Received: 10 August 2021 | Accepted: 22 October 2021

DOI: 10.1002/er.7448

RESEARCH ARTICLE

INTERNATIONAL JOURNAL OF ENERGY RESEARCH WILEY

The potential of deep learning to reduce complexity in energy system modeling

Clara Sophie Köhnen[1] | Jan Priesmann[1] | Lars Nolting[1] | Leander Kotzur[2,3] | Martin Robinius[4] | Aaron Praktiknjo[1,5]

[1]Chair for Energy System Economics, Institute for Future Energy Consumer, Needs and Behavior (FCN), E.ON Energy Research Center, RWTH Aachen University, Aachen, Germany

[2]Techno-economic Systems Analysis (IEK-3), Forschungszentrum Jülich, Institute of Energy and Climate Research, Jülich, Germany

[3]JARA-ENERGY, Jülich Aachen Research Alliance, Jülich, Germany

[4]Umlaut Energy GmbH, Aachen, Germany

[5]JARA-ENERGY, Jülich Aachen Research Alliance, Aachen, Germany

Correspondence
Aaron Praktiknjo, Chair for Energy System Economics (FCN-ESE), E.ON Energy Research Center, RWTH Aachen University, Mathieustr. 10, 52074 Aachen, Germany.
Email: aaron.praktiknjo@rwth-aachen.de

Funding information
Bundesministerium für Wirtschaft und Energie, Grant/Award Numbers: 03EI1022A, (Jan Priesmann, Lars Nolting, Aaron Praktiknjo); Bundesministerium für Wirtschaft und Energie, Grant/Award Number: 03ET4064 (Leander Kotzur)

Summary

In order to cope with increasing complexity in energy systems due to rapid changes and uncertain future developments, the evaluation of multiple scenarios is essential for sound scientific system analyses. Hence, efficient modeling approaches and complexity reductions are urgently required. However, there is a lack of scientific analyses going beyond the scope of traditional energy system modeling. For this reason, we investigate the potential of metamodels to reduce the complexity of energy system modeling. In our explorative study, we investigate their potential and limits for applications in the fields of electricity dispatch and design optimization for heating systems. We first select a suitable metamodeling approach by conducting pre-tests on a small scale. Based on this, we selected artificial neural networks due to their good performance compared to other approaches and the multiple possibilities of network topologies and hyperparameter settings. As for the dispatch model, we show that a high accuracy of price replication can be achieved while substantially reducing the runtimes per investigated scenario (from 2 hours on average down to less than 30 seconds). With the design optimization model, we find double-edged results: while we also achieve a substantial reduction of runtime in this case (from ∼0.8 hours to less than 30 seconds), the simultaneous forecasting of several interdependent variables proved to be problematic and the accuracy of the metamodel shows to be insufficient in many cases. Overall, we demonstrate that metamodeling is a suitable approach to complemement traditional energy system modeling rather than to replace them: the loss of traceability in (black-box) metamodels indicates the importance of hybrid solutions that combine fundamental models with metamodels.

KEYWORDS

design optimization, dispatch optimization, energy system models, machine learning, metamodeling

Abbreviations: ANN, artificial neural network; CHP, combined heat and power; ESM, energy system model; FiTCHP, feed-in tariff combined heat and power; FiTPV, feed-in tariff photovoltaics; HP, heat pump; IPCC, intergovernmental panel on climate change; LSTM, long short-term memory; MAPE, mean absolute percentage error; MBDO, metamodel-based design optimization; MSE, mean squared error; PV, photovoltaics; RF, random forest; RMSE, root mean squared error; TPE, tree Parzen Estimator.

1 | INTRODUCTION

As defined by the Intergovernmental Panel on Climate Change (IPCC), energy systems comprise the "production, conversion, delivery, and use of energy."[1] Each of these components is subject to major changes and highly uncertain developments.[2,3] Energy system models (ESMs) are used to depict energy system components and their behavior to simulate this transition in a laboratory-like setting. In the current energy transition, energy supply, demand, and policies are changing at an unprecedented rate.[4] Hence, ESMs have recently grown in popularity.[5] The results of ESMs are often intended to serve as guidance for decision-makers from politics and industry regarding the future development of energy systems.[6] Therefore, a broad variety of future scenarios must be depicted by such models to account for the high levels of inherent uncertainty.[7]

As the complexity of energy systems increases, so does the complexity of the models that represent them.[5,8] Changing framework conditions in the energy sector, combined with further developments in information and data science, encourage the development of highly complex models that are capable of depicting certain parts of the energy system with a high degree of detail, such as temporal and spatial resolutions.[9,10] However, the aim of energy system modeling should be to create models that are as simple as possible and as complex as necessary to achieve the goal of "parsimony" as defined by DeCarolis et al.[11] The virtue of parsimony in energy system modeling does not lose its importance as computing power increases. Increasing uncertainty due to increasing complexity in (real world) energy systems must be addressed by covering large scenario spaces which restrict the computational complexity of models.

One promising approach to counteract the increasing complexity of ESMs is metamodeling.[12] As is shown in Figure 1, metamodeling describes the process of replacing an ESM by directly depicting the relationships of the input and output vectors using less complex methods of approximation. Most techniques for complexity reduction are applied to either input data (data aggregation), the number of model components, or the component relationships (simplification or omission of system relationships).[5] In contrast, metamodeling is applied to achieve sufficiently accurate solutions in shorter runtimes, thereby enabling the assessment of a broader variety of scenarios, motivating the application of metamodels.

A recent review of studies on the subject of metamodeling by Zaibi et al[13] shows that various approaches

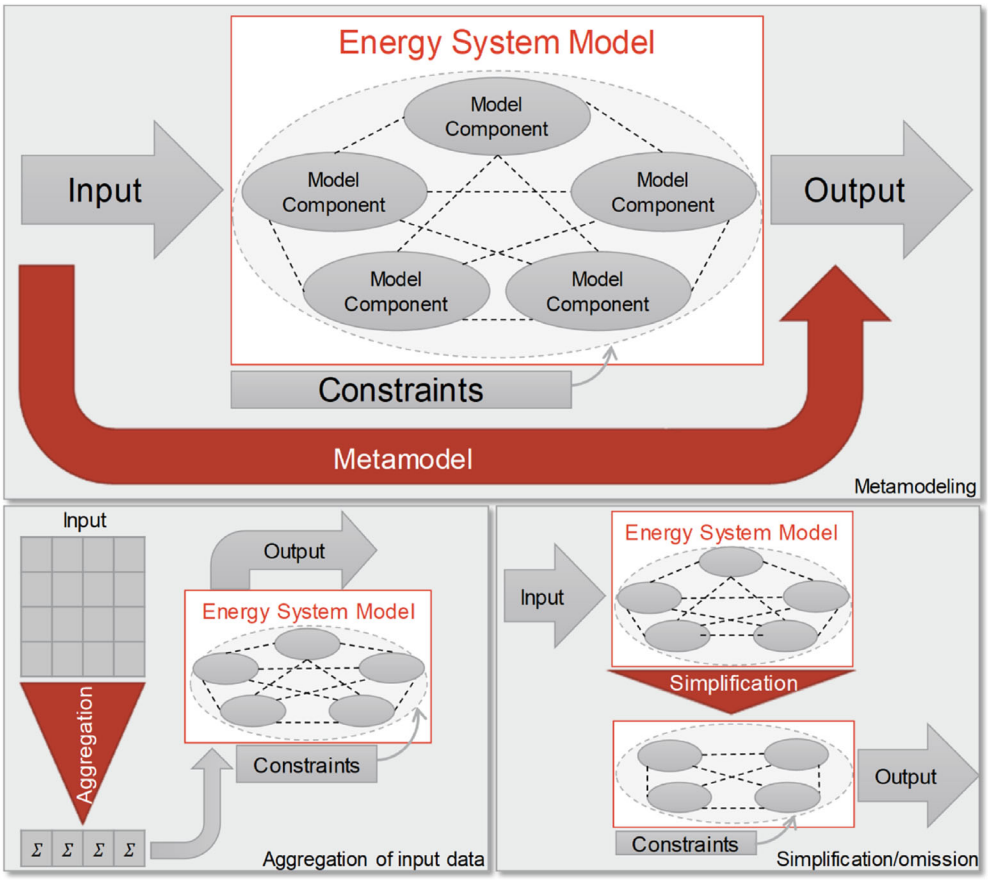


FIGURE 1 Energy system model (ESM) and metamodeling

are being pursued, such as metamodels based on classical polynomial functions,[14,15] stochastic metamodels,[16–18] and those using artificial neural networks (ANNs).[19–22] We provide a more comprehensive overview of existing approaches to metamodeling in Section 2.

In this study, we focus on metamodels using artificial intelligence, more specifically deep learning. Whereas other studies have focused on particular applications such as building energy systems (see References 15,18,22), we go beyond the scope of existing analyses by comparing the benefits and drawbacks of different metamodels for different applications. Our fields of application comprise (simplified) unit commitment models and design optimization models. We ask the following research questions:

- Can a reduction in complexity be achieved by replacing classical energy system optimization models with deep learning-based metamodels?
- How can metamodels be applied to increase the number of scenarios investigated under given computational restrictions?

The remainder of this work is structured as follows. First, we summarize relevant literature on metamodeling, ANNs, random forests, and challenges in deep learning in Section 2. We then introduce our methodology in Section 3. Thereinafter, we demonstrate our results, first for pre-tests and subsequently for full analyses. We provide a discussion and derive general modeling recommendations in Section 5. The paper concludes in Section 6.

2 | THEORETICAL BACKGROUND

Having introduced our work, this section provides an overview of relevant literature on existing approaches to metamodeling. Furthermore, we introduce ANNs and random forests and analyze their potential for reducing the complexity of ESMs. Finally, we summarize challenges inherent to processes based on deep learning.

2.1 | The concept of and approaches to metamodeling in energy system analysis

The prefix "*meta*" originates from Greek and means that something is *on a higher level*, standing *above* or *behind* another category.[23] According to Kühne[24], the prefix "*meta*" is used to indicate that a term is applied twice. A metamodel is, therefore, "a model of models".[25]

Several approaches to metamodeling can be found in the literature, for example, Simpson et al.[26] demonstrate the potential of metamodels to enable immersive engagement in energy system modeling. However, to the authors' knowledge, no general assessment regarding a metamodel strategy for energy system optimization models (in particular for dispatch and design optimization models) has been published to date.* For this reason, we summarize the previous fields of applications and the possible transferability of metamodels to our case in Table 1.

