Data-Driven Copy-Paste Imputation for Energy Time Series

Moritz Weber, Marian Turowski, Hüseyin K. Çakmak, Ralf Mikut, Uwe Kühnapfel, *Member, IEEE*, and Veit Hagenmeyer, *Member, IEEE*

***Abstract*—A cornerstone of the worldwide transition to smart grids are smart meters. Smart meters typically collect and provide energy time series that are vital for various applications, such as grid simulations, fault-detection, load forecasting, load analysis, and load management. Unfortunately, these time series are often characterized by missing values that must be handled before the data can be used. A common approach to handle missing values in time series is imputation. However, existing imputation methods are designed for *power* time series and do not take into account the total energy of gaps, resulting in jumps or constant shifts when imputing *energy* time series. In order to overcome these issues, the present paper introduces the new Copy-Paste Imputation (CPI) method for *energy* time series. The *CPI* method copies data blocks with similar characteristics and pastes them into gaps of the time series while preserving the total energy of each gap. The new method is evaluated on a real-world dataset that contains six shares of artificially inserted missing values between 1 and 30%. It outperforms the three benchmark imputation methods selected for comparison. The comparison furthermore shows that the *CPI* method uses matching patterns and preserves the total energy of each gap while requiring only a moderate run-time.**

***Index Terms*—Time series imputation, energy time series, missing values.**

I. Introduction and State of the Art

In the course of the worldwide transition to an energy system mainly based on renewable energy sources, a key is the implementation of smart grids [1]. Smart meters are a cornerstone of these smart grids and are thus installed in an increasing number worldwide. They record and transmit a variety of data such as voltage, reactive power, or the electricity consumption of consumers [2]. The collected data is an essential input to various applications supporting and enabling the transition to energy systems from renewable energies. For example, the collected data enables grid operators to perform grid simulations [3] for stability analysis, grid development, fault-detection, and efficiency improvements. The collected data is also needed for load forecasting [4], load analysis, and load management [5]. Moreover, the collected data allows research facilities to develop technologies for the grid of the future.

The results of these applications highly depend on the quality of the input data. The data quality, in turn, is highly influenced by two key challenges in the smart grid infrastructure: the *accuracy* of data acquisition and the *reliability* of data transmission and data storage [6]. The accuracy of data acquisition refers to the correctness of the recorded data. It is reduced by problems causing, for example, noise and outliers in the data [7], [8]. For further processing, outliers in particular are often detected and labeled as missing values as a first step [2], [9]. The reliability of data transmission and data storage, however, mainly relates to the completeness of the recorded data. In implemented smart meter systems, recorded data contain between 3 and 4% of missing values, for example due to planned outages [6], [10]. Due to these two key challenges of smart grids, missing values in recorded data are a common problem. Although some applications are able to handle incomplete data [11], most applications require that the missing values are handled by pre-processing the data.

A common method to handle missing data is *imputation*. Imputation replaces missing values with values that should resemble the actual data [12]. Since missing values are a common problem in real-world datasets, many imputation methods exist for time series: They range from very basic methods such as linear interpolation and Last Observation Carried Forward (LOCF) [12] over time series analysis-based methods [9], [13] to learning-based methods [14], [15].

To further improve the imputation, it is a common approach to focus on time series from a particular domain and to consider their characteristics as additional information. In the context of smart meters, the recorded time series of electricity consumption or generation are typically influenced by factors such as weather, human routines, social norms (e.g., weekends or holidays) and many others [16], [17]. These factors often lead to the commonly known characteristic patterns with daily, weekly, and yearly periodicities, which can be utilized by imputation methods. For example, daily and weekly patterns are exploited in [18]. The pattern frequency of a power time series is estimated with the auto-correlation function and the mean values of the estimated pattern frequency are used to impute missing values. In another work [19], the similarity between days is used by filling larger gaps in a power time

Manuscript received December 23, 2020; revised April 16, 2021 and June 15, 2021 ; accepted July 28, 2021. Date of publication August 2, 2021; date of current version October 21, 2021. This work was supported by the Helmholtz Association under the Joint Initiative "Energy System 2050—A Contribution of the Research Field Energy." Paper no. TSG-01899-2020. *(Moritz Weber and Marian Turowski contributed equally to this work.) (Corresponding author: Moritz Weber.)*

The authors are with the Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, 76344 Eggenstein-Leopoldshafen, Germany (e-mail: moritz.weber@kit.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSG.2021.3101831.

Digital Object Identifier 10.1109/TSG.2021.3101831

series with the average values of validated reference days. Very short gaps with a length of two hours or less are imputed with a linear interpolation, as this often fits the very short-term characteristics of smart meter time series. The *Optimally Weighted Average* approach in [16] utilizes daily and weekly patterns as well as seasonality to select appropriate historical values. With these values, the historical averages of a power time series are calculated, before they are combined with a linear interpolation for smooth transitions between actual and imputed values.

Other methods utilize even more additional data or information to impute missing values in smart meter time series. An example is the method for imputation, de-noising, and outlier removal based on *Principal Component Pursuit* in [20]. It utilizes the spatial correlations in the power load profiles of adjacent substations. In [21], the time series measured by smart meters in a factory are used to impute missing values in other time series from smart meters located in the same factory with clustering and k-nearest neighbors. In [22], the imputation of substation data is formulated as a forecasting problem. The forecast uses the collected power data of nearby substations as well as weather data, which often has an impact on power consumption and generation.

While all of these imputation approaches are specifically designed for smart meter time series, all of them except [21] are limited to the imputation of *power* time series and none of the approaches utilizes the inherent properties of *energy* time series. In a *power* time series P, every entry p_t contains the average power consumption or generation between two time steps $t-1$ and t. However, smart meters typically provide *energy* time series by default. In an *energy* time series E, every entry e_t contains the meter reading, i.e., the energy that has been consumed or generated up to time step t. Therefore, – unlike in power time series – if, for example, entries between the entries e_t to e_{t+3} are missing in an *energy* time series, the next existing entry e_{t+4} still contains the information about the total energy, which was consumed or produced between $t-1$ and $t+4$. As a consequence, a *power* time series can be derived from an *energy* time series with missing values but not vice versa.

Thus, in the present paper, we propose the novel Copy-Paste Imputation (CPI) method for univariate *energy* time series. It uses an energy time series as input and copies blocks of data with similar characteristics into gaps. By copying blocks of matching data, the inherent patterns of the time series are preserved, even in time series with pattern changes. For this purpose, the *CPI* method utilizes the information about the total energy of each gap that energy time series contain in contrast to power time series. It can, therefore, guarantee that the total recorded energy remains unchanged during the imputation. To the best of our knowledge, no other method in literature has so far used this property of energy time series for imputation. By this imputation, the *CPI* method achieves complete energy time series, which also allows deriving a complete power time series. Both energy and power time series can then serve as input for all of the aforementioned applications that support and enable the transition to energy systems from renewable energies.

The remainder of the present paper is structured as follows. The proposed method is explained in detail in Section II and evaluated against three benchmark methods on a real-world dataset in Section III. Concluding remarks and an outlook are given in Section IV.

II. Novel Copy-Paste Imputation Method

In this section, the newly proposed Copy-Paste Imputation (CPI) method[1] is described. As illustrated in the exemplary application in Figure 1, the *CPI* method uses an energy time series with gaps, i.e., one or multiple consecutive missing values, as input and imputes the missing values by filling them with the best matching days of the same time series. In the following, we describe each step of the *CPI* method in detail and reference the corresponding lines in Algorithm 1 and the corresponding part in Figure 1.

