

ORIGINAL RESEARCH

Open access

Generating Synthetic Power Consumption Patterns with VA-Creator: A Machine Learning Approach for Smart Homes

Sofia Alvarez^{1*}, Miguel Herrera¹, Lucia Vega², Pablo Torres¹, Carla Rios²

Abstract

With the rise of the Smart Home sector and the growing adoption of Machine Learning (ML), the acquisition of power consumption data is becoming increasingly significant. Gathering real-world energy measurements using sensors is often time-intensive, costly, error-prone, and, in some cases, entirely infeasible. To address this, we introduce the VA-Creator, a framework designed for generating Virtual Appliances (VAs). These VAs produce synthetic power consumption patterns (PCPs) using Neural Networks (NNs) that dynamically adjust their architecture according to the structure of the training data, thereby facilitating the development of additional VAs. To cover every appliance category commonly found in a typical household, we employ a range of neural network types—including Multilayer Perceptrons (MLPs), Long Short-Term Memory networks (LSTMs), and a tailored Generative Adversarial Network (GAN)—alongside other ML approaches such as XGBoost, choosing the most suitable method based on the specific properties of each appliance. The outputs of these ML models are subsequently compared with actual measurements and evaluated using Dynamic Time Warping (DTW) and MLP-based discriminator classification accuracy as evaluation metrics. Furthermore, to verify that the VAs enable effective training of ML models, we generate synthetic datasets with them and conduct an in-depth evaluation by training Non-Intrusive Load Monitoring (NILM) models. The results of this comprehensive assessment confirm that the VA models yield realistic and practically useful outputs.

Keywords Machine learning, Smart home, Synthetic sensor data, Energy data, Virtual appliance, Neural networks

*Correspondence:

Sofia Alvarez

sofia.alvarez@unam.mx

¹ Department of Grid Analytics and Optimization, Faculty of Engineering, National Autonomous University of Mexico, Mexico City, Mexico

² Department of Smart Energy Systems, Faculty of Engineering, Monterrey Institute of Technology, Monterrey, Mexico

Introduction

Energy constitutes an essential resource across numerous modern processes, prompting ongoing efforts to create energy-saving solutions. Conventional strategies focus on improving energy production efficiency or expanding the use of renewable sources, while more recent innovations leverage Artificial Intelligence (AI) and Machine Learning (ML) techniques. ML models support various applications, including energy monitoring [1-3], power forecasting [4-6], and energy disaggregation [7-9]. It is widely recognized that effective training and robust ML models require large volumes of data. In reality, acquiring substantial quantities of power consumption data is a laborious, costly undertaking that is further restricted by legal regulations and privacy concerns.

In light of these obstacles, a natural solution is to use simulations that produce meaningful synthetic power consumption data, which can either supplement existing datasets or generate realistic data from the ground up. Approaches to simulating aggregated power consumption can be divided into two categories based on their underlying strategy. Top-

down methods construct aggregated consumption figures from a broad, macro-level viewpoint [10]. In contrast, bottom-up methods build aggregate consumption from a detailed, micro-level perspective by combining appliance-specific data [11–13]. Appliance-level simulations function as digital twins of households, modeling virtual occupants whose customizable behaviors dictate appliance usage and resulting energy consumption. This mechanism produces realistic household-level aggregate power traces. Provided with adequate initial data, such simulations can, in principle, produce an unlimited supply of labeled data.

To leverage these simulations effectively, however, a sufficient volume of reliable appliance-level power consumption data is required—data typically obtained in real environments, for example, through smart plugs. Yet such measurements frequently suffer from inconsistent sampling rates, delays, or recording inaccuracies, rendering them prone to errors. Only a limited number of publicly accessible datasets—such as GeLaP [14], REFIT [15], and ENERTALK [16]—provide genuine appliance-level ground truth. Nevertheless, the quantity of actual collected data remains constrained, and relying on it as the basis for generating large-scale synthetic data would result in highly repetitive patterns. As later sections of this paper will illustrate, access to a diverse and extensive set of appliance-level power consumption patterns is highly advantageous.

A compelling strategy for expanding the availability of appliance-level signatures is synthetic data generation. This challenge gives rise to the following core research questions:

- How can realistic synthetic appliance-level data be created for every type of power consumption pattern found in a standard residential household?
- How can the quality of synthetic appliance-level data be assessed dependably?

In this paper, we introduce the Virtual Appliance Creator (VA-Creator), a comprehensive framework comprising methods and tools for producing synthetic appliance-level power consumption data derived from real-world appliance recordings. We describe techniques for synthesizing power consumption patterns (PCPs) that encompass all appliance categories typical of a household, utilizing a variety of ML approaches. Synthetic data generation generally involves several difficulties and unresolved issues that must be tackled. It is critical to confirm that the produced data is meaningful and faithfully represents the target domain. In many instances, determining the appropriate evaluation method for synthetic data is not straightforward. The present study proposes a method for comparing synthetic appliance-level power consumption data with real measurements by combining an ML classifier with Dynamic Time Warping (DTW).

In the context of bottom-up data simulation, once synthetic appliance-level data has been successfully generated, an additional question emerges: Can this synthetic appliance-level data be used to produce meaningful synthetic aggregated smart meter readings? Here, “meaningful data” refers to data suitable for training robust ML models. To compare aggregated synthetic data generated by the VA-Creator and aggregated synthetic data built from real appliance-level recordings with actual measurements, we adopt the empirical evaluation methodology introduced in [17] for rigorous dataset comparison. This approach involves training an adequate number of ML models—particularly Non-Intrusive Load Monitoring (NILM) models—on the datasets under examination and then analyzing their performance. This work shows that NILM models trained on aggregated synthetic data produced by the VA-Creator achieve superior performance compared to models trained on aggregated real data and to models trained on aggregated synthetic data derived from real appliance-level recordings.

The key contributions of this research are as follows:

- We introduce and examine a variety of Machine Learning (ML) methods and techniques designed to produce appliance-level data covering all appliance categories commonly encountered in a typical household.

- We propose a systematic methodology for assessing the quality of the generated synthetic appliance-level data in comparison to real-world measurements.
- We show that Non-Intrusive Load Monitoring (NILM) models trained on synthetic data produced by Virtual Appliances (VAs) achieve better performance than models trained on equivalent real-world data and models trained on synthetic data derived from real power consumption patterns (PCPs).

Related Work

As previously noted, smart meter data can be produced through various approaches. One category involves generating complete smart meter readings directly in a model-based fashion. Representative examples of such model-based techniques include [18-25]. However, these methods, which operate solely on aggregated smart meter data without regard to individual appliance behavior, face limitations when simulating consumption under specific conditions or scenarios.

Alternatively, smart meter data can be generated using a bottom-up approach that first synthesizes appliance-level power consumption data and then aggregates it into household-level smart meter readings. This bottom-up method is adopted in the present study to generate synthetic smart meter data for NILM evaluation. Unlike purely model-based techniques, the bottom-up approach enables the explicit modeling of targeted situations and behaviors. Several recent concepts for producing synthetic appliance-level data have appeared in the literature and are reviewed in greater detail below.

SmartSIM [26] was initially developed as a tool for synthesizing household smart meter data; to achieve this, it also creates individual appliance power consumption patterns (PCPs). SmartSIM relies on the parameterized models outlined in Barker et al. [27], which group appliances into categories and define PCPs based on each group's characteristics. This parameterization approach demands considerable manual tuning and requires deep domain expertise, making the addition of new appliance models labor-intensive and dependent on expert input.

ANTgen [28] is a framework specifically designed to generate appliance-level power consumption traces. It extracts operational cycles from an appliance's usage, eliminates inactive periods, and segments each cycle into portions that are then approximated using one of four mathematical functions: a constant power level, a linear transition between an initial and final power value, a logarithmic growth/decay curve, or an exponential growth/decay curve [28]. While effective for coarse approximation, these simplified functional forms cannot capture the fine-grained oscillations and transient details typically present in real-world PCPs.

SynD [29] likewise produces synthetic appliance-specific PCPs. Still, its method is template-based: it relies on a library of pre-recorded real PCPs, which are randomly selected and placed into designated time slots. Although this technique yields reasonably realistic power traces, the resulting patterns are restricted to those already observed in the training data, potentially leading ML models to overfit to a limited set of existing signatures.

The Automated Model Builder for Appliance Loads (AMBAL) [30] generates synthetic PCPs by repeatedly segmenting real consumption events and subsequently optimizing those segments to minimize the Mean Absolute Percentage Error (MAPE). This iterative segmentation and optimization process can result in excessively fine-grained segments, which in turn produce highly repetitive patterns. The authors assess the quality of their synthetic data using the Combinatorial Optimization disaggregation algorithm provided by NILMTK, an approach that has since become outdated [31].

With TraceGAN [32], synthetic power consumption patterns (PCPs) can be generated using a Generative Adversarial Network (GAN) that produces slightly varied PCPs for multiple appliances, intended to support training in Non-Intrusive Load Monitoring (NILM) applications. This is achieved by assigning a single label to each appliance type. However, relying on a single unified model for all appliances imposes several limitations. Using a single label per appliance prevents the intentional generation of a specific program or a specific combination of behaviors within a single appliance.

Moreover, incorporating additional appliances requires retraining the entire model from scratch, which is inefficient. Furthermore, the realism of the generated patterns could still be enhanced.

In addition, other GAN-based approaches exist, such as SGAN [33], which produces appliance-level data that closely mimics real-world observations. Nevertheless, this work is not comprehensive, as it does not cover all appliance categories typically present in a household. Although the authors state that their patterns are developed for NILM purposes, they do not include an evaluation in that context.

In contrast to these prior approaches, our method introduces a flexible framework that enables the creation of Virtual Appliances capable of generating highly realistic power consumption data by following the workflow outlined in this paper. The framework supports users in conveniently training new Virtual Appliance models whenever sufficient training data is available. We supply tailored ML techniques for each of the six appliance pattern categories described in the next section.

Furthermore, to the best of our knowledge, none of the previously discussed studies—or any other existing work—has conducted a reliable assessment of whether synthetically generated PCPs can produce aggregated smart meter data suitable for effectively training machine learning models.

Categorization of Appliances

A central goal of this paper is to enable the generation of synthetic data for every appliance type commonly found in a typical household. To accomplish this, a clear categorization of appliance types is required first. Alberternst *et al.* [34] refined and extended the appliance categorization for households originally proposed by Klemenjak *et al.* [29]. They established six distinct categories based on the specific generation techniques needed to produce realistic PCPs, which are elaborated below.

Single Patterns cover appliances such as a toaster or an electric kettle, which generate PCPs that may vary in duration but consistently exhibit highly similar shapes.

Programmatic Patterns encompass appliances such as dishwashers and washing machines that support multiple distinct programs—for instance, short versus long wash cycles. Unlike Single Patterns, each program can differ significantly in both pattern structure and length, effectively treating each program as a separate class.

Periodical Patterns include appliances such as refrigerators, freezers, or robotic vacuum cleaners, which operate autonomously and display recurring, periodic consumption behavior.

Composed Patterns are appliances that produce PCPs by combining elements from the previously described categories (Single Patterns and Programmatic Patterns), such as coffee makers and stoves. For example, a filter coffee machine exhibits multiple phases in its PCP: initial water heating, gradual release for brewing, and sustained use of a keep-warm plate, whose duration is manually controlled by the user turning the device off. The resulting complexity prevents these from being treated as simple single patterns.

Constantly-On Patterns involve appliances such as routers that maintain a steady power draw with no interruptions.

Duration-Dependent Patterns include devices such as televisions or computers, whose PCPs consist of diverse sub-patterns but whose total energy use is directly tied to the duration of human interaction. Because consumption depends heavily on duration, simply defining separate classes (as with Programmatic Patterns) is insufficient; the variable duration must be explicitly modeled.

Beyond this primary categorization, appliance types can be further grouped by the degree of human interaction into two broader classes: human-induced and permanent appliances. Appliances exhibiting Single, Programmatic, Composed, or

Duration Dependent patterns are classified as human-induced, since they require manual activation to consume energy. In contrast, Periodical and Constantly-On pattern appliances qualify as permanent, as they draw power independently once switched on. Note, however, that even some Periodical appliances are indirectly influenced by human behavior—for instance, leaving a refrigerator door open for an extended period typically alters its cooling cycles.

In this work, we develop specialized methods and ML-based approaches tailored to generate realistic PCPs for each of these pattern types.

Data Preprocessing

A primary goal of this study is to enable the generation of synthetic data for all appliance categories commonly found in a typical household. To illustrate this within the scope of the paper, we create representative examples for each pattern type based on real-world recordings. All appliances of the same pattern type can be generated using the same procedures as those applied to the representative examples shown here. The chosen appliances serve merely as characteristic illustrations of their respective pattern categories. The generation techniques described in this work for producing appliance-level data remain applicable to other appliances within the same categories.

The data for all appliances except one are drawn from the GeLaP dataset [14], specifically from household 14 (the television data come from household 04, as household 14 contains only 6 PCPs for this appliance). We selected the GeLaP dataset due to its public availability, which supports reproducibility and comparability, and its high-resolution recordings (1 Hz). This resolution is essential for accurately replicating real consumption behavior when the synthetic data are later used to construct synthetic smart meter readings. The sole exception is the router, a constantly-on appliance type absent from GeLaP; for this, we extracted data from the publicly available REFIT dataset [15].

To produce synthetic power consumption patterns (PCPs), we first preprocess the real-world appliance-level measurements, followed by the data transformations needed as input to the machine learning models.

Data preparation

Initially, the recordings for each considered appliance are resampled to a uniform 1 Hz frequency. Any isolated missing values are filled using forward filling (i.e., carrying forward the value from the previous timestamp). Next, we segment the continuous time series to isolate individual PCPs based on defined thresholds [13]. The core rule is: whenever the power exceeds w_{min} watts, the appliance is considered active; below this threshold, it is inactive. A PCP is deemed complete if the power remains below w_{min} for at least tb consecutive seconds. The patience parameter tb accommodates patterns in which the appliance briefly dips below w_{min} during operation. These thresholds can be adjusted to better suit the behavior of different appliances. The segmentation process yields the following counts of extracted PCPs: 51 from the toaster, 115 from the kettle, 70 from the washing machine, 14 from the vacuum cleaner robot, 84 from the coffee machine, 0 from the television, and 14 from the router. Table 1 summarizes both the number of extracted patterns and their maximum durations.

Since several of the appliance-level pattern generation models presented in this paper require input sequences of uniform length, all PCPs for a given appliance are zero-padded to match the maximum length (Maxlength) observed for that appliance. As shown in Table 1, the maximum durations vary substantially across appliances, resulting in distinct sequence lengths for each appliance dataset (notably, the PCPs of the vacuum cleaner robot and the router are extracted over a full day).

Table 1. Appliances used from the GeLaP dataset.

Appliance	Number of occurrences	Maximum duration

Toaster	51	130 s
Kettle	115	388 s
Washing machine	70	14,914 s
Vacuum cleaner robot	14	86,400 s
Coffee machine	84	13,806 s
Television	16	32,263 s
Router	14	86,400 s

Data transformation

A common issue observed across nearly all considered appliances is a pronounced bimodal distribution in their power values. This bimodality is especially pronounced for appliances that are either fully active with relatively stable power draw or completely off with zero consumption—examples include the toaster, kettle, coffee machine, and television (**Figure 1**). Due to the sharp separation and clustering at the extremes of the distribution, models in preliminary experiments struggled to capture subtle oscillations and instead produced flat, constant power predictions during “on” phases. Such outputs lack realism, as a human observer can readily distinguish genuine fluctuating patterns from artificially uniform ones.

To mitigate this problem, we apply a MinMax scaler with a target range of [0, 2]. We then apply a distribution-equalizing transformation, which also motivates the choice of 2 as the upper bound of the scaler (**Eq. 1**):

$$f(x) = \begin{cases} \sqrt{x} & \text{if } x \leq 1 \\ x^2 & \text{if } x > 1 \end{cases} \quad (1)$$

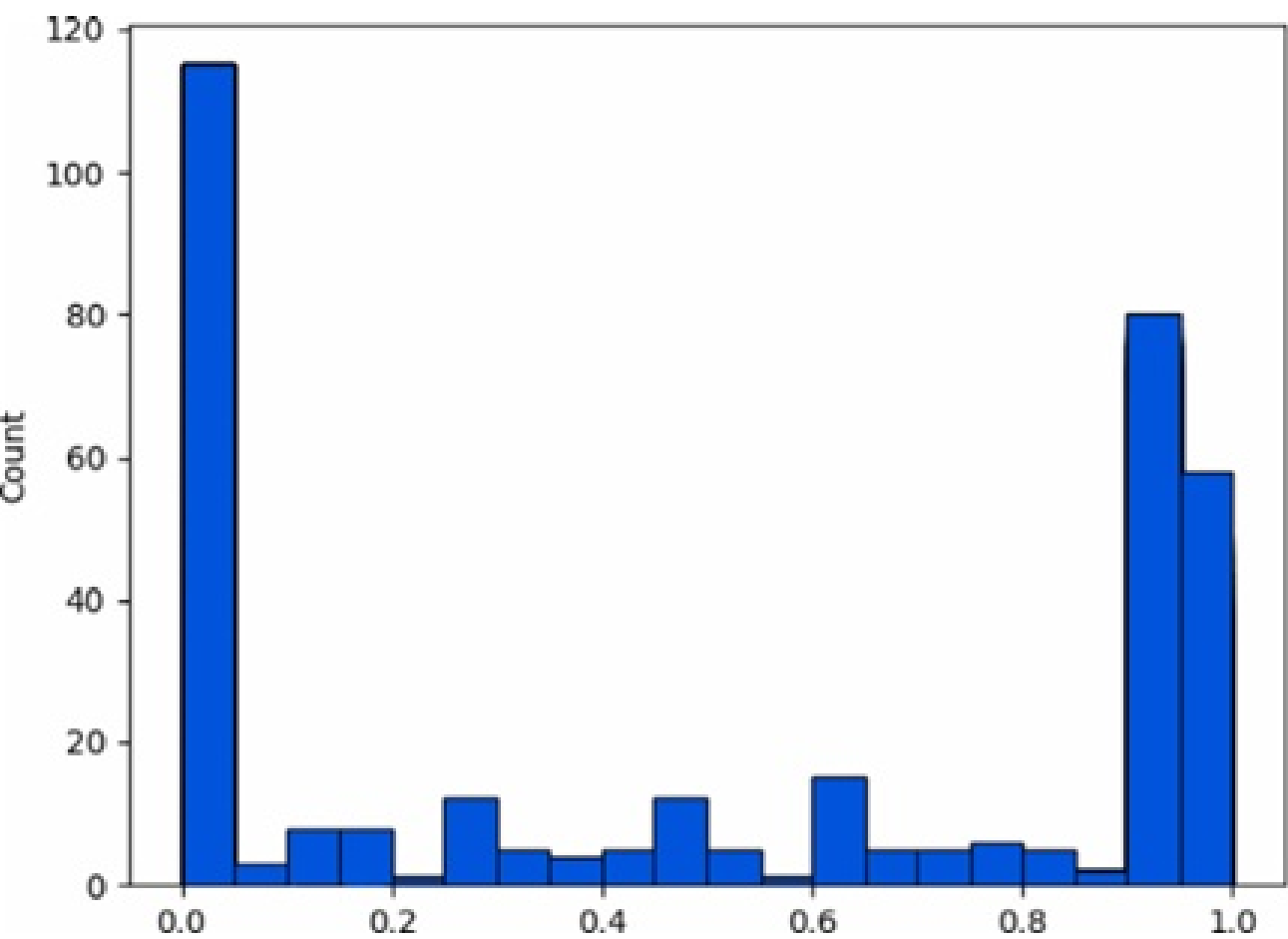


Figure 1. Distribution of power values in the kettle dataset after normalization.