2.2 | Method portfolio for metamodeling of ESMs

In the following, we present the theory behind the selection of methods that we use as metamodels in our explorative study.

2.2.1 | Random forests

Random forests consist of a multitude of individual decision trees and therefore belong to the ensemble learning techniques family.[42] Decision trees are a machine learning method that relies on the *wisdom of the crowd*, which means that the aggregated answer of thousands of random people is, in many cases, more likely to be correct than an expert's answer.[43] Transferred to random forests, this indicates that even if a single decision tree is highly sensitive to existing noise in the training data, the average of a large number of decision trees is not (given that the trees are not correlated).

To ensure the diversity of the decision trees, they are focused on random subsets of the training set. Sampling of the subsets can be done with and without replacement. Replacement means that samples can be drawn multiple times from the training set when creating the random subsets. No replacement means that the subsets consist of unique elements (no duplicates). Sampling performed with replacements is called *bagging* (short for *bootstrap aggregating*) whereas that performed without replacement is called *pasting*.[43] The overall result of the random forest is calculated by aggregating the results of the individual decision trees. The procedure for training and training the set sampling of a random forest is shown in Figure 2. The training processes of the individual decision trees can take place in parallel, and the same applies to the predictions.[43]

During the training process, the decision tree and its branches are constructed for each of the random samples. The root of the tree contains the entire training dataset.

TABLE 1 Literature overview

Metamodel type	References	Field of application	Methodology	Transferability to energy system optimization models
Inductive learning	27	Process control, diagnostic systems	Automation of knowledge-acquisition processes. Partitioning of data into discrete categories. "Training" via backpropagation, using artificial neural networks (ANNs) or random forests.	*Transferability is questionable* as this method performs better with discrete-valued data.
Stochastic models (Kriging, Gaussian process regression)	12,16,17	Geostatic, non-linear problems, modeling of deterministic computer responses	Application of a global optimization method to identify maximum likelihood estimators. Values at locations with no sample can be approximated using surrounding measured values.	*Transferability is questionable* as we focus on the metamodeling of linear optimization models. No consideration of scenario probabilities in our case.
Dual estimation models	28,29	Performance monitoring, fault detection, mass flow monitoring	The state of a dynamic system and the underlying model are estimated simulations. Different types of algorithms, such as dual Kalman filters are used. The unknown model can be approximated using neural networks, among other methods.	*Transferability is questionable* as we focus on the metamodeling of white-box models for which equation systems are known.
Artificial neural networks	19,21,26,27–30	Wide fields of application, tested in the context of energy systems, for example, for assessments of the security of electricity supply	Training of the ANN usually done via backpropagation. Further explanations in Section 2.2.	*Transferability is possible* as the adaptation of structure, size and hyperparameter settings allows for application as comprehensive approximators.
Polynomial function models	12,13,17,21,26	Wide fields of application, for example, metamodel-based design optimization (MBDO) and multiple regression	Depicting relationships between several exogenous and endogenous variables.	*Transferability is questionable* as polynomial functions are easy to use and clear in parameter sensitivity, but not sufficiently accurate and limited by the chosen function type.
Random forests	31,32	Wide fields of application, typical use cases being classification and regression	Several slightly differently trained decision/regression trees with subsets of training data. While a single tree is highly sensitive to the noise in the training set, the average of many trees is not. Further	*Transferability is possible* due to the strong performance in highly heterogeneous data sets.

(Continues)

TABLE 1 (Continued)

Metamodel type	References	Field of application	Methodology	Transferability to energy system optimization models
			explanations in Section 2.2.	
Spline interpolation models	13	Metamodel-based design optimization (MBDO)	Interpolation of given nodes using piecewise polynomials of a low degree. The results of spline models do not oscillate due to unfavorably-determined knots.	*Transferability is questionable* as a dynamic simulator and high number of system simulations are required for finding the optimal configurations.

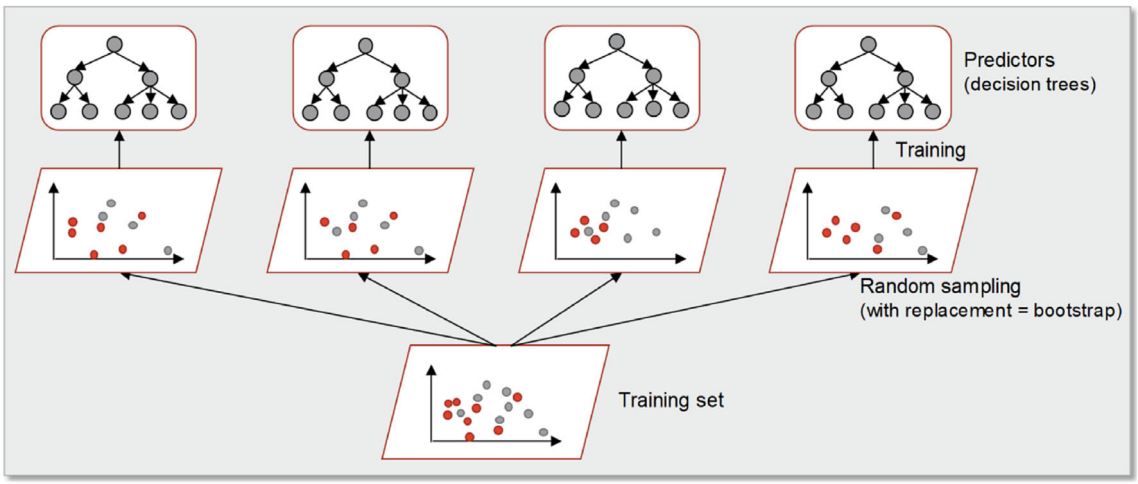


FIGURE 2 Random forest training with bagging training set sampling (based on Reference 43)

To obtain the next level, a selection tuple σ_i, containing the corresponding input feature i and threshold t_i, is applied:

$$\sigma_i = \langle i, t_i \rangle \, with \, i \in [1, n], t_i \in \left[min\left(x^{(i)}\right), max\left(x^{(i)}\right) \right] \quad (1)$$

For each tuple, a decision tree is constructed. At each tree node, the tuple is divided based on a threshold that is set during the training process. This results in two subsets (branches), one with values smaller than the threshold and one with values larger or equal than the threshold. The goal of training the decision tree is to find the optimal selection tuples and thresholds to avoid unbalanced and very deep decision trees.[44] For further information regarding the precise procedure of the training process of a decision tree, see Geron[43] and Bonnaccorso.[44]

In this study, besides neural networks, random forest models are applied as a possible approach to metamodeling, as they often exhibit better performance on heterogeneous datasets. This is because random forests can represent very different data in different decision trees.[31]

2.2.2 | Feed-forward ANN

According to the *Encyclopedia of Machine Learning* by Kakas et al,[33] an ANN "is a computational model based on biological neural networks." The simplest form of ANN, the perceptron, was first developed by Rosenblatt in 1958.[34] As for adaptive systems, ANNs are used extensively in various fields of application. They are information processing systems that consist of a large number of simple units, or neurons, each of which passes

information through directed connections. Complex problems can be represented by different network topologies, that is, different ways of connecting individual neurons. An essential aspect of ANNs is their ability to learn a task independently during the training process without having to explicitly program the neural network.[30,35,36] As the systemic relationships are learned rather than explicitly specified, the programming effort required for a neural network is significantly lower than the implementation of an ESM. This will be discussed in detail in Section 5.[†]

The fundamental elements of a neural network are the neurons. These calculate an output signal by applying the given input to the so-called activation function. The neurons are connected by directed, weighted edges. The weights inhibit or enhance the input signals and are iteratively adjusted during training. Adjustments are made based on the input weighted with the previous error and learning rate.[30]

Figure 3 shows the structure of a simple feed-forward ANN consisting of an input layer, two hidden layers, and one output layer. This network topology is characterized by information simply traveling forward. No path leads back from a neuron, either directly or indirectly through an intermediate neuron.[30]

2.2.3 | Long short-term memory neural network

Aside from the feed-forward neural network, another form of network topology is applied in this study, namely the long short-term memory (LSTM) neural network, which is a special type of recurrent ANN that was first introduced by Hochreiter and Schmidhuber in 1997.[38] In contrast to feed-forward neural networks, recurrent neural networks have edges that return to themselves, enabling that information to travel through time. Therefore, the neurons receive current information from the previous layer and, additionally, from themselves from the previous pass.[39] The problem, however, with recurrent networks is that the error signals flowing back in time tend to either blow up ("exploding gradients") or vanish ("vanishing gradients"). The weights have an exponential influence on the temporal evolution of the backpropagated error. Thus, exploding gradients may lead to oscillating weights and, in the case of vanishing gradients, the networks cannot carry information forward from earlier time steps to later ones if the sequence is long enough. This means that the networks suffer from short-time memory, an issue that is addressed by adding a "forget gate" to LSTM neural networks.[38]

An exemplary simple LSTM neural network is depicted in the left part of Figure 4. The structure is similar to a feed-forward neural network, with the additional returning edges as explained above. The network topology can be optionally extended by further hidden and LSTM layers. The right part of Figure 4 shows the structure of a single LSTM neuron, including the three different gates: input, output and forget.

