector containing all the effects of the estimated GAM $f_k(X_{J_k,t})$, for $k = 1, \dots, K$; we consider the state-space model approach introduced by Obst et al. (2021) and assume, for $t = 2, \dots$,

$$\begin{cases} Y_t &= \theta_t^\top \mathbf{f}(X_t) + \varepsilon_t, & \text{with } \varepsilon_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2) \\ \theta_t &= \theta_{t-1} + \eta_t, & \text{with } \eta_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \bar{Q}). \end{cases}$$

The variance σ^2 and covariance $K \times K$ matrix $\bar{Q}$ of the Gaussian white noises $(\varepsilon_t)_t$ and $(\eta_t)_t$, respectively, parameterize the model. This is the setting of Kalman filtering so we may use the recursive formulae of Kalman providing the expectation of θ_t given the past observations, yielding to a forecast for Y_t . As described in Obst et al. (2021), the Kalman filter is crucially dependent on the hyperparameter $Q = \bar{Q}/\sigma^2$. It also requires some initialization parameters that we fix in this work. Thus an adaptive version of a GAM relies on the choice of the parameter Q , a $K \times K$ -matrix. In all that follows, we consider only diagonal matrices.

Overall, the formula f of a GAM and the parameter Q of the associated state-space model for the adaptive version can be considered as hyperparameters from the standpoint of machine learning; they cannot be modified during training but are significantly tied to a model's accuracy.

3.3 Encoding through DRAGON

As mentioned above, we used the DRAGON package to build our search space. DRAGON provides a set of variables for encoding different computer objects. Each element of the search space is represented by a fixed size array containing f and potentially Q in the adaptive setup. The variables f and Q are arrays of variable size, with a dimension K equal to the number of effects in f . Each element of Q is a float, while each element of f corresponds to an effect. Each effect is made up of several elements, depending on the type of effect. First, one or two categorical variables corresponding to the features (depending on whether we are in a univariate or bivariate case), from the covariate set Z . Depending on the chosen feature(s), components of different types (categorical, floats or integers) can also be added. The variables corresponding to the features and

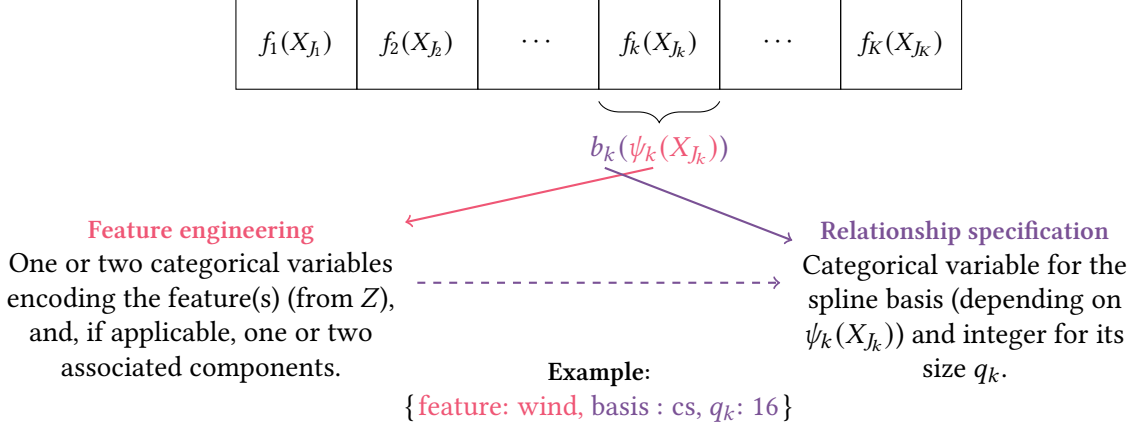


Figure 2: Encoding of a GAM formula f using the DRAGON package.

their components, if any, are used to construct the value $\psi_k(X_{J_k})$ for each time step t . Depending on the value of $\psi_k(X_{J_k})$, several spline basis b_k can be used. A categorical variable specifies the one chosen for the effect k . Finally, a last attribute q_k , an integer, determines the basis size. The search space is hierarchical in the sense that the elements of a given effect k depend on the feature(s) chosen for that effect. Moreover, at the scale of the variable-size array encoding the f formula, the addition of a new effect is done in such a way as to avoid any duplication of a given $\psi_k(X_{J_k})$. An illustration of the encoding of a formula f can be found Figure 2. The process of constructing a GAM formula using DRAGON’s tools is fully described in Algorithm 1 in the appendix and works as follows: first, the covariate, or the pair of covariates for bi-variate effects, is selected. Then comes the feature engineering part, and subsequently, the relationship is chosen. Finally, the coefficients Q are also generated. Tuning a GAM with the P-IRLS algorithm can be a long and tedious job. We then aim to apply efficient model selection algorithms to optimize them. We just define the search space Ω that contains all the possible models we may consider, so basically the pairs of formulae and hyperparameters (f, Q) . We now have to define a loss function $\ell : \Omega \rightarrow \mathbb{R}$ so that we can compare the models with each other.

4 Performance evaluation

We recall that we aim to find the best pair $(f^*, Q^*) \in \Omega$ and that this is done by minimizing a loss function $\ell : \Omega \rightarrow \mathbb{R}$. For a model $(f, Q) \in \Omega$, we denote by $\hat{f}$ the fixed generalized additive model obtained by running the P-IRLS algorithm implemented in the `mgcv`-package on a training data set $\mathcal{D}_{\text{TRAIN}} = (Y_t, X_t)_{t=1, \dots, \tau_1}$. This algorithm estimates the coefficients of Equation (1) by performing a regularized over-parameterized linear regression. The regularization factor, optimized using a cross-validation criterion, avoids over-fitting and ensures that covariate effect functions are smoothed. This criterion involves the effective degrees of freedom of a GAM. It is defined by the trace of the influence matrix: the matrix A such that, for any $t = 1, \dots, \tau_1$, $\hat{f}(X_t) = AY_t$. We refer to Chapter 4 of Wood (2017) for further details. The effective degrees of freedom of a GAM $\hat{f}$ may be interpreted as a (possibly non-integer) number of parameters of the model and is thus a good measurement of its complexity; it is denoted by $\text{DF}(\hat{f})$. For any time step t , this fixed model provides the forecast $\hat{Y}_t = \hat{f}(X_t) = \sum_{k=1}^K \hat{f}_k(X_{J_k, t})$. To evaluate the quality of the forecasts provided by the model (f, Q) on a validation data set $\mathcal{D}_{\text{VALID}} = (Y_t, X_t)_{t=\tau_1+1, \dots, \tau_2}$ in an adaptive setting, the forecasts are modified. Indeed, for each time step t of the data set, the model provides the forecast $\hat{Y}_t = \sum_{k=1}^K \hat{\theta}_t \hat{f}_k(X_{J_k, t})$, where the adaptive vector $\hat{\theta}_t$ was previously updated using the Kalman filter equations. This calculation involves, among other things, the prediction error $(Y_{t-1} - \hat{Y}_{t-1})$ and