A. Linear Interpolation of Single Missing Values

In the first step of the *CPI* method, single missing values, i.e., individual meter readings, are imputed in the given energy time series (line 1). For this imputation, a linear interpolation is used because it provides sufficiently correct estimates for individual missing values. We consider only single missing values in this step to limit the number of consecutively linearly interpolated values and thus potentially unrealistic imputations, while still benefiting from these easily imputable values. Indeed, the resulting imputed values are considered as correct in the subsequent steps to increase the number of days without missing values that are available for copying.

B. Energy Consumption Estimation

The second step of the *CPI* method is the energy consumption estimation for days with gaps (lines 2 - 6). The total energy consumption[2] during gaps can be determined because the *CPI* method uses an energy time series as input. In energy time series, the first entry after a gap still contains the information about the total energy consumed during the gap (see Figure 1(a)). For this reason, to obtain the total energy consumption E_i of the gap i from time step t to time step $t+k$, we calculate the energy difference, i.e.,

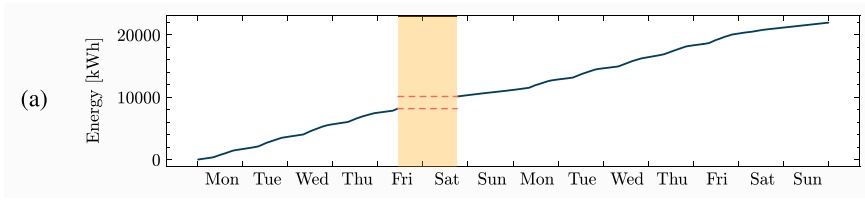
$$E_i = e_{t+k+1} - e_{t-1}, \tag{1}$$

where e_{t+k+1} and e_{t-1} are the energy consumption at the time steps $t+k+1$ and $t-1$ respectively. This equation is the basis of the estimation of the missing energy in line 5 of the *CPI* algorithm.

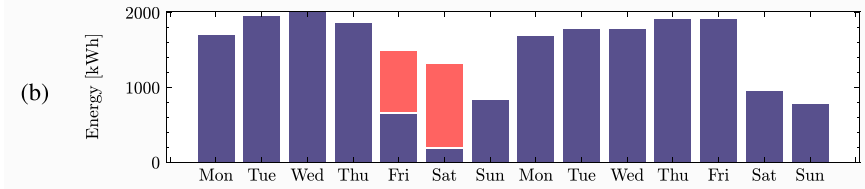
However, for gaps longer than one day, the calculated energy consumption must be allocated to the respective days appropriately. For this purpose, firstly, the calculated energy consumption of the gap is distributed to the respective days according to their share of missing values. Figure 1(b) illustrates this distribution of the calculated energy consumption

[1] A *Python* implementation of the *CPI* method is available on https://github.com/KIT-IAI/CopyPasteImputation.

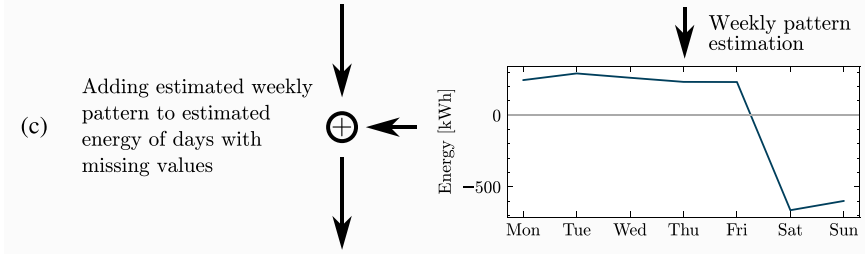
[2] In the following, we refer to consumption data only, but the same principles apply to generation data.



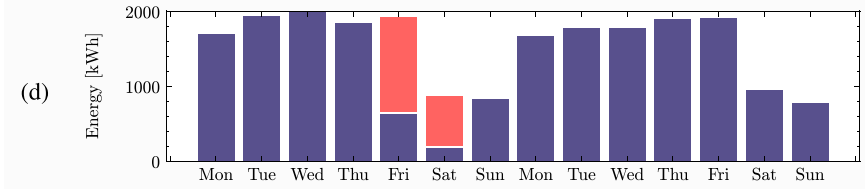
Initial energy consumption time series with a gap, i.e. missing values on the first Friday and Saturday, in yellow. The dashed lines indicate the energy difference between the last known value before and first known value after the gap.



Energy consumption per day with an estimated consumption in red for the days with missing values proportional to their share of missing values.



Estimated weekly pattern of the daily energy consumption that models the weekly pattern such that the values of all weekdays add up to zero. The estimated weekly pattern is added to the estimated energy consumption per day in (b), resulting in a new estimated energy consumption per day in (d).



Energy consumption per day with an estimated consumption according to the weekly pattern in red for the days with missing data.

(e)

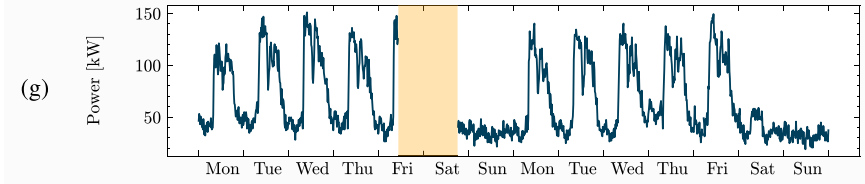
total energy	1703	1946	2012	1854	1924	876	834	1679	1779	1777	1908	1911	952	781
weekday	1	2	3	4	5	6	7	1	2	3	4	5	6	7
seasonal position	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Characteristics of all shown days.

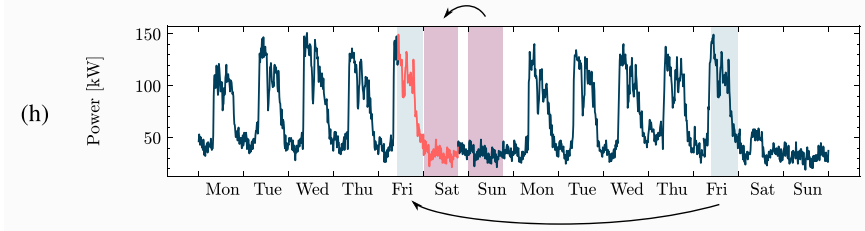
(f)

D_e	0.179	0.018	0.071	0.057	-	-	0.885	0.199	0.117	0.119	0.013	0.011	0.79	0.929
D_w	0.5	0.5	0.5	0.5	-	-	1.0	0.5	0.5	0.5	0.5	0.0	1.0	1.0
D_s	0.022	0.017	0.011	0.006	-	-	0.011	0.017	0.022	0.028	0.033	0.039	0.044	0.049
D	4.2	0.945	1.982	1.66	-	-	18.75	4.56	2.96	3.02	0.93	0.4	17.01	19.82

Resulting distances between the first day with missing values d_4, i.e. the first Friday, and the other days. The dissimilarity calculation is parameterized with $w = (20, 1, 5)$.



The power time series derived from (a) showing typical patterns of a load curve.



Summary of the *CPI* method's procedure that imputes the missing values on Friday and Saturday. As indicated in (f), the second Friday with $D = 0.4$ is selected to fill the first gap.