> *The forget gate* is the main difference to a conventional recurrent neural network. During the training process, this gate learns to decide what information should be kept and what should be forgotten.[40] The use of the forget gate prevents gradients from vanishing and allows both short-term and long-term sequences to be reproduced.
>
> *The input gate* updates the cell state by passing the previous hidden state and current input into the activation function, which is analogous to the conventional neuron. The new cell state results from the outputs of the forget and input gates.[40]
>
> *The output gate* determines the new hidden state analogously to a conventional neuron. The hidden state contains information about previous inputs and is also used for predictions. Finally, the updated cell and hidden states are carried over to the next time step.[40]

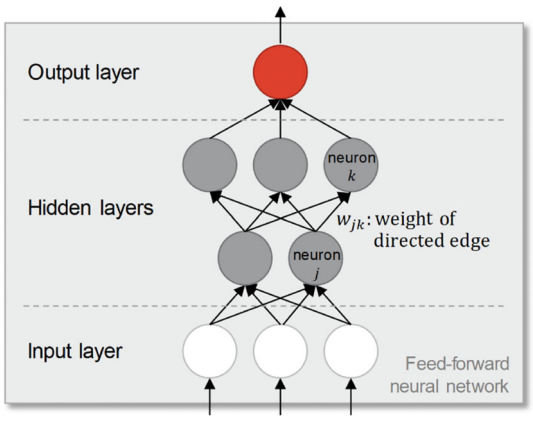


FIGURE 3 Simple feed-forward neural network (based on Reference 30)

In addition to the network topologies explained in this section (feed-forward neural network and LSTM neural network), there are various other network topologies. Further approaches have been tested in this study (eg, convolutional and concatenated neural networks)

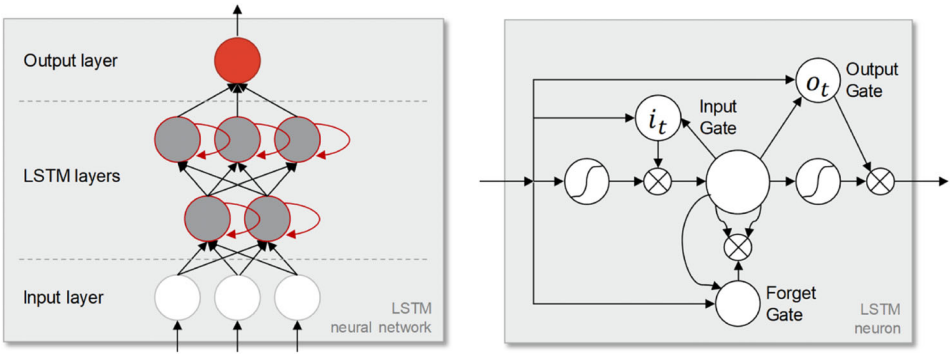


FIGURE 4 LSTM neural network (l.) and LSTM neuron (r.) (based on References 39,41)

but these did not lead to sufficiently good results. Therefore, these approaches are not explained here and, for further information, the interested reader is referred to Reference 39.

3 | METHODS

Figure 5 depicts our process of building and training the metamodels. Step one, the "method selection," is based on the literature research presented in Section 2.1 and results in selecting random forest, feed-forward neural networks, and LSTM neural networks as the most promising methods. We use two fundamentally different types of ESOMs, namely a unit commitment model for simulating wholesale electricity prices for Germany and a design optimization model for defining the energy supply of a building. Both models are white-box or fundamental models, that describe the system behavior via a mathematical system of variables and equations. The assumption is that the underlying systemic mechanisms are known and can therefore be modeled. In the second step, the respective input and output data of the models is preprocessed. This process includes validation, encoding, and normalization and is further described in the following sections of Chapter 3. The third step comprises setting up the metamodels. Before fully training the models, pre-tests are conducted based on simplified models to validate the approaches and to decide which approach to pursue further. Following the pre-tests, the methodology was conveyed and extended to the non-simplified models.

For all applications shown in this study, we use the open-source platform *KNIME*,[47] which was first developed at the University of Konstanz in 2004. *KNIME* offers the possibility of integrating a large number of the most common machine learning packages, including *Keras* and *Tensorflow*.[47,48] An extract of the workflows built into *KNIME* can be found in Appendix A (see Figure A1).

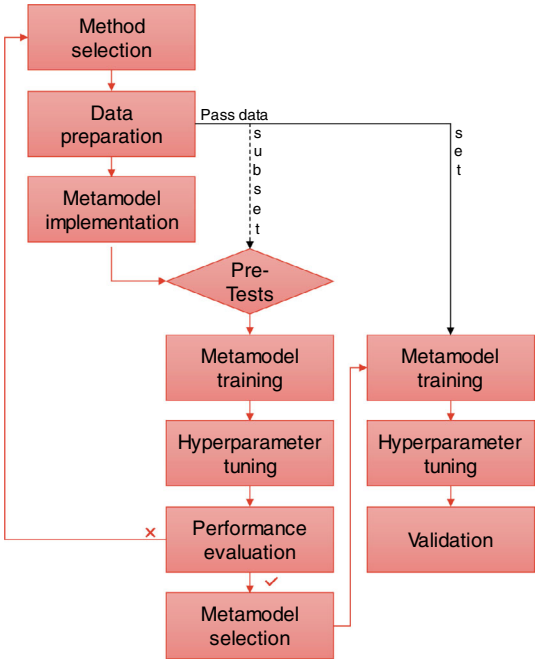


FIGURE 5 Schematic representation of the process for building the metamodels

3.1 | Metamodeling of the unit commitment model

In the following, we introduce the relevant input and output data of the unit commitment model. With $T = \{1,2,...,8760\}$ defining the optimization time horizon, $G = \{1,...,25\}$ defining the considered set of generation technologies, and $N = \{1,...,401\}$ defining the number of considered nodes.

It should be noted that the unit commitment model requires additional input data, such as fuel prices and power plant efficiencies. As the availability of renewable feed-in and demand constitute the main drivers of the electricity price and all other inputs are held constant, the metamodel is limited to input categories shown in Table 2.[‡] 80% of the data is used for training, 20% for validation.

TABLE 2 Input and output data of the unit commitment model used in the metamodeling approach

Name	Description	Unit
Input data		
$D(n,t) \in \mathbb{R}^+ \ \forall n \in N, \forall t \in T$	Electricity demand during hour t	MWh
$C(n,g,t) \in \mathbb{R}^+ \forall n \in N, \forall g \in G, \forall t \in T$	Installed capacity at node n of generator type g during hour t	MW
$\alpha(n,g,t) \in \mathbb{R}^+ \forall n \in N, \forall g \in G, \forall t \in T$	Availability factor at node n of generator type g during hour t	%
Output data		
$P(t) \in \mathbb{R}^+ \forall t \in T$	Electricity price during hour t (equivalent: day-ahead spot price)	EUR/MWh

The input factor time (ie, hour of the year) was further decomposed. Figure 6A) shows the general procedure for encoding time stamps. For the day of the week and hour of the day, we use one-hot encoding. For each day of the week or hour of the day, a binary column is created that is shown exemplarily for the day of the week encoded in Figure 6B). We include time as an input factor because electricity prices differ during certain periods or for certain events, such as weekday against weekend prices or peak against off-peak prices.

For monthly seasonality, the one-hot encoding method should not be applied because the influence of the order of the months would be lost. This means that the metamodel would not acknowledge that February and March are adjacent and that December is continuously followed by January. The intention is to represent seasonal trends rather than representing monthly differences. We solve this by employing periodic encoding. The idea is that the metamodel is not only informed about the current month but also on the temporal proximity of other months. Therefore, during encoding, certain energy values (between 0.0 and 1.0) are applied to each month depending on the distance between the current training day and month.[51] Although the days of the week and the hours of the day have a certain order as well, the more valuable information here is the overall difference between certain days and hours rather than a weekly or daily trend.

In summary, the input data for the metamodel consists of the following features: $D(n,t)$ [MWh], $C(n,g,t)$ [MW], $\alpha(n,g,t)$ [MW], year, month (encoded), day (encoded), hour (encoded). The value to be predicted is the electricity price in EUR/MWh.

To be able to make a robust statement about the performance, the availabilities, installed capacities,[52] and load for the years 2015-2017[53] are used as input. The day-ahead spot prices from 2015-2017[53] and the equivalent model prices, respectively, are used as benchmarks. Furthermore, a 10-fold cross-validation is carried out (see Section 3.4 for more details). Prior to this, the hyperparameters are optimized using Bayesian optimization (using the Tree Parzen Estimator [TPE]) based on the real data. (For further information about the optimization method, refer to References 55,56). The optimized hyperparameters are adopted for the metamodel, which is trained on the model data.[§]

3.2 | Metamodeling of the design optimization model

The next step is to apply the metamodel to another type of model, namely a design optimization model.[57] The main differences from the dispatch model are, first, that future investments are predicted *ex-ante*, that is, there is no real data for comparison, and second, the input and output data have different dimensions. The input data consists of four different time series (each comprising 8,760 time steps, that is, 1 year at hourly resolution) while the output data is a single value per optimized feature. Furthermore, the output data includes continuous as well as binary and integer values with different units. We introduce the relevant input and output data for the design optimization model (Table 3).

The design optimization model was run for 733 different types of buildings and the results serve as the data basis for our metamodeling approach. In our approach, we train different models for each output feature. An alternative approach would have been to train a model predicting all output features as a single feature vector. This can be challenging in terms of model fitting if certain output features are hard to predict based on the input features we selected. This assumption was tested and confirmed during the pre-tests. Therefore, we decided to build individual models to gain further insides into which of the output features is suitable for metamodeling.

The hyperparameter tuning is done manually to maximize the coefficient of determination of each output feature. These hyperparameters were considered: number of epochs, number of hidden layers, number of hidden neurons, and activation function. The evaluation was performed in two separate runs. In each of these, 80% of the data set was randomly selected for training and the remaining 20% for evaluation.

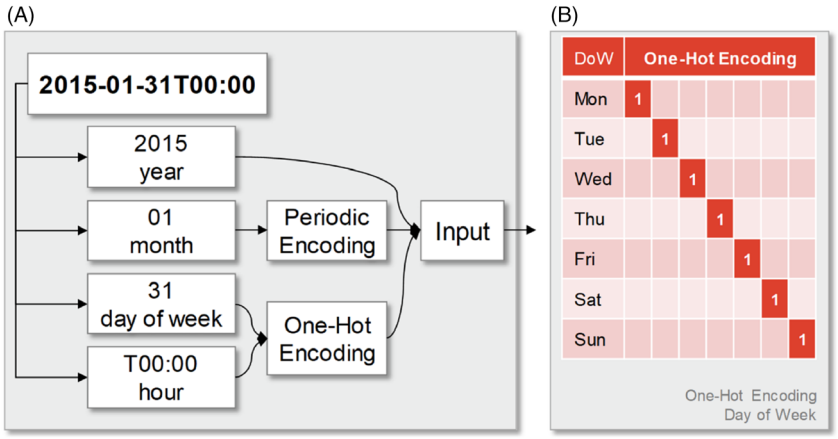


FIGURE 6 (A) Encoding time stamp general procedure; (B) one-hot encoding (based on Reference 51)

3.3 | Pre-testing based on simplified models for narrowing the method scope

By conducting pre-tests, we intend to narrow down the methods applied for metamodeling to the most promising ones. We use simplified versions of the unit commitment and the design optimization models and reduced data sets during the pre-tests. During the pre-tests, no cross-validation is performed.

For the simplified unit commitment model, we reduce the data sets to $T=\{1,2,...,24\}$ defining the optimization time horizon, $G=\{1,...,8\}$ defining the considered set of generation technologies, and $N=\{1\}$ defining the number of considered nodes. The electricity prices as model output to be represented by the metamodels are determined using the simplified dispatch model. According to this procedure 2,400 data points are determined to train the metamodel. 80% of the data is used for training, 20% for validation. For pre-testing the unit commitment model, we build the metamodels based on the following types of methods: Random forest and feed-forward neural networks.