For each value x in the dataset: if $x \leq 1$, compute the square root; if $x > 1$, compute the square. Finally, the transformed values are normalized to serve as input to the neural networks.

Figure 2 shows the resulting distribution after applying this transformation. Compared to **Figure 1**, the values around the midpoint (near 1) are significantly more evenly spread, enabling the models to learn and reproduce fine oscillations much more effectively. Further equalization is possible—for instance, by replacing the square/square-root operations with cube/cube-root functions—depending on the specific power consumption characteristics and distribution shape of each appliance’s PCPs.

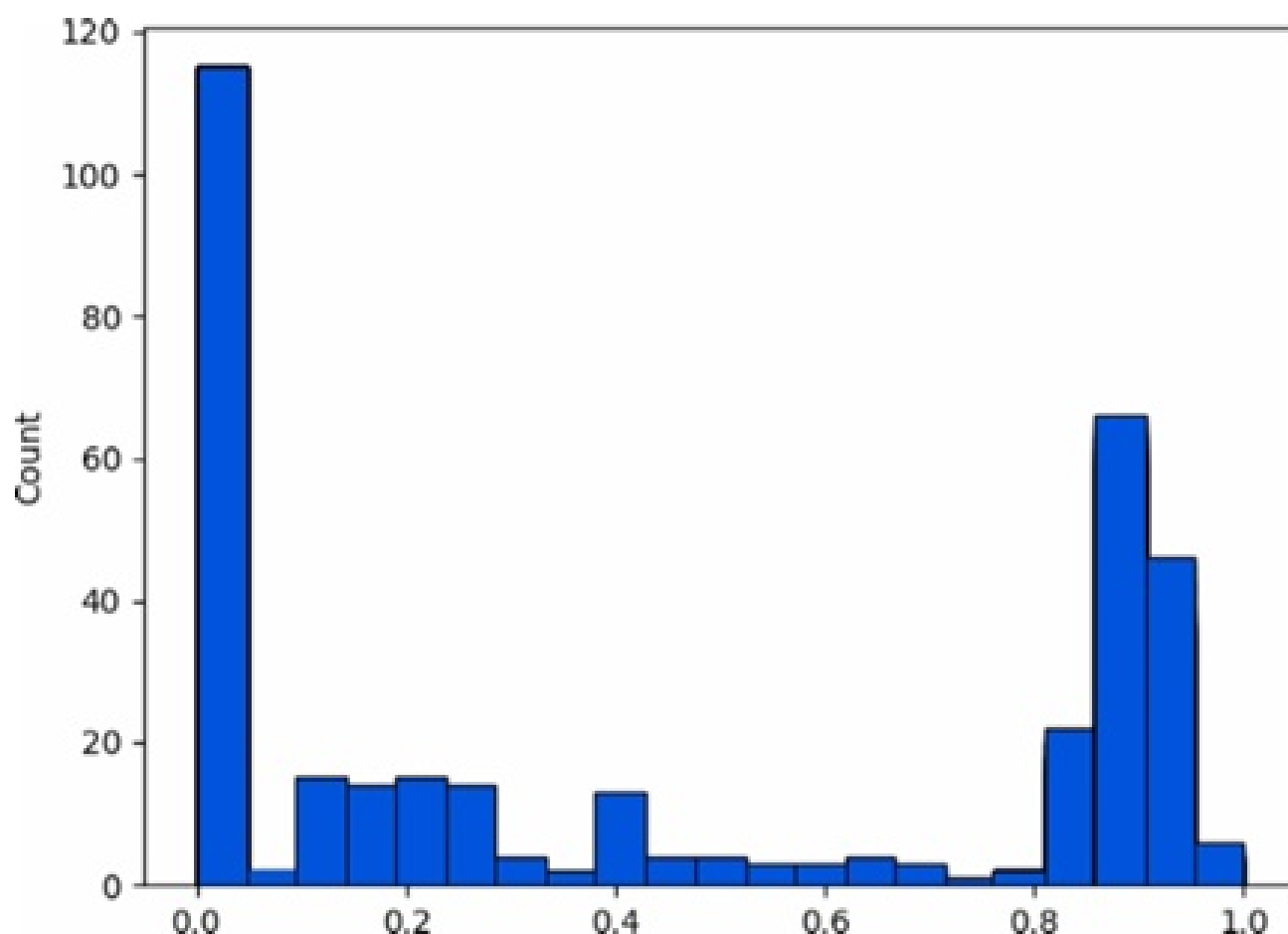


Figure 2. Distribution of power values in the kettle dataset after the MD-transformer.

Evaluation Metrics

As previously outlined, the goal is to produce power consumption patterns (PCPs) that are not exact copies of the real ones but exhibit a controlled level of variation. After generation, it becomes essential to apply appropriate measures to assess the quality of the synthetic data. Determining the realism of synthetic data is inherently challenging, as no single, universally accepted metric exists [35]. One option is qualitative assessment through visual inspection by a human observer. While feasible for a small number of patterns, this method quickly becomes impractical when evaluating large volumes of PCPs.

A central aim of this research is to leverage synthetic appliance-level data to construct aggregated power consumption traces suitable for training machine learning models. The evaluation strategy for the appliance-level data is therefore closely aligned with this purpose. Specifically, we use a Multilayer Perceptron (MLP) classifier to distinguish between real and synthetic patterns. In addition, we apply Dynamic Time Warping (DTW) to perform a statistical comparison of the real and synthetic sequences.

Evaluation data preparation

To evaluate each appliance, we create at least one synthetic dataset corresponding to each real dataset. Within each synthetic dataset, each real PCP is matched with a synthetic counterpart, ensuring that the real and synthetic datasets for a given appliance contain the same number of PCPs. For evaluation using the MLP classifier, each dataset (real and

synthetic) is split into a training subset and a test subset. The training subset consists of the first x PCPs, while the test subset comprises the remaining PCPs.

MLP classifier

To assess the fidelity of the generated PCPs for each appliance, we construct a Multilayer Perceptron (MLP) classifier model, referred to hereafter simply as the Classifier. During training, the model receives input from both the training portion of the real dataset for a given appliance and the corresponding synthetic dataset. The hidden layers use sigmoid activation functions and dropout regularization to mitigate overfitting. The number of neurons in each layer is scaled to the maximum PCP length for the appliance under consideration, making the architecture flexible and adaptable to any appliance type included in the study.

The model outputs a continuous value ranging from 0 (indicating a synthetic PCP) to 1 (indicating a real PCP). For every evaluation, the Classifier is trained for 100 epochs using the Adam optimizer with a learning rate of 0.0001 and binary cross-entropy as the loss function.

Through this training process, the Classifier learns to discriminate between real and synthetic PCPs. When tested on unseen data, an output near 1 suggests the model perceives the input as real, while a value near 0 indicates it perceives the input as synthetic. Values close to 0.5 indicate uncertainty, indicating that the model cannot confidently classify the pattern as either real or synthetic. Consequently, our target outcome for a successfully generated synthetic PCP is an output value around 0.5, signifying that the Classifier is unable to reliably distinguish synthetic from real data. This outcome aligns with the desired realism and variability.

Importantly, the evaluation protocol in this work involves training multiple Classifier instances rather than relying on a single model. For each appliance and each synthetic dataset under examination, a fresh Classifier is trained from scratch. This repeated training process enables a robust assessment of whether—and to what extent—a given generation method produces synthetic data that remains indistinguishable from real data across different appliances and generation approaches.

To verify that the developed classifier can effectively differentiate between synthetic and real power consumption patterns (PCPs), we train and evaluate it using both real and synthetic data from a representative single-pattern appliance (the kettle) and a programmatic-pattern appliance (the washing machine). The synthetic data are created by superimposing a normal (Gaussian) distribution onto each real PCP. The distribution has a mean of 0, and its standard deviation is scaled to reflect the specific characteristics of each pattern: specifically, it equals the PCP's maximum watt value divided by 100. To maintain comparability between real and synthetic datasets, noise is applied exclusively to the portions of the PCP where the appliance is actively consuming energy.

Results show that, for the Single Pattern appliance (kettle), the Classifier assigns an average output of 0.860 to real PCPs and 0.257 to synthetic PCPs, clearly demonstrating its ability to distinguish between the two classes. Similarly, for the Programmatic Pattern appliance (washing machine), the Classifier yields 0.803 for real PCPs and 0.187 for synthetic ones, reinforcing its discriminative capability. These outcomes confirm that the classifier-based approach functions reliably and consistently separates real from synthetic PCPs. At the same time, the results highlight that simply adding Gaussian noise to existing patterns is insufficient to produce realistic synthetic data. This finding emphasizes the need for more advanced generation techniques and structured frameworks.

Dynamic time warping

For a complementary statistical evaluation of the generated data, we use Dynamic Time Warping (DTW) as the similarity measure for time-series comparison. The core principle of DTW is to allow one-to-many and many-to-one alignments that minimize the cumulative distance between two time series: $X = (x_1, x_2, \dots, x_n)$ of length $N \in \mathbb{N}$ and $Y = (y_1, y_2, \dots, y_m)$ of length $M \in \mathbb{N}$ [36].

A warping path $p = (p_1, p_2, \dots, p_l)$ of size (N, M) establishes a correspondence between the two sequences $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_m)$ by mapping each element x_{nl} from X to an element y_{ml} from Y . The total cost $c_p(X, Y)$ of a given warping path p is computed as the sum of the local costs c along the path.

$$c_p(X, Y) = \sum_{l=1}^L c(x_{nl} - y_{ml}) \quad (2)$$

The optimal warping path p^* is the one that achieves the lowest total cost among all feasible paths. The DTW distance (X, Y) between sequences X and Y is then defined as the total cost associated with this optimal path p^* [36].

$$\begin{aligned} DTW(X, Y) &= c_p(X, Y) \\ &= \min \{c_p(X, Y) \mid p \text{ is an } (N, M) \text{-warping path}\} \end{aligned} \quad (3)$$

A major benefit of DTW, particularly valuable for our evaluation, is that the sequences being compared do not need to have identical lengths (N and M can differ). This flexibility enables direct and meaningful comparison of PCPs—even when they vary in duration [36].

To perform a trustworthy side-by-side comparison of the synthetic data against the genuine measurements, the DTW distances listed below are computed:

- Each real PCP is compared to every other real PCP within the real dataset (real/real)
- Each synthetic PCP is compared to every other synthetic PCP within the synthetic dataset (synth/synth)
- Each synthetic PCP is compared to every real PCP across the real dataset (synth/real)
- Each synthetic PCP is matched and compared directly, one-to-one, against its corresponding real PCP (paired by label) (synth/real 1vs1)

Studying the resulting DTW distance distributions for real/real, synth/synth, and synth/real allows us to assess whether the synthetic dataset exhibits a structural resemblance to the real one. The paired 1-vs-1 comparisons (synth/real 1vs1) give a precise indication of how closely each generated pattern matches its real counterpart.

Performing 1-vs-1 DTW also helps confirm that the synthetic patterns are not simply duplicates of the originals. When real patterns are compared against identical copies of themselves, DTW naturally returns 0, which aligns with theoretical expectation. We confirmed this behavior experimentally for both the kettle and the washing machine.

To better benchmark the effectiveness of our proposed generation techniques, we also computed DTW distances between the real PCPs and synthetic versions generated solely by adding Gaussian noise, for both the kettle and the washing machine. The noise parameters matched those used earlier for the classifier tests (mean = 0, standard deviation = maximum watt value of the PCP $\div$ 100). Across both appliances, these noise-based synthetic patterns produced DTW distances that were dramatically higher—at minimum ten times larger—than the distances achieved by the generation models introduced in this paper.

From these observations, we can infer that DTW serves as a robust indicator for detecting unwanted over-similarity between synthetic and real patterns. At the same time, unusually large DTW values provide strong evidence that a synthetic pattern deviates substantially from its real counterpart. Overall, DTW stands out as a practical and informative

metric in the initial stages of pattern synthesis because it quantifies the actual closeness (or separation) between generated and observed sequences.

Synthetic Appliance-Level Data Generation

With the real household appliance data now properly preprocessed and suitable evaluation procedures in place, the following sections explain in depth how synthetic PCPs are created for each defined pattern category.

Single and programmatic pattern generation

Here, we first outline the end-to-end procedure for synthesizing realistic PCPs of single-pattern and programmatic-pattern appliances starting from authentic power consumption traces. We present two generative architectures—an MLP-based generator and a GAN-based generator—and provide a balanced discussion of their benefits and drawbacks. We then quantitatively judge the quality of the resulting synthetic PCPs using the previously described Classifier and DTW measures.

Data generation models

Below, we describe the two neural network architectures used to generate synthetic power consumption patterns (PCPs) for single- and programmatic-appliance types. One is a Generative Adversarial Network (GAN), the other is a standalone generator model we term the lonely Generator (IGen), which draws inspiration from a GAN's generator component but operates without any discriminator. Both architectures were implemented in Keras, utilizing TensorFlow as the underlying deep learning backend.

Lonely generator

The IGen is constructed as a Multilayer Perceptron (MLP), a classic feedforward neural network comprising at least three layers: an input layer, one or more hidden layers, and an output layer [37]. Like most deep learning models, it is trained via backpropagation.

The input layer receives a discrete label for each distinct pattern (e.g., 0–50 for the various toaster PCPs). These labels are first embedded into a dense vector and then combined with random Gaussian noise. This noise injection prevents the network from memorizing individual patterns too precisely, thereby preserving meaningful variability in the outputs. Hidden layers employ Leaky ReLU activation functions [38]. The number of neurons per layer is dynamically scaled according to the longest PCP observed for the appliance in question, as illustrated in **Figure 3**. This design yields an architecture resembling a decoder. It enables the same network structure to accommodate a wide range of appliance types without requiring manual reconfiguration, greatly enhancing its flexibility. The output layer applies a sigmoid activation, producing values normalized to 0-1 that collectively form the complete PCP. Training is performed using the Adam optimizer with a learning rate of 0.0001 and mean squared error as the loss function.

Although the toaster and kettle fall into the single-pattern category, and the washing machine into the programmatic category, we successfully trained a single IGen configuration (following the rules above) on all three appliances. This illustrates the model's adaptability. Because the toaster and kettle exhibit simpler, more uniform patterns, satisfactory results were obtained after fewer training epochs (toaster: 29,000 epochs; kettle: 25,000 epochs). In contrast, the considerably more intricate and variable PCPs of the washing machine required significantly longer training (200,000 epochs). A modest dropout rate of 0.1, combined with the injected input noise, effectively prevents overfitting and avoids overly faithful reproduction of individual real patterns.

Conditional Wasserstein GAN with gradient penalty

As a benchmark against the standalone MLP-based IGen described above, we also implemented a conditional Wasserstein GAN with gradient penalty (cWGAN-GP).

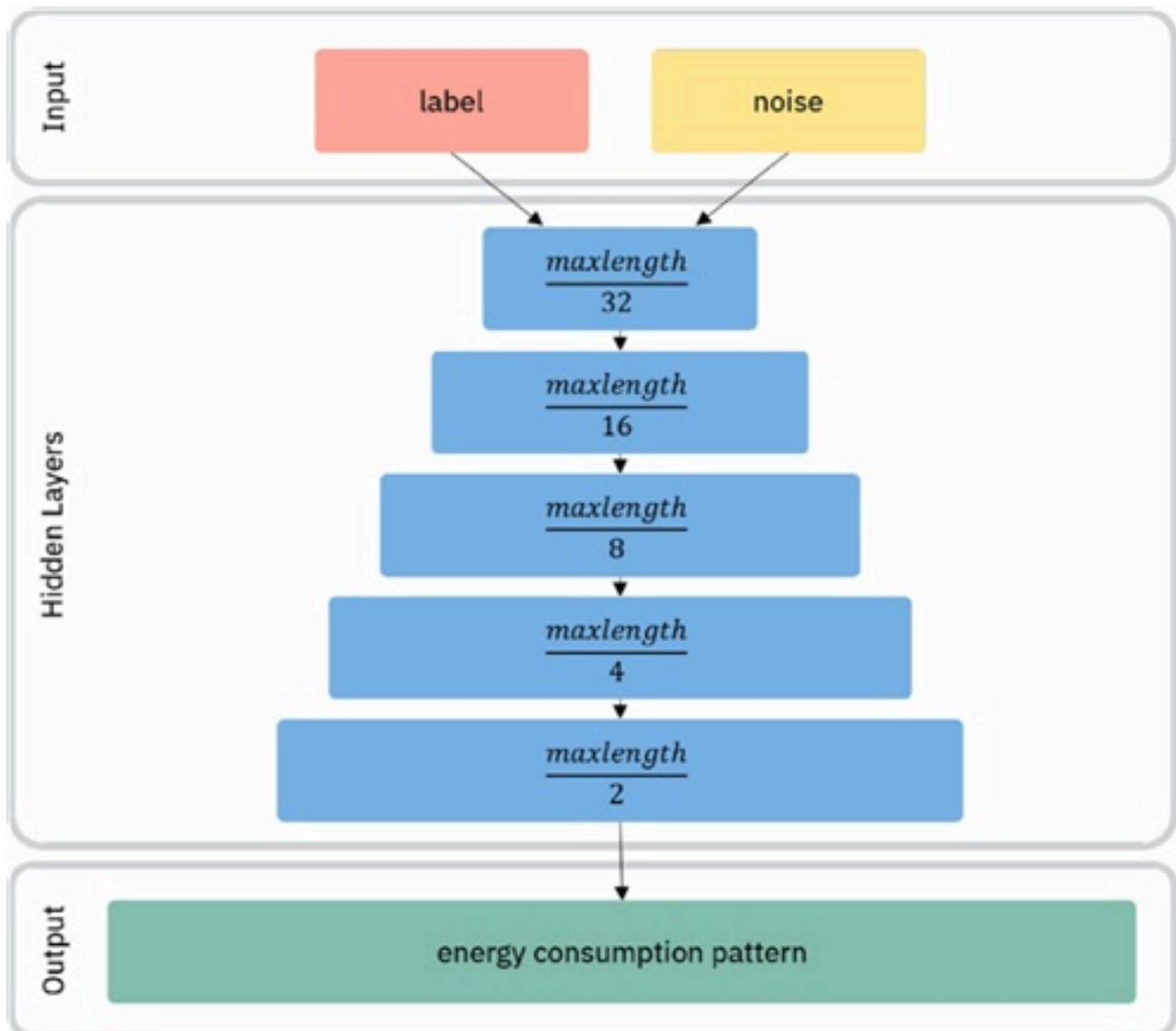


Figure 3. Abstract architecture of the MLP model described in 6.1.1.1.