the hyperparameter Q (for further details, see, e.g. Obst et al., 2021). We consider the Root Mean Squared Error (RMSE) to evaluate the prediction on $\mathcal{D}_{\text{VALID}}$. Maximizing prediction accuracy while avoiding overcomplicated models led us to define, for any model (f, Q) , the following loss function:

$$\ell(f, Q, \mathcal{D}_{\text{VALID}}) = \sqrt{\frac{1}{\tau_2 - \tau_1} \sum_{t=\tau_1+1}^{\tau_2} (Y_t - \hat{Y}_t)^2} + \eta \text{DF}(\hat{f}). \quad (2)$$

and $\eta \in \mathbb{R}^+$ is a regularization parameter which balances the two terms of the loss: the RMSE on the validation data set and the degrees of freedom. We emphasize that running the Kalman filter on the whole validation data set is computationally expensive: in our model selection framework, both training and testing are costly.

5 Search algorithms

We now focus on the AutoML strategies we consider to solve the minimization problem $\text{argmin} \ell(f, Q)$ over Ω . It should first be pointed out that once a formula f has been selected, there are several statistical approaches to optimize the choice of the hyperparameter Q . For example, in Chapter 5 of de Villemarest (2022), Algorithm 5 proposes an iterative grid search procedure to tune Q . We denote by $Q_{\text{IGS}}(\hat{f}, \mathcal{D}_{\text{TRAIN}}, Q_0, i)$ the matrix output by this algorithm, after i iterations, for a fixed trained model $\hat{f}$ and an initial matrix Q_0 . In the following, we explore algorithms which make use of this iterative grid search optimization and others which view Q as a classical hyperparameter. We used the tools in the DRAGON package to encode the various elements of the search space as detailed in Section 3. We then modified the search operators to use our representations with the package’s search algorithms. We first designed a procedure to randomly generate a model (f, Q) and thus create an initial population. It is detailed in Algorithm 1 of Appendix B. Then, we defined specific mutation and crossover operators useful for the Evolutionary Algorithm. In comparison with the search operators that are already present in DRAGON, the ones that have been implemented prevent a covariate from intervening more than once in the formula. This is done to guarantee interpretability. Below, we detail these two processes.

Crossover. We used a two-point crossover between two GAMs (f^1, Q^1) and (f^2, Q^2) to create two offsprings (f^{12}, Q^{12}) and (f^{21}, Q^{21}) . It can be done on the formulae of respectively $K^1 < K^2$ terms and eventually on the adaptation parameters as illustrated below. The effects from the formulae and the elements from the matrix are swapped, just like bits in the regular two-points crossover. This is illustrated in Figure 3.

$$\begin{array}{ll} f^1(X) = f_1^1(X_{J_1^1}) + f_2^1(X_{J_2^1}) + f_3^1(X_{J_3^1}) + \dots + f_{K_1}^1(X_{J_{K_1}^1}) & Q^1 = \text{diag}(q_1^1, \dots, q_{K_1}^1) \\ f^2(X) = f_1^2(X_{J_1^2}) + f_2^2(X_{J_2^2}) + f_3^2(X_{J_3^2}) + \dots + f_{K_2}^2(X_{J_{K_2}^2}) & Q^2 = \text{diag}(q_1^2, \dots, q_{K_2}^2) \\ \downarrow & \downarrow \\ f^{12}(X) = f_1^1(X_{J_1^1}) + f_2^2(X_{J_2^2}) + f_3^2(X_{J_3^2}) + \dots + f_{K_1}^1(X_{J_{K_1}^1}) & Q^{12} = \text{diag}(q_1^1, \dots, q_{K_1}^2) \\ f^{21}(X) = f_1^2(X_{J_1^2}) + f_2^1(X_{J_2^1}) + f_3^1(X_{J_3^1}) + \dots + f_{K_2}^2(X_{J_{K_2}^2}) & Q^{21} = \text{diag}(q_1^2, \dots, q_{K_1}^1, \dots, q_{K_2}^2) \end{array}$$

Figure 3: Illustration of the crossover.

Mutation. The mutation operator creates a new model from an existing one (f, Q) . It may impact a given effect in the GAM formula f by randomly changing the covariate(s), the relationship with the target variable (and, when it is relevant, the number of splines in the basis), or the feature

engineering function. Multiple characteristics can be changed at once. The mutation can also be an effect addition (the effect is then randomly generated) or deletion. We ensure that the new formula is non-empty and that none of the effects appear twice. The coefficient in the matrix Q can be randomly mutated as well. The size of Q matrix is automatically adjusted to the number of terms in the formula.

We are now able to extend the search algorithms from DRAGON to GAM selection. For more information on these algorithms, please refer to Keisler (2025). In the experiments of Section 6, we will consider an evolutionary algorithm (EA). We propose two versions, which are fully described in Algorithm 2 of Appendix B. For the first version, the EA is performed only on the formulae. This first algorithm, referred to as $EA(f) + Q_{IGS}$, trains the models with the P-IRLS algorithm so the evolutionary algorithm outputs a fixed model. We finalize the optimization by running the iterative grid search until convergence. For the second one, the adaptive hyperparameter is considered to be part of the search space (so f and Q are optimized at the same time) and we refer to it as $EA(f, Q)$.