Fig. 1. Exemplary application of the novel Copy-Paste Imputation (CPI) method to two weeks of a typical real-world energy consumption time series with missing values.

between Friday and Saturday for the given example. Secondly, we consider a weekly pattern in the daily energy consumption. For this pattern, we use the weekly pattern of the input energy time series estimated by the *Prophet* method [11]. It models the weekly pattern such that the values of all weekdays add up to zero. If some of these values are positive, others need

Algorithm 1: Copy-Paste Imputation (CPI)

Input: energy time series *ets* with missing values (i.e. NaNs)
Result: *ets* without missing values, optionally power time series *pts*

```
1  ets ← single_value_linear_interpolation(ets)
2  energy_per_day ← calculate_energy_per_day(ets)
3  non_complete_days ← determine_days_with_missing_values(ets)
   // each entry in this list describes whether a day of ets has missing values
4  weekly_pattern ← estimate_weekly_pattern_with_prophet(energy_per_day, non_complete_days)
   // only consider the daily energy consumption of the days without missing values
5  missing_energy_per_day ← estimate_missing_energy_per_day(ets, weekly_pattern)
6  estimated_energy_per_day ← energy_per_day + missing_energy_per_day
7  complete_days ← compile_list_of_complete_days(ets.time, energy_per_day, non_complete_days)
8  pts ← derive_power_ts_from_energy_ts(ets)
9  foreach day with missing values do
10 |  best_matching_day ← find_day_with_min_dissimilarity(day, complete_days)
11 |  pts[day] ← pts[best_matching_day]
12 end
13 foreach gap in ets do
14 |  scaling_factor ← actual_energy_of_gap / imputed_energy_of_gap
15 |  pts[gap] ← pts[gap] · scaling_factor
16 end
17 ets ← calculate_energy_ts_from_power_ts(pts)
```

to be negative. For the estimation of these values, the *Prophet* method only considers the daily energy consumption of the days without missing values, i.e., one value per day. For the given example, Figure 1(c) visualizes the weekly pattern. Lastly, the estimated weekly pattern is added to all days of the gap as shown in Figure 1(d). When adding the weekly pattern, the added energy is summed up. The sum of the added energy is then divided by the number of days in the gap to obtain the average. This average is subtracted from each day of the gap to preserve the total energy consumption of the gap.

C. Compilation of Available Complete Days

In the third step of the *CPI* method, a list of the available complete days (i.e., days without missing values) is compiled (line 7). Assuming daily patterns, a weekly cycle, and a yearly seasonality in the energy consumption, each day is listed with its following characteristics: its total energy consumption (d_e), its weekday ($d_w \in \{1 \dots 7\}$), and its seasonal position (d_s). Under the assumption of a yearly seasonality, i.e., 365 days or 366 days for leap years, it follows that d_s is in $\{1 \dots 366\}$. An example of a list with days and their characteristics is shown in Figure 1(e).

D. Calculation of Dissimilarity Between Days

In the fourth step, the *CPI* method calculates a dissimilarity criterion between each day with gaps and all complete days (line 10), which is used to select the best matching days for filling gaps in the next step. For the dissimilarity criterion, the *CPI* method uses the three previously introduced characteristics of days: total energy, weekday, and seasonal position. Since these characteristics are already computed for all complete days, they only have to be determined for the days with missing values in this step. More specifically, three distance measures, i.e., D_e, D_w, and D_s, are calculated for each day with gaps d_i and each available complete day d_j.

The first distance measure D_e describes the distance between the total energy consumption of a day with gaps d_i and a complete day d_j. The total energy consumption can serve as a distance measure because the *CPI* method uses an energy time series as input and thus can calculate the energy consumed during a gap. D_e is defined as

$$D_e(d_i, d_j) = \frac{|d_{i,e} - d_{j,e}|}{e_{max} - e_{min}}, \tag{2}$$

where e_{max} and e_{min} are the maximum and minimum energy consumption of a day in the time series and $d_{i,e}$ and $d_{j,e}$ are the total energy consumption of the days d_i and d_j. For the day with gaps d_i, the previously estimated energy consumption is used. Dividing by the difference between e_{max} and e_{min} ensures that the distance measure D_e is in [0, 1].

The second distance measure D_w is based on the assumption of a weekly pattern in the time series and describes the distance between the weekday of a day with gaps d_i and a complete day d_j. It is defined as

$$D_w(d_i, d_j) = \begin{cases} 0.0, & \text{if } d_{i,w} = d_{j,w} \\ 0.5, & \text{if } d_{i,w} \in \{1..5\} \wedge d_{j,w} \in \{1..5\} \\ & \quad \vee\, d_{i,w} \in \{6, 7\} \wedge d_{j,w} \in \{6, 7\} \\ 1.0, & \text{else,} \end{cases} \tag{3}$$

where $d_{i,w}$ and $d_{j,w}$ are integer representations for the weekday of days d_i and d_j. One to five represent the workdays Monday to Friday, whereas 6 and 7 represent the weekend

days Saturday and Sunday. This distance measure D_w assigns smaller distances to days of the same weekday or days of the same class (i.e., workday or weekend) and higher distances to days of different classes.

The third distance measure D_s captures the underlying seasonal patterns and describes the distance between the seasonal position of a day with gaps d_i and a complete day d_j. It is defined as

$$D_s(d_i, d_j) = \begin{cases} \frac{|d_{i,s} - d_{j,s}|}{\lfloor \frac{\mathbf{s}}{2} \rfloor}, & \text{if } |d_{i,y} - d_{j,y}| \leq \lfloor \frac{\mathbf{s}}{2} \rfloor \\ \frac{\mathbf{s} - |d_{i,s} - d_{j,s}|}{\lfloor \frac{\mathbf{s}}{2} \rfloor}, & \text{else,} \end{cases} \tag{4}$$

where $\mathbf{s}$ is the length of the seasonal cycle and $d_{i,s}$ and $d_{j,s}$ are the position of days d_i and d_j in this cycle. For a yearly seasonality, $\mathbf{s}$ can be set to 365 or 366 to reflect the number of days in a year. This distance measure ensures that two days from the same season are considered as more similar than two days from different seasons. For example, January 1 and December 31 of the same year are almost one year apart but have a minimal distance D_s. In contrast, January 1 and July 1 are only half a year apart and have a maximal distance D_s.

In order to determine the dissimilarity between a day with gaps and a complete day, the three individual distance measures are combined into a single criterion. The resulting dissimilarity criterion D is the weighted sum of the three individual distance measures D_e, D_w, and D_s. It is defined as

$$D = w_e D_e + w_w D_w + w_s D_s, \tag{5}$$

where w_e, w_w, and w_s are the weights and D_e, D_w, and D_s are the normalized distance measures. The individual distance measures are normalized to the interval [0,1] for an easier interpretation of these weights. The specification of these weights is necessary once before applying the *CPI* method. To find suitable weights, one possible approach is to conduct a grid search on a representative set of time series (see Section III-C3 for an exemplary grid search). For the given example, Figure 1(f) shows the individual distances between the first Friday as a day with gaps and all the other days and the resulting dissimilarity values D for given weights.

E. Copy and Paste of Matching Days

In the last step, the *CPI* method copies the best matching days, pastes them into gaps (line 11), and scales the imputed values to preserve the energy of the respective gaps (lines 13 - 16). In order to determine the best matching days, the previously generated list of complete days is used. For a day with gaps d_i, the day d_j with the smallest dissimilarity $D(d_i, d_j)$ is chosen. Since the entire list of complete days is used, days from the future of the day with gaps are also considered. In the given example in Figure 1(f), the most similar day is the second Friday of the time series because that Friday has the lowest dissimilarity value.

Based on the determined best matching days, the actual copying and pasting of the best matching days into gaps is done. For this purpose, the power time series, as shown in Figure 1(g), serves as basis. It can be derived from the input energy time series (line 8) by calculating the average power p_t between time steps $t-1$ and t, i.e.,

$$p_t = \frac{e_t - e_{t-1}}{\Delta t}, \tag{6}$$

where Δt is the time between two time steps, e_t and p_t are the energy and power at time step t, and e_{t-1} is the energy at time step $t-1$. In the derived power time series, every missing value in each day with gaps is replaced by the corresponding value of the previously determined best matching complete day (see Figure 1(h)).