For the design optimization model, the given dataset, which consists of 733 different buildings, a test data set with 250 buildings was extracted via random sampling. Based on this dataset, different metamodels with different network topologies were evaluated. For this purpose, the metamodels were trained with 80% of the test dataset, determined by random sampling, and the remaining 20% was used to evaluate the performance of the different approaches. For pre-testing the design optimization model, we build the metamodels based on the following types of methods: Feed-forward neural networks, LSTM neural networks, convolutional neural networks, and concatenated neural networks).

3.4 | Ensuring the generalizability of the metamodels

In this study, we focus on supervised learning processes. The machine learning algorithms (in our case random forest, feed-forward neural networks, and LSTM neural networks) receive both the input and output data from the original model. During the training process, the metamodel learns the relationships between the input and output data based on the given training dataset by minimizing the error e between the given model output $y=(y_1,...,y_N)$ and the output predicted by the metamodel $\hat{y}=(\hat{y}_1,...,\hat{y}_N)$ for a total of N data points:

$$min\, e(y,\hat{y}) \tag{2}$$

The aim is to ensure that the relationships learned during the training process can be applied to input data unknown to the metamodel and thus an output with high accuracy can be determined.[45] If this generalization ability is achieved, the metamodeling approach will be considered successful.

A typical performance measure for regression tasks is the *mean squared error* (MSE), respectively the *root mean squared error* (RMSE),[43] which is calculated as follows:

$$e(y,\hat{y})=MSE(y,\hat{y})=\frac{1}{N}\sum_{n=1}^{N}(y_n-\hat{y}_n)^2 \tag{3}$$

$$e(y,\hat{y})=RMSE(y,\hat{y})=\sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_n-\hat{y}_n)^2} \tag{4}$$

Another performance measure is the coefficient of determination R^2:

$$R^2(y,\hat{y})=1-\frac{\sum_{n=1}^{N}(y_n-\hat{y}_n)^2}{\sum_{n=1}^{N}(y_n-\bar{y})^2}\text{with}\,\bar{y}=\frac{1}{N}\sum_{n=1}^{N}y_n, \tag{5}$$

where $\bar{y}$ is the average value in the reference dataset. Also, the *mean average percentage error* (MAPE) is

TABLE 3 Input and output data of the design optimization model used in the metamodeling approach

Name	Description	Unit
Input data		
$AC_{building}(t) \in \mathbb{R}^{+} \forall t \in T$	Power consumption of the building	kW_{el}
$AC_{hot\ water}(t) \in \mathbb{R}^{+} \forall t \in T$	Power consumption for hot water	kW_{el}
$H_{building}(t) \in \mathbb{R}^{+} \forall t \in T$	Heat consumption of the building	kW_{th}
$H_{hot\ water}(t) \in \mathbb{R}^{+} \forall t \in T$	Heat consumption for hot water	kW_{th}
Output data		
$AC_{node} \in \mathbb{R}^{+}$	Maximal internal electricity load	kW_{el}
$H_{node} \in \mathbb{R}^{+}$	Maximal internal heat load	kW_{th}
$E_{supply} \in \mathbb{R}^{+}$	Electricity supply	kW_{el}
GS	Gas supply	kW_{th}
LS	Log supply	kW_{th}
$PV_{tariff} \in \mathbb{R}^{+}$	Feed-in tariff for photovoltaics	kW_{el}
HP_{tariff}	Heat pump tariff	kW_{el}
FC	Fuel cell	kW_{el}
PV_1	Photovoltaic first roof	kW_{el}
PV_2	Photovoltaic second roof	kW_{el}
ST_1	Solar thermal first roof	m^2
ST_2	Solar thermal second roof	m^2
DH	District heating	kW_{th}
EH	Electric heater	kW_{th}
FP	Fireplace	kW_{th}
GB	Gas boiler	kW_{th}
HP	Heat pump	kW_{th}
B	Building design heat load	kW_{th}
BAT	Electric battery	kWh_{el}
HS	Heat storage	kWh_{th}

Note: See Appendix C, Table C1 for a more detailed overview of the model's properties.

commonly used for measuring the performance of models that forecast prices and is therefore well-transferable to our application. It is calculated as follows:

$$MAPE(y,\hat{y}) = \frac{1}{N}\sum_{n=1}^{N}\left|\frac{y_n - \hat{y}_n}{y_n}\right| \quad (6)$$

Some difficulties during the design and training process of machine learning algorithms have been identified by Reference 43. These mainly comprise the problems of over- and underfitting, which are illustrated in Figure 7. Underfitting indicates that the model representing the system's relationship is too simple to represent the system dynamics.[44] The model can neither map the system relationship nor the samples, which leads to a high training error and low generalization ability. Overfitting, on the contrary, indicates that the noise of the samples has been learned beyond the system relationship and the model is too complex regarding the number and noise of the training set.[43] Although overfitting leads to a small training error, it also leads to the low generalizability of the model.

To counteract underfitting, the model can be extended. Counteracting in the case of overfitting requires model simplification, reducing the noise in the training data, or extending the training dataset to reduce the effect of existing data noise.[43] One possible solution to this is the adjustment of the hyperparameters.

Hyperparameters are initialized prior to the training and do not change due to the learning process. These parameters control the training algorithm and must be optimized to achieve the best possible performance in the model.

Random forests, as well as ANNs, possess a multitude of hyperparameters that can be optimized, whereby the number of hyperparameters of the neural network exceeds the number of random forests.[31,46] Due to the large number of possible hyperparameters to optimize, a selection of hyperparameters to be optimized is made in this study. These include the following:

ANN: number of epochs, number of hidden layers, number of hidden neurons, activation function.

Random forest: number of decision trees, minimum number of data points of a node to branch, minimum number of data points of a leaf, maximum tree depth.

Other hyperparameters are held constant at their predefined standard levels.

To ensure generalizability, validation of the hyperparameter tuning should not be limited to the training error, as this is very low in the case of overfitting. Therefore, the *k-fold cross-validation* method is used here. This method prevents the loss of prediction accuracy due to a smaller training dataset caused by the separation of the validation data and, at the same time,[48] enables a robust estimation of the model's performance. The *k-fold cross-validation* is displayed schematically in Figure 8.

4 | RESULTS AND DISCUSSION

In the following section, we present our metamodeling results for the unit commitment and the design

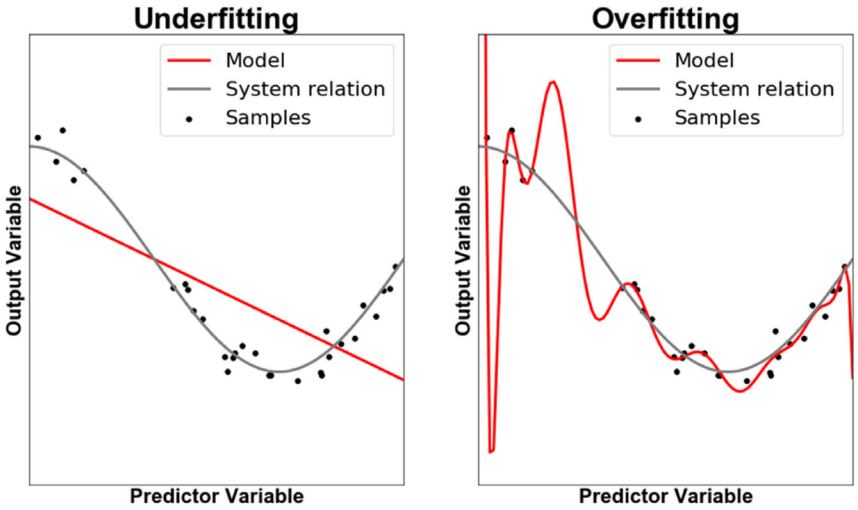


FIGURE 7 Under- and overfitting (own illustration, based on Reference 43)

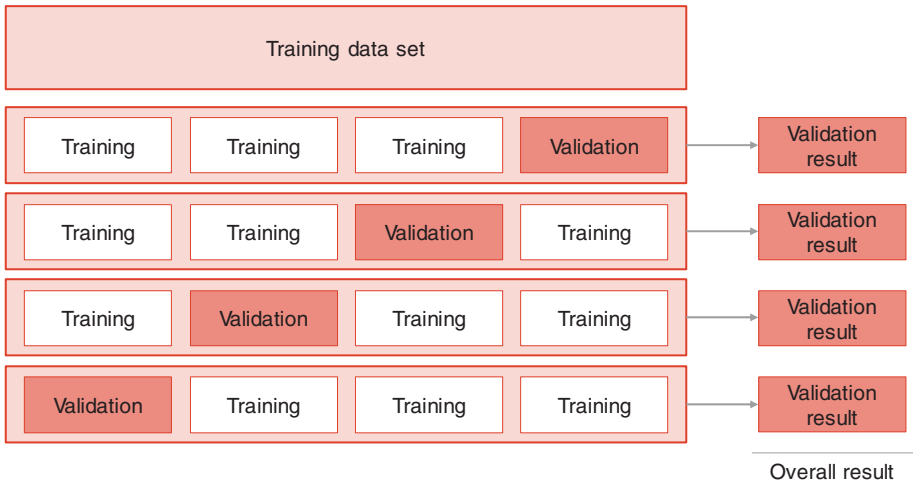


FIGURE 8 K-fold cross-validation (k = 4) (own illustration, based on Reference 43)

optimization models. First, the results from the pre-tests are presented, followed by the extensive analyses based on the non-simplified models. The unit commitment model was investigated to evaluate whether metamodels are capable of forecasting the model's main output, namely the electricity price. The evaluation was carried out *ex-post* using real data. The second metamodel building upon the design optimization model tries to emulate the recommendations made for optimal future investments in the energy supply system of a single building. This evaluation is carried out *ex-ante* using the model's output.

4.1 | Pre-tests based on the simplified dispatch model

In the following, we present the results of the pre-tests. The results for the unit commitment model are shown in detail to illustrate the general procedure.