6 Experiments: short-term electricity demand

6.1 Experiment design

Data sets and adaptation procedure. Our experiments consist in forecasting the French hourly electricity demand on a short-term horizon. The open source data set comes from the French Transmission System Operator. The set of covariates contains weather (temperature, cloud cover and wind) and calendar covariates. We work within an operational framework: each week we receive electricity demand data with a delay. Specifically, we assume that every Monday at 8:00 we access new electricity demand records: data from Saturday 00:00 nine days earlier to the previous Friday 23:00. Then, we update our models (namely the vector θ_t introduced in Section 3.2) using these new data points and we predict the demand for the next seven days (from Monday 8:00 to next Monday 7:00). Models were fitted on the training data set $\mathcal{D}_{\text{TRAIN}}$, which contains electricity demand, weather and calendar information from 2017 to 2020. Since they completely differ from the rest of the data and to avoid some bias on the forecast, we remove the periods of lockdown due to the COVID-19 pandemic from the training data set. The validation data set $\mathcal{D}_{\text{VALID}}$ used to evaluated the models (i.e. to compute the loss defined in Section 4) consists of data from the years 2021 and 2022. Finally, models were tested on $\mathcal{D}_{\text{TEST}}$: the years 2023 and 2024.

Algorithms. The performance of the models found by the proposed approach is compared with that of a state-of-the-art handcrafted model detailed in Appendix C and with that of the model found by GAGAM (the R-package developed by Cus, 2020). As the effect of the hour $H_t \in \{0, \dots, 23\}$ is crucial to forecast electricity demand, it is often more efficient to consider a model per hour (see Fan and Hyndman, 2011 and Goude et al., 2013). Thus, we run the algorithms for each hour, independently. The training of the handcrafted model (referred to as “SotA” in what follows) and the one output by GAGAM (“GAGAM”) is completed with the optimization of the adaptation parameter using the iterative grid-search algorithm (“+ Q_{IGS} ”). We executed the proposed AutoML algorithm with a budget of $T = 200$, which means that 200 calls to the P-IRLS algorithm will be made during the entire runtime. All experiments were made with a population of $M = 20$ individuals. Our search space, well designed for our problem, allows us to represent the handcrafted model in order to propose it in the initial population of both EAs. For these search algorithms, the tournament selection size was chosen equal to $k = 5$ (see Algorithm 2 in Appendix).

Search space. The search space defined in section 3 is parameterized by several elements, including features, feature modifications, spline bases and the size of these bases. We have chosen these elements based on our industrial knowledge to represent the largest number of GAM formulae that

we know are relevant to our problem. These choices impact the performance, particularly as we have had to keep the size and complexity of our GAM formulae to a minimum in order to avoid excessive computation time.

Performance evaluation. The evaluation of a model (f, Q) starts with its training on $\mathcal{D}_{\text{TRAIN}}$. To get the forecasts $\hat{Y}_t$ defined in Section 4, the operational adaptation procedure described above involving Q and Kalman filtering is then run. For $EA(f, Q)$, we emphasize that both training and forecasts computation (because of the adaptation) can be time consuming. In contrast, for GAGAM $(+Q_{\text{IGS}})$ and $EA(f) + Q_{\text{IGS}}$, the forecasts are computed in the fixed setting (namely with $\hat{\theta}_t = 1$ for all t so $\hat{Y}_t = \tilde{Y}_t$). Once the forecasts on $\mathcal{D}_{\text{VALID}}$ have been computed, the loss of the model defined in Equation (2) can be obtained. We recall that the loss function balances two terms: the RMSE on $\mathcal{D}_{\text{VALID}}$ and the complexity of the model through the degrees of freedom. To give importance to both terms and find a good trade-off, based on computations of the two terms of several models, we empirically set $\eta = \sqrt{\text{Var}[Y]}/5000$, where the variance is the empirical variance on $\mathcal{D}_{\text{VALID}}$.

6.2 Results

We compared the performance of the state-of-the-art handcrafted model (“SotA” and “SotA + Q_{IGS} ”) and the one found by GAGAM (“GAGAM” and “GAGAM + Q_{IGS} ”) with the two versions of the evolutionary algorithm (EA): $EA(f) + Q_{\text{IGS}}$ and $EA(f, Q)$. Table 1 presents the RMSE and the MAPE (Mean Absolute Percentage Error) for each data set (training, validation and testing). Figure 4 shows the RMSE for each hour on the testing set. For the fixed setting, GAGAM and $EA(f)$ perform

Model	Training data set RMSE · MAPE	Validation data set RMSE · MAPE	Testing data set RMSE · MAPE
SotA	2510 MW · 3.44 %	2951 MW · 4.33 %	4397 MW · 7.95 %
GAGAM	2162 MW · 2.82 %	2776 MW · 4.25 %	4240 MW · 7.73 %
$EA(f)$	1814 MW · 2.33 %	2440 MW · 3.58 %	4290 MW · 8.03 %
SotA + Q_{IGS}	3301 MW · 4.60 %	3051 MW · 4.22 %	3364 MW · 5.07 %
GAGAM + Q_{IGS}	11208 MW · 10.67 %	6636 MW · 7.76 %	5976 MW · 7.78 %
$EA(f) + Q_{\text{IGS}}$	1805 MW · 2.32 %	2466 MW · 3.64 %	4320 MW · 8.10 %
$EA(f, Q)$	1594 MW · 2.10 %	1292 MW · 1.79 %	1552 MW · 2.20 %

Table 1: RMSE and MAPE on training, validation and testing data.

quite similarly to the SotA model on the testing data set. Errors on the validation data set highlight the efficiency of the evolutionary algorithms and of the proposed search space modeling. For the adaptive setting, GAGAM + Q_{IGS} and $EA(f) + Q_{\text{IGS}}$ perform much worse than the SotA model. Figure 4 reveals that, for GAGAM + Q_{IGS} , most of this is due to a few hours: GAGAM probably selects over-complicated models. Then, either Q_{IGS} does not converge to a relevant minimum, or the adaptation process is no longer identifiable at all, leading to bad performances on the three datasets. Comparing the performances of GAGAM + Q_{IGS} and $EA(f) + Q_{\text{IGS}}$, specifically on Figure 4, legitimates the addition in the loss - see Equation (2) - of the second term: less-complicated models are selected. Finally, it is clear that $EA(f, Q)$ largely outperforms all the previous models and improves the SotA model’s performance by 54%. It therefore seems crucial to optimize both the formula and the adaptation parameter simultaneously. These promising results come with new questions and possible improvements. Indeed, with this first work, f and Q are treated by EA as two independent parameters. Doing “smart” mutations and crossing-overs by considering the effect of the GAM f_k “attached” with the associated coefficient in the matrix Q could probably improve the results. Moreover, we believe that Q_{IGS} (which is time-consuming) could be used partially (a

few iterations) during the AutoML algorithm, and not only at the end. Finally, we highlight an overfitting of our AutoML algorithms on the validation test. We should consider an evaluation process involving methods close to cross-validation.