Finally, the imputed power values are scaled in order to preserve the actual energy of each gap. The scaling is based on the actual energy and the imputed energy. Both can be determined because the *CPI* method uses energy time series as input and thus can calculate the energy consumed during a gap. The actual energy E_i of the gap i is calculated according to Equation (1). The imputed energy E_i' is calculated by accumulating the imputed power values. To preserve the energy, the imputed power values of gap i are multiplied with the ratio of the actual energy and the imputed energy, i.e.,

$$\hat{p}_t = \hat{p}_t' \cdot \frac{E_i}{E_i'}, \tag{7}$$

where $\hat{p}_t'$ is the power value calculated by the *CPI* method and $\hat{p}_t$ is the scaled power value. After this scaling, the imputed power time series can be used to calculate an imputed energy time series by solving Equation (6) for e_t. With the calculated energy time series, the *CPI* method finally returns a complete energy time series whose initially missing values are completely imputed.

III. Evaluation

In this section, the proposed *CPI* method is evaluated on real-world data and its performance is compared to benchmark methods. Therefore, the used dataset is introduced followed by the selected benchmark methods. After describing the experimental setting, the results are presented.

A. Dataset

The dataset used for the evaluation is the *ElectricityLoadDiagrams20112014* dataset[3] from the *UCI Machine Learning Repository* [23]. The dataset consists of power time series with complete consumption data from 370 different smart meters over a period of up to four years. The time series contain quarter-hourly average power values in kW, resulting in 35,040 values per year. Of these 370 time series, 50 differently shaped time series with a length of one year are selected as a representative sample. The selected time series vary greatly in terms of seasonal, weekly, and daily patterns as illustrated in Figure 2.

For the evaluation of the *CPI* method, the selected power time series, that do not contain any missing values, are converted to energy time series by accumulating the power values. Due to their completeness, we artificially insert missing values in the used time series by replacing values with NaNs. In order to decide which values are replaced, we follow four steps.

[3]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

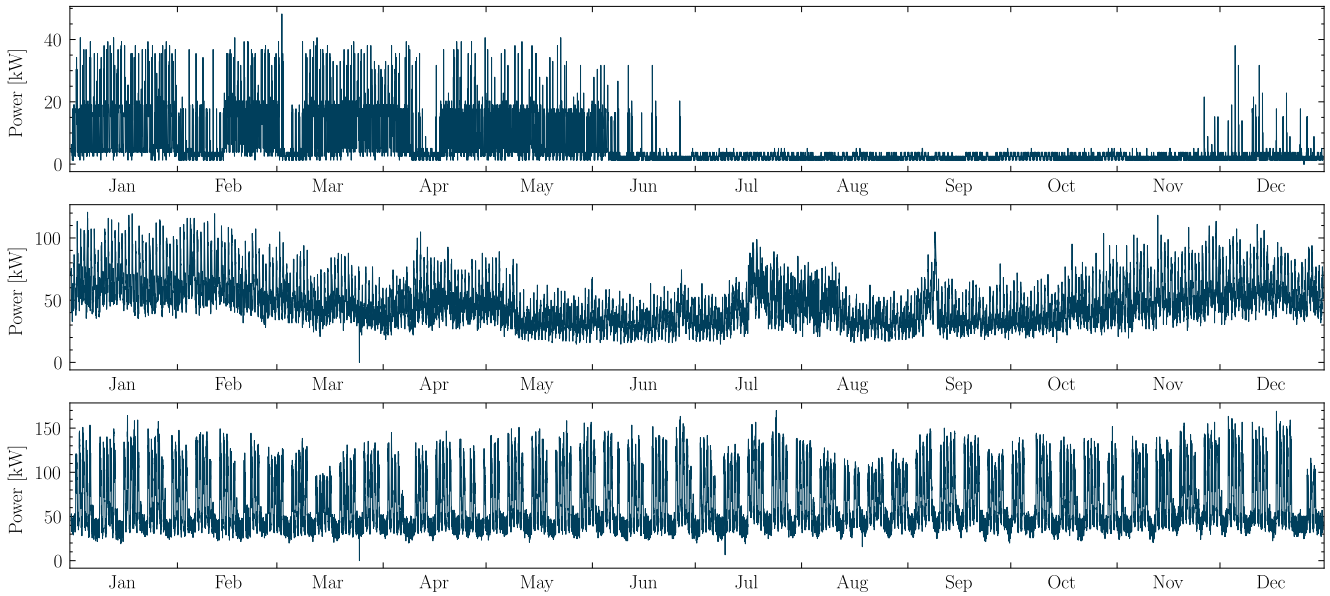


Fig. 2. Three exemplary time series from the *UCI* dataset, containing variations of different seasonal, weekly, and daily patterns.

Firstly, we determine the longest sequence of the time series without missing values T_c. Secondly, we define the number of consecutive values to be replaced in this sequence T_c by choosing uniformly between 2 and the minimum of the specified maximum number of consecutive missing values, the length of T_c, and the remaining number of values to be replaced. Thirdly, we randomly select a starting index for the determined number of consecutive values to be replaced such that all values to be replaced are contained in T_c. Lastly, we replace each selected value with NaN. These four steps are repeated until the total number of values to be replaced is reached.

For the evaluation, we consider the number of values to be replaced in the form of shares of missing values. In the evaluation, six shares of missing values are used between 1% and 30%, i.e., 1, 2, 5, 10, 20, and 30%. In order to consider both larger gaps and single missing values, 5% of each share of missing values are single missing values. The indices for the single missing values are determined randomly after creating the larger gaps.

B. Benchmark Methods

In order to compare the performance of the proposed *CPI* method, we apply benchmark methods to the dataset. As suitable benchmark methods, we generally consider all imputation methods for energy time series that utilize the time series and its characteristics only. Due to the lack of imputation methods for energy time series – to the best knowledge of the authors –, we include imputation methods for power time series and time series in general despite their disadvantage of not utilizing energy data. Methods requiring additional data or information such as weather data [9], [13] or validated reference days [19] and methods designed for multivariate time series only [14], [20], [22] are discarded due to their lack of comparability. Furthermore, during the evaluation, the method in [15] is excluded due to its excessive run-time.

In this context, we select three methods as benchmarks in view of comparison complexity and fairness. We derive these methods from literature [11], [12], [16], [18] and adapt them where necessary. To establish a fair comparison, the evaluated benchmark methods receive their data input in the same way as the *CPI* method. They sequentially get the 50 time series and can use each time series completely but independently from the others.

The first benchmark method is a commonly applied linear interpolation [12], [16]. This method represents a lower baseline and should be outperformed in any case. It imputes missing values $\hat{p}_t$ by linearly interpolating the first and last known power value before and after a gap, i.e.,

$$\hat{p}_t = \frac{t - t_1}{t_2 - t_1} \cdot \left(p_{t_2} - p_{t_1}\right) + p_{t_1}, \tag{8}$$

where t_1 and t_2 are the time steps before and after the gap. The linear interpolation is thus the only evaluated method that uses two values for imputing a gap.

The second benchmark method is the *Optimally Weighted Average* (OWA) [16]. Assuming a weekly pattern, this method calculates a historical average

$$\hat{p}_t^{\mathrm{HA}} = \frac{1}{|H|} \Sigma_{i \in H} p_i, \tag{9}$$

where H contains all values of the hour before and after t as well as of the same two hours of the previous and of the next week. As long as H is empty, the considered weeks are iteratively extended by one in each direction to consider additional values from the same two hours in more weeks. To ensure smooth transitions between actual and imputed values, this average is combined with a linear interpolation $\hat{p}_t^{\mathrm{LI}}$ (8). The combination results in

$$\hat{p}_t = w_t \hat{p}_t^{\mathrm{LI}} + (1 - w_t) \hat{p}_t^{\mathrm{HA}}, \tag{10}$$

where w_t weighs the influence of the two imputation methods. The weight w_t is designed to decrease with increasing distance

to the actual values, i.e.,

$$w_t = e^{-\alpha d_t}, \tag{11}$$

where d_t describes the distance from t to the nearest actual value in time steps and α determines the rate of decay for w_t. Since α has a negligible influence on the imputation results in the present evaluation, we use a global $\alpha = 0.1387$ for the evaluation as determined in [16].