In the classic GAN framework, introduced by Goodfellow *et al.* [39], a generator and a discriminator are trained concurrently in a minimax contest to produce increasingly realistic samples. This paradigm has become a cornerstone of synthetic data generation, especially in image synthesis tasks [40–44]. The generator creates outputs from random noise, while the discriminator evaluates their realism by comparing them to real examples and provides a feedback signal that guides further improvement of the generator. Over time, the discriminator sharpens its ability to separate real from fake samples, while the generator learns to fool it more effectively.

The Wasserstein GAN with gradient penalty (WGAN-GP) is an advanced variant that replaces the standard discriminator loss with the Earth Mover’s (Wasserstein) distance [45], turning the discriminator into a so-called critic. This change promotes more stable convergence and yields higher-quality outputs compared with vanilla GANs [46]. The gradient penalty mechanism further improves training stability by replacing weight clipping (used in earlier WGAN formulations) with a penalty term that constrains the critic’s gradient norm with respect to its input [47]. Finally, the conditional aspect of cWGAN-GP enables class-specific generation: by conditioning both the generator and the critic on class labels, the model can produce samples belonging to a specified category rather than purely random ones.

In our implementation, the generator is structurally identical to the MLP used in the IGen. It takes random noise concatenated with a pattern label as input and outputs a complete PCP. The critic, also an MLP, receives either a real or generated PCP along with the corresponding label and outputs a scalar realism score. To maintain training balance, the critic is deliberately designed to be shallower and narrower (fewer layers and fewer neurons) than the generator. Because distinguishing real from fake is generally an easier task than synthesizing realistic patterns from scratch, a weaker critic helps prevent the discriminator from overpowering the generator and ensures more reliable adversarial learning.

As with the lonely Generator (IGen), a single cWGAN-GP architecture was successfully trained on both the single-pattern appliances (toaster and kettle) and the programmatic-pattern appliance (washing machine). This demonstrates the model's uniformity and strong generalization capability across pattern types. In comparison with the IGen, achieving comparable quality for the simpler single-pattern appliances required substantially more training epochs in the cWGAN-GP setup (toaster: 150,000 epochs; kettle: 150,000 epochs). This indicates that the conditional Wasserstein GAN with gradient penalty requires greater computational resources and longer training times than the simpler MLP-based IGen to achieve equivalent performance. For the more complex programmatic patterns of the washing machine, the required number of epochs remained identical across the two models (200,000). However, because each training iteration in the cWGAN-GP takes considerably longer than in the IGen, the overall training duration is still significantly higher for the GAN-based approach. In general, the learning dynamics of the cWGAN-GP closely resemble those of the IGen. Thanks to the conditional labeling, the model rapidly captures the typical durations of PCPs, yet it requires more epochs to accurately reproduce the finer transient oscillations. As with the IGen, we applied a modest dropout rate of 0.1, along with input noise, to avoid excessive memorization and overly precise replication of individual real patterns.

Evaluation

To carry out the assessment, we generated two separate synthetic datasets for each real dataset: one using IGen and the other using cWGAN-GP. For every appliance considered (toaster, kettle, and washing machine), the real dataset and both synthetic datasets contain the same number of PCPs, with each unique pattern label appearing precisely once. The chosen appliances serve as prototypical representatives of their respective pattern categories rather than being evaluated solely as individual devices. Each dataset—real and synthetic—was subsequently partitioned into a training subset and a test subset. The training subset comprises the first x PCPs, while the test subset contains all remaining PCPs (Table 2). This results in three datasets per appliance (one real and two synthetic), each further split into equally sized training and test portions specific to that appliance. Throughout the evaluation process, no samples were modified or exchanged between datasets, ensuring full comparability.

Following the evaluation framework, we trained two separate Classifier models for each appliance: one using the real dataset paired with the IGen-generated synthetic dataset, and another using the real dataset paired with the cWGAN-GP-generated synthetic dataset. In both cases, the Classifier was trained on a balanced combination of real and synthetic examples. To complement the Classifier results, we also computed the full set of DTW distances for each generated dataset.

Table 2. Partitioning of the real and the synthetic datasets for the single and programmatic pattern type appliances.

Appliance	Total	Training	Test
Toaster	51	25	26
Kettle	115	55	60
Washing machine	70	35	35

The evaluation results for the Classifier outputs and the DTW distance distributions are presented below for the toaster, the kettle, and the washing machine. Within each subsection, we first discuss the synthetic data produced by the IGen, followed by the results for the cWGAN-GP-generated data.

Single pattern type appliance I

We begin the evaluation with the toaster as the first exemplar of a single-pattern appliance. For synthetic test data generated by the IGen, the Classifier produces output values that are closely centered around 0.5, indicating high uncertainty and an inability to confidently classify the patterns as either real or synthetic. When comparing the histogram of Classifier outputs for the real test data with that of the synthetic test data, the distributions appear nearly identical. This visual similarity is further supported by the almost equal mean values (real: 0.425; synthetic: 0.428). Overall, these findings show that the Classifier fails to differentiate between the real and IGen-synthetic datasets for the toaster.

A similar pattern emerges when examining the Classifier results for the cWGAN-GP-generated synthetic test data alongside the corresponding real test data. The output distributions are again highly comparable (real mean: 0.451; synthetic mean: 0.445), closely mirroring the behavior observed with the IGen synthetic data. Consequently, the Classifier is equally unable to separate the cWGAN-GP synthetic test samples from the real ones.

Regarding the DTW analysis for the IGen-generated toaster data, the three key distance distributions—real/real, synth/synth, and synth/real—exhibit strong overlap, confirming that the synthetic dataset structurally resembles the real dataset to a high degree. The one-to-one paired comparisons (synth/real 1vs1) yield consistently low DTW distances, further evidencing close similarity between matched synthetic and real patterns. **Figure 4** illustrates the two toaster PCP pairs that exhibit the largest DTW distances among all pairs. Even in these cases of relatively greatest divergence, visual inspection reveals that the real and synthetic patterns remain strikingly alike.

The DTW distance distributions obtained from synthetic toaster data produced using cWGAN-GP exhibit virtually no divergence from those produced using IGen. Likewise, the pairwise (1vs1) comparison yields highly comparable outcomes, providing no evidence of meaningful differences between the performance of the two generative approaches.

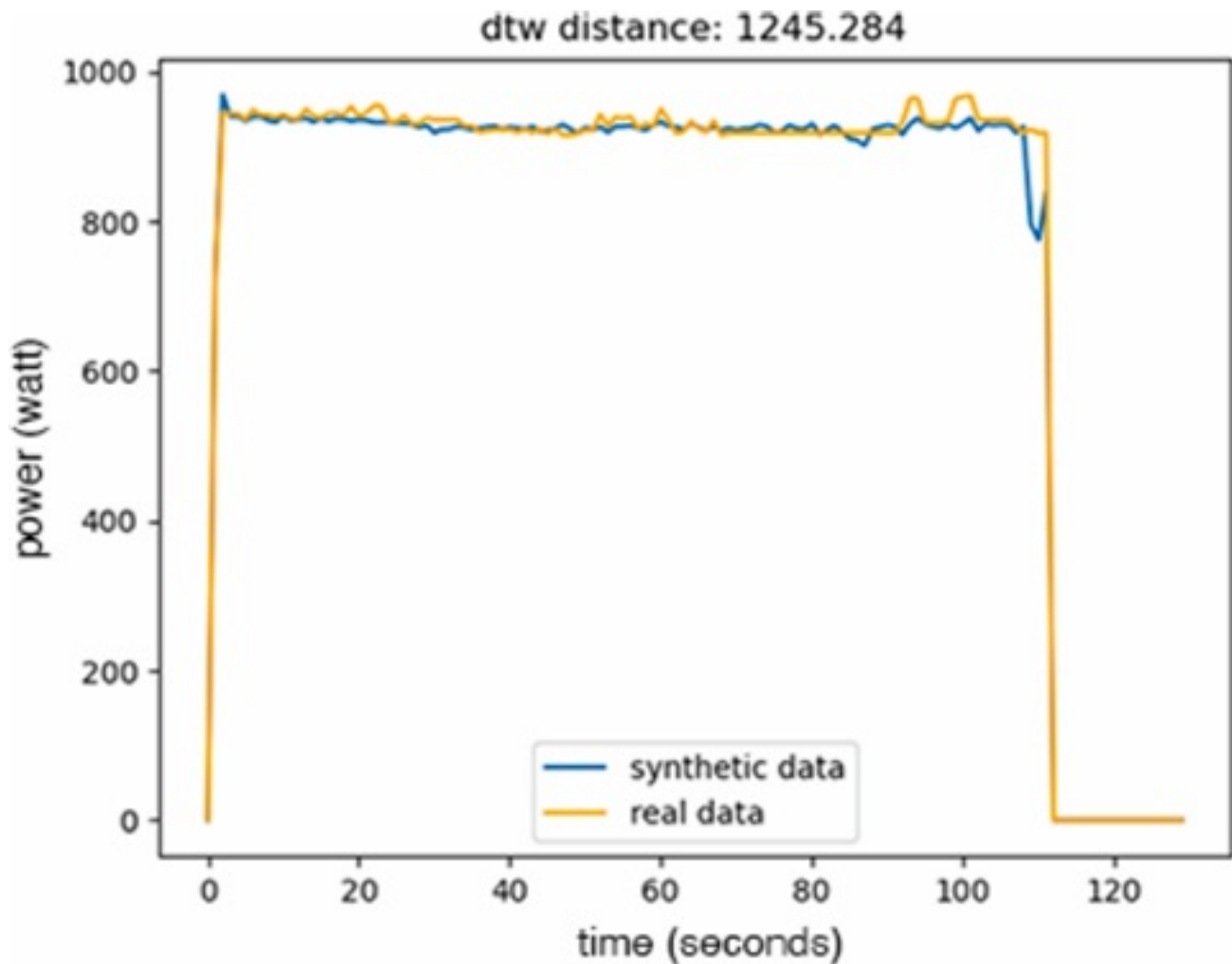


Figure 4. Synthetic PCP created by the IGen model (shown in blue) juxtaposed against a real PCP (shown in orange) from the toaster that recorded the largest observed DTW distance (1245.285).

Single pattern type appliance II

For the second representative of a single-pattern appliance category, we examine the fidelity of synthesized kettle load profiles. The histogram depicting Classifier scores for kettle data synthesized by IGen reveals that 58 out of 60 PCPs cluster near 0.5. In contrast, the Classifier assigns scores close to 0 to the other two PCPs. This pattern mirrors the histogram derived from the actual test set, in which 58 of the 60 PCPs also concentrate around 0.5, the remaining two lie near 0, and the mean scores are nearly identical (real: 0.431, synthetic: 0.433). As with the toaster, the Classifier fails to separate most instances across the two datasets, strongly suggesting that the synthetic kettle profiles closely resemble their real counterparts. The fact that two PCPs—originating from both the real and synthetic test sets—are flagged as synthetic points indicates the presence of two atypical profiles in the test data that deviate noticeably from the training distribution.

Consistent with the toaster findings, Classifier outputs for kettle data synthesized via cWGAN-GP closely match those obtained with IGen. The score distribution remains almost unchanged (58 values near 0.5 and 2 values near 0), and the mean scores across the test sets show only a marginal difference (real: 0.447, synthetic: 0.446). We therefore infer that the Classifier is unable to reliably discriminate between real and synthetic kettle data.

Parallel to the toaster results, the DTW distance distributions confirm strong alignment between the real kettle dataset and the versions synthesized by IGen. When directly comparing synthetic and real PCPs in a 1vs1 matching setup, DTW

distances exceed those observed for the toaster. This outcome is anticipated, however, given that DTW values are absolute and kettle profiles characteristically involve higher power draws together with extended duration.

The DTW distance distributions for kettle data synthesized by cWGAN-GP are likewise nearly identical to those produced by IGen. The 1vs1 comparison remains very close across both models, with a modest advantage apparent for cWGAN-GP. **Figure 5** illustrates two kettle PCPs that exhibit a notably large DTW distance. Yet, a side-by-side visual inspection still reveals substantial structural and shape similarity between the real and synthetic versions.

Programmatic pattern type appliance

We now turn to a typical example from the programmatic pattern category and examine how well the washing machine load profiles are reproduced. Among the synthetic PCPs created by IGen, Classifier scores are distributed with some near 0.5, others approaching 0, and a considerable share peaking at 1. These patterns closely align with the classification results from genuine data, with average scores almost identical (real: 0.638, synthetic: 0.636). The Classifier's tendency to label certain PCPs as synthetic—even within the real dataset—arises from the diverse program options available on washing machines, which exhibit far greater variability than the consistent usage profiles of toasters or kettles. As a result, the test set contains program behaviors unfamiliar to the Classifier from its training phase, causing it to flag them as artificial.

A comparison of Classifier outputs reveals that cWGAN-GP produces synthetic washing machine profiles of noticeably lower fidelity than IGen. In the synthetic test set from cWGAN-GP, numerous PCPs receive synthetic classifications, in stark contrast to the real test set, where scores remain more realistic. This discrepancy is evident in the mean values (real: 0.583, synthetic: 0.357). Hence, for cWGAN-GP-generated washing machine data, the Classifier achieves at least some degree of success in separating genuine from fabricated profiles.

The presence of multiple distinct washing programs causes the DTW distance distributions to differ significantly from those seen in single-pattern devices such as the toaster and kettle. The real/real, synth/synth, and synth/real distributions each display three clear modes reflecting the variety of programs, yet the overall shapes stay reasonably comparable.

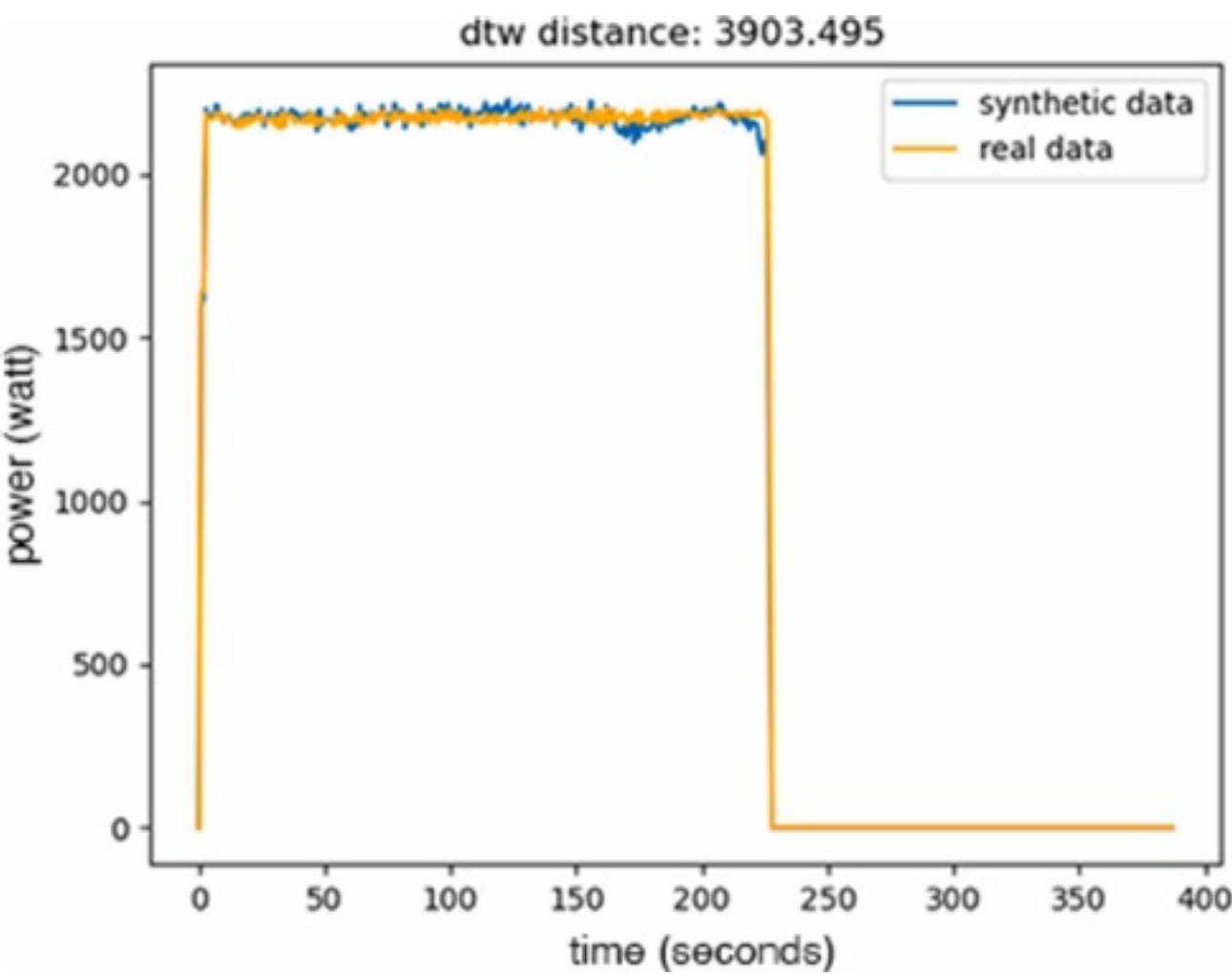


Figure 5. Synthetic PCP generated by the cWGAN-GP model (blue color) in comparison with a real PCP (orange color) of the kettle with a high DTW distance (3903.495).