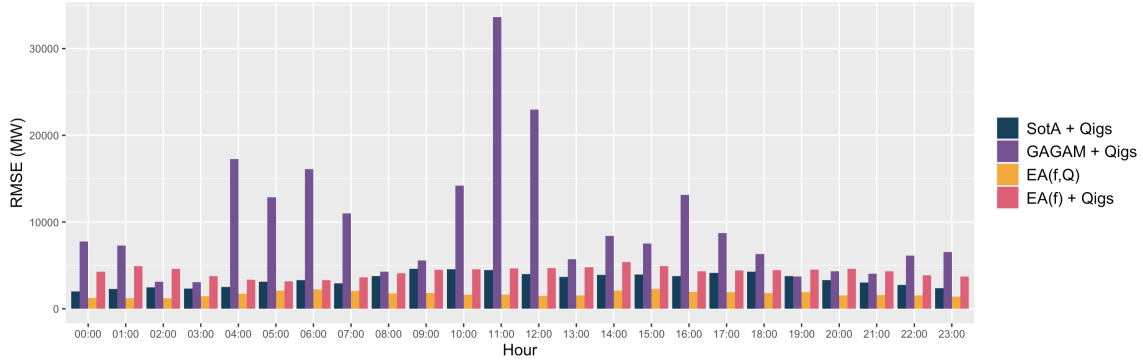


Figure 4: Hourly RMSE on testing data set for the adaptive models SotA + Q_{IGS} , GAGAM + Q_{IGS} , $EA(f) + Q_{IGS}$ and $EA(f, Q)$.

7 Conclusion

This work proposes a framework for automated online generalized additive model selection. It proposes an efficient modeling of the search space which is compatible with the DRAGON framework. Application to short-term electricity demand forecasting attests the relevance of the approach and highlights how it is crucial to optimize both the formula of the GAM and the adaptation parameter simultaneously. In practice, it is possible to greatly improve electricity demand forecasts by using mixtures of models (see for example Gaillard et al. (2016)). The idea is to have several good predictors, called “experts”, who make quite different errors from one another, and to make time-evolving weighted averages of their predictions. This AutoML work on GAMs could be improved by trying to optimize not a single formula but a mix of formulae, or even by integrating different types of models into the search space for the best mix. The homogenization, through the DRAGON package, of the representation of the GAMs in this work with that of the deep learning models proposed in EnergyDragon (Keisler et al., 2024a; Keisler and Br  g  re, 2024), which are also applied to electricity demand forecasting, would make it easy to design search spaces containing both neural networks and GAMs. These two types of models naturally model electricity demand in very different ways and therefore do not make the same errors, which would probably produce a good mix. Finally, these model choices, whether in a mix or not, could vary over time. While online tuning can produce gains, too much change in the data distribution may require online AutoML.

8 Broader Impact Statement

Our work automates the creation of reliable load forecasting models. It could provide different actors in the electricity system with access to reliable electricity demand forecasts without the need to spend a lot of time building models by hand. While this work is limited to short-term national forecasts, GAMs can be used for different forecast horizons and spatial aggregation scales (see, for example, the applications of GAMs to mid-term forecasting Zimmermann and Ziel 2025 and to local load signals Lambert et al. 2023). These different electricity demand forecasts play a crucial role in the fight against global warming. On the one hand, they facilitate the massive integration of renewable energy (by reducing uncertainty about demand, we can afford to increase uncertainty about electricity production). They also make it possible to adjust the demand of both individual and industrial actors, for example, to run large calculations or start machines at times when demand is low and production is guaranteed by low-carbon emission sources.

References

- Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.
- Ba, A., Sinn, M., Goude, Y., and Pompey, P. (2012). Adaptive learning of smoothing functions: Application to electricity load forecasting. *Advances in neural information processing systems*, 25.
- Brégère, M. and Keisler, J. (2024). A Bandit Approach with Evolutionary Operators for Model Selection. working paper or preprint.
- Cus, M. (2020). Simultaneous variable selection and structure discovery in generalized additive models.
- de Vilmarrest, J. (2022). *Modèles espace-état pour la prévision de séries temporelles. Application aux marchés électriques*. PhD thesis, Sorbonne université.
- Fan, S. and Hyndman, R. J. (2011). Short-term load forecasting based on a semi-parametric additive model. *IEEE transactions on power systems*, 27(1):134–141.
- Feurer, M., Eggenberger, K., Falkner, S., Lindauer, M., and Hutter, F. (2022). Auto-sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research*, 23(261):1–61.
- Feurer, M., Klein, A., Eggenberger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28.
- Gaillard, P., Goude, Y., and Nedellec, R. (2016). Additive models and robust aggregation for gefcom2014 probabilistic electric load and electricity price forecasting. *International Journal of forecasting*, 32(3):1038–1050.
- Goude, Y., Nedellec, R., and Kong, N. (2013). Local short and middle term electricity load forecasting with semi-parametric additive models. *IEEE transactions on smart grid*, 5(1):440–446.
- Hastie, T. and Tibshirani, R. (1986). Generalized Additive Models. *Statistical Science*, 1(3):297 – 310.
- Keisler, J. (2025). *Automated Deep Learning: algorithms and software for energy sustainability*. PhD thesis, University of Lille.
- Keisler, J. and Brégère, M. (2024). Automated spatio-temporal weather modeling for load forecasting. In *International Ruhr Energy Conference*.
- Keisler, J., Claudel, S., Cabriel, G., and Brégère, M. (2024a). Automated deep learning for load forecasting. In *International Conference on Automated Machine Learning*, pages 16–1. PMLR.
- Keisler, J., Talbi, E.-G., Claudel, S., and Cabriel, G. (2024b). An algorithmic framework for the optimization of deep neural networks architectures and hyperparameters. *Journal of Machine Learning Research*, 25(201):1–33.
- Lambert, G., Hamrouche, B., and De Vilmarrest, J. (2023). Frugal day-ahead forecasting of multiple local electricity loads by aggregating adaptive models. *Scientific Reports*, 13(1):15784.
- Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52.