The third benchmark method is based on the *Prophet* method for time series forecasting [11]. *Prophet* uses a modular regression model that can be described as

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t, \tag{12}$$

where g is a model for the trend, s for seasonality, h for holidays, and ε_t for changes that are not represented in the model. The imputation method based on this model exploits *Prophet*'s capability to estimate a time series model on irregularly spaced data [11] and imputes missing values with the corresponding values of the model. The model is learned on all values available in the time series to be imputed. In contrast to its application in the *CPI* method, the benchmark imputation method based on *Prophet* receives, like all other benchmark methods, the aforementioned quarter-hourly values as input, i.e., 96 values per day.

C. Experimental Setting

This subsection describes the used hard- and software platform, introduces the error measures to evaluate the imputation methods, and explains the calculation of the weights in the dissimilarity measure of the evaluated *CPI* method.

1) Hard- and Software Platform: For the evaluation of the *CPI* and the benchmark methods, we compare the quality of the imputation and the required run-time. For a better comparability of the results, all methods are implemented in *Python* and evaluated on the same hardware. The evaluation hardware is a desktop PC running *Ubuntu 20.04* with an *AMD Ryzen 5 3600* processor and 16GB of memory.

2) Error Measures: In order to evaluate the quality of an imputation in energy time series, we examine both the usage of matching patterns to fill gaps and the conservation of the total energy in the gaps. To evaluate the usage of matching patterns, we determine how well imputed patterns match the actual patterns. For this purpose, we measure the deviation between every single actual power value and the corresponding imputed power value using the Mean Absolute Percentage Error (MAPE). It is defined as

$$\mathrm{MAPE_p} = \frac{1}{|T_m|} \Sigma_{t \in T_m} \left| \frac{\hat{p}_t - p_t}{p_t} \right|, \tag{13}$$

where p_t and $\hat{p}_t$ are the actual and imputed power values at time step t and T_m is the set of time steps with missing values. To evaluate the conservation of the total energy in gaps, we measure the difference between the actual and imputed energy while ignoring the fine granular patterns that are used for the imputation. The difference is determined using the Weighted Absolute Percentage Error (WAPE), which is defined as

$$\mathrm{WAPE_E} = \frac{\Sigma_{i=1}^{N} \left| \hat{E}_i - E_i \right|}{\Sigma_{i=1}^{N} E_i}, \tag{14}$$

where E_i and $\hat{E}_i$ are the actual and imputed energy of gap i in a time series with N gaps. In contrast to the $\mathrm{MAPE_p}$ (13), the weighting of the individual absolute errors is necessary in the $\mathrm{WAPE_E}$ (14) to account for gaps of different sizes.

3) Weights in the CPI Dissimilarity Measure: Before copying and pasting matching days, the *CPI* method calculates the dissimilarity measure between two days (5). For this calculation, it is necessary that the weights of the three distance measures regarding total energy consumption, weekday, and seasonal position are specified beforehand. To determine these weights for the *CPI* method applied in the evaluation, we thus perform a grid search.

This grid search is conducted before the actual evaluation on a separate dataset that is used only for calibration. To compile the calibration dataset, we consider the remaining 320 time series given in the aforementioned dataset. Based on a visual inspection, we choose five time series for the calibration dataset, which are different from each other but have similar characteristics to the 50 time series in the evaluation. Each of the five time series from the calibration dataset is evaluated with six different shares of missing values, ranging from 1% to 30%, which results in 30 time series in total.

Using this calibration dataset, we test 1,000 combinations of the three weights $w = (w_e, w_w, w_s)$ with each weight in the range $[1 \ldots 10]$. Each combination of weights is evaluated with both error measures $\mathrm{MAPE_p}$ (13) and $\mathrm{WAPE_E}$ (14) on all time series of the calibration dataset. Based on the results, we determine the overall minimum and maximum of both error measures. With them, we min-max normalize the values of both error measures. The min-max normalized $\mathrm{MAPE_p}$ is calculated with

$$\mathrm{MAPE_{p,n}}(w, i) = \frac{\mathrm{MAPE_p}(w, i) - \min \mathrm{MAPE_p}}{\max \mathrm{MAPE_p} - \min \mathrm{MAPE_p}}, \tag{15}$$

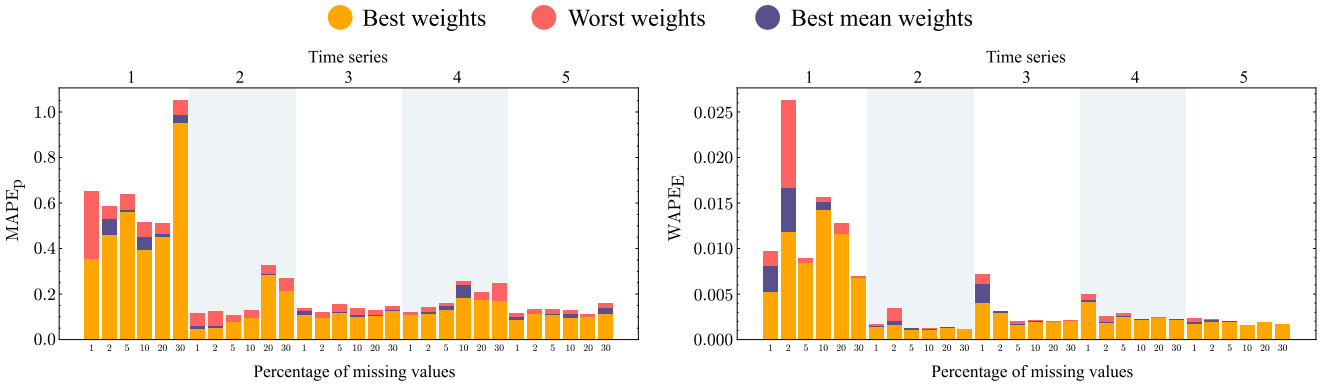
where w is the tested weight combination and i is the identifier of the time series from the calibration dataset. The min-max normalized $\mathrm{WAPE_E}$ is determined analogously to (15). To obtain the average sum of both normalized error measures $\overline{TE}$, we first add the normalized results of both error measures together for each time series from the calibration dataset. Afterward, we sum this total over all time series and divide it by the number of time series, i.e.,

$$\overline{TE}(w) = \frac{\sum_i^n \mathrm{MAPE_{p,n}}(w, i) + \mathrm{WAPE_{E,n}}(w, i)}{n}, \tag{16}$$

where n is the number of time series in the calibration dataset. Finally, we determine the weights with the minimum average sum of both normalized error measures, i.e.,

$$w_{opt} = \arg\min_w \overline{TE}(w), \tag{17}$$

representing the optimal weights w_{opt}.



(a) MAPE_p (13) (b) WAPE_E (14)

Fig. 3. The two error measures of the best, worst, and overall best weights for the five time series from the calibration dataset used in the grid search. Every time series is evaluated with six different shares of missing values ranging from 1% to 30%, resulting in a total number of 30 tested time series. The orange bars indicate the errors of the best weights, the red bars the errors of the worst weights, and the purple bars the errors resulting from the determined optimal weight combination w_{opt}.