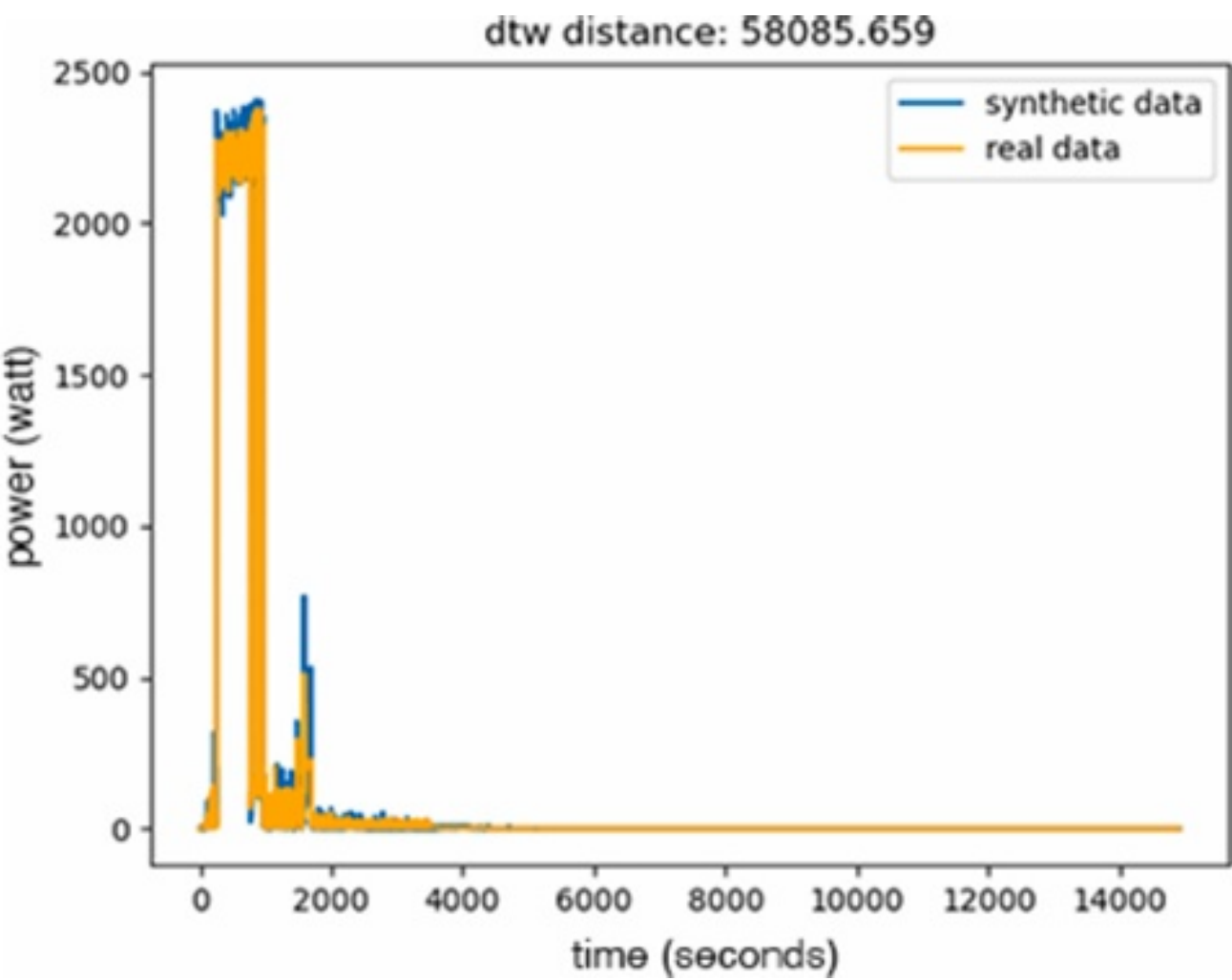


Figure 6. Synthetic PCP generated by the IGen model (blue color) in comparison with a real PCP (orange color) of the washing machine with a small DTW distance (58085.659).

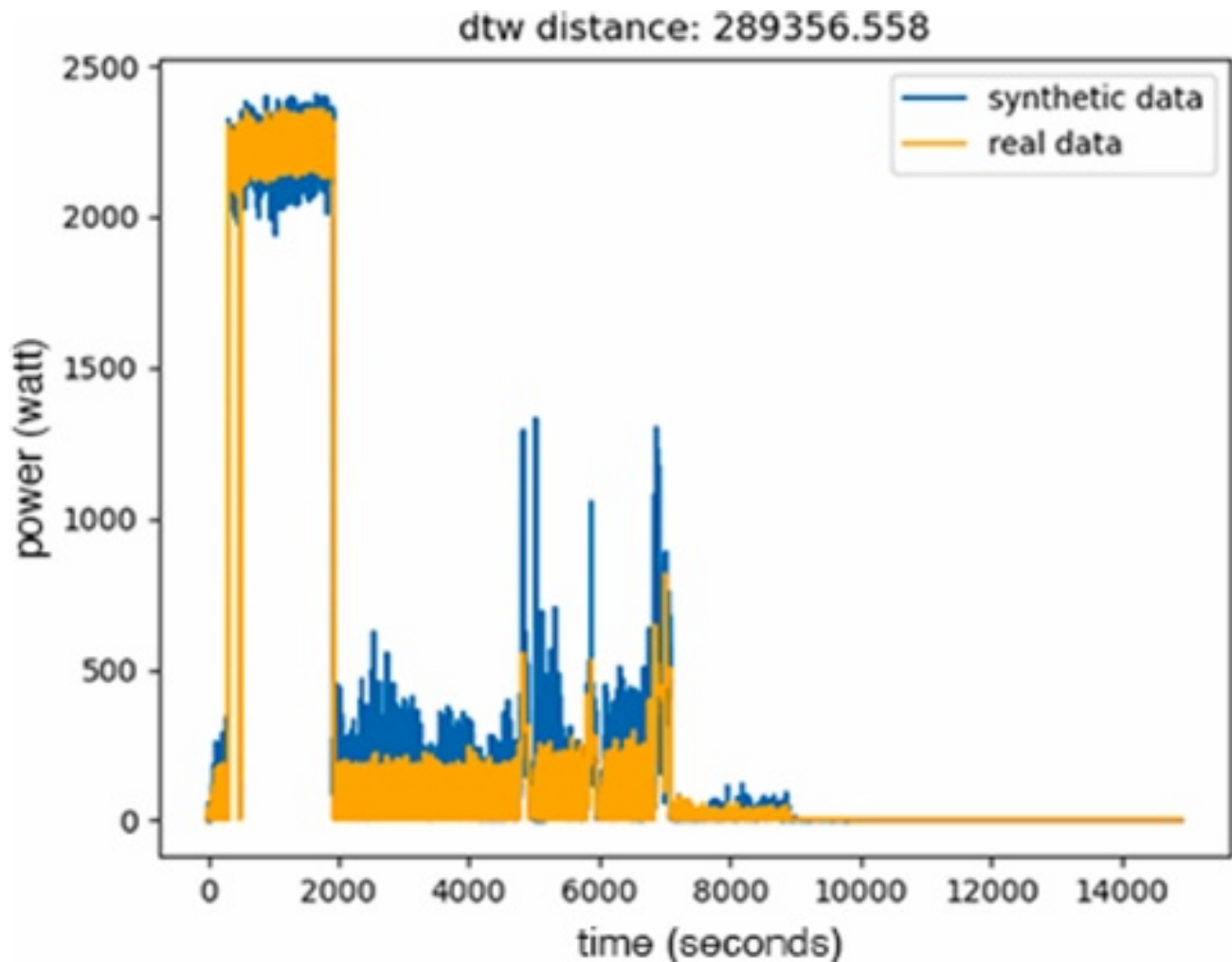


Figure 7. Synthetic PCP generated by the cWGAN-GP model (blue color) in comparison with a real PCP (orange color) of the washing machine with a high DTW distance (289356.558).

Across the board, DTW distances are considerably higher than those measured for the toaster or kettle. This is primarily because DTW computes absolute differences, and washing machine cycles last much longer than those of smaller appliances. An example of a close synth/real 1vs1 match is illustrated in [Figure 6](#).

The DTW distance distributions for cWGAN-GP-generated washing machine data mirror the three-peak structure observed with IGen. In the pairwise (1vs1) evaluation, however, distances associated with the GAN-based model are substantially larger than those from IGen in nearly every case, except for one outlier. This pattern suggests that the synthetic profiles from cWGAN-GP are overall less faithful to the real data.

Periodical and composed pattern generation

Having covered the synthesis of single-pattern and programmatic PCPs, the following section explains the approach for constructing synthetic PCPs for appliances characterized by periodic or composite patterns, drawing on authentic real-world electricity usage traces.

Periodical as well as composed PCPs are composed of recurring subunits—either individual single patterns or full programmatic sequences—that repeat after fixed or semi-fixed intervals. A clear illustration is the robotic vacuum cleaner, whose profile comprises nearly identical cleaning bursts at predictable intervals. The coffee machine follows a similar

logic. However, its consumption trace combines two separate single-pattern types: one for brewing and several for intermittent heating. Given that both IGen and cWGAN-GP can already synthesize high-quality single patterns, the key missing component is a mechanism that accurately determines the temporal placement of these patterns.

We therefore designed the Activity Prediction Model (APM) to fulfill this role; a detailed description appears later. With the APM in place, we proceed to synthesize periodical and composed PCPs and then assess their quality .

Data generation models

As previously noted, both periodical and composed PCPs can be broken down into smaller recurring units—specifically single patterns for periodical cases and programmatic patterns for composed cases—that appear at regular or predictable intervals. These insertion timestamps are forecasted by the Activity Prediction Model (APM), which is introduced in detail in the current section.

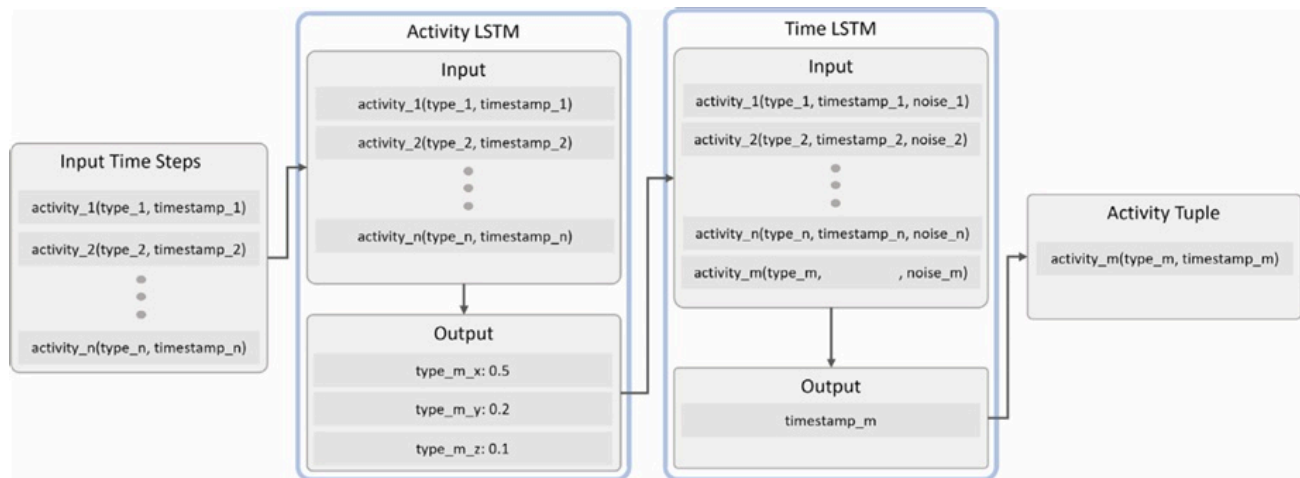


Figure 8. Schematic structure of the Activity Prediction Model.

Activity prediction model

For the synthesis of periodical and composed patterns, we have adapted and implemented the APM originally described in [34]. This model forecasts upcoming activities by drawing on historical activity sequences, employing two separate long short-term memory (LSTM) networks [48]. The first network, referred to as the Activity LSTM, takes the most recent activities, along with their start times, to determine the type of the next activity. Subsequently, the second network, the Time LSTM, uses the same sequence of the last n activities, along with the newly predicted activity type, to estimate the corresponding start time.

Figure 8 depicts the step-by-step iterative construction of activity tuples using the APM. At each prediction step, the model considers the previous n tuples ($activity_i(type_i, timestamp_i)$ for $i = [1, 2, \dots, n]$) as input (**Figure 8**, Input Time Steps). The Activity LSTM computes a probability distribution over possible next activity types based solely on the sequence of the preceding n activity types, following a mechanism akin to an N -gram model [49]—a well-established probabilistic technique in computational linguistics that estimates the likelihood of the next token from the frequency counts of prior n tokens in the training corpus. The key improvement in our LSTM implementation lies in incorporating not only the sequence of activity types but also their associated timestamps, thereby enriching the input feature space (**Figure 8**, Activity LSTM input). The Activity LSTM outputs a probability distribution; in the example shown in **Figure 8**, this yields 0.5 for $type_x$, 0.2 for $type_m_y$, and 0.1 for $type_m_z$, leading to the selection of $type_m_x$ (the highest-probability class) as the next activity.

Structurally, the Activity LSTM mirrors the architectures of both IGen and the Classifier: it is a feed-forward neural network comprising an input layer, several hidden layers with leaky ReLU activations, and an output layer with a softmax activation to produce probabilities that sum to 1. Training is performed using the Adam optimizer (default learning rate 0.001) and categorical cross-entropy as the loss function.

The prediction of the activity's start timestamp is handled by the Time LSTM. This network receives the same base input as the Activity LSTM, augmented by an additional triple representing the just-predicted activity (`activity_m(type_m, _, noise_m)`), where the type is taken from the Activity LSTM output (in the example, `type_m = type_m_x`), a placeholder is used for the unknown timestamp, and a noise term is injected to introduce controlled variability and generate diverse PCP variants. The Time LSTM then outputs the predicted `timestamp_m`, thereby completing the new activity tuple (`activity_m(type_m, timestamp_m)`), which is appended to the growing sequence.

Like the Activity LSTM, the Time LSTM is designed as a feed-forward neural network. Still, it includes a small dropout rate to prevent overly precise timestamp predictions and thereby ensure natural variation across generated profiles. It is likewise optimized with Adam (default settings) and trained using mean absolute error as the loss function. Each newly generated tuple is immediately added to the input history for the subsequent prediction step, enabling fully autoregressive, iterative sequence building. In this manner, the APM can successively determine the timestamps at which single or programmatic patterns should be placed to form complete periodical or composed PCPs.

The APM supports training on either real-world or synthetic activity sequences. Input data for the model are obtained via a segmentation algorithm that identifies both the activity type and the start time of each energy consumption event. Once the APM provides the predicted timestamps, the corresponding single or programmatic patterns are inserted at those points, ultimately yielding realistic periodical or composed PCPs for the target appliance.

Data generation

Below, we provide a more detailed account of how synthetic profiles are created for a periodical appliance (a robotic vacuum cleaner) and a composed-pattern appliance (a filter coffee machine).

Consistent with the earlier explanation, producing these PCP categories begins with decomposing the original appliance consumption traces into their basic building blocks—the individual single patterns—via the segmentation procedures. For the robotic vacuum cleaner, this yields a collection of repeated charging cycles; for the coffee machine, it produces distinct brewing segments and multiple heating segments. Applying these methods to the selected 14-day portion of the GeLaP dataset yielded 264 charging patterns for the vacuum cleaner robot and 84 full profiles for the coffee machine. Within those 84 coffee machine profiles, segmentation identified 84 brewing instances and 2986 heating instances, corresponding to an average of roughly 35.5 heating cycles per brewing event.

Beyond isolating the patterns, the approach also captures the exact onset times of every single pattern relative to the start of its parent profile. These are encoded as structured triples containing: (i) a sequence index to link the pattern to its overall profile, (ii) the pattern's activity label, and (iii) the elapsed time (in seconds) from the profile's beginning. An illustrative sequence for the coffee machine, therefore, looks like this:

- (0, "brewing", 0)
- (0, "heating", 388)
- (0, "heating", 523)
- (0, "heating", 654)
- ...

Such abstracted timelines form the training corpus for the Activity Prediction Model. The Activity LSTM component learns to anticipate the class of the upcoming single pattern, whereas the Time LSTM estimates the appropriate insertion timestamp. Training for both appliance types involved 20 epochs for the Activity LSTM and 50 epochs for the Time LSTM.

With timestamp and type predictions available from the APM, the next step is to synthesize the required single-pattern waveforms. Since cWGAN-GP requires substantially longer training and greater computational effort to achieve satisfactory quality compared to IGen, we relied exclusively on IGen for waveform generation: charging patterns were trained for 20000 epochs, brewing patterns for 40000 epochs, and heating patterns for 20000 epochs. The length-adaptive design of IGen ensures it can handle the differing durations of these patterns without issue. Collectively, this produces three queryable Virtual Appliances, each specialized in delivering synthetic versions of one specific pattern type.

Equipped with the trained APM and the three Virtual Appliances, we can now construct complete synthetic consumption profiles for the robotic vacuum cleaner (periodical behavior) and the filter coffee machine (composed behavior). This is accomplished through a dedicated orchestration layer that consults the APM to decide the sequence and timing of pattern insertions. Whenever the APM indicates that a given pattern type should appear at a particular moment, the orchestrator retrieves a matching synthetic waveform from the appropriate Virtual Appliance and places it accordingly, thereby assembling realistic full-length PCPs for either the periodical or the composed appliance category.

The overall pipeline—including the interplay between the APM, the IGen-based Virtual Appliances, and the orchestration process—is schematically depicted in **Figure 9**, which also shows an example synthetic coffee machine profile assembled in this way.

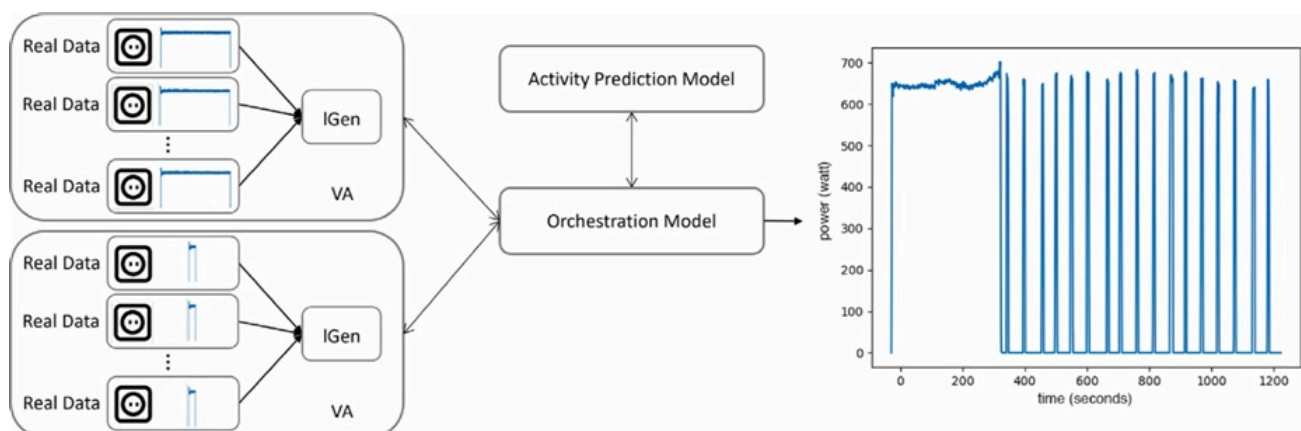


Figure 9. Schematic view of the orchestration model to create a synthetic coffee machine PCP example generated by the Activity Prediction Model and the IGen model of the brewing and the heating plate patterns.

Evaluation

Assessing generated periodic and composed PCPs directly with DTW poses challenges, as intentional variability is introduced both within individual patterns and in the timing predictions provided by the APM. Over the extended duration typical of periodical or composed profiles, this deliberate variation can accumulate dramatically, leading to excessively high DTW values that offer little informative value.