- Obst, D., De Vilmares, J., and Goude, Y. (2021). Adaptive methods for short-term electricity load forecasting during covid-19 lockdown in france. *IEEE transactions on power systems*, 36(5):4754–4763.
- Pierrot, A. and Goude, Y. (2011). Short-term electricity load forecasting with generalized additive models. *Proceedings of ISAP power*, 2011.
- Tang, Z., Fang, H., Zhou, S., Yang, T., Zhong, Z., Hu, C., Kirchhoff, K., and Karypis, G. (2024). Autogluon-multimodal (automm): Supercharging multimodal automl with foundation models. In *International Conference on Automated Machine Learning*, pages 15–1. PMLR.
- White, C., Safari, M., Sukthankar, R., Ru, B., Elsken, T., Zela, A., Dey, D., and Hutter, F. (2023). Neural architecture search: Insights from 1000 papers. *arXiv preprint arXiv:2301.08727*.
- Wood, S. N. (2017). *Generalized Additive Models, An Introduction with R*. Chapman and Hall/CRC.
- Wood, S. N., Goude, Y., and Shaw, S. (2015). Generalized additive models for large data sets. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 64(1):139–155.
- Zimmermann, M. and Ziel, F. (2025). Efficient mid-term forecasting of hourly electricity load using generalized additive models. *Applied Energy*, 388:125444.

Submission Checklist

1. For all authors...

- (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The aim of the paper was to assess the relevance of applying AutoML on GAMs for electricity demand forecasting. The results show the ability of our framework to find better formulae than the handcrafted one.
- (b) Did you describe the limitations of your work? [Yes] We explained which variables could be acted upon to potentially significantly change the results in the conclusion.
- (c) Did you discuss any potential negative societal impacts of your work? [Yes] We suggested ways to reduce computation time: being mindful before running the algorithm and searching manually for some basic GAM formulae to input as part of its initial population and to leverage resource allocation.
- (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? <https://2022.automl.cc/ethics-accessibility/> [Yes] The paper conforms to the guidelines

2. If you ran experiments...

- (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? [Yes] GAGAM did not have the same loss function: it relied solely on the RMSE to rank its models. The resources given were the same for all AutoML frameworks: 200 iterations for a population of 20 individuals.
- (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [Yes] The "Experiment design" section gives all these details and the chosen scaling factor is mentioned in section 3.4.
- (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [No] We did not want to launch too many experiments. But we ran our framework for the 24 hours of the day, and for each hour, our approach managed to find a better solution than GAGAM or the handcrafted model. We think it is enough to attest to the pertinence of our approach.
- (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [No] We did not report the variance as we only launched each experiment once, but we set the global seed to 0 to ensure reproducibility.
- (e) Did you report the statistical significance of your results? [Yes] We report that our approach improve by 54% the current state of the art.
- (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [N/A] We used an industrial use case.
- (g) Did you compare performance over time and describe how you selected the maximum duration? [No] The maximum duration was set due to time constraints. We believe that allowing more resources could lead to even better results.
- (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] This information is given Section 6.
- (i) Did you run ablation studies to assess the impact of different components of your approach? [Yes] We ran two versions of our framework to show the impact of the joint selection of f and Q , which is very relevant in our context.

3. With respect to the code used to obtain your results...
 - (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive README with installation, and execution commands (either in the supplemental material or as a URL)? [\[Yes\]](#) The code can be found in the supplementary material. It requires both a Python and an R environment.
 - (b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [\[Yes\]](#) We provide a notebook to run the algorithms for a small number of iterations (20 iterations and a population of 5 models, which can be modified).
 - (c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [\[Yes\]](#) We did, the only tricky part is to manage both Python and R environments.
 - (d) Did you include the raw results of running your experiments with the given code, data, and instructions? [\[Yes\]](#) We did.
 - (e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [\[Yes\]](#) We did.
4. If you used existing assets (e.g., code, data, models)...
 - (a) Did you cite the creators of used assets? [\[Yes\]](#) The creator of GAGAM, that of the DRAGON package, and that of the data set were all cited.
 - (b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [\[Yes\]](#) GAGAM and DRAGON are open-source, and so is the RTE data set.
 - (c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [\[Yes\]](#) The RTE data is open-source and free from personally identifiable information or offensive content.
5. If you created/released new assets (e.g., code, data, models)...
 - (a) Did you mention the license of the new assets (e.g., as part of your code submission)? [\[Yes\]](#) The code in the supplementary material is open-source.
 - (b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [\[Yes\]](#) The code in the supplementary material is open-source.
6. If you used crowdsourcing or conducted research with human subjects...
 - (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [\[N/A\]](#) We did not use crowdsourcing or conduct research with subjects.
 - (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [\[N/A\]](#) We did not use crowdsourcing or conduct research with subjects.
 - (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [\[N/A\]](#) We did not use crowdsourcing or conduct research with subjects.
7. If you included theoretical results...

- (a) Did you state the full set of assumptions of all theoretical results? [N/A] There are no theoretical results in this paper.
- (b) Did you include complete proofs of all theoretical results? [N/A] There are no theoretical results in this paper.

A Detailed structure of generalized additive models (fixed setting)

In the following, we detail the two steps required to set a GAM formula in our set-up. We recall that the target variable is a time series $(Y_t)_{t \geq 1}$ where Y_t represents the amount of electricity demanded between times t and $t + 1$. The latter depends on a multitude of variables Z_t , including weather (temperature, wind, etc.) and calendar-related (public holidays, weekdays, etc.) factors. The set of covariates X_t used in the GAM formula includes actual and past covariates, so $X_t = (Z_s)_{s=1, \dots, t}$ and we obtain the equation

$$\mathbb{E}[Y_t|X_t] = \sum_{k=1}^K f_k(X_{J_k,t}).$$

The specification of the smooth functions f_k is divided into two: feature engineering (to extract useful information from $X_{J_k,t}$) and relationship specification (to link this information with the target variable).