D. Results

Based on the 50 selected time series, the selected benchmark methods, and the experimental setup described above, this section explains the evaluation results. It first covers the grid search, before describing the usage of matching patterns and the conservation of energy, quantified by MAPE_p (13) and WAPE_E (14) respectively. The presented values of these error measures are the truncated means for the 50 evaluated time series, which omit the two best and worst values to obtain less outlier-sensitive results. Afterward, the run-time of the evaluated methods and its decomposition is addressed before the usage of matching patterns and the run-time are put in comparison. Lastly, the output of an exemplary imputation visually illustrates the evaluation results.

1) Grid Search: The grid search evaluates the aforementioned 1,000 weight combinations in 91 minutes with a total system memory usage of approximately 3.5GB. Figure 3 shows the MAPE_p (13) and WAPE_E (14) for the tested time series. For each time series and each share of missing values, an orange bar depicts the result with the best weights. Accordingly, the red bars show the worst results. The dark purple bars describe the results with the overall best weights $w_{opt} = (5, 1, 10)$ determined with Equation (17). The grid search reveals that every time series has its own optimal weight combination. Nevertheless, the difference between the results with the best and the worst weights is often very small. Similarly, the difference between the results with the best weights and the overall best weights w_{opt} is often negligible. w_{opt} is, therefore, used for all 50 time series to be evaluated.

2) Usage of Matching Patterns: The usage of matching patterns – as defined in Equation (13) – by the evaluated methods is presented in Figure 4. For the six different shares of artificially inserted missing values, the figure shows the MAPE_p (13) of all evaluated methods. For most of the shares of missing values, the *CPI* method performs better than the OWA method as the best benchmark method. Both methods perform overall about 10 - 12% better than the *Prophet*-based method. The linear interpolation performs by far the worst

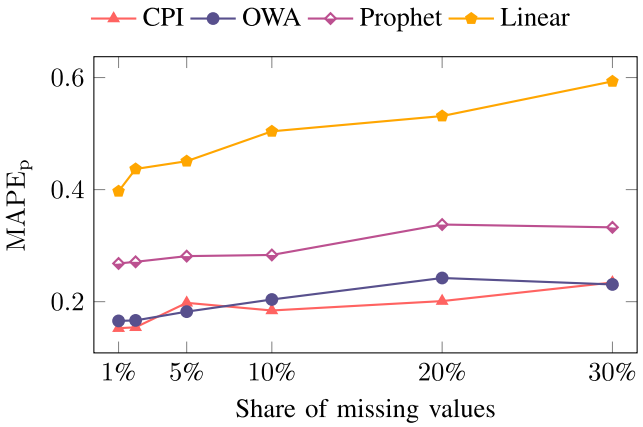


Fig. 4. The MAPE_p (13) of the *CPI* method and the three benchmark methods with different shares of missing values. As the scaling of imputed values does not noticeably affect the results of the *CPI* method, it is omitted in this figure.

for all shares of missing values. All methods tend to higher errors with higher shares of missing values. This trend is most distinct for the linear interpolation. With regard to the errors of individual time series, the benchmark methods are more prone to extreme errors with a maximum MAPE_p (13) of 5.88 and above while the *CPI* method has a maximum MAPE_p (13) of 2.37.

3) Conservation of Energy: For the four evaluated methods, the conservation of energy in the gaps – as defined in Equation (14) – is shown in Figure 5. The figure presents the WAPE_E (14) for the six shares of missing values. The *CPI* method performs best regardless of the share of missing values. To allow a better comparability with the benchmark methods that all do not use scaling, the dashed line indicates the error values for the *CPI* method without scaling. Without scaling, the *CPI* method performs on average 4.4% better than the OWA method, which is the second best method. We assume that this result is strongly related to using energy as a distance measure, which is possible thanks to the use of energy time series as input. The scaling even reduces the error to nearly zero for all shares of missing values, so the

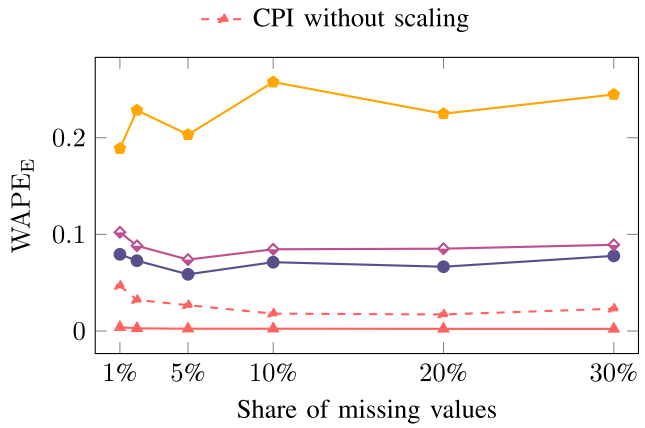


Fig. 5. The WAPE$_E$ (14) of the CPI method and the three benchmark methods with different shares of missing values. For better comparability with the benchmark methods that all do not use scaling, the dashed line indicates the WAPE$_E$ of the *CPI* method without scaling the imputed values to preserve the energy of a gap.

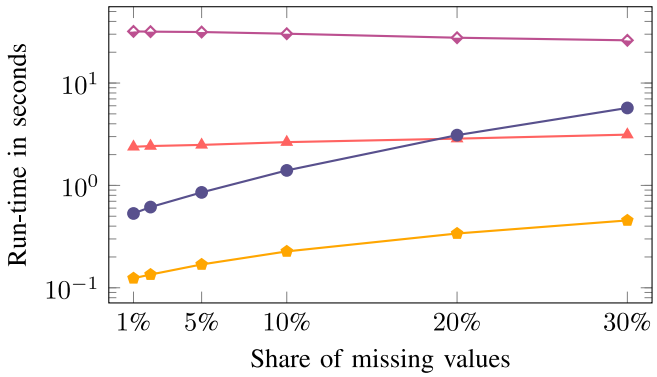


Fig. 6. The average run-times required by the *CPI* method and the three benchmark methods for the imputation of the 50 selected one-year time series. Note the logarithmic time scale, which visually compresses *Prophet*'s run-time decrease by 5.8 seconds from 1 to 30% of missing values.

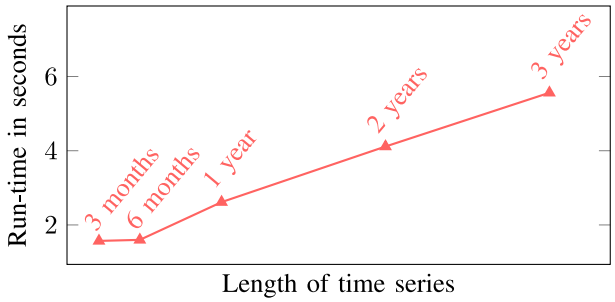


Fig. 7. Run-time of the *CPI* method for time series with different lengths from three months (8,832 values) up to three years (105,120 values).

CPI method performs even better. In view of the already good results without scaling, the contribution of scaling is, however, relatively small. The linear interpolation again performs worst for all shares of missing values. The *Prophet*-based method and the OWA method perform very similarly with an average advantage of 1.6% for the OWA method.

4) Run-Time: Figure 6 shows the average run-times required by the evaluated methods for the imputation of the 50 selected one-year time series with 35,040 values each. The linear interpolation is by far the fastest method. The OWA

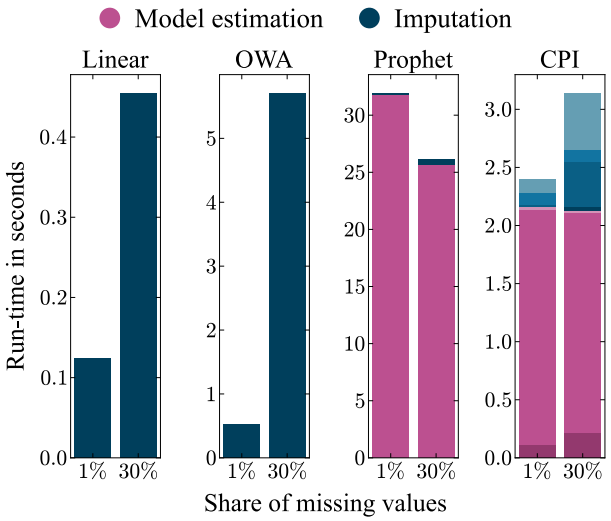


Fig. 8. Run-time decomposition of the *CPI* method and the three benchmark methods into model estimation including training and fitting as well as imputation.