Nevertheless, we can still apply the DTW metric and the Classifier model effectively to the constituent single and programmatic patterns that form these larger PCPs. In other words, the individual building-block patterns are evaluated following the same procedures. In addition, we apply the Classifier model to the entire periodical and to the composed PCPs themselves. Finally, visual inspection of plotted examples provides a qualitative assessment of the synthetic profiles.

For this evaluation, we created synthetic datasets using the IGen model specifically for the charging pattern of the robotic vacuum cleaner (GeLaP dataset, hh14, label_008) and the brewing and heating patterns of the coffee machine (GeLaP dataset, hh14, label_003). Separately, we generated full synthetic datasets for both the robotic vacuum cleaner and the coffee machine by combining the APM with IGen. Each real and corresponding synthetic dataset contains the same number of PCPs, with each unique label represented exactly once. Consistent with the approach used for single and programmatic patterns, we split each dataset into a training portion (the first x PCPs) and a test portion (the remaining PCPs), as summarized in **Table 3**. This yields paired real and synthetic datasets for each single-pattern type and for the periodical and composed appliance categories under study. All datasets were split equally into training and test sets, and no samples were altered during evaluation to maintain strict comparability across pairs.

For each appliance, a dedicated Classifier model was trained on a balanced combination of real and synthetic data from that appliance.

Table 3. Partitioning of the real and the synthetic datasets for the periodical and composed pattern type appliances.

Appliance	Total	Training	Test
Vacuum cleaner robot	14	12	12
Charging	264	132	132
Coffee machine	84	42	42
Brewing	84	42	42
Heating	2986	1493	1493

The sections that follow present the Classifier outcomes and DTW distance analyses for the robotic vacuum cleaner and for the coffee machine profiles.

Periodical pattern type appliance

We examine the robotic vacuum cleaner as a representative of the periodic pattern category and assess the fidelity of the patterns it generates. To judge the quality of a complete synthetic vacuum cleaner PCP, we begin by scrutinizing the individual charging patterns that constitute it.

The Classifier scores for the real test data exhibit a distribution that closely matches the scores obtained from the synthetic charging patterns produced by IGen. The mean scores across the two test sets are the same (real: 0.524, synthetic: 0.524), indicating that the Classifier cannot differentiate between the real and synthetic charging datasets.

Examination of the DTW distance distributions—real/real, synth/synth, and synth/real—for the IGen-generated vacuum cleaner charging data reveals strong alignment between the synthetic and real distributions. In the direct 1vs1 comparison of synthetic PCPs against real PCPs sharing the same labels, the overwhelming majority of DTW distances are very small, confirming high overall similarity.

Having confirmed the high quality of the isolated charging patterns, we next evaluate the full synthetic PCP of the robotic vacuum cleaner, assembled using both IGen and the APM. Here too, the Classifier results are encouraging: the distribution closely resembles that of the real full profiles, with average scores centered around 0.5 (real: 0.432, synthetic: 0.432). This outcome demonstrates that the Classifier is unable to reliably distinguish synthetic periodic PCPs from their real counterparts.

Composed pattern type appliance

We now assess the filter coffee machine as a representative example of the composed pattern category, focusing on the quality of its synthesized patterns. To judge the overall fidelity of a complete synthetic coffee machine PCP, we first examine the individual brewing and heating components that together form the full profile.

For the brewing process patterns generated by IGen, the Classifier scores on the synthetic test set cluster around 0.5; the same holds for the heating period patterns. This central tendency around 0.5 indicates that the Classifier remains uncertain about whether a given PCP is real or artificial. When these results are compared with the histograms from the corresponding real test data—for brewing and for heating—the distributions appear nearly identical in both cases. The close match is further confirmed by the almost equal average scores across the paired test sets (brewing real: 0.525, brewing synthetic: 0.525; heating real: 0.506, heating synthetic: 0.506). Consequently, the Classifier proves unable to differentiate between real and IGen-synthesized brewing patterns, or between real and synthetic heating patterns.

Regarding the DTW distance distributions for both brewing and heating data produced by IGen, the real/real pairs, synth/synth pairs, and synth/real pairs all exhibit strong congruence between synthetic and real profiles. In the direct 1vs1 matching of synthetic PCPs against real ones sharing identical labels, the DTW distances remain consistently very small for both brewing and heating, reinforcing the impression of high similarity. Having established the strong quality of the isolated brewing and heating patterns, we proceed to evaluate the complete coffee machine PCP assembled using IGen in combination with the APM. Here again, the Classifier's outputs on the synthetic test set hover around 0.5, reflecting the model's uncertainty in classifying data as real or synthetic. The histogram of the real test data displays a distribution that closely parallels the synthetic one, a similarity underscored by nearly identical mean scores (real: 0.493; synthetic: 0.494). Thus, even at the level of the full composed profile, the Classifier fails to separate the synthetic from the real dataset.

Duration-dependent and constantly-on pattern generation

This section explores in greater depth the synthesis of duration-dependent and constantly-on pattern appliances. Duration-dependent appliances require explicit control over pattern length because we aim to produce PCPs whose number of timesteps—and hence overall duration—varies and may differ from lengths observed in the training data, while still maintaining realism. Achieving this flexibility calls for approaches distinct from those used by IGen or cWGAN-GP. To address this requirement, we therefore developed two alternative iterative methods: an MLP-based generator and an Extreme Gradient Boosting (XGBoost) algorithm [50].

Appliances exhibiting constantly-on behavior can be treated analogously to duration-dependent ones, since their active periods could—in principle—extend indefinitely. Accordingly, we apply the same generation techniques developed for duration-dependent patterns to this category as well.

Importantly, the methods introduced here for duration-dependent and continuous appliance use also support the creation of synthetic PCPs with arbitrary durations for single-pattern appliances (e.g., a toaster or kettle). A key benefit over the previously presented models is their ability to generate profile lengths that do not appear in either the training or test datasets.

In the sections that follow, we demonstrate and assess the application of these approaches by generating and evaluating examples of: a duration-dependent appliance (TV), a duration-dependent single-pattern appliance (kettle), and a constantly-on appliance (router).

Generation architecture

For the creation of duration-dependent patterns, we adopt an iterative generation strategy that closely resembles the n-gram-style mechanism employed by the APM. In this process, each new timestep of a PCP is predicted based on the preceding n timesteps, an injected noise component, and both the absolute total duration and the relative position (expressed in timesteps) within the target PCP length. Once generated, this new timestep is incorporated—along with the

other inputs—into the feature set for the subsequent prediction step, and the process continues iteratively until the desired overall length is achieved (**Figure 10**).

To introduce controlled diversity and prevent repetitive outputs, we supply a noise term as an extra input feature. Additionally, by including both the total duration and the relative position of the current timestep within the full pattern, the model learns to adapt its predictions to the location of each value in the sequence. For the PCPs generated in our examples, we set $n = 10$ (using the last 10 timesteps), applied uniform random noise in the range $[0, 1]$, normalized the total and relative durations to $[0, 1]$ via MinMax scaling, and similarly scaled the power (watt) values using the MinMax procedure to mitigate the bimodal nature of typical PCP distributions.

This iterative method enables the synthesis of PCPs with arbitrary lengths—either shorter or longer than those observed in the training set—or with lengths identical to the training examples when the target number of timesteps is deliberately matched to the training distribution (analogous to the fixed-length approach used for single and programmatic patterns).

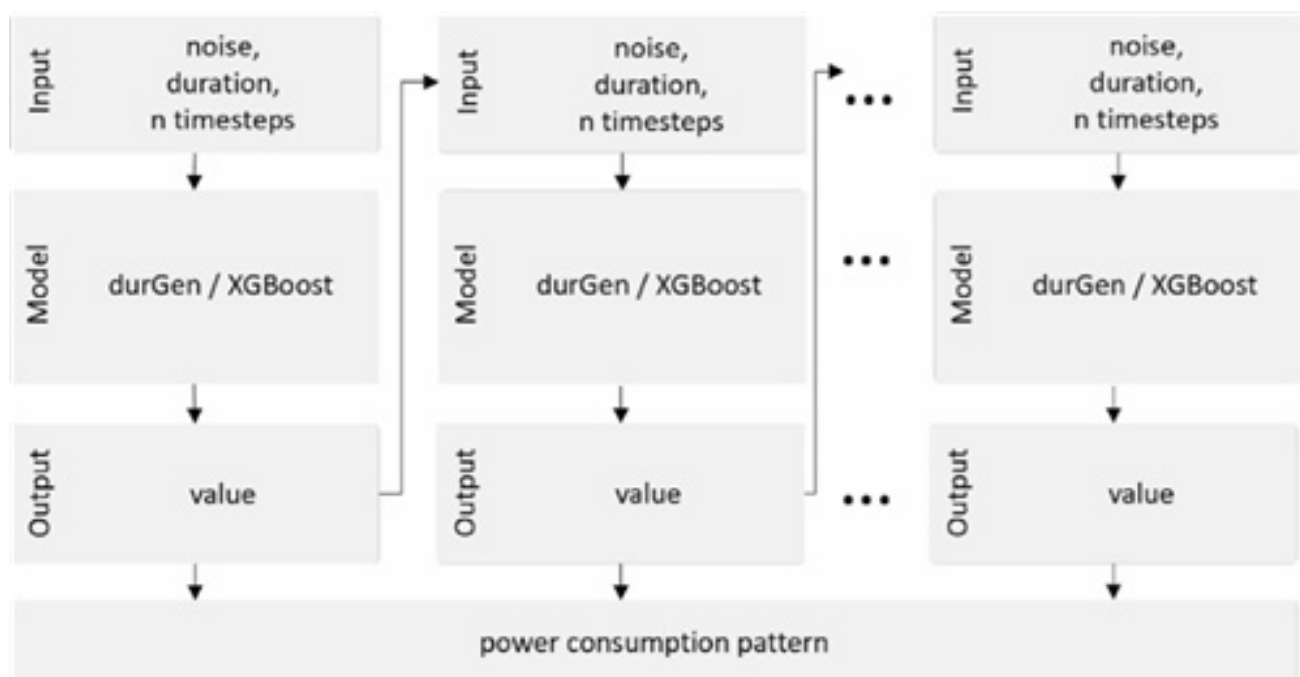


Figure 10. Generation architecture.

To carry out these sequential predictions using the described input features, we designed and implemented two distinct machine learning models: an MLP-based duration generator referred to as durGen and an iterative Extreme Gradient Boosting procedure based on XGBoost [50]. Both techniques are elaborated further in the subsections below.

Duration generator

The Duration Generator (durGen) is a flexible multilayer perceptron (MLP) whose architecture scales dynamically with the chosen number of history timesteps n . Similar to IGen, it is structured as a feed-forward neural network comprising at minimum an input layer, one or more hidden layers, and an output layer. The size and configuration of the hidden layers are adapted to the input dimensionality, which includes the n prior timesteps, the noise term, and both the total and relative durations (in timesteps) of the target PCP.

This adaptive design makes durGen suitable for appliances whose PCPs span only a few timesteps (such as a kettle) as well as those with considerably longer profiles (such as a television). Between hidden layers, we apply Leaky ReLU activation functions and omit dropout. Training is performed using the Adam optimizer with a learning rate of 0.0001 and

mean squared error as the loss function. For the television, kettle, and router examples presented here, the durGen model was trained for 1000 epochs.

Figure 11 illustrates a representative synthetic television PCP produced by durGen (shown in blue), which contains roughly 10% more timesteps than a comparable real profile (shown in orange).

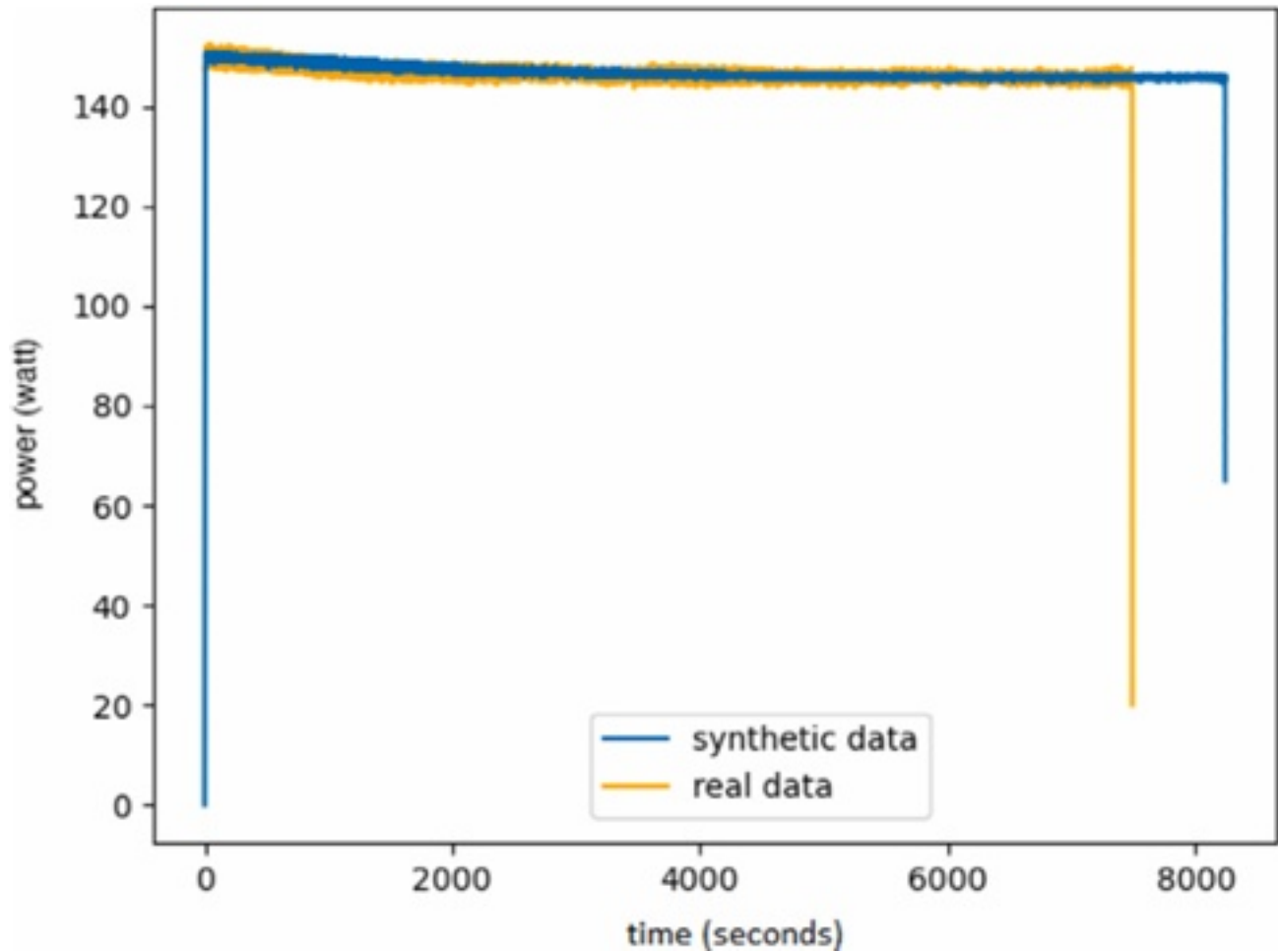


Figure 11. Synthetic PCP of the television generated by the durGen model (blue color), which has approximately 10% more timestamps compared to a corresponding real PCP (orange color).

XGBoost

Extreme Gradient Boosting (XGBoost) is a widely adopted, open-source machine learning technique for tree-based boosting, available publicly on GitHub.³ It is renowned for its efficiency, strong predictive accuracy, and scalability. The method relies on ensembles of decision trees and supports both regression and classification tasks. That said, XGBoost tends to be resource-demanding in terms of computation and memory, particularly with sizable datasets.

An XGBoost model comprises an ensemble of classification and regression trees (CARTs), where the final predicted score $\hat{y}_i$ is obtained by summing the outputs of each tree [50].

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), f_k \in F \quad (4)$$

Eq. 4 expresses the prediction $\hat{y}_i$ as the additive combination of K CART functions f_k , each drawn from the functional space $\mathcal{F}$ (the collection of all valid regression trees).

As with any supervised learning framework, an objective function, obj , must be minimized (**Eq. 5**).

$$obj = \sum_i^n l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^K w(f_k) \quad (5)$$

Here, l denotes a convex, differentiable loss function that quantifies the discrepancy between the predicted value $\hat{y}_i$ and the true target y_i , while $w(f_k)$ serves as a regularization term that penalizes model complexity.

$$w(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2 \quad (6)$$

In this term, T is the number of leaf nodes in the tree, and w represents the leaf weight vector [50].

In our implementation, we utilize the standard XGBoost Python library. The same hyperparameter configuration was applied across the television, kettle, and router experiments to underscore the method's robustness and ease of transfer. Specifically, we configured a maximum tree depth of 20 (to accommodate greater model expressiveness), a subsample ratio of 0.75 (to mitigate overfitting risk), and 300 boosting rounds, using squared error as the learning objective.

Figure 12 presents an example synthetic PCP of a duration-dependent kettle profile produced by the XGBoost model (blue trace), which contains roughly 15% more timesteps than a matching real profile (orange trace).