Feature engineering. For any $k = 1, \dots, K$, feature engineering for covariates X_{J_k} may be relevant. We detail in the following the two examples that we consider.

Example 1: Exponential smoothing of the temperature. The incorporation of exponentially smoothed temperatures, which model the thermal inertia of buildings, is likely to improve electricity demand forecasts; see, e.g. Goude et al. (2013). The creation of such a variable is based on a smoothing parameter $\alpha \in [0, 1]$ and requires access to all past temperature values, so we consider, for any time step t , $X_{J_k,t} = (T_1, \dots, T_t)^\top$, where T_t denotes the temperature at time step t . The exponentially smoothed temperature parameterized by α is defined by:

$$\psi_k(T_t) = \begin{cases} T_1 & \text{if } t = 1 \\ \alpha\psi_k(T_{t-1}) + (1 - \alpha)T_t & \text{else.} \end{cases}$$

By induction, for any time step $t = 1, 2, \dots$, we get $\psi_k(X_{J_k,t}) = \sum_{s=0}^{t-1} (1 - \alpha)\alpha^s T_{t-s} + \alpha^t T_1$. We emphasize that the function ψ_k depends only on the smoothing parameter α .

Example 2: Categorical variable selection. We may consider a categorical variable D taking integer values between 1 and m , and possibly a default value 0. Among the m modalities, the objective is to select only a subset of them. Let v a vector in $\{0, 1\}^m$ such that $v_j = 1$ if the modality j is selected and 0 otherwise. We define the engineering function ψ_k for the feature D that depends only on v by

$$\psi_k(D_t) = D_t \mathbf{1}_{v_{D_t}=1}.$$

When no feature engineering is required, the function ψ_k is the identity. To consider bi-variate effects, with, e.g. some exponentially smooth temperature and categorical variable selection, the above functions of Examples 1 and 2 can be combined.

Relationship specification. Let $\tilde{f}_k$ be the GAM effect associated to the engineered feature $\psi_k(X_{J_k})$, namely the function such that $f_k(X_{J_k}) = \tilde{f}_k(\psi_k(X_{J_k}))$. In the GAM framework, we take $\tilde{f}_k$ in the linear span of a set of q_k basis functions. More precicely, with $b_{k,i}$ the i^{th} basis function, and any x belonging to set of the values of the covariate $\psi_k(X_{J_k})$, $\tilde{f}_k$ is assumed to have the representation

$$\tilde{f}_k(x) = \sum_{i=1}^{q_k} \beta_{k,i} b_{k,i}(x),$$

for some values of the unknown parameters $\beta_{k,i}$. For a univariate smooth effect, that is, when $\psi_k(X_{J_k})$ is a quantitative real variable, classical basis functions are obtained using cubic splines, cyclic cubic splines, thin plate splines, etc. - see Wood et al. (2015) for further details. We emphasize

that the above notation also works for tensor products, i.e. when $\tilde{f}_k$ takes any number m of covariates as input. By defining a spline basis $(b_i)_{i=1,\dots,q_j}$ for each component j of the already engineered feature vector $\psi_k(X_{J_k})$, with x_j belonging to the set of values of feature j , we consider

$$\tilde{f}_k(x_1, \dots, x_m) = \sum_{i_1=1}^{q_1} \sum_{i_2=1}^{q_2} \dots \sum_{i_m=1}^{q_m} \beta_{i_1, i_2, \dots, i_m} b_{i_1}(x_1) b_{i_2}(x_2) \dots b_{i_m}(x_m) = \sum_{i=1}^{q_1 q_2 \dots q_m} \tilde{\beta}_i \tilde{b}_i(x_1, x_2, \dots, x_m),$$

where $\tilde{\beta}_i = \beta_{i_1, i_2, \dots, i_m}$ and $\tilde{b}_i(x_1, x_2, \dots, x_m) = b_{i_1}(x_1) b_{i_2}(x_2) \dots b_{i_m}(x_m)$ with, for $l = 1, \dots, m$,

$$i_l = \left\lceil \frac{r(i, q_1 q_2 \dots q_{l-1}) - 1}{q_{l+1} q_{l+2} \dots q_m} \right\rceil + 1, \text{ where } r(i, q) \text{ is the remainder of the Euclidean division of } i \text{ by } q.$$

Finally, when $\psi_k(X_{J_k})$ is a categorical variable, taking integer values between 1 and m , we set $q_k = m$ and $b_i(x) = \mathbf{1}_{x=i}$, for $i = 1, \dots, m$. Thus, a coefficient $\beta_{k,i}$ is estimated of each modality i .

B Algorithms

We aim to find the best adaptive Generalized Additive Model (GAM) among a search-space Ω defined in Section 3. We recall that we used the P-IRLS algorithm from the `mgcv` R-package to train the GAMs. In what follows, for a formula f and a training data set $\mathcal{D}_{\text{TRAIN}}$, we denote by $\hat{f} = \text{P-IRLS}(f, \mathcal{D}_{\text{TRAIN}})$ the trained GAM. In Chapter 5 of de Villemarest (2022), Algorithm 5 proposes an iterative grid search procedure to optimize the matrix which parameterizes the adaptive version of a model $\hat{f}$. In the following, we denote by $Q_{\text{IGS}}(\hat{f}, \mathcal{D}_{\text{TRAIN}}, Q_0)$ the matrix output of this algorithm, for a model $\hat{f}$ and an initial matrix Q_0 . Once the model $(f, Q) \in \Omega$ has been trained, it is possible to evaluate its performance. To do so, we use the loss function defined in Equation (2) which takes as arguments the formula f , the hyperparameter Q and the data set $\mathcal{D}$. As detailed in Section 4, $\ell(f, Q, \mathcal{D})$ requires the calculation of forecasts in the adaptive setting. In what follows, if $Q = -\infty$, the forecasts are computed in the fixed setting (namely with $\hat{\theta}_t = 1$ for all t). Table 2 contains the notations used in the algorithms defined below. Algorithm 1 describes the generation of a model $(f, Q) \in \Omega$. Algorithm 2 is declined in two versions: $\text{EA}(f) + Q_{\text{IGD}}$ and $\text{EA}(f, Q)$. It explores T models by evolving a population of constant size M . To do so, it starts with M sampled models (using Algorithm 1). At each round, it creates two new models from two good ones using evolutionary operators and then deletes the two models with the highest loss among the entire population (including the two new ones). In the first version $\text{EA}(f) + Q_{\text{IGD}}$, the evolutionary algorithm is used only to optimize the GAM formula, and in a second step, the hyperparameter for its adaptive version is optimized using the iterative grid-search Q_{IGD} . In contrast, in $\text{EA}(f, Q)$, we do not use the iterative grid search and the evolutionary algorithm optimizes the formula and the hyperparameter of the adaptive version both at the same time.