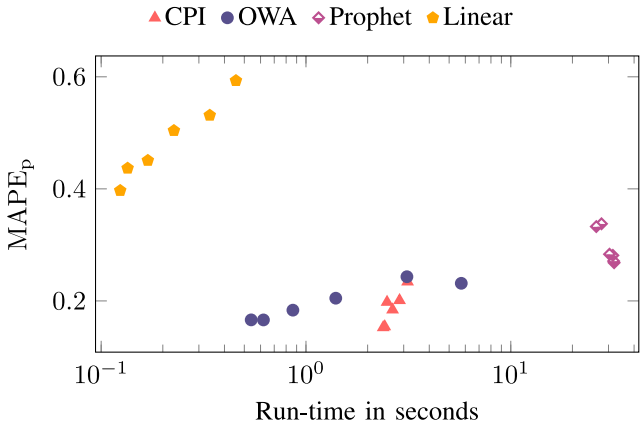


Fig. 9. Comparison of the usage of matching patterns and the average run-time needed of the *CPI* method and the three benchmark methods for the imputation of 50 one-year time series. The x-axis shows the required average run-times on a logarithmic scale and the y-axis the MAPE$_p$ (13).

method is similarly fast for small shares of missing values but increases more drastically in run-time than the other methods for increasing shares of missing values. The *CPI* method requires about 10 to 20 times more time than the linear interpolation but is faster than the OWA method for 20 and 30% of missing values. The *Prophet*-based method requires much more time than all other methods and is 9 to 10 times slower than the *CPI* method. Its longer run-time compared to the *CPI* method, wherein the *Prophet* method is also used, is caused by a more time-consuming training due to its larger input. When applied as benchmark method, the *Prophet* method receives 96 values per day as input as opposed to only one value per day when used in the *CPI* method.

In addition to the run-time evaluation for the one-year time series, we briefly evaluate how the *CPI* method's run-time relates to the number of input values. Figure 7 shows the run-time of the *CPI* method for time series with different lengths from one quarter of a year (8,832 values) up to three years (105,120 values). The *CPI* method scales approximately

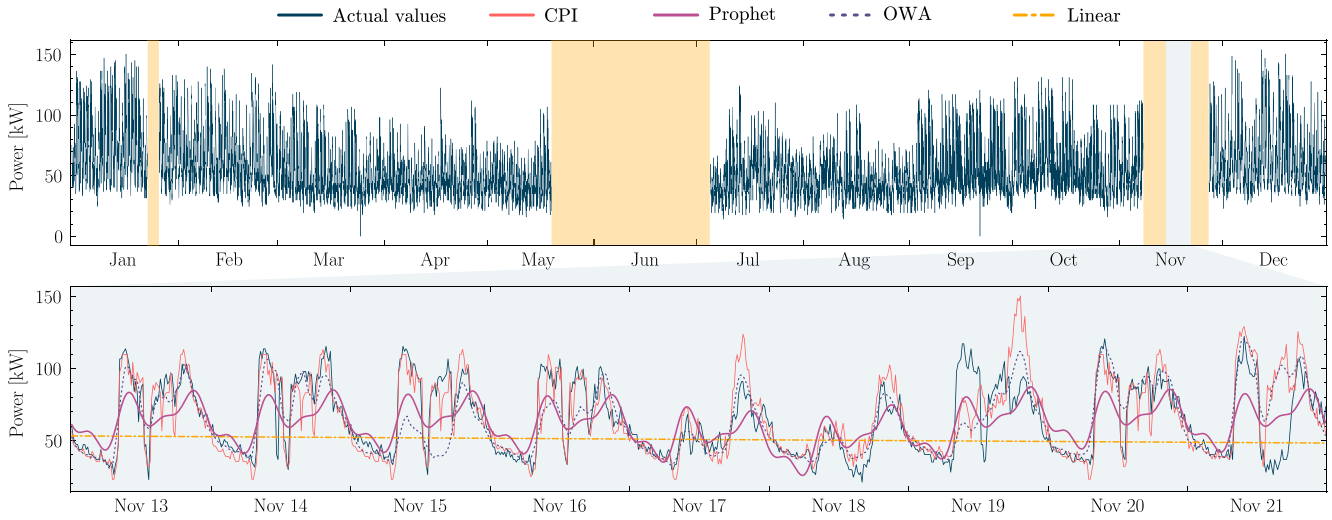


Fig. 10. The upper figure shows an exemplary one-year time series with 20% of missing values. For a multi-day excerpt of a gap in November, the lower figure presents the resulting imputations by the *CPI* method and the three benchmark methods in comparison to the actual values. Metrics of the *CPI* method and the best benchmark method for this example: MAPE_p (13): 0.194 (OWA: 0.211), WAPE_E (14): 0.003 (OWA: 0.065), run-time: 2.89s (Linear: 0.35s).

linearly to the number of input values with an average run-time of 5.56 seconds for time series with 105,120 values.

5) Run-Time Decomposition: Figure 8 decomposes the run-time of the evaluated methods for 1% and 30% of missing values. Model estimation including training and fitting are depicted in purple whereas the actual imputation is depicted in blue. For the *CPI* method, the run-time is even further differentiated with respect to its steps. From bottom to top, the purple colors respectively describe the steps: linear interpolation of single missing values, the energy consumption estimation, and the compilation of available complete days. The blue colors respectively indicate the steps of matching of the most similar days, pasting the values into the gaps, scaling the imputed values, and calculating the completed energy time series. Figure 8 shows that the energy consumption estimation for gaps, that relies on the *Prophet* method, is the dominating part of the *CPI* method's run-time. Similarly, the model estimation also dominates the run-time of the *Prophet*-based method. In contrast, the linear interpolation and the OWA method do not comprise any model estimation. Their run-time thus consists entirely of the imputation itself.

6) Usage of Matching Patterns vs. Run-Time: In Figure 9, the obtained results regarding the usage of matching patterns is put in relation to the run-time needed with a scatter plot showing the required average run-times on the x-axis and the MAPE_p (13) on the y-axis. Smaller values indicate a better performance. While the linear interpolation provides fast and inaccurate results, the *CPI* method and the OWA method deliver the most accurate results with a reasonable run-time. The *Prophet*-based method yields mediocre results while taking much longer to calculate than the other methods.

7) Exemplary Imputation Results: For all evaluated imputation methods, Figure 10 illustrates an exemplary imputation of a time series with 20% of artificially inserted missing values, resulting in large gaps. The imputation of the linear interpolation fails to capture the patterns of the time series. The imputations by the OWA method and the *Prophet*-based method capture the essential patterns but lack details. The imputation by the *CPI* method mostly fits the actual values but it shifts and increases some peaks. Despite not explicitly addressing the transitions between existing and imputed values, the *CPI* method also generally provides smooth transitions on both ends of gaps. Compared to the three benchmark methods, the imputation by the novel *CPI* method thus comes closest to the actual values.