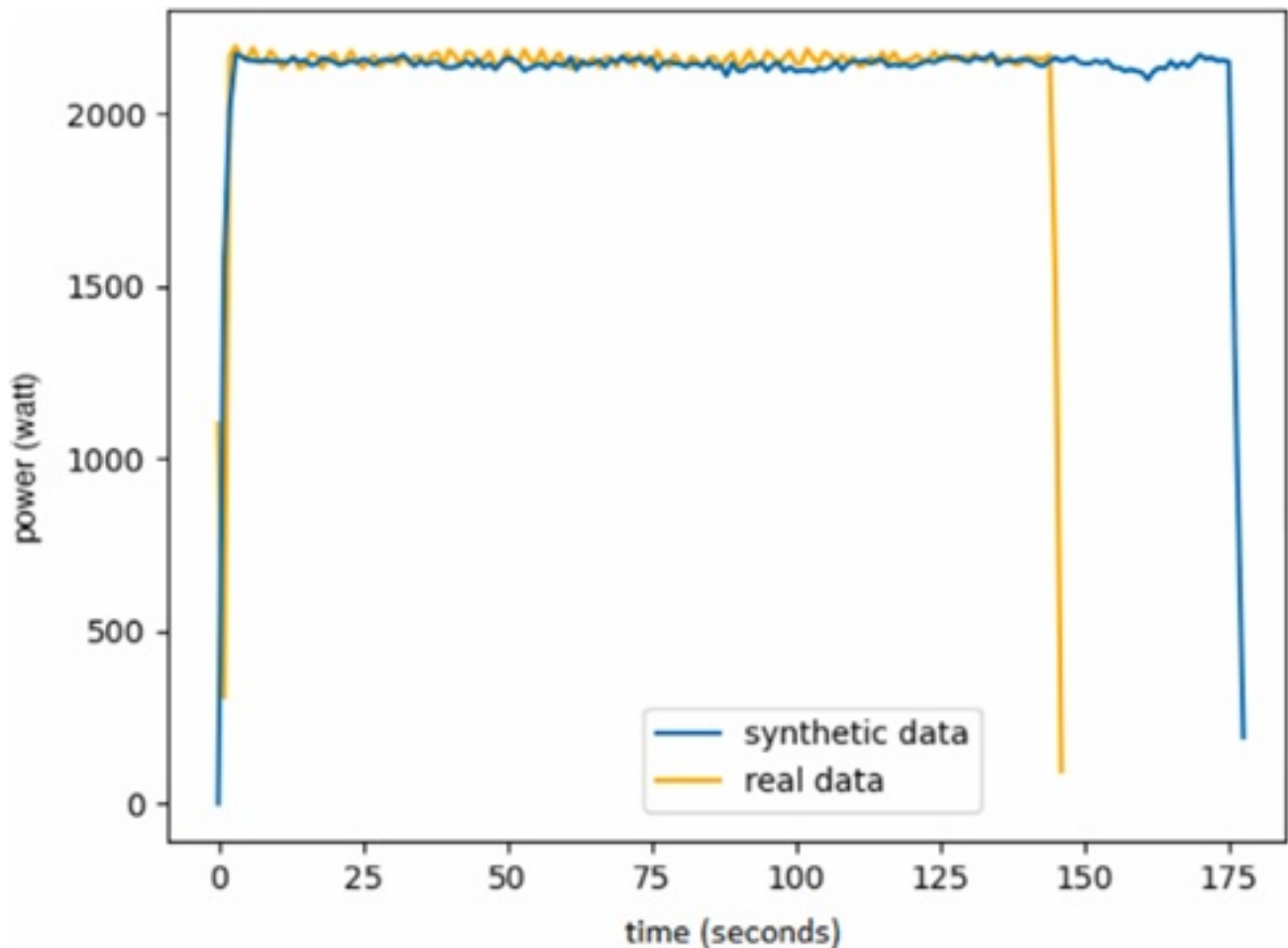


Figure 12. Synthetic PCP of the duration-dependent kettle generated by the XGBoost model (blue color), which has approximately 15% more timestamps compared to a corresponding real PCP (orange color).

Evaluation

When synthesizing duration-dependent PCPs whose length (i.e., number of timesteps) does not match any profile present in the training data, direct application of DTW becomes impractical, as there are no corresponding real PCPs available for pairwise comparison. This situation parallels the challenges encountered with periodical and composed patterns, where intentional deviations from training-set lengths are expected. Since the generated PCPs invariably differ in the number of timesteps—and therefore in overall duration—from the training examples, DTW distances would become excessively large and largely uninformative.

Instead, we rely on the Classifier model developed earlier to assess whether it can reliably separate synthetic PCPs from real ones. Complementing this quantitative evaluation, we perform visual inspections of selected synthetic examples to confirm that the generated profiles appear plausible and realistic. The same evaluation strategy applies to constantly-on appliances, which—by nature—may also exhibit durations absent from the training set and are therefore synthesized using identical iterative techniques.

As a representative duration-dependent appliance, we generated synthetic television profiles. However, the hh14 television data proved unsuitable for training, containing only 6 segmented PCPs—an insufficient sample size. We therefore selected the television from hh04 (GeLaP, hh04, label_007), which provided 16 segmented PCPs (Table 4). To illustrate duration-dependent synthesis applied to a single-pattern appliance, we reused the previously employed kettle

data (GeLaP, hh14, label_004). In both cases, the synthetic PCPs were deliberately varied by 5–20% in length relative to their real counterparts for evaluation.

Table 4. Partitioning of the real and synthetic datasets for duration-dependent and constantly-on pattern-type appliances.

Appliance	Total	Training	Test
Television	16	8	8
Kettle	115	55	60
Router	14	7	7

We additionally synthesized router profiles as an exemplar of constantly-on behavior. Since no suitable constantly-on appliance could be identified within the GeLaP dataset, we turned to the Refit dataset [15] and selected the router from House_11, Appliance_8. To prepare the data, appliance-level Refit recordings were resampled to 1 Hz and segmented daily. Router power consumption remains highly stable, fluctuating narrowly between 10 and 12 watts. For evaluation, we generated one full day of synthetic router PCPs.

Duration-dependent pattern type appliance I

We start with the television as our initial case study for duration-dependent appliances and examine how faithfully the synthetic patterns reproduce the originals.

Looking at the Classifier performance on the synthetic television test profiles from durGen and from XGBoost, the scores in both cases cluster tightly around 0.5. This midpoint concentration indicates that the Classifier struggles to decide whether a profile is real or generated. When juxtaposed with the histograms from the real test sets, the synthetic distributions appear almost indistinguishable. The equivalence is underscored by the nearly matching mean scores (durGen real: 0.487, durGen synthetic: 0.488; XGBoost real: 0.525, XGBoost synthetic: 0.525). Overall, these results indicate that the Classifier cannot meaningfully distinguish synthetic television data from authentic recordings.

On a qualitative level, side-by-side visual checks show that the television PCPs generated by durGen and XGBoost both look convincingly natural and believable.

Duration-dependent pattern type appliance II

Next, we turn to the kettle—treated here as a single-pattern, duration-variable appliance—and assess the realism of the profiles it generates.

Classifier outputs for the synthetic kettle test data created with durGen and with XGBoost likewise settle predominantly near 0.5, reflecting ongoing indecision about the origin (real versus synthetic) of each PCP. The histograms derived from the real kettle test data exhibit patterns that are virtually identical to those of the synthetic counterparts. This strong resemblance is quantitatively supported by the almost equal average values (durGen real: 0.512, durGen synthetic: 0.513; XGBoost real: 0.512, XGBoost synthetic: 0.512). In short, the Classifier proves ineffective at distinguishing the real kettle profiles from those synthesized by either method.

Mirroring the television outcome, visual evaluation confirms that the kettle PCPs from both durGen and XGBoost maintain a high degree of realism.

Constantly-on pattern type appliance

We conclude this evaluation series with the router, chosen as a prototypical constantly-on device, and review the quality of its synthetic versions.

For the router profiles generated by durGen and by XGBoost, the Classifier assigns scores that remain centered around 0.5, further demonstrating uncertainty in classifying the data as authentic or fabricated. The real test data histograms closely match the synthetic distributions. The close match is further evidenced by the comparable mean scores (durGen real: 0.515, durGen synthetic: 0.514; XGBoost real: 0.518, XGBoost synthetic: 0.517). Therefore, the Classifier is unable to reliably distinguish synthetic router PCPs from genuine ones.

Summary

In this section, we presented generation techniques for all appliance pattern types typically found in a household. We created representative appliance-level data for each pattern type and assessed their quality. The findings indicate that synthetic PCPs cannot be differentiated from real ones either visually or using the established performance metrics. Consequently, the proposed methods enable the creation of a Virtual Appliance for every pattern type. As the Virtual Appliances are designed to produce aggregated synthetic power consumption data suitable for training machine learning models, we further investigate this application in the sections that follow.

NILM Transferability Evaluation

To determine whether Virtual Appliances can produce aggregated power consumption data suitable for training ML models, we adopt the evaluation methodology introduced in [17]. This approach empirically assesses and compares the quality of various datasets using NILM model performance. For this purpose, we employ SynTiSeD, a simulation tool that generates synthetic data from Virtual Appliances and real appliance-level measurements. The Virtual Appliances (VAs) are constructed using data from multiple households in the GeLaP dataset. Both the GeLaP data and the resulting VAs are then utilized to create several synthetic datasets. Five different NILM model architectures (sequence2point, sequence2sequence, DAE, RNN, and windowGRU) are trained on these synthetic datasets, and their performance is compared both among themselves and against models trained on real data. This thorough evaluation focuses on a single pattern-type appliance (kettle) and a single programmatic pattern-type appliance (washing machine), as these are the most prevalent appliances in typical households.

Synthetic data generation

The following section provides a more detailed description of how the various synthetic datasets were produced.

Data preparation

For the NILM evaluation, both the real appliance-level data and the data used to build the Virtual Appliances are taken from the publicly available GeLaP dataset [14]. This dataset was selected because it provides appliance-level ground truth data alongside high-resolution smart meter data (1 Hz), which enables the creation of precise Virtual Appliances. The GeLaP dataset includes 20 households, each with smart meter data and ground-truth measurements for 10 different appliances.

Given the scope of this study, we focus on generating one Virtual Appliance for a single-pattern-type appliance (kettle) and one for a programmatic-pattern-type appliance (washing machine), both of which have sufficient PCPs available. As noted in [17], the GeLaP dataset contains 7 usable households for the kettle and 11 usable households for the washing machine. We synthesized a total of 5 households (hh01, hh04, hh07, hh14, hh15), with four synthetic households containing a kettle and four containing a washing machine; these serve as training and validation data for the NILM evaluation. For all selected households, the smart meter data and appliance-level data were resampled to 1-second intervals (using the mean) and forward-filled.

To construct a Virtual Appliance, we pooled the segmented PCPs from the synthesized households (kettle: hh01, hh04, hh07, hh14; washing machine: hh01, hh04, hh14, hh15) and trained a unified model using IGen. In total, 247 PCPs were used for the kettle Virtual Appliance and 45 PCPs for the washing machine Virtual Appliance. The generated synthetic

PCPs were subsequently validated using the Classifier model. Training a Virtual Appliance on data from multiple households in the GeLaP dataset ensures that the resulting PCPs naturally reflect the characteristics of all households included in the training data. The households that were not synthesized are reserved as test households for the NILM evaluation (kettle test households: hh08, hh11, hh17; washing machine test households: hh03, hh05, hh06, hh09, hh13, hh19, hh20).

To produce synthetic datasets that effectively support the training of NILM models capable of generalizing across different scenarios, incorporating unassigned power consumption (UPC) data is indispensable; without it, the modeled energy patterns become unrealistically simplistic. UPC represents the residual power draw in a household for which appliance-specific ground truth is unavailable. It is computed for each household by deducting the combined consumption of all appliances with known ground truth from the total household smart meter readings (Eq. 7).

$$upc(t) = pc(t) - \sum_{i=1}^{N_{ha}} ap_i(t) \quad (7)$$

At any timestamp t , the UPC value (i) equals the household's overall power draw $p(t)$ minus the total consumption summed over all N_{ka} appliances with available ground truth ap .

SynTiSeD

SynTiSeD is a multi-agent-based simulation platform that synthesizes realistic smart meter readings by either using Virtual Appliances or authentic appliance-level recordings [13]. Users can define a set of households (hh) denoted as $H = h1, h2, \dots, hj$ and equip them with selected appliances $A = a1, a2, \dots, al$. Resident agents $R = r1, r2, \dots, rk$ then perform actions inside these virtual homes, engaging with appliances according to specified sequences. Each activation causes the corresponding appliance to draw power, and the collective load across all appliances within a household yields the generated synthetic aggregate smart meter trace.

During the PCP extraction and segmentation step for an appliance, the precise occurrence times of each PCP within the ground truth record can be captured. These timestamps are subsequently employed to build action sequences as that emulate genuine occupant usage patterns. That said, the framework also permits users to manually design entirely custom action sequences without relying on empirical data.

The operational logic of SynTiSeD is summarized in pseudocode Algorithm 1. Users begin by providing the households $hj \in H$, residents $rk \in R$, appliances $al \in A$, the desired simulation duration in days Nd , and the target time resolution res . The engine then steps through every timestamp across the simulated period, evaluating at each step whether any resident triggers an appliance. This mechanism covers human-initiated pattern categories, such as single-event, programmatic, composite, and duration-based. Once an activation occurs, SynTiSeD either pulls a pre-stored PCP from the appliance's repository or sends a request to the associated Virtual Appliance through a REST endpoint. The VA supplies an appropriate PCP, which is then incorporated into the appliance's ongoing consumption profile. SynTiSeD also accommodates always-present or cyclically operating appliances (constant and periodic pattern classes) by periodically sampling their PCPs from the linked VAs using the same procedure. In the end, the household's total aggregated power consumption apc emerges from the combined contributions of human-triggered appliances, permanent/background appliances, and the UPC component.

```

Input : households  $H$ , residents  $R$ , appliances  $A$ , number of days  $N_d$ , resolution  $res$ 
Output: aggregated power consumption  $apc$ , action sequences  $as$ , active phases  $ap$ 
1 Function SIMULATE( $H, R, A, N_d, res$ ):
2    $H \leftarrow h$ 
3    $R \leftarrow r$ 
4    $A \leftarrow a$ 
5    $N_d \leftarrow n$ 
6    $N_t \leftarrow 86400/res$  //  $N_t$  := number of timestamps
7   for  $d \leftarrow 1$  to  $N_d$  do
8     for  $t \leftarrow 1$  to  $N_t$  do
9       for  $r$  in  $R$  do
10        if  $r_a$  and  $a \in A$  then //  $r_a$  := action of a resident
11          activate appliance
12        end if
13      end for
14      for  $pc(a)$  in  $A$  do //  $pc_a$  := power cons. of appl.
15         $apc \leftarrow apc + p(a)$ 
16      end for
17    end for
18  end for
19  return  $apc, as, ap$ 
    
```

Algorithm 1. SynTiSeD workflow

Synthetic dataset generation

One key aim of the present research is to verify whether Virtual Appliances together with synthetically generated PCPs can yield synthetic load traces suitable for training NILM disaggregation models that demonstrate strong transferability to unseen data. To explore this question, three distinct synthetic datasets were created using SynTiSeD and the earlier developed Virtual Appliances corresponding to the single-pattern appliance (kettle) and the programmatic-pattern appliance (washing machine).

The initial dataset—labeled noSPCP (no synthetic power consumption patterns)—was constructed to function as the reference synthetic baseline. For each household included, the simulation spans the same number of days and uses the same 1 Hz sampling frequency as the original real-world recordings. Furthermore, resident action sequences are copied directly from the real dataset so that every known appliance activates at precisely the moments observed in the actual traces. The PCPs assigned during these activations are randomly sampled from the genuine PCP collection of that appliance, thereby ensuring that the full diversity of real consumption patterns is preserved in the synthetic version. On top of that, unassigned power consumption (UPC) is included to account for the contribution of appliances lacking labeled measurements. UPC is derived daily by subtracting the sum of the ground-truth consumption for all labeled appliances from the household's corresponding smart meter series. Because slight temporal misalignments between the ground truth and aggregate meter recordings can arise from differing sensor characteristics, the computed UPC may sometimes take on negative values; following established practice, all such implausible negative entries are clamped to zero [17].

The second dataset, referred to hereafter as SPCP (synthetic power consumption patterns), builds directly upon the noSPCP dataset. Unlike in the noSPCP case, the PCPs for both the kettle and the washing machine are not derived from real measurements; instead, they are synthetically generated using the IGen method. To achieve this, all available PCPs of the single-pattern-type appliance (kettle) and the programmatic-pattern-type appliance (washing machine) from the training households were pooled as input for IGen, resulting in the construction of corresponding Virtual Appliances. During the simulation, whenever one of these appliances was triggered, the associated VA was queried via its interface, and a newly generated PCP was added to the appliance's simulated consumption profile. Because each VA had been trained on the complete set of PCPs from all relevant training households, it is possible—and indeed expected—that a PCP originally belonging to one household may appear in the simulation of a different synthetic household.

This SPCP dataset is specifically designed to test whether machine learning models gain greater robustness and transferability when trained and validated on synthetic data that includes artificially created PCPs. The underlying hypothesis is that NILM model performance will benefit from the added variance in PCP shapes within the aggregate smart meter traces, as this increased diversity makes the disaggregation task more demanding and thus encourages the development of more generalizable algorithms.

The third dataset, labeled hereafter as augm. (augmented data), is created by combining elements of the noSPCP and SPCP datasets. As a result, the augm. dataset contains twice as many data points as either noSPCP or SPCP alone, and it incorporates both the realistic PCPs from noSPCP traces and the synthetic PCPs generated for SPCP traces. Given that a major objective of this study is to leverage synthetic PCP generation to expand the amount of training data available for building more robust ML models, this augmented dataset serves to investigate whether increasing data quantity and diversity leads to improved transferability in NILM models. The working hypothesis is that NILM performance will be enhanced when models are trained on a broader, more varied set of PCP instances.

Experimental environment

After employing Virtual Appliances to produce a variety of synthetic datasets, we now provide a detailed account of the experimental framework used to assess and compare them. We begin by presenting the five NILM disaggregation algorithms selected for evaluation. Next, we outline the established evaluation protocol for measuring synthetic data quality via transferable NILM performance, which closely follows the approach originally proposed in [17]. Finally, we specify the concrete experimental configuration employed for our tests.

NILM disaggregation algorithms

The assessment of the generated aggregate power consumption traces follows the benchmarking methodology introduced in [17]; accordingly, we adopt the same five widely recognized state-of-the-art NILM algorithms used in transferability studies. All implementations are sourced from the publicly accessible NILMTK-contrib repository [51]. To enable NILMTK-contrib to process the synthetic datasets, they are first converted into the required HDF5 (.h5) format. Below is a concise overview of the five chosen algorithms.

One prominent algorithm optimized for transferability is the sequence-to-point (seq2point) approach [52]. The seq2point implementation in NILMTK employs a neural network that takes sliding windows of aggregate smart meter readings as input and predicts the ground-truth power demand of the target appliance, outputting specifically the value at the center of each input window [53].

We also include the sequence-to-sequence (seq2seq) model, which operates comparably. This neural architecture learns a regression mapping from the full aggregate power sequence to the corresponding target appliance power sequence over the same time span [53].

Another model under consideration is the Denoising Autoencoder (DAE), a specialized deep learning architecture [54]. The DAE treats the household's aggregate consumption as a noisy version of the individual appliance signal and is trained to reconstruct a cleaner estimate of the target appliance's consumption by suppressing contributions from other devices.

We further evaluate Recurrent Neural Networks (RNNs) that use Long Short-Term Memory (LSTM) units. Thanks to their ability to retain information across extended time spans, RNNs with LSTMs are especially well-suited for processing sequential time-series data [54].

The final algorithm tested is the Window Gated Recurrent Unit (windowGRU) model, another leading method in the field. This architecture is built around Gated Recurrent Units (GRUs), and processes input windows to capture temporal dependencies [55].

For all experiments involving these NILM algorithms, we retain the default configuration parameters provided by NILMTK-contrib, including both model hyperparameters and evaluation metrics. While these defaults may not yield the absolute best possible performance for each model, optimization is not the primary objective of this study. Using standardized settings ensures greater comparability across algorithms and facilitates straightforward reproduction of the results.

Methodology

To judge how well the synthetic datasets perform, we train each of the five NILM algorithms outlined earlier using both genuine household recordings and the artificially created datasets, then measure how effectively the resulting models generalize (i.e., their transferability). The chosen performance indicator is mean absolute error (MAE). This choice is driven by the fact that NILMTK implements all evaluated disaggregation methods as regression tasks. MAE remains one of the most widely endorsed and frequently reported metrics in the NILM regression literature [11, 52-62].