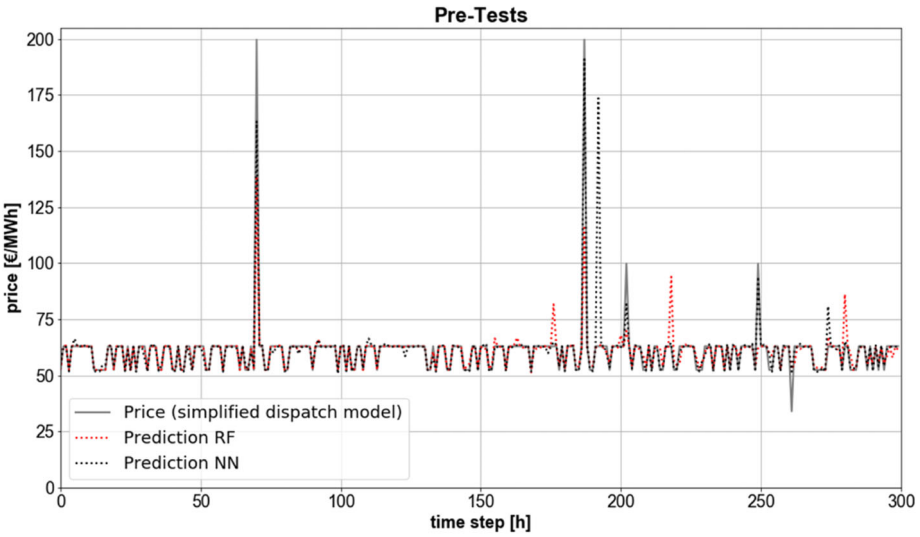
The results of the pre-tests for the unit commitment model are depicted in Figure 9. Shown in red are the prices determined by the simplified dispatch model. The dotted red line represents the price prediction of the metamodel based on random forest and the dotted black line the prediction of the metamodel based on a feed-forward neural network. The prices in the range from 50 EUR/MWh to 60 EUR/MWh are met fairly well, but the prediction for the peaks is not always reliable. We perform a hyperparameter optimization using the coefficient of determination R^2 as a performance measure to be maximized. A random search[49] is applied to search a previously defined space of hyperparameter configurations to determine the best combination.

The pre-tests result in:

$$\text{Random forest}: \; R2 = 0.74, MSE = 35.33 \left[\frac{EUR^2}{MWh^2}\right],$$

$$RMSE = 5.94 \left[\frac{EUR}{MWh}\right], \; MAPE = 2.18\%$$

FIGURE 9 Results of the pre-tests



$$\text{Neural network}: R^2 = 0.73,\ MSE = 32.11\left[\frac{EUR^2}{MWh^2}\right],$$
$$RMSE = 5.67\left[\frac{EUR}{MWh}\right],\ MAPE = 1.88\%$$

Given that the results of the pre-tests are based on a relatively small number of data points, it can be shown that random forests and feed-forward neural networks seem to be generally suitable for the metamodeling of a dispatch model. Although the performance measures of the two tested approaches do not significantly differ, the ANN approach is used in the following analysis. This decision is based on the fact that the variety of possible network topologies of neural networks significantly exceeds that of modeling random forests,[44] allowing for even further improved performance. Also, several studies (eg, Reference 50) reveal that the performance of random forests in forecasting time series is weaker than the performance of neural networks. Therefore, we only examine feed-forward neural networks as metamodeling approaches in our subsequent analysis for the unit commitment model. The results are shown in Section 4.2.

For the design optimization model, we trained metamodels based on a simplified version of the design optimization model. We tested the prediction of all 20 output values and the prediction of single output values using separate metamodels. The results of the pre-tests showed that the simultaneous prediction of all output features based on a single hyperparameter optimization does not lead to the expected metamodel performance. We, therefore, use individual models for each output feature for the non-simplified case. The most promising approaches were feed-forward neural networks and an LSTM neural networks. These two methods are applied to the entire dataset and the results are shown in Section 4.3.

4.2 | Forecasting of price time series using metamodels

We compare the price forecast accuracy of the metamodel using multiple benchmarks: the historic electricity prices, the prices computed by the unit commitment model, and a prediction model trained not on the model output but directly on the historic values.

Figure 10 shows the four different electricity price time series for a given week. The historic (real) prices are represented by the red solid line and the model-determined prices by the gray solid line. Equivalent to this is the dotted lines, the red one representing the price prediction of the metamodel trained on the real data and the gray one the price prediction of the metamodel trained on the model data.

As mentioned in Section 3.1, the unit commitment model has a lower variety of prices as model output compared to the historic (real) data. This makes it easier for the metamodel to depict the model prices than the real ones, which is also reflected by the performance measures in Table 4.

In summary, it can be concluded that the metamodel is suitable for application to the unit commitment model and that sufficiently good results can be achieved, sometimes even better than those of the model being replaced.

4.3 | Determination of the optimal system design using metamodels

For the case of the design optimization model, we analyze the accuracy of our metamodel for reproducing the model's design decisions. As explained in Section 3.2, individual metamodels are trained for the 20 design

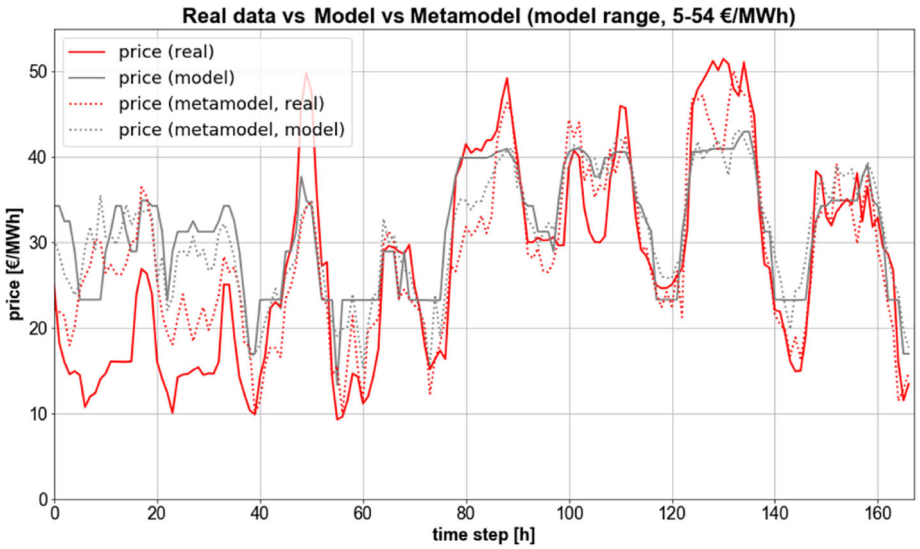


FIGURE 10 Results of the metamodel's performance on real data vs model data

Comparison	R^2 [%]	MSE $\left[\frac{EUR^2}{MWh^2}\right]$	RMSE $\left[\frac{EUR}{MWh}\right]$	MAPE [%]
Real - Model	78.8	22.82	4.78	13.60
Real - $\text{Metamodel}_{\text{real}}$	82.9	16.21	4.03	12.08
Model - $\text{Metamodel}_{\text{model}}$	88.0	7.21	2.69	8.54

TABLE 4 Performance measures of the unit commitment model and the metamodel

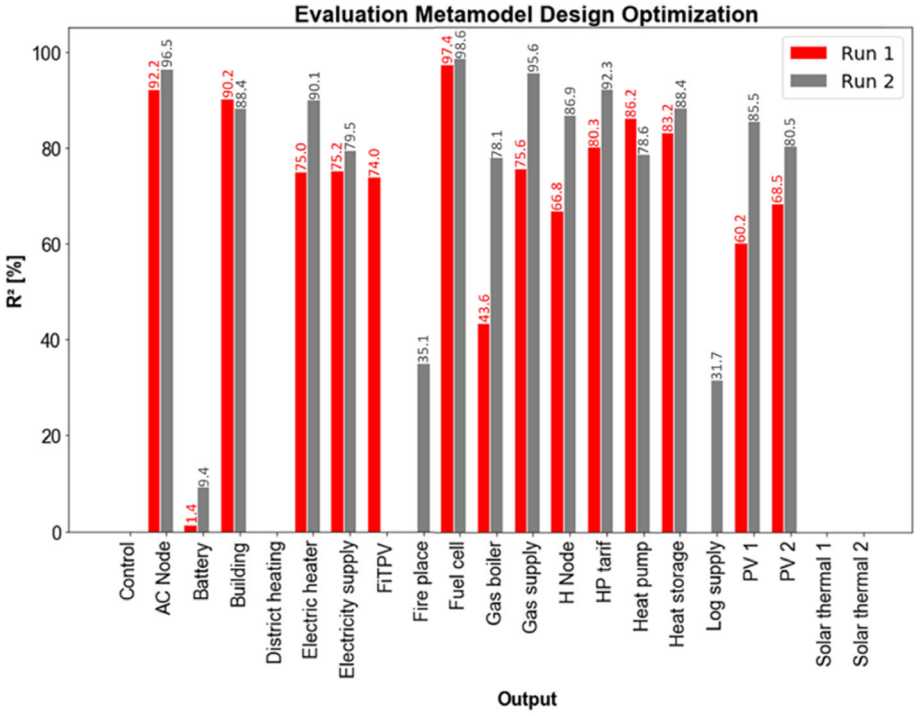


FIGURE 11 Evaluation of the metamodel for the design optimization model based on a feed-forward neural network for two random training datasets (gray and red)

decisions made by the models. We conduct two separate runs with different training data (each run including hyperparameter optimization) to objectify our results.