C Experiments: state-of-the-art handcrafted model

We refer, among others, to Goude et al., 2013 and Gaillard et al., 2016 for an exhaustive presentation of the generalized additive models used to forecast power demand. Our state-of-the-art handcrafted model takes into account some meteorological variables at an hourly time step: the temperature T_t and the smoothed temperature $\tilde{T}_t$, the cloud cover C_t , and the wind speed W_t ; and some calendar variables: the day of the week D_t , the hour of the day $H_t \in \{1, \dots, 24\}$ and the position in the year $P_t \in [0, 1]$, which takes linear values between $P_t = 0$ on January 1st at 00:00 and $P_t = 1$ on December the 31st at 23:59. We also add a “Break” variable B_t : a categorical variable referring to various moment in the year (Winter time, Summer time, August and Christmas break, etc.). The

t	Time step within the time series
i	Iteration of a search algorithm
f	Formula of a Generalize Additive Model (GAM)
Q	Hyperparameter of the adaptive version of a GAM ($K \times K$ -matrix, where K is the number of effects of the GAM)
Ω	Search space containing the GAM formulae and the associated matrices
$\mathcal{D}_{\text{TRAIN}}$	Data set used for model training (P-IRLS and Q_{IGS} algorithms)
$\mathcal{D}_{\text{VALID}}$	Data set used for model evaluation
$\mathcal{D}_{\text{TEST}}$	Data set used for AutoML algorithms' performance evaluation

Algorithms and operators	
$\mathcal{B}(p)$	Function to generate a realization of a random variable that follows a Bernoulli distribution of parameter p
$\text{P-IRLS}(f, \mathcal{D}_{\text{TRAIN}})$	GAM output by P-IRLS algorithm (mgcv R-package) for a formula f and a training data set $\mathcal{D}_{\text{TRAIN}}$
$\ell(f, Q, \mathcal{D}_{\text{VALID}})$	Loss defined in Equation (2)
$Q_{\text{IGS}}(\hat{f}, \mathcal{D}_{\text{TRAIN}}, Q_0)$	Adaptive version hyperparameter for trained model $\hat{f}$ output by Algorithm 5 of de Villemarest (2022) for an initial hyperparameter Q_0 and a training data set $\mathcal{D}_{\text{TRAIN}}$
$\mathcal{M}(f, Q)$	Mutant model created from $(f, Q) \in \Omega$ using the mutation operator described in Section 5
$\mathcal{CO}((f_1, Q_1), (f_2, Q_2))$	Children models created from models (f_1, Q_1) and $(f_2, Q_2) \in \Omega$ using the crossing-over operator described in Section 5

Table 2: Table of Notation

state-of-the-art handcrafted model is then the sum of 24 daily models, one for each hour of the day. Thus, at any time step t , with $h = H_t$, the following model is considered:

$$\hat{f}^h(X_t) = \hat{s}_T^h(T_t^h) + \hat{s}_{\bar{T}}^h(\bar{T}_t^h) + \hat{s}_C^h(C_t^h) + \hat{s}_W^h(W_t^h) + \sum_{d=1}^D \hat{\delta}_d^h \mathbf{1}_{D_t=d} + \sum_{b=1}^B \hat{\beta}_b^h \mathbf{1}_{B_t=b} + \hat{s}_P^h(P_t^h).$$

The $\hat{s}_T^h$, $\hat{s}_{\bar{T}}^h$, $\hat{s}_C^h$, $\hat{s}_W^h$, and $\hat{s}_P^h$ functions catch the effect of the meteorological variables and of the yearly seasonality. They are cubic splines: $\mathcal{C}^2$ -smooth functions made up of sections of cubic polynomials joined together at points of a grid. The coefficients $\hat{\delta}_d^h$ and $\hat{\beta}_b^h$ model the influence of the day of the week and the period of the year respectively; these effects are represented as a sum of indicator functions. As we consider a model per hour, all the coefficients and splines are indexed by h . In our automated model selection approach, we consider other variables such as the month of the year, the maximum and minimum temperatures per day, the binary variable Weekend, and so on.

Algorithm 1 Generation of an (adaptive) GAM model

Inputs:

Covariate list L_{VAR}
Categorical covariates list $L_{\text{CAT_VAR}} \subset L_{\text{VAR}}$
Spline list L_{SP} (for the uni-variate effects), Tensor list L_{TE} (for the bi-variate effects)
Maximum number of effects in the GAM K_{max} . (If None, use the number of features),
Minimum / Maximum degree of freedom for a spline $k_{\text{min}} / k_{\text{max}}$,
Probability of generating a bi-variate effect $p_{\text{bi_var}}$,
Minimum / Maximum smoothing coefficient for the temperature $\alpha_{\text{min}} / \alpha_{\text{max}}$,
List of days / offsets to be calculated $L_{\text{DAY}} / L_{\text{OS}}$,
Boolean specifying whether Q should also be generated or not KALMAN
Minimum / Maximum value for the Q matrix coefficients $Q_{\text{min}} / Q_{\text{max}}$,
Minimum / Maximum value for the Q matrix coefficients' standard deviation $\sigma_{\text{min}} / \sigma_{\text{max}}$