Nevertheless, previous work has shown that NILM models exhibit noticeable run-to-run variability in cross-domain generalization—even with fixed hyperparameters and identical training material—owing to the inherent randomness in neural network optimization [17]. In light of this, we conduct 100 separate training runs for every model configuration and compute the average outcome. The final reported figure is therefore the mean absolute error across unseen households (MAEUH), formally defined in Eq. 8.

$$\text{MAEUH} = \frac{1}{N_{uh}} \sum_{u=1}^{N_{uh}} \left(\frac{1}{n_0} \sum_{e=1}^{n_0} \text{MAE}_e \right)_u \quad (8)$$

MAEUH is obtained by summing the MAE scores over the complete set of unseen households N_{uh} , applying Cochran's sample-size adjustment with parameter n_0 [63], dividing the adjusted sum by n_0 , and then dividing again by the total number of unseen households N_{uh} . We deliberately fix $n_0 = 100$ so that the resulting confidence interval approximates 95%.

Experiment setup

The study concentrates on two representative appliance categories: a single-pattern-type device (kettle) and a programmatic-pattern-type device (washing machine). Every NILM algorithm is trained independently on each of the synthetic datasets previously described, as well as on the authentic real-world recordings. Importantly, the test portion is kept constant throughout all trials and always comprises unseen real households extracted from the GeLaP collection. This design decision reflects the practical objective: the final disaggregation models must function reliably when confronted with true aggregate meter readings from real homes.

The largest training sets (those based on r+r and augm. configurations) supply up to 9,676,800 individual timestamps (derived from 8 households, each contributing 1,209,600 timestamps). When evaluating disaggregation performance for the programmatic-pattern-type appliance (washing machine), testing occurs over 8,467,200 timestamps (corresponding to 7 households $\times$ 1,209,600 timestamps per household).

Altogether, the experimental campaign involves training and assessing 2,500 distinct NILM models. This total stems from performing 100 independent training runs per algorithm across 5 algorithms (s2p, s2s, DAE, RNN, wGRU), 5 dataset variants (real, r+r, noSPCP, SPCP, augm.), and 2 target appliances.

All computations were carried out with the NILMTK experiment API harness on a system featuring a Tesla V100-PCIE-32 GB GPU and TensorFlow-GPU 1.15.0. For efficiency, a downsampling factor of 2 is applied to all algorithms except windowGRU, which uses a factor of 8; higher resolution for windowGRU would have led to prohibitively long runtimes and excessive resource usage.

Table 5. Kettle – performance summary (average MAE in watts, computed over 100 runs) for kettle disaggregation models trained and validated on real, synthetic, and augmented datasets, then evaluated on held-out real data.

	test hh	real	r+r	noSPCP	SPCP	augm.
S2P	hh08	39.077	35.375	29.169	26.046	26.333
	hh11	17.257	15.834	10.631	7.624	8.819
	hh17	16.731	16.517	10.188	9.770	9.787
	MAEUH	24.355	22.575	16.663	14.480	14.980
S2S	hh08	33.975	32.661	29.134	24.267	24.973
	hh11	15.689	15.310	9.748	7.186	8.254
	hh17	15.581	15.698	9.474	9.248	9.629
	MAEUH	21.748	21.223	16.119	13.567	14.285
DAE	hh08	37.811	35.637	40.746	33.787	32.844
	hh11	13.662	13.099	11.845	9.099	11.942
	hh17	16.339	16.343	13.854	12.923	13.934
	MAEUH	22.604	21.693	22.148	18.603	19.573
RNN	hh08	48.764	44.325	40.887	34.940	35.802
	hh11	20.703	19.254	12.710	9.128	10.386
	hh17	18.829	17.264	9.989	9.159	9.441
	MAEUH	29.432	26.948	21.195	17.742	18.543
wGRU	hh08	24.589	22.404	25.436	23.805	23.681
	hh11	12.216	11.355	9.752	7.335	9.097
	hh17	11.035	10.677	9.487	9.908	10.181
	MAEUH	15.947	14.812	14.892	13.683	14.319

Table 6. Washing Machine – performance summary (average MAE in watts, computed over 100 runs) for washing machine disaggregation models trained and validated on real, synthetic, and augmented datasets, then evaluated on held-out real data.

	test hh	real	r+r	noSPCP	SPCP	augm.
S2P	hh03	13.459	12.751	12.671	12.912	11.816
	hh05	23.734	23.201	21.746	24.173	23.297
	hh06	20.628	20.038	19.897	21.342	18.690
	hh09	26.759	26.690	26.779	24.539	26.880
	hh13	27.896	28.261	32.341	29.524	30.616
	hh19	39.713	40.211	35.009	31.333	32.214
	hh20	16.151	15.554	15.688	15.157	15.131
	MAEUH	24.049	23.815	23.447	22.712	22.664
S2S	hh03	18.587	16.979	19.860	21.825	17.915
	hh05	22.637	22.634	21.088	22.487	21.365
	hh06	17.630	17.978	19.316	17.654	16.926
	hh09	29.687	29.416	28.601	27.996	29.011
	hh13	28.430	27.962	33.866	26.677	31.979
	hh19	34.984	34.604	31.064	31.285	31.478
	hh20	15.051	15.128	16.333	14.750	14.968
	MAEUH	23.858	23.529	24.304	23.239	23.377
DAE	hh03	25.437	24.327	24.620	24.975	24.674
	hh05	32.878	31.467	31.279	31.129	29.799
	hh06	26.445	25.969	26.068	27.131	25.662
	hh09	25.585	25.790	24.679	24.704	25.058
	hh13	40.359	39.067	41.087	39.421	38.794
	hh19	38.644	37.453	37.654	38.240	38.254
	hh20	15.884	16.166	16.131	15.364	15.340
	MAEUH	29.319	28.606	28.788	28.709	28.226
RNN	hh03	13.410	13.434	15.394	15.292	14.430
	hh05	24.970	24.999	26.105	26.259	26.124
	hh06	22.771	22.662	23.127	23.711	23.694
	hh09	24.979	25.109	24.024	23.250	23.008

wGRU	hh13	31.618	31.681	34.040	32.672	32.478
	hh19	45.686	46.613	35.466	35.156	35.336
	hh20	14.770	14.787	15.130	14.672	14.575
	MAEUH	25.458	25.612	24.755	24.430	24.235
	hh03	22.753	17.902	23.221	21.327	17.039
	hh05	30.960	28.279	29.194	28.903	26.706
	hh06	27.561	24.238	25.496	25.017	22.992
	hh09	27.246	27.302	25.523	25.085	25.598
	hh13	38.304	35.439	38.312	37.641	33.659
	hh19	34.938	33.309	32.549	33.619	33.289
	hh20	15.558	13.451	14.829	14.618	12.132
	MAEUH	28.189	25.703	27.018	26.602	24.488

Results

We assess both the real-world dataset and the quality of our synthetic datasets by training the seq2point, seq2seq, DAE, RNN, and windowGRU NILM disaggregation models on each and analyzing their comparative performance. When comparing the outcomes of models trained on the noSPCP dataset with those trained on authentic real data, we find that for the single-pattern-type appliance, performance actually improves when the models are tested on synthetic traces (Table 5 and Figure 13).

For the programmatic-pattern-type appliance, the four algorithms—s2p, DAE, RNN, and wGRU—trained on noSPCP also outperform their real-data-trained counterparts (Table 6 and Figure 14), albeit with a smaller margin of improvement than for the single-pattern appliance. These observations align with and reinforce the findings reported in [17], which demonstrated that synthetic data can enable NILM models to surpass the performance of models trained solely on real measurements.

Further comparison between models trained on real data or noSPCP and those trained on SPCP reveals additional gains across all tested NILM algorithms and for both appliance categories. In the single-pattern-type appliance case, every NILM model trained on the SPCP dataset delivers the strongest overall results. This pattern strongly suggests that Virtual Appliances offer a valuable means of producing synthetic aggregate smart meter traces that support the effective training of high-performing NILM models.

When we contrast models trained on genuine real data with those trained on an equivalent dataset twice the real data volume (r+r), we observe modest improvements for both single- and programmatic-pattern-type appliances (except for the RNN applied to the programmatic-pattern appliance). This outcome is consistent with the widely held view that machine learning models generally benefit from larger training sets that expose them to more data points. That said, simply increasing the number of training epochs by a factor of two would produce roughly the same effect as duplicating the dataset, since the model would encounter each example twice as often.

The true strength of Virtual Appliances lies in their capacity to synthesize entirely new PCPs that never appeared in the original real measurements. This enables the creation of expanded synthetic aggregate traces that enrich the diversity of PCPs available during training. The superior performance of models trained on the augmented data. dataset provides clear support for this idea: despite having the same total number of data points as the r+r dataset, the augmented. The results are markedly better for both the single-pattern-type and programmatic-pattern-type appliances.

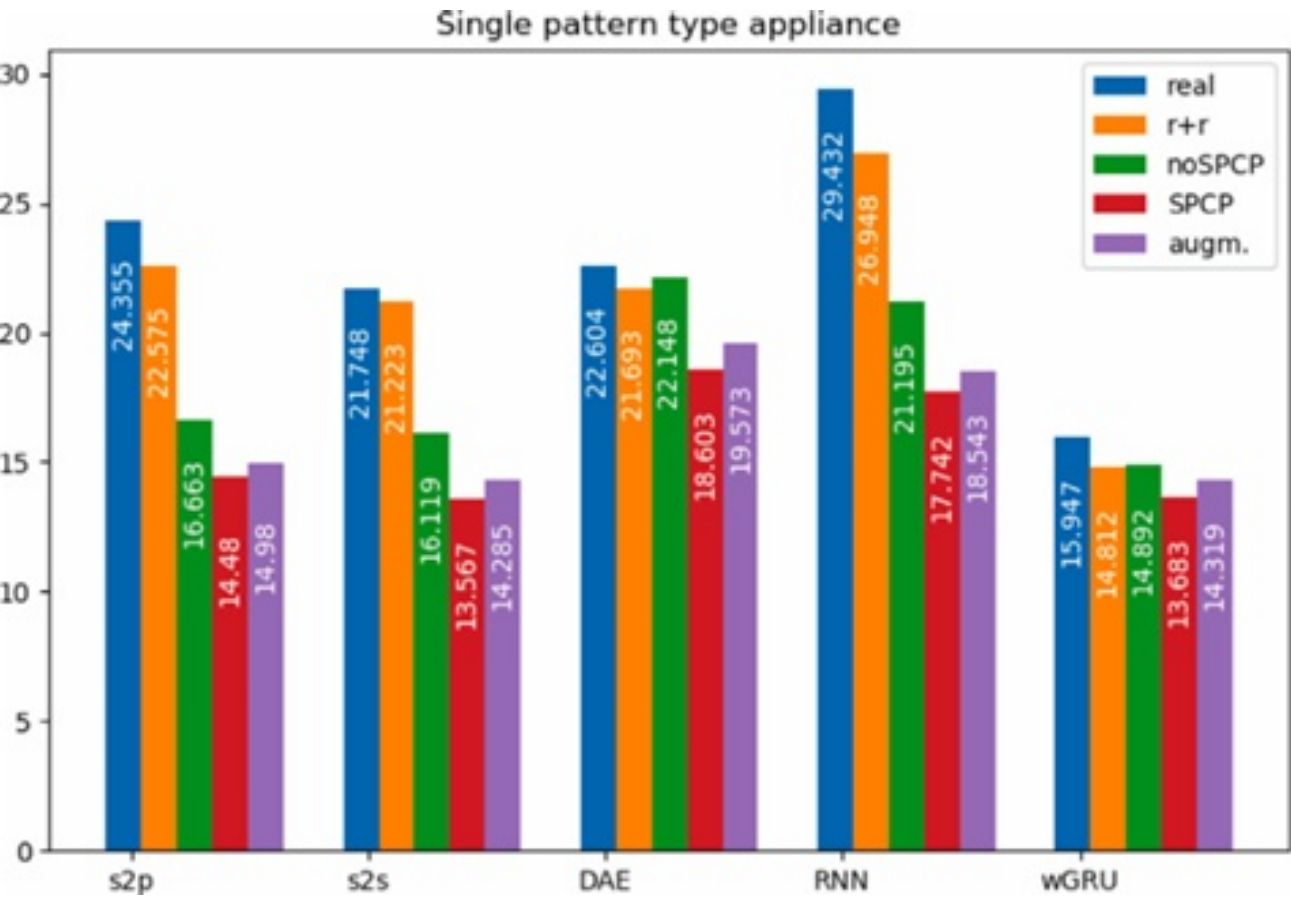


Figure 13. Comparison of MAE values across all evaluated datasets and NILM algorithms for the single-pattern-type appliance.

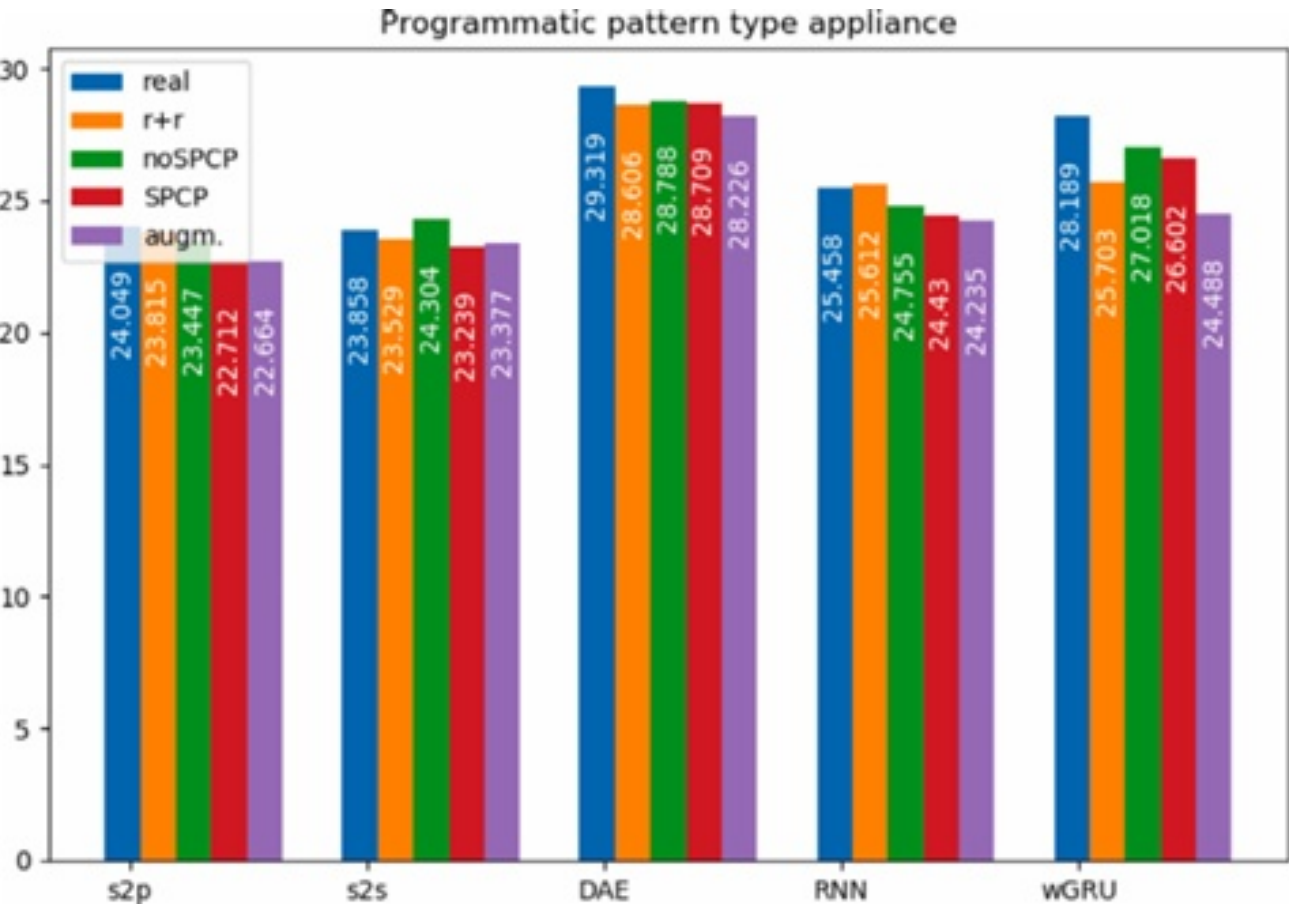


Figure 14. Comparison of MAEH values across all evaluated datasets and NILM algorithms for the programmatic-pattern-type appliance.

Practical Use

Thanks to the self-adapting nature of the PCP generation models presented earlier, it becomes feasible to establish a straightforward, user-friendly pipeline for creating new Virtual Appliances, which in turn can be used to produce realistic synthetic smart meter data (Figure 15).

Virtual Appliances can be formally described using a Web of Things (WoT) Thing Description (TD), a W3C standard. A TD provides a machine-readable, semantic specification of a thing’s metadata and interaction interfaces, enabling seamless integration of heterogeneous devices into applications. Building on this, the authors of [34] propose the Thing Registry (TR), a framework for cataloging and managing collections of TDs, enabling their metadata to be queried and accessed via RESTful services. It is therefore practical to represent the developed Virtual Appliances as TDs and register them within a TR, thereby making them discoverable and invocable. For instance, a simple GET request to the appropriate endpoint can trigger a VA to generate and return the desired number of synthetic PCPs from its stored model.

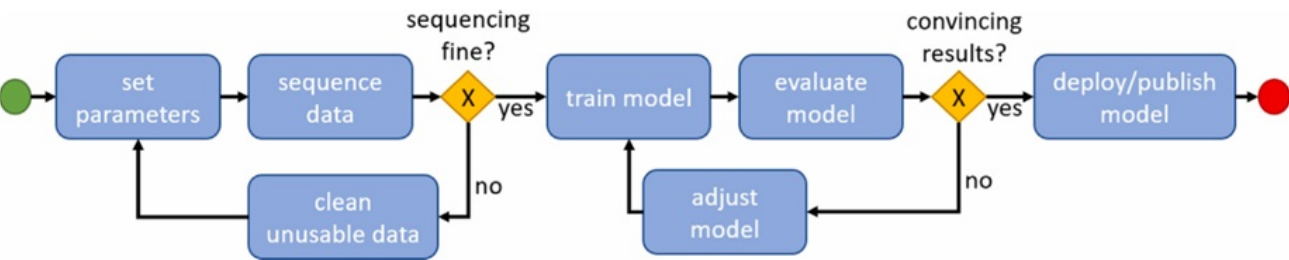


Figure 15. Simplified BPMN diagram illustrating a workflow for deploying Virtual Appliance models.

By integrating Virtual Appliances into an orchestration-capable simulation environment that aggregates smart meter readings from appliance-level inputs, it becomes possible to emulate entire households for purposes such as training and validating energy forecasting systems or human activity recognition algorithms—without deploying sensors or collecting real consumption data in those households. This capability proves particularly valuable for generating data on rare or critical scenarios that occur too infrequently in real-world recordings to adequately train data-driven machine learning models.