Looking at the metamodel results for the design optimization model, the feed-forward neural network shows better results than the LSTM neural network. Figure 11

shows the results for the feed-forward neural network for the two separate runs. The various output features are plotted on the x-axis and the bars indicate the coefficient of determination (R^2) separately for each run and feature. The quality of the prediction of the individual features varies significantly, whereby certain groups can be distinguished. The first group of features is predicted well in both runs, whereas the second group is predicted poorly. Finally, for the third group of features, the predictions varied strongly depending on the training sample. A

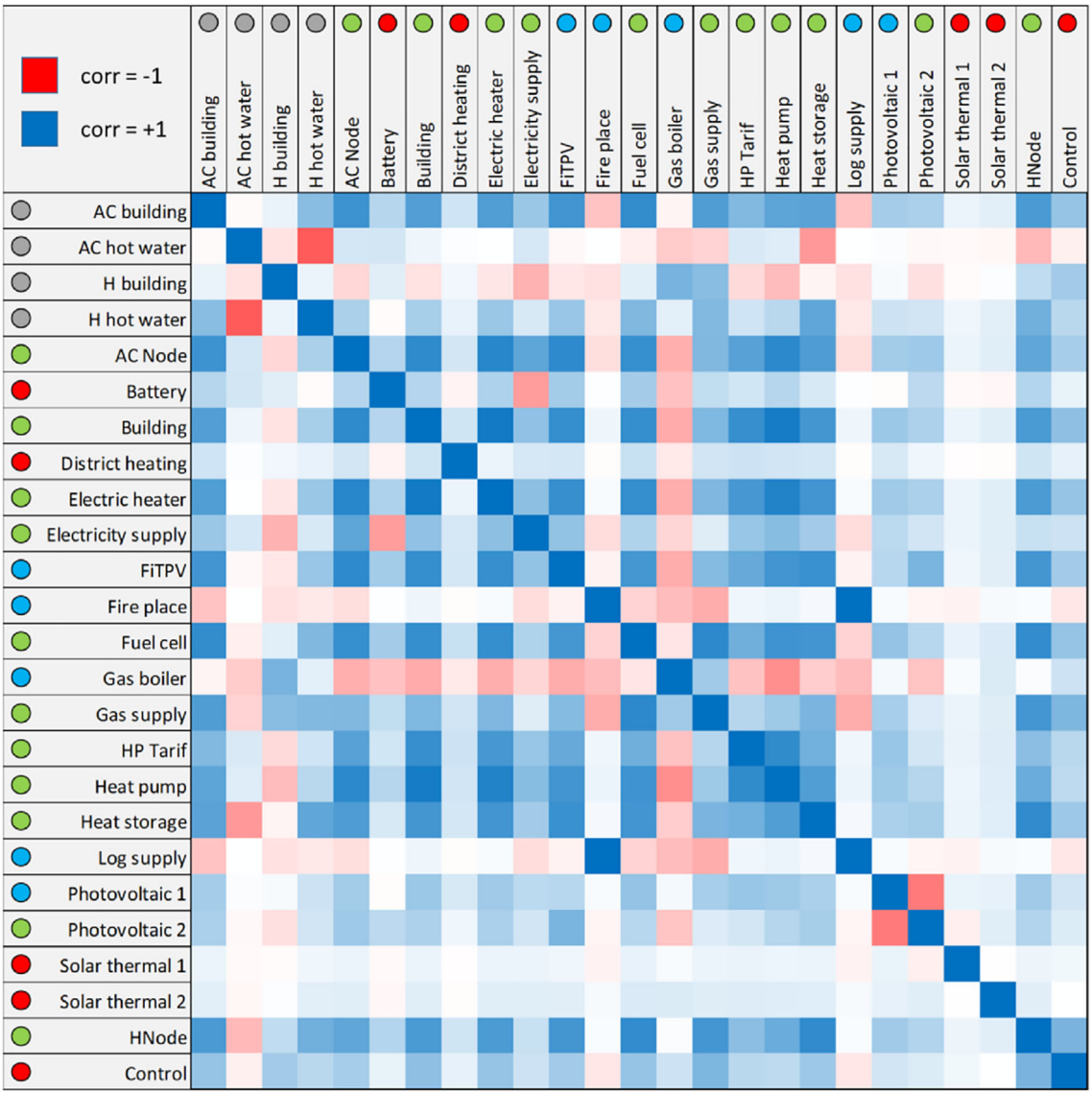


FIGURE 12 Correlation analysis of the design optimization model

TABLE 5 Runtimes of the white-box dispatch model and the black box-metamodel

Model type	Metamodel training and hyperparameter tuning	Runtime per scenario	Hardware specifications
White-box: non-simplified dispatch model	—	~2 h	• Intel® Core(TM) i5-8250U CPU @ 1.60 GHz • 32.00 GB RAM
Black-box: metamodel	~0-2 h for training ~6-72 h for hyperparameter tuning, depending on tuning strategy	<30 s	• AMD A6-5200 APU with Radeon(TM) HD Graphics 2.00 GHz • 8.00 GB RAM

value of the coefficient of determination close to zero indicates that the metamodel cannot replicate the outcome of the model correctly regarding this feature.

To understand the background of this pattern, correlation analyses of the input and output features, as well as the output features themselves are performed. The results are shown in Figure 12. Input features are marked with gray dots. Features that are predicted well in both runs are marked with green dots and features that cannot be predicted well in both runs are marked with red dots. For the case that features are predicted well in one of the two runs, they are marked with blue dots. Within the matrix, red represents a negative and blue a positive correlation between the features.

Firstly, it can be seen that most output features are primarily correlated with the electricity demand "*AC building*" and the hot water demand "*H hot water*". Most of the well-predicted output features are strongly correlated to these two input features. The poor performance in predicting some of the outputs indicates that the chosen input set is too small to determine the optimal configuration of the metamodel during the training process. The sample-dependent outputs exhibit correlations with many of the other outputs, indicating that these outputs cannot be predicted based on the limited input set alone, for example, roof inclinations, roof availability, or geographical location would be required to properly predict the optimal photovoltaic capacities.

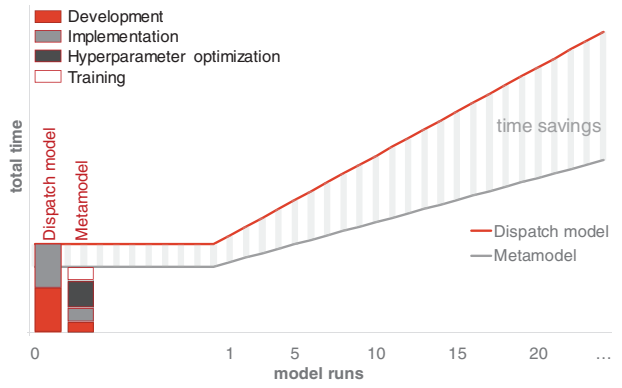


FIGURE 13 Complexity reduction dispatch model vs metamodel (illustrative)

5 | DISCUSSION: THE VALUE-ADDED OF METAMODELS FOR ENERGY SYSTEM MODELING

In this section, we summarize and discuss our findings and derive general modeling recommendations for the application of metamodels. As the number of ESMs used in practice is large and their scopes and objectives differ significantly, we now abstract from our results and derive statements of a more general validity.

Our results indicate that the approach of metamodeling using a feed-forward neural network is highly applicable to the dispatch model. In practice, this leads to a reduction in the complexity, as the neural network-based metamodel can save runtimes for each scenario run once it is sufficiently trained. A comparison of the runtimes of the unit commitment model (white-box) against the metamodel (black-box) is shown in Table 5.

The resulting achievement of complexity reduction is shown in Figure 13. It should be noted that the figure is only a schematic illustration and the values may vary depending on the model. In general, we assume that the time for the development and implementation of the metamodel is significantly shorter, as not every relationship between the smallest components of the system needs to be modeled individually. Even though the hyperparameter optimization and the training can require a lot of time, once the metamodel is ready for use, time is saved with each simulation performed by the metamodel. The runtime of the metamodel, once configured, is substantially shorter than that of the unit commitment model. Even if the configuration of the metamodel should take longer than the modeling of the unit commitment model, the metamodel will be faster as the number of model runs increases. This feature of the metamodel will be of great advantage if a large number of different scenarios need to be considered. Each modeler must, however, evaluate the trade-off between

TABLE 6 Runtimes of the white-box system design model and black-box metamodel

Model type	Metamodel training and hyperparameter tuning	Runtime per scenario	Hardware specifications
White box: system design model	—	~ 0.8 h in a single core	• Two Intel(R) Xeon(R) Platinum 8180 CPUs and • 512 GB RAM
Black box: metamodel	~ 0-2 h for training ~ 6-72 h for hyperparameter tuning. Depending on tuning strategy	<30 s	• AMD A6-5200 APU with Radeon(TM) HD Graphics 2.00 GHz • 8.00 GB RAM

modeling effort and runtime savings to achieve an optimum reduction in complexity.

In the application of our metamodeling approach to the more complex design optimization models, we encountered several obstacles. Initially, the results showed that the simultaneous prediction of all output features based on a single hyperparameter optimization does not lead to the expected metamodel performance. This may be because the relationships between the outputs cannot be correctly represented by the chosen metamodel. Hence, other network topologies must be tested. Moreover, it could be possible that the hyperparameters for the individual outputs must be optimized separately.

The second challenge is that the design optimization model is used to forecast future investments *ex-ante*, which means that, unlike the dispatch model, there is no real data that the metamodel can be trained on. Instead, a white-box model is required first, based on which training data must be generated for the metamodel. In that case, the metamodel will only be able to depict those relationships that are correctly represented in the white-box model. Still, the benefit of the metamodel is to enhance the number of possible scenario variants to be investigated, thereby increasing the robustness of the model results. Table 6 displays the runtimes of the white-box and black-box models.

A hybrid approach would be a conceivable solution which is shown schematically in Figure 14. As before, the illustrated values may vary depending on the model. First, a fundamental model is developed and some scenarios are considered. In this way, training data for the metamodel is generated. The metamodel could then be used to consider several more scenarios. At this point, it must be considered whether the time saved during the model runs can outweigh the additional work involved in configuring the metamodel.

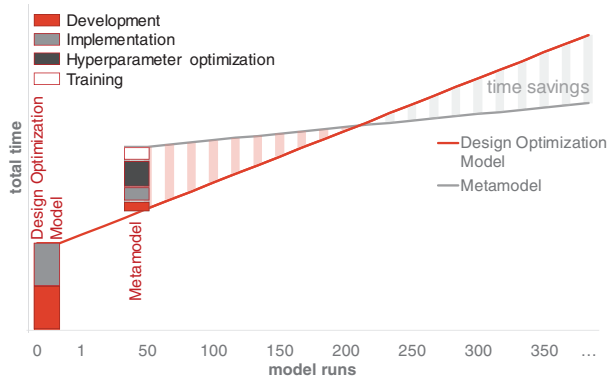


FIGURE 14 Complexity reduction hybrid approach - design optimization model vs metamodel (illustrative)

Furthermore, it should be noted that the metamodel approaches analyzed in this study are black-box models. This means that the metamodels are highly applicable for results-oriented studies. However, if the exact relationships between the individual components in the ESMs are to be investigated, black box metamodels are not practical and fundamental models should instead be used. Hence, the modeler must keep in mind the tradeoff between reduced model complexity and increasing the loss of information when applying metamodeling approaches to energy system analysis. A hybrid approach combining white- and black-box models would again offer a promising solution to this transparency issue.