Initialization

Sample a number of effects $K \in \{1, \dots, K_{\text{max}}\}$
Create an empty list L_{GAM} that will contain the effects of the generated GAM
If KALMAN = TRUE:
 Uniformly choose a standard deviation σ between σ_{min} and σ_{max} ,
 Logarithmically choose the Q matrix's diagonal coefficients between Q_{min} and Q_{max} ,
 Append a list containing σ and the diagonal coefficients to L_{GAM}
For $k = 1, \dots, K$:
 Sample $X_k \sim \mathcal{B}(p_{\text{bi_var}})$
 If $X_k = 1$, a bi-variate is generated:
 Select two distinct covariates from L_{VAR} while making sure they do not already exist as a bi-variate effect in L_{GAM} and that they are not both categorical using $L_{\text{CAT_VAR}}$,
 Sample a relationship in L_{SP} if one of the covariates is in $L_{\text{CAT_VAR}}$ and a degree of freedom between k_{min} and k_{max} ; or in L_{TE} and two degrees of freedom otherwise while ensuring compatibility with the chosen relationship
 Else:
 Select a covariate while making sure it does not already exist in L_{GAM}
 Sample a relationship in L_{SP} and a degree of freedom while ensuring compatibility with the chosen relationship
 If the covariate(s) are temperature, days or offsets, generate a temperature smoothing coefficient between α_{min} and α_{max} , or a list among L_{DAY} or L_{OS}
 Add the effect to L_{GAM}

Output: L_{GAM}

Algorithm 2 Steady state evolutionary algorithm (EA) for adaptive GAM selection

Inputs:

Training data set $\mathcal{D}_{\text{TRAIN}}$, validation data set $\mathcal{D}_{\text{VALID}}$, loss function ℓ
 P-IRLS algorithm, Mutation operator $\mathcal{M}$, Crossover operator $\mathcal{CO}$,
 Population size M , Size of the tournament selection $m \in \{1, \dots, M-1\}$,
 Parameter for the initialization of the iterative grid search q_0 , Number of tested models T ,
 $V_{\text{EA}} \in \{\text{EA}(f) + Q_{\text{IGD}}, \text{EA}(f, Q)\}$ the version of EA considered

Initialization

For $i = 1, 2, \dots, M$:

If $V_{\text{EA}} = \text{EA}(f, Q)$:

 Sample a model (f_i, Q_i) using Algorithm 1 with $\text{KALMAN} = \text{TRUE}$

 Train the model: $\hat{f}_i = \text{P-IRLS}(f_i, \mathcal{D}_{\text{TRAIN}})$

Else:

 Sample a formula f_i using Algorithm 1 with $\text{KALMAN} = \text{FALSE}$

 Train the associated fixed model: $\hat{f}_i = \text{P-IRLS}(f_i, \mathcal{D}_{\text{TRAIN}})$

$Q_i = -\infty$ (when $V_{\text{EA}} = \text{EA}(f) + Q_{\text{IGD}}$)

 Get the loss $\ell_i = \ell(f_i, Q_i, \mathcal{D}_{\text{VALID}})$

For $i = 1, \dots, \lfloor \frac{T-M}{2} \rfloor$:

 Sample a partition $\mathcal{I} \in \{1, \dots, M\}$ of size m and select $j_A \in \arg\min_{j \in \mathcal{I}} \ell_j$

 Sample a partition $\mathcal{J} \in \{1, \dots, M\} \setminus \{j_A\}$ of size m and select $j_B \in \arg\min_{j \in \mathcal{J}} \ell_j$

 Apply the crossover operator on models j_A and j_B :

$((f^{\text{NEW1}}, Q^{\text{NEW1}}), (f^{\text{NEW2}}, Q^{\text{NEW2}})) = \mathcal{CO}((f_{j_A}, Q_{j_A}), (f_{j_B}, Q_{j_B}))$

 Mutate the two new models:

$(f^{\text{NEW1}}, Q^{\text{NEW1}}) = \mathcal{M}(f^{\text{NEW1}}, Q^{\text{NEW1}})$ and $(f^{\text{NEW2}}, Q^{\text{NEW2}}) = \mathcal{M}(f^{\text{NEW2}}, Q^{\text{NEW2}})$

 Train two the new models: $\hat{f}^{\text{NEW1}} = \text{P-IRLS}(f^{\text{NEW1}}, \mathcal{D}_{\text{TRAIN}})$, $\hat{f}^{\text{NEW2}} = \text{P-IRLS}(f^{\text{NEW2}}, \mathcal{D}_{\text{TRAIN}})$

If $V_{\text{EA}} = \text{EA}(f, Q)$:

$Q^{\text{NEW1}} = Q_{\text{IGD}}(\hat{f}^{\text{NEW1}}, \mathcal{D}_{\text{TRAIN}}, Q^{\text{NEW1}}, N)$ and $Q^{\text{NEW2}} = Q_{\text{IGD}}(\hat{f}^{\text{NEW2}}, \mathcal{D}_{\text{TRAIN}}, Q^{\text{NEW2}}, N)$

 Get the losses: $\ell^{\text{NEW1}} = \ell(f^{\text{NEW1}}, Q^{\text{NEW1}}, \mathcal{D}_{\text{VALID}})$, $\ell^{\text{NEW2}} = \ell(f^{\text{NEW2}}, Q^{\text{NEW2}}, \mathcal{D}_{\text{VALID}})$

If $\ell^{\text{NEW1}} < \max_{j \in \{1, \dots, M\}} \ell_j$:

 Replace $(f_{j_{\text{OLD}}}, Q_{j_{\text{OLD}}})$ where $j_{\text{OLD}} \in \arg\max_{j \in \{1, \dots, M\}} \ell_j$ by $(f^{\text{NEW1}}, Q^{\text{NEW1}})$

If $\ell^{\text{NEW2}} < \max_{j \in \{1, \dots, M\}} \ell_j$:

 Replace $(f_{j_{\text{OLD}}}, Q_{j_{\text{OLD}}})$ where $j_{\text{OLD}} \in \arg\max_{j \in \{1, \dots, M\}} \ell_j$ by $(f^{\text{NEW2}}, Q^{\text{NEW2}})$

Select the best model $(f_{j_{\text{BEST}}}, Q_{j_{\text{BEST}}})$ with $j_{\text{BEST}} \in \arg\min_{j=1, \dots, M} \ell_j$

If $V_{\text{EA}} = \text{EA}(f) + Q_{\text{IGD}}$:

$Q_{j_{\text{BEST}}} = Q_{\text{IGD}}(\hat{f}_{j_{\text{BEST}}}, \mathcal{D}_{\text{TRAIN}}, Q_0)$ with $Q_0 = q_0 I_d$

Output: $(f_{\text{BEST}}, Q_{\text{BEST}})$