IV. Conclusion and Outlook

The present paper introduces a new *Copy-Paste Imputation* method for energy time series. Using an energy time series as input, it copies blocks of data with similar characteristics and pastes them into gaps of the time series. This approach enables realistic imputations even for large gaps with several weeks of consecutively missing values. In contrast to all other methods in the literature – to the best knowledge of the authors –, the *CPI* method utilizes the often provided *energy* time series, i.e., the actual meter readings, instead of *power* time series, i.e., the average power per interval. Using an energy time series allows including the information on the total energy consumed or produced during gaps. Utilizing this information enables a robust selection of matching blocks of data and ensures that the overall energy per gap remains unchanged while imputing the missing values with realistic patterns. Through imputing missing values, the *CPI* methods increases the completeness of collected energy time series and of the derivable power time series, which are then available to applications relying on complete input data. For the imputation itself, the *CPI* method does not require any additional information such as weather data or consumption data from spatially close smart meters.

The proposed *CPI* method is applied to a real-world dataset and compared to three benchmark methods. For the evaluation, six shares of artificially inserted missing values between 1 and 30% are used. For all shares of missing values, the *CPI* method outperforms the benchmark methods. The evaluation

confirms that the *CPI* method uses matching patterns for the imputations and that it conserves the overall energy of every imputed gap. In comparison to the benchmark methods, the *CPI* method requires only a moderate run-time that scales well with increasing shares of missing values and length of input time series. Overall, the *CPI* method offers a good trade-off between the usage of matching patterns and the run-time.

Based on these results, *future work* could follow three directions. First, the robustness of the *CPI* method could be analyzed and improved regarding aperiodic events or time series with other temporal resolutions and periodicities such as residential solar power generation or fast charging of electrical vehicles. Similarly, time series containing both power consumption and generation from renewable energy sources could be of interest for further investigation. A robustness analysis could further include the selection of the weights in the dissimilarity measure – for example in dynamic environments – and the compilation of the calibration data set used for determining the weights. Furthermore, the transitions between existing and imputed values on both ends of gaps could be further investigated. Second, a trend analysis could enhance the *CPI* method's selection of matching days especially for longer gaps. Similarly, additional information such as voltage magnitude [24] and spatial temporal correlations [25] could be used to improve the matching days selection. Third, the *CPI* method could be integrated in applications that rely on complete input data such as grid simulation, load forecasting, and load management. Anomaly or error detection functions could also be included in the *CPI* method itself to repair implausible values. Moreover, a reporting and analysis tool could use the *CPI* method to estimate the imputation quality based on artificially inserted missing values.

REFERENCES

[1] D. Alahakoon and X. Yu, "Smart electricity meter data intelligence for future energy systems: A survey," *IEEE Trans. Ind. Informat.*, vol. 12, no. 1, pp. 425–436, Feb. 2016.

[2] T. Alquthami *et al.*, "Analytics framework for optimal smart meters data processing," *Elect. Eng.*, vol. 102, no. 3, pp. 1241–1251, Sep. 2020.

[3] V. Hagenmeyer *et al.*, "Information and communication technology in energy lab 2.0: Smart energies system simulation and control center with an open-street-map-based power flow simulation example," *Energy Technol.*, vol. 4, no. 1, pp. 145–162, 2016.

[4] B. Heidrich, M. Turowski, N. Ludwig, R. Mikut, and V. Hagenmeyer, "Forecasting energy time series with profile neural networks," in *Proc. 11th ACM Int. Conf. Future Energy Syst. (e-Energy)*, 2020, pp. 220–230.

[5] Y. Wang, Q. Chen, T. Hong, and C. Kang, "Review of smart meter data analytics: Applications, methodologies, and challenges," *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 3125–3148, May 2019.

[6] C. King and J. Strapp, "Chapter 11—Software infrastructure and the smart grid," in *Smart Grid: Integrating Renewable, Distributed and Efficient Energy*, F. P. Sioshansi, Ed. Amsterdam, The Netherlands: Academic, 2012, pp. 259–288.

[7] W. Chen, K. Zhou, S. Yang, and C. Wu, "Data quality of electricity consumption data in a smart grid environment," *Renew. Sustain. Energy Rev.*, vol. 75, pp. 98–105, Aug. 2017.

[8] L. Wang *et al.*, "Point and contextual anomaly detection in building load profiles of a university campus," in *Proc. IEEE PES Innov. Smart Grid Technol. Europe (ISGT-Europe)*, Oct. 2020, pp. 11–15.

[9] H. N. Akouemo and R. J. Povinelli, "Data improving in time series using ARX and ANN models," *IEEE Trans. Power Syst.*, vol. 32, no. 5, pp. 3352–3359, Sep. 2017.

[10] J. Peppanen, M. J. Reno, M. Thakkar, S. Grijalva, and R. G. Harley, "Leveraging AMI data for distribution system model calibration and situational awareness," *IEEE Trans. Smart Grid*, vol. 6, no. 4, pp. 2050–2059, Jul. 2015.

[11] S. J. Taylor and B. Letham, "Forecasting at scale," *Amer. Stat.*, vol. 72, no. 1, pp. 37–45, Jan. 2018.

[12] S. Moritz and T. Bartz-Beielstein, "imputeTS: Time series missing value imputation in R," *R J.*, vol. 9, no. 1, pp. 207–218, 2017.

[13] H. N. Akouemo and R. J. Povinelli, "Time series outlier detection and imputation," in *Proc. IEEE PES Gen. Meeting Conf. Exposit.*, 2014, pp. 1–5.

[14] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "BRITS: Bidirectional recurrent imputation for time series," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6775–6785.

[15] N. Bokde, M. W. Beck, F. Martínez Álvarez, and K. Kulat, "A novel imputation methodology for time series based on pattern sequence forecasting," *Pattern Recognit. Lett.*, vol. 116, pp. 88–96, Dec. 2018.

[16] J. Peppanen, X. Zhang, S. Grijalva, and M. J. Reno, "Handling bad or missing smart meter data through advanced data imputation," in *Proc. IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf. (ISGT)*, 2016, pp. 1–5.

[17] J. Á. G. Ordiano, S. Waczowicz, V. Hagenmeyer, and R. Mikut, "Energy forecasting tools and services," *WIREs Data Min. Knowl. Disc.*, vol. 8, no. 2, 2018, Art. no. e1235.

[18] M. Friese *et al.*, "UniFIeD univariate frequency-based imputation for time series data," Inst. Inf., Technische Hochschule Köln, Cologne, Germany, Rep. 2194-2870, 2013. [Online]. Available: https://cos.bibl.th-koeln.de/frontdoor/index/index/docId/36

[19] D. Matheson, C. Jing, and F. Monforte, "Meter data management for the electricity market," in *Proc. Int. Conf. Probab. Methods Appl. Power Syst.*, 2004, pp. 118–122.

[20] G. Mateos and G. B. Giannakis, "Load curve data cleansing and imputation via sparsity and low rank," *IEEE Trans. Smart Grid*, vol. 4, no. 4, pp. 2347–2355, Dec. 2013.

[21] Y. Ang, Y. Qian, and S. Gao, "Factory energy data imputation by nearest neighbor search with clustering," in *Proc. IEEE Int. Conf. Adv. Elect. Eng. Comput. Appl. (AEECA)*, Aug. 2020, pp. 302–307.

[22] C. E. Borges, O. Kamara-Esteban, T. Castillo-Calzadilla, C. M. Andonegui, and A. Alonso-Vicario, "Enhancing the missing data imputation of primary substation load demand records," *Sustain. Energy Grids Netw.*, vol. 23, Sep. 2020, Art. no. 100369.

[23] D. Dua and C. Graff. (2019). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[24] Y. Gao, B. Foggo, and N. Yu, "A physically inspired data-driven model for electricity theft detection with smart meter data," *IEEE Trans. Ind. Informat.*, vol. 15, no. 9, pp. 5076–5088, Sep. 2019.

[25] J. Shi, Y. Liu, and N. Yu, "Spatio–temporal modeling of electric loads," in *Proc. North Amer. Power Symp. (NAPS)*, 2017, pp. 1–6.