6 | CONCLUSIONS

In this study, we investigated the question of whether a reduction in complexity can be achieved by metamodeling ESMs using approaches based on deep learning. We selected two types of energy system optimization model and evaluated the performance of deep learning-based metamodeling approaches to come up with general recommendations for the application of metamodels in energy system modeling.

We applied metamodel based on a feed-forward neural network to the unit commitment model and demonstrated that metamodeling is highly applicable to predictive energy system optimization models. We found that, as the metamodel is a black-box approach, the implementation effort and the run time per scenario were considerably lower compared to conventional white-box modeling approaches. By reducing the implementation effort as well as the runtime, a complexity reduction could be achieved while maintaining the high accuracy of the results. This could also help increase the robustness of the modeling results, as metamodels enable the investigation of a larger number of scenarios, thus depicting the uncertainty of future developments more precisely. A promising approach for handling uncertainty is combining metamodeling with the design of the experiment approach, as done by Nolting et al.[21]

The drawback of the black-box approach compared to the underlying fundamental model lies in the traceability of its results. Unlike in the case of a comparably straightforward unit commitment model, some difficulties arose when applying the metamodel to the design optimization model. We found that the metamodeling of more ESMs that are used for ex-ante analysis (often in the form of a greenfield analysis), including design optimization models, is accompanied by higher demands on the input data for training the metamodel. Additional information alongside

the input data of the fundamental model may also be required. Furthermore, the simultaneous forecasting of different interdependent outputs proved to be problematic.

Our results indicate that for *ex-ante* analyses, the application of a hybrid approach is a promising strategy for applying metamodels. For this, a selection of representative scenarios is initially evaluated using a fundamental model, which is then followed by the training of a metamodel to consider a variety of other scenarios with shorter solving times. Thereby, the choice of an efficient learning set is key whereby future works should identify efficient sampling methods that result in the improved training of the metamodel.

As discussed above, there is a need for further research, especially in the metamodeling of design optimization models. Research should be extended to models with the same characteristics, namely: the forecasting of future data without a historic data basis and the simultaneous optimization of several (interdependent) variables. The metamodeling approaches considered in this study demonstrate substantial potential for complexity reductions and can be transferred to various types of ESMs. For future research, it would be of particular interest to further evaluate and quantify the tradeoff between the modeling effort of the metamodel and the time savings and complexity reduction achieved.

NOMENCLATURE

Indices

g	generation technology
i	input feature
n	output feature
t	time step
y	year

Symbols

σ_i	tuple decision tree for input feature i
t_i	threshold value for input feature i
$x^{(i)}$	value of input feature i
y_n	reference value of output feature n
$\widehat{y}_n$	predicted value of output feature n
$\overline{y}$	average reference value
$availability_{g,t}$	availability [MW] of generation tech g during time step t
$capacity_{g,y}$	installed capacity [MW] of generation tech g per year y
$demand_t$	electricity demand [MW] in time step t
$price_t$	day-ahead spot price [EUR/MWh] in time step t

ACKNOWLEDGMENT

Open access funding enabled and organized by Projekt DEAL.

CONFLICT OF INTEREST

The authors declare no conflicts of interest.

ENDNOTES

[*] The application of metamodels based on artificial neural networks to energy system optimization models on a large scale is amongst the goals of ongoing research projects such as *UNSEEN* (funded by the German Federal Ministry of Economic Affairs and Energy).

[†] For a brief introduction of ANNs, see also Reference 44.

[‡] See Appendix B, Table B1 for a more detailed on the unit commitment model.

[§] An analysis of the price time series, both real and model-determined, shows that the model prices are preprocessed capped to a certain price range due to model limitations. Working with preprocessed data is common in machine learning and not necessarily a problem.[38] Therefore, the metamodel is trained on the data in the price data representing the feasible model range. The filtered data sets contain 24 307 of the original 26 304 data points.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available on request from the corresponding author.

ORCID

Jan Priesmann https://orcid.org/0000-0002-9127-0545
Lars Nolting https://orcid.org/0000-0001-8800-5807
Leander Kotzur https://orcid.org/0000-0001-5856-2311
Martin Robinius https://orcid.org/0000-0002-5307-3022
Aaron Praktiknjo https://orcid.org/0000-0002-2151-4241

REFERENCES

1. Allwood JM, Bosetti V, Dubash NK, Gómez-Echeverri L, von Stechow C. Contribution of Working Group III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change: glossary. In: Edenhofer O, Pichs-Madruga R, Sokona Y, Farahani E, Kadner S, et al., eds. *Climate Change 2014: Mitigation of Climate Change*. New York, NY: Cambridge University Press; 2014.
2. Schellong W. Herausforderungen der künftigen Energieversorgung. In: Schellong W, ed. *Analyse und Optimierung von Energieverbundsystemen*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2016:1-32.
3. Herbst A, Toro F, Reitze F, Jochem E. Introduction to energy systems modelling. *Swiss J Econ Stat*. 2012;148(2):111-135. doi:10.1007/BF03399363
4. Weijermars R, Taylor P, Bahn O, Das SR, Wei Y-M. Review of models and actors in energy mix optimization – can leader visions and decisions align with optimum model strategies for our future energy systems? *Energy Strat Rev*. 2012;1(1):5-18. doi:10.1016/j.esr.2011.10.001
5. Martinsen D, Krey V. Compromises in energy policy—Using fuzzy optimization in an energy systems model. *Energy Policy*. 2008;36(8):2983-2994. doi:10.1016/j.enpol.2008.04.005
6. nationalgridESO Future Energy Scenarios (FES). https://www.nationalgrideso.com/insights/future-energy-scenarios-fes.

7. Priesmann J, Nolting L, Praktiknjo A. Are complex energy system models more accurate? an intra-model comparison of power system optimization models. *Appl Energy*. 2019;255:113783.
8. Lopion P, Markewitz P, Robinius M, Stolten D. A review of current challenges and trends in energy systems modeling. *Renew Sustain Energy Rev*. 2018;96:156-166. doi:10.1016/j.rser.2018.07.045
9. Ridha E, Nolting L, Praktiknjo A. Complexity profiles: a large-scale review of energy system models in terms of complexity. *Energy Strat Rev*. 2020;30:100515. doi:10.1016/j.esr.2020.100515
10. Priesmann J, Nolting L, Kockel C, Praktiknjo A. Time series of useful energy consumption patterns for energy system modeling. *Scientific Data*. 2021;8(1) doi:10.1038/s41597-021-00907-w
11. DeCarolis J, Daly H, Dodds P, et al. Formalizing best practice for energy system optimization modelling. *Appl Energy*. 2017;194:184-198. doi:10.1016/j.apenergy.2017.03.001
12. Wang GG, Shan S. Review of metamodeling techniques in support of engineering design optimization. *J Mech Des*. 2007;129(4):370-380. doi:10.1115/1.2429697
13. Zaibi M, Layadi TM, Champenois G, et al. A hybrid spline metamodel for photovoltaic/wind/battery energy systems. Paper presented at: IREC2015 The Sixth International Renewable Energy Congress, 2015 Sixth International Renewable Energy Congress (IREC); March 24-26, 2015; Sousse, Tunisia: IEEE; 1-6. doi: 10.1109/IREC.2015.7110909
14. Chlela F, Husaunndee A, Inard C, Riederer P. A new methodology for the design of low energy buildings. *Energy Build*. 2009;41(9):982-990. doi:10.1016/j.enbuild.2009.05.001
15. Yong S-G, Kim J, Cho J, Koo J. Meta-models for building energy loads at an arbitrary location. *J Build Eng*. 2019;25:100823. doi:10.1016/j.jobe.2019.100823
16. Coelho RF, Bouillard P. Multi-objective reliability-based optimization with stochastic metamodels. *Evolutionary Computation*. 2011;19(4):525-560. doi:10.1162/evco_a_00034
17. Wang H, Li E, Li GY, Zhong ZH. Development of metamodeling based optimization system for high nonlinear engineering problems. *Adv Eng Softw*. 2008;39(8):629-645. doi:10.1016/j.advengsoft.2007.10.001
18. Chen J, Gao X, Hu Y, Zeng Z, Liu Y. A meta-model-based optimization approach for fast and reliable calibration of building energy models. *Energy*. 2019;188:116046. doi:10.1016/j.energy.2019.116046
19. Betiku E, Taiwo AE. Modeling and optimization of bioethanol production from breadfruit starch hydrolyzate vis-à-vis response surface methodology and artificial neural network. *Renew Energy*. 2015;74:87-94. doi:10.1016/j.renene.2014.07.054
20. Fang K, Li R-z, Sudjianto A. *Design and Modeling for Computer Experiments*. Boca Raton, FL: Chapman & Hall/CRC; 2006:290.
21. Nolting L, Spiegel T, Reich M, Adam M, Praktiknjo A. Can energy system modeling benefit from artificial neural networks? Application of two-stage metamodels to reduce computation of security of supply assessments. *Comput Ind Eng*. 2020;142:106334. doi:10.1016/j.cie.2020.106334
22. Prada A, Gasparella A, Baggio P. On the performance of metamodels in building design optimization. *Appl Energy*. 2018;225:814-826. doi:10.1016/j.apenergy.2018.04.129
23. Dudenredaktion (o.J.) "meta" auf Duden online. https://www.duden.de/rechtschreibung/meta_. Retrieved January 23, 2020.
24. Kühne T. Matters of (meta-) modeling. *Softw Syst Model*. 2006;5(4):369-385. doi:10.1007/s10270-006-0017-9
25. Belaunde M, Burt C, Casanave C. *MDA Guide Version 1.0.1*. 1st ed. Needham, MA; 2003. https://www.omg.org/news/meetings/workshops/UML_2003_Manual/00-2_MDA_Guide_v1.0.1.pdf
26. Simpson TW, Poplinski JD, Koch PN, Allen JK. Metamodels for Computer-based Engineering Design: survey and recommendations. *Eng Comput*. 2001;17(2):129-150. doi:10.1007/PL00007198
27. Rumelhart DE, Widrow B, Lehr MA. The basic ideas in neural networks. *Commun ACM*. 1994;37(3):87-92. doi:10.1145/175247.175256
28. Baldi S, Yuan S, Endel P, Holub O. Dual estimation: Constructing building energy models from data sampled at low rate. *Applied Energy*. 2016;169:81-92. doi:10.1016/j.apenergy.2016.02.019
29. Baldi S, Quang TL, Holub O, Endel P. Real-time monitoring energy efficiency and performance degradation of condensing boilers. *Energy Conversion and Management*. 2017;136:329-339. doi:10.1016/j.enconman.2017.01.016
30. Schellong W, ed. *Analyse und Optimierung von Energieverbundsystemen*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2016.
31. Hutter F, Xu L, Hoos HH, Leyton-Brown K. Algorithm runtime prediction: methods & evaluation. Paper presented at: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015), 2015:79-111.
32. Hutter F, Xu L, Hoos HH, Leyton-Brown K. Online Appendix for AIJ Article "Algorithm Runtime Prediction: Methods & Evaluation". http://www.cs.ubc.ca/labs/beta/Projects/EPMs/EPMs-online-appendix-opt.pdf.
33. Kakas AC, Cohn D, Dasgupta S, Barto AG, Carpenter GA, et al. Artificial neural networks. In: Sammut C, Webb GI, eds. *Encyclopedia of Machine Learning*. Boston, MA: Springer US; 2011:44.
34. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958;65(6):386-408.
35. Uther W, Mladenić D, Ciaramita M, Berendt B, Kołcz A, et al. Topology of a neural network. In: Sammut C, Webb GI, eds. *Encyclopedia of Machine Learning*. Boston, MA: Springer US; 2011:988-989.
36. Zeil A. *Simulation neuronaler Netze*. München, Germany: R. Oldenburg Verlag München Wien; 1994.
37. Behm C, Nolting L, Praktiknjo A. How to model European electricity load profiles using artificial neural networks. *Appl Energy*. 2020;277:115564. doi:10.1016/j.apenergy.2020.115564
38. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Computation*, 1997:1735-1780.
39. van Veen F. Neural Network Zoo Prequel: Cells and Layers. https://www.asimovinstitute.org/author/fjodorvanveen/. Retrieved January 25, 2020.
40. Nguyen M. Illustrated Guide to LSTM's and GRU's: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. Retrieved January 26, 2020.
41. Maer S. Recurrent Neural Networks und LSTM. https://www.ai-united.de/recurrent-neural-networks-und-lstm/. Retrieved January 25, 2020.
42. Breiman L. Random forests. *Machine Learning*, 2001:5-32. doi:10.1023/A:1010933404324
43. Géron A. *Hands-on Machine Learning with Scikit-Learn and TensorFlow. Concepts, Tools, and Techniques to Build Intelligent Systems*. Vol 11. Sebastopol, CA: O'Reilly Media; 2017.

44. Bonaccorso G. *Mastering Machine Learning Algorithms. Expert Techniques to Implement Popular Machine Learning Algorithms and fIne-Tune your Models*. Birmingham, England: Packt Publishing; 2018.
45. Chollet F, Allaire JJ, eds. *Deep Learning mit R und Keras. Das Praxis-Handbuch; von Entwicklern von Keras und RStudio = Deep learning with R*. 1st ed. Frechen, Germany: mitp; 2018.
46. Radhakrishnan P. What are Hyperparameters? and How to tune the Hyperparameters in a Deep Neural Network? https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a.
47. KNIME, "Introductory Course to Data Science," URL: https://www.knime.com/knime-introductory-course.
48. Ng A, Soo K, eds. *Data Science - was ist das eigentlich?! Algorithmen des maschinellen Lernens verständlich erklärt*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2018.
49. Bergstra J, Bengio Y. Random search for hyper-parameter optimization. *J Mach Learn Res*. 2012;13:281-305.
50. Tyralis H, Papacharalampous G. Variable selection in time series forecasting using random forests. *Algorithms*. 2017;10(4):114. doi:10.3390/a10040114
51. Scientific Strategy. Advanced Demand Forecasting Neural Networks. https://scientificstrategy.com/kn-302/.
52. Neon Neue Energieökonomik. Technical University of Berlin, ETH Zürich, DIW Berlin, Open Power System Data. https://open-power-system-data.org/about/.
53. entso-e. Power Statistics. https://www.entsoe.eu/data/power-stats/.
54. Danish Ministry of Climate, Energy and Utilities (Energinet). Energi Data Service. https://www.energidataservice.dk/collections/whole-sale-market.
55. Dewancker I, McCourt M, Clark S. Bayesian Optimization Primer; 2015. https://app.sigopt.com/static/pdf/SigOpt_Bayesian_Optimization_Primer.pdf
56. Koehrsen W. A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning: The concept behind efficient hyperparameter tuning using Bayesian optimization. https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f. Retrieved January 29, 2020.
57. Kotzur L, Markewitz P, Robinius M, et al. Bottom-up energy supply optimization of a national building stock. *Energ Build*. 2020;209:109667. doi:10.1016/j.enbuild.2019.109667
58. Kotzur L. *Future Grid Load of the Residential Building Sector. Die zukünftige elektrische Netzlast der Wohngebäude*. Vol. 213. Jülich, Germany: Forschungszentrum Jülich GmbH, Zentralbibliothek, Verlag; 2018.
59. Kannengießer T, Hoffmann M, Kotzur L, et al. Reducing computational load for mixed integer linear programming: an example for a district and an Island energy system. *Energies*. 2019;12(14):2825. doi:10.3390/EN12142825

How to cite this article: Köhnen CS, Priesmann J, Nolting L, Kotzur L, Robinius M, Praktiknjo A. The potential of deep learning to reduce complexity in energy system modeling. *Int J Energy Res*. 2022;46(4):4550-4571. doi:10.1002/er.7448

APPENDIX A.

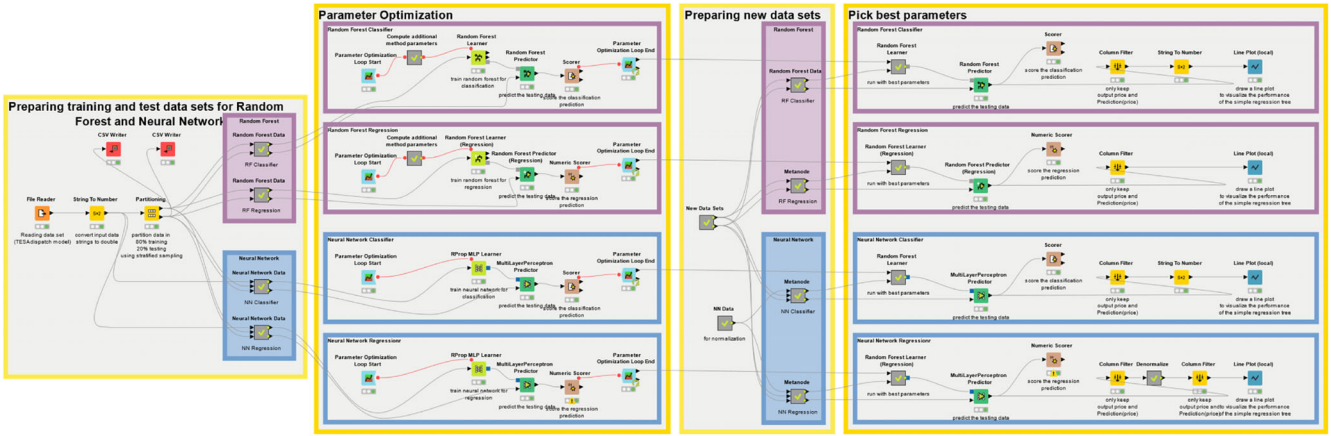


FIGURE A1 KNIME workflow for random forests and artificial neural networks

TABLE B1 Relevant model properties of the non-simplified dispatch model (see Reference 7 for a detailed model description)

Target function	Restrictions (extract)	Temporal resolution	Spatial resolution	Representation of generator types
Minimize dispatch costs: $\min_{x(g,t)} \sum_{g,n,t} \left[x(g,n,t) \cdot \left(\frac{FP(g,n)+EF(g,n)*EC(t)}{\eta(g,n)} + OVC(g,n) \right) \cdot l(t) \right]$,	1. Cover demand: $\sum_g x(g,n,t) - \sum_{m \in N/n} [y(n,m,t) + y(m,n,t)] = d(n,t) \forall n,t \in N,T$ 2. Keep capacity limits: $x(g,n,t) \leq Cap(g,n) \forall g,n,t \in G,N,T$ 3. Keep operational limits (extract): $x(g,n,t) \geq P_{min}(g.n) \mid x(g,n,t) = 0 \forall g,n,t \in G,N,T$ 4. Keep transmission limits: $y(n,m,t) < Cap(n,m) \forall (n,m), t \in (N,N),T$ 5. Non-negativity constraint: $x(g,t) \geq 0 \quad \forall g,t \in G,T$,	8760 time steps representing 1 y	Up to 401 regions within Germany and eight neighboring countries	25 different generator types: $G = \{1,\ldots,25\}$
With $x(g,n,t)$:= Dispatched power of generator type g at node n during hour t [MW] $y(n,m,t)$:= Transmitted power from node n to node m during hour t [MW] $FP(g,n)$:= Fuel price for generator type g at node n $\left[\frac{\text{EUR}}{\text{MWh}_{\text{thermal}}}\right]$ $EF(g,n)$:= Emission factor for generator type g at node n $\left[\frac{t_{CO_2}}{\text{MWh}_{\text{thermal}}}\right]$ $EC(t)$:= Emission costs during hour t $\left[\frac{€}{t_{CO_2}}\right]$ $\eta(g,n)$:= efficiency of generator type g at node n [/] $OVC(g,n)$:= Other variable costs of generator type g at node n $\left[\frac{\text{EUR}}{\text{MWh}_{\text{electric}}}\right]$ $l(t)$:= Length of time interval t (fixed at 1 h)[h]	With $d(t)$:= Demand during hour t [MW] $P_{min}(g.n)$:= *Minimum load* of generator type g at node n [MW] $Cap(g,n)$:= *A*vailable capacity of generator type g at node n [MW] $Cap(n,m)$:= *A*vailable capacity of transmission line from node n to node m [MW]			

APPENDIX C.

TABLE C1 Relevant model properties of the design optimization model. All equations are described in Reference 58, to which the numbering refers

Target function	Restrictions (Extract)	Spatial resolution	Temporal resolution
Minimize annualized energy cost including discounted investment in efficiency measures, new supply technologies and energy purchase costs	1. Cover heat, electricity and hot water demand 2. Retain energy balances 3. Limit energy conversion by efficiency 4. State of charge of storage technologies between time steps 5. Charging and discharging limitations of the storage 6. Retain capacity limits of all technologies	Single building as a single node model in a single location. Different buildings relate to different geospatial locations and weather conditions, as described in Reference 56.	8760 hours in a single representative year. The resolution varies with two levels with first an aggregation to 12 typical days and the full temporal resolution on the second level, as described in detail in Reference 59.