Moreover, the proposed Virtual Appliance approach offers substantial practical advantages in terms of cost and deployment speed. It eliminates the requirement to install dedicated power-monitoring hardware (such as smart plugs) on every appliance and then wait for an extended period to accumulate sufficient labeled data for each device.

The training and deployment of new Virtual Appliance models can be organized into a straightforward workflow. The process begins with collecting a comprehensive dataset for a given appliance, ideally containing the widest possible variety of PCPs. This dataset is then processed through segmentation and preparation steps, applying the specified parameters. Once prepared, the model can be trained in its initial phase without any modifications to its underlying architecture. Training is facilitated by an intuitive cockpit interface that enables domain experts—who may lack deep machine learning expertise—to perform the task comfortably. The specific architecture employed depends on the appliance's pattern type: single and programmatic pattern types rely on an IGen or GAN-based model; periodic and composed pattern types combine IGen with the APM model; duration-dependent and constantly-on pattern types utilize an iterative durGen model or an XGBoost-based approach.

Should the initial training outcomes prove satisfactory, they undergo evaluation using the techniques—for instance, a neural-network-based Classifier that assesses whether generated PCPs resemble known real ones. Upon successful validation, the model is deployed for integration into a simulation tool that coordinates multiple models. When the current architecture lacks sufficient adaptability, a human expert must review and fine-tune it until the synthetic outputs closely match real-world behavior. Over time, it is conceivable to implement a reinforcement learning system that observes the expert's interventions, learns from them, and gradually automates pattern recognition decisions—reducing or eventually eliminating the need for manual adjustments. Once a Virtual Appliance model delivers reliable performance, it is encapsulated in a Thing Description (TD) and registered in a publicly accessible Thing Registry (TR), making it available for interaction via REST services.

Manufacturers of household appliances could readily carry out these steps themselves, thereby offering the resulting Virtual Appliance model as an additional value-added service to their customers.

Discussion

To fully exploit the advantages of energy consumption simulations—such as modeling arbitrary combinations of appliances and synthesizing data from infrequent but critical events—bottom-up simulation strategies are essential. However, bottom-up approaches demand large volumes of appliance-level ground truth data, which are expensive, time-consuming, and often impractical to acquire in real-world settings. To overcome this barrier, we introduce the VA-Creator framework: a practical system that empowers users to construct Virtual Appliances capable of producing diverse synthetic PCPs. These PCPs can then be orchestrated to form realistic virtual smart meter traces suitable for training and validating machine learning models.

As evidenced by our evaluation, the proposed methods successfully generate synthetic PCPs for all major appliance categories typically present in residential settings. Visually, these synthetic PCPs are indistinguishable from real ones to the human observer—an impression corroborated quantitatively by the DTW distance metric and by classification

performance using an NN-based classifier model. Here, we define “realistic” as the inability of either a human inspector or an automated classifier to reliably differentiate synthetic from genuine PCPs.

For single-pattern-type and programmatic-pattern-type Virtual Appliances, we recommend the IGen model. It trains and generates faster, consumes fewer computational resources than the evaluated GAN alternative, and produces synthetic appliance-level PCPs that support meaningful training of downstream ML models. Moreover, IGen exhibits strong generalizability with only minor adaptations, making it broadly applicable across appliances. We therefore also employed IGen (in combination with APM) for periodic and composed pattern types. Among the pattern categories examined, single and programmatic types incur the lowest computational cost when generated with IGen. This cost is heavily influenced by PCP length—that is, the number of timestamps during which an appliance draws power. A typical kettle PCP spans far fewer timestamps than a washing machine PCP, resulting in a smaller network for the kettle and correspondingly lower training and inference times. For periodic and composed types, the additional overhead of the APM component must be factored in; resource demands then depend on both the overall simulation duration and the length of individual energy-consumption phases.

Duration-dependent and constantly-on pattern types, due to their iterative generation process, demand the highest computational resources. For these categories, we advocate the XGBoost approach over the tested durGen model, as XGBoost achieved convincing results with significantly shorter training times and offers a highly flexible architecture that generalizes readily to different appliances.

Across all pattern types, the appliance-level evaluation confirms that the presented generation models produce synthetic data statistically and visually comparable to real measurements. Further performance gains for individual models would likely be achievable with access to larger appliance-level datasets for training. This assumption, however, remains to be validated through additional targeted experiments.

We also performed the NILM evaluation to determine the extent to which Virtual Appliances and their synthetic PCPs can produce synthetic aggregate smart meter traces suitable for effectively training NILM disaggregation models. The findings clearly indicate that synthetic data created with the evaluated VAs is both competitive and practically useful. Remarkably, NILM models trained exclusively on this synthetic data achieve substantially better performance than those trained on real measurements—and they also surpass models trained on equivalent volumes of synthetic data that lack VA-generated PCPs (i.e., data without synthetic power consumption patterns). One plausible explanation for these strong results is the inherent variability we introduce: each generated PCP is never an exact duplicate of any previous one; instead, every instance exhibits slight, natural differences, thereby ensuring a consistent level of diversity in the training material.

Even when we combine synthetic data produced without VAs with VA-generated synthetic PCPs to form the augmented (augm.) dataset—resulting in a larger overall training volume—the NILM models trained on augmoutperform those trained on non-VA synthetic data (noSPCP dataset) as well as those trained on genuine real data (real dataset). The superior outcomes observed on the augmentation. The dataset provides strong evidence that Virtual Appliances enable the creation of high-quality, meaningful synthetic smart meter traces. More broadly, these results illustrate that VA-based generation theoretically supports the production of arbitrarily large quantities of diverse, fully labeled synthetic aggregate data, opening the door to more robust training of machine learning models.

Should the automatic adaptation of a model's architecture prove inadequate for a particular appliance, a domain expert can inspect and manually refine the structure until the generated outputs sufficiently match real-world behavior. Such a workflow could, for instance, be implemented by an appliance manufacturer seeking to offer the resulting Virtual Appliance model as a supplementary service to customers or partners.

In principle, the generation models and the overall procedure can be applied to any household appliance. At this stage of the research, however, we cannot yet quantify how frequently manual architectural adjustments are required to achieve

satisfactory results. A definitive assessment would require applying the full process to a significantly broader range of appliances.

Broadly speaking, the appliance-level data generation techniques presented here are adaptable to different temporal resolutions. We have shown that high-resolution (1 Hz) synthetic appliance-level data can be successfully produced and subsequently downsampled to coarser intervals (e.g., 15 minutes). Conversely, attempting to generate synthetic appliance-level traces directly from low-resolution real data (e.g., 15-minute intervals) becomes problematic for appliances whose operation is shorter than the sampling interval or involves low energy draw over brief periods; in such cases, meaningful reconstruction is generally not feasible.

Synthetic data generation of this kind empowers developers to prototype and validate applications or to adapt pre-trained machine learning services to novel settings. By simulating an entire household—including realistic daily routines of its occupants—developers can produce representative data for fine-tuning models to better fit new environments or usage contexts. This strategy delivers improved model accuracy without the lengthy and costly process of collecting months of real-world measurements in the target setting.

Conclusion

Bottom-up energy simulation methods require substantial quantities of appliance-level ground truth data to generate realistic aggregate consumption traces, yet obtaining such data in practice remains challenging and resource-intensive. In this paper, we introduce VA-Creator, a user-oriented framework for creating Virtual Appliances that generate diverse synthetic PCPs. These PCPs can then be combined and orchestrated to form synthetic smart meter data suitable for training and evaluating machine learning models.

In this work, we developed generation techniques that cover all major appliance pattern categories commonly encountered in residential settings (single, programmatic, periodical, composed, constantly-on, and duration-dependent). To assess the fidelity of the synthetic outputs, we proposed and applied Dynamic Time Warping (DTW) as a robust statistical measure of similarity between real and generated PCPs. We further validated these findings using a neural-network-based Classifier. Additionally, we employed NILM disaggregation models as an end-to-end benchmark to gauge the practical utility of VA-generated synthetic data.

For single-pattern-type and programmatic-pattern-type appliances, we devised two generation approaches: a GAN-based model and a lightweight MLP architecture named IGen. We detailed mechanisms for automatic architectural adjustment, enabling non-experts in neural networks or machine learning to train new appliances with minimal manual intervention. We also addressed appropriate data transformation strategies for handling strongly bimodal distributions, which are characteristic of many real PCPs. The evaluation metrics confirm that the proposed models—particularly IGen—produce synthetic PCPs that are highly comparable to real measurements in both qualitative and quantitative terms.

By integrating an LSTM-based Activity Prediction model with the IGen framework, we can also successfully generate synthetic PCPs for appliances exhibiting periodic and composite pattern types. Once more, assessments conducted using the Classifier model and the DTW metric confirm that these generated PCPs cannot be reliably distinguished from their real counterparts, thereby qualifying as realistic.

For constantly-on and duration-dependent pattern types, we developed both an iterative MLP architecture and an XGBoost-based method. These approaches further allow the creation of diverse synthetic instances even for single-pattern-type appliances. In both pattern categories, the resulting synthetic PCPs exhibited very high similarity to actual measurements, indicating that realistic generation was achieved.

Given that Virtual Appliances are primarily intended to provide labeled synthetic aggregate smart meter data to enhance the robustness of machine learning algorithms, we conducted a comprehensive NILM evaluation. The findings show that

synthetic data produced via Virtual Appliances is highly effective for building more robust NILM models. Notably, NILM models trained on this synthetic data consistently outperform equivalent models trained on comparable volumes of real-world data.

The ideas and procedures presented in this paper establish a complete workflow within the VA-Creator for constructing Virtual Appliances. Whenever a suitable collection of real power consumption recordings is available, users need only configure a small number of parameters to initiate the segmentation process. The segmented PCPs are then assessed for quality; if the results are adequate, model training proceeds fully automatically—no prior expertise in neural networks is required. Subsequently, the trained models are evaluated using DTW distance calculations and classification testing with a dedicated classifier. When the synthesized PCPs demonstrate sufficient realism, the resulting Virtual Appliances are published, enabling their integration into a simulation environment that coordinates multiple models to synthesize realistic aggregate smart meter traces. Collectively, the concepts and techniques outlined here enable the generation of Virtual Appliances representing virtually any appliance found in a typical residential household.

Abbreviations

ML: Machine Learning

VA-Creator: Virtual Appliance Creator

MLP: Multilayer Perceptron

GAN: Generative Adversarial Network

DTW: Dynamic Time Warping

NILM: Non-Intrusive Load Monitoring

SynTiSeD: Synthetic Time Series Data Generator

UPC: Unassigned Power Consumption

PCP: Power Consumption Pattern

IGen: Lonely Generator

durGen: Duration Generator

MAE_{UH}: Mean Absolute Error on Unseen Houses

S2P: Sequence to Point

S2S: Sequence to Sequence

DAE: Denoising Autoencoder

RNN: Recurrent Neural Network

wGRU: Window Gated Recurrent Units

Acknowledgements

None

Conflict of interest

None

Financial support

This research was funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) as part of the ForeSightNEXT project and by the German Federal Ministry of Education and Research (BMBF) as part of the ENGAGE project.

Ethics statement

None

Received: 12 Oct 2023

Revised: 18 Nov 2023

Accepted: 27 Dec 2023

Published online: 20 March 2024

Rights and permissions

Open Access The author(s) retain copyright. This article is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. It may be shared and adapted for non-commercial purposes with appropriate attribution, an indication of changes, and distribution of adaptations under the same license. Third-party material may be subject to separate terms identified in its credit line. View the license at <https://creativecommons.org/licenses/by-nc-sa/4.0/>.

References

- Zini M, Carcasci C. Machine learning-based monitoring method for the electricity consumption of a healthcare facility in Italy. *Energy*. 2023;262:125576.
- Behzadi A, Gram A, Thorin E, Sadrizadeh S. A hybrid machine learning-assisted optimization and rule-based energy monitoring of a green concept based on low-temperature heating and high-temperature cooling system. *J Clean Prod*. 2023;384:135535.
- Rashid RA, Chin L, Sarijari MA, Sudirman R, Ide T. Machine learning for smart energy monitoring of home appliances using IoT. In: 2019 Eleventh International Conference on Ubiquitous and Future Networks. IEEE; 2019. p. 66-71.
- Gonzalez-Briones A, Hernandez G, Corchado JM, Omatu S, Mohamad MS. Machine learning models for electricity consumption forecasting: a review. In: 2019 2nd International Conference on Computer Applications & Information Security. IEEE; 2019. p. 1-6.
- Moon J, Park J, Hwang E, Jun S. Forecasting power consumption for higher educational institutions based on machine learning. *J Supercomput*. 2018;74:3778-800.
- Yan K, Wang X, Du Y, Jin N, Huang H, Zhou H. Multi-step short-term power consumption forecasting with a hybrid deep learning strategy. *Energies*. 2018;11(11):3089.

Kelly J, Knottenbelt W. Neural nilm: Deep neural networks applied to energy disaggregation. In: 2nd ACM Int Conf on Embedded Systems for Energy-Efficient Built Environments; 2015. p. 55-64.

Jiang J, Kong Q, Plumbly MD, Gilbert N, Hoogendoorn M, Roijers DM. Deep learning-based energy disaggregation and on/off detection of household appliances. *ACM Trans Knowl Discov Data*. 2021;15(3):1-21.

Konstantopoulos C, Sioutas S, Tsihlias K. Machine learning techniques for regression in energy disaggregation. In: IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer; 2022. p. 356-66.

Manera M, Marzullo A. Modelling the load curve of aggregate electricity consumption using principal components. *Environ Model Softw*. 2005;20(11):1389-400.

Delfosse A, Hebrail G, Zerroug A. Deep learning applied to NILM: is data augmentation worth for energy disaggregation? In: ECAI 2020. IOS Press; 2020. p. 2972-7.

Chen D, Irwin D, Shenoy P. Smartsim: A device-accurate smart home simulator for energy analytics. In: 2016 IEEE Int Conf on Smart Grid Communications. IEEE; 2016. p. 686-92.

Meiser M, Duppe B, Zinnikus I. SynTiSeD—synthetic time series data generator. In: 2023 11th Workshop on Modelling and Simulation of Cyber-Physical Energy Systems (MSCPES). IEEE; 2023. p. 1-6.

Wilhelm S, Jakob D, Kasbauer J, Ahrens D. GeLaP: German labeled dataset for power consumption. In: Proceedings of the Sixth International Congress on Information and Communication Technology. Springer Singapore; 2021. p. 21-33.
https://doi.org/10.1007/978-981-16-2377-6_5.

Firth S, Kane T, Dimitriou V, Hassan T, Fouchal F, Coleman M, et al. REFIT Smart Home dataset. 2017.

Shin C, Lee E, Han J, Yim J, Rhee W, Lee H. The ENERTALK dataset, 15 Hz electricity consumption data from 22 houses in Korea. *Sci Data*. 2019;6(1):193.

Meiser M, Duppe B, Zinnikus I. Generation of meaningful synthetic sensor data—Evaluated with a reliable transferability methodology. *Energy and AI*. 2024;15:100308.

Binderbauer PJ, Kienberger T, Staubmann T. Synthetic load profile generation for production chains in energy intensive industrial subsectors via a bottom-up approach. *J Clean Prod*. 2022;331:130024.

Sandhaas A, Kim H, Hartmann N. Methodology for generating synthetic load profiles for different industry types. *Energies*. 2022;15(10):3683.

Hong T, Macumber D, Li H, Fleming K, Wang Z. Generation and representation of synthetic smart meter data. In: Building Simulation. Vol 13. Springer; 2020. p. 1205-20.

Zhang C, Kuppannagari SR, Kannan R, Prasanna VK. Generative adversarial network for synthetic time series data generation in smart grids. In: 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids. IEEE; 2018. p. 1-6.

Ifthikhar N, Liu X, Nordbjerg FE, Danalachi S. A prediction-based smart meter data generator. In: 2016 19th International Conference on Network-Based Information Systems. IEEE; 2016. p. 173-80.

Behm C, Nolting L, Praktijnjo A. How to model European electricity load profiles using artificial neural networks. *Appl Energy*. 2020;277:115564.

Fischer D, Härtl A, Wille-Haussmann B. Model for electric load profiles with high time resolution for German households. *Energy Build*. 2015;92:170-9.

Brodén DA, Paridari K, Nordström L. Matlab applications to generate synthetic electricity load profiles of office buildings and detached houses. In: 2017 IEEE Innovative Smart Grid Technologies-Asia. IEEE; 2017. p. 1-6.

Chen D, Irwin D, Shenoy P. SmartSim: A device-accurate smart home simulator for energy analytics. In: 2016 IEEE International Conference on Smart Grid Communications. 2016. p. 686-92.
<https://doi.org/10.1109/SmartGridComm.2016.7778841>.

Barker S, Kalra S, Irwin D, Shenoy P. Empirical characterization and modeling of electrical loads in smart homes. 2013:1-10.
<https://doi.org/10.1109/IGCC.2013.6604512>.

Reinhardt A, Klemenjak C. How does load disaggregation performance depend on data characteristics? insights from a benchmarking study. In: Proceedings of the Eleventh ACM International Conference on Future Energy Systems. 2020:167-77.

Klemenjak C, Kovatsch C, Herold M, Elmenreich W. A synthetic energy dataset for non-intrusive load monitoring in households. *Sci Data*. 2020;7(108).
<https://doi.org/10.1038/s41597-020-0434-6>.

Buneeva N, Reinhardt A. AMBAL: Realistic load signature generation for load disaggregation performance evaluation. In: 2017 IEEE International Conference on Smart Grid Communications. 2017:443-8.
<https://doi.org/10.1109/SmartGridComm.2017.8340657>.

Reinhardt A, Klemenjak C. How does load disaggregation performance depend on data characteristics? insights from a benchmarking study. In: Proceedings of the Eleventh ACM International Conference on Future Energy Systems. 2020:167-77.

Harell A, Jones R, Makonin S, Bajić IV. Tracegan: Synthesizing appliance power signatures using generative adversarial networks. *IEEE Trans Smart Grid*. 2021;12(5):4553-63.
<https://doi.org/10.1109/TSG.2021.3078695>.

Gkoutroumpi C, Gkalinikis NV, Vrakas D. SGAN: Appliance signatures data generation for NILM applications using GANs. In: Science and Information Conference. Springer; 2024:325-39.

Alberternst S, Anisimov A, Antakli A, Hoffmann H, Meiser M, Duppe B, et al. Orchestrating heterogeneous devices and AI services as virtual sensors for secure cloud-based IoT applications. *Sensors*. 2021;21(22):7509.

Borji A. Pros and cons of GAN evaluation measures: New developments. *Comput Vis Image Underst*. 2022;215:103329.

Müller M. Dynamic time warping. *Inf Retr Music Motion*. 2007;69-84.

Marius P, Balas V, Perescu-Popescu L, Mastorakis N. Multilayer perceptron and neural networks. *WSEAS Trans Circuits Syst*. 2009;8.

Maas AL, Hannun AY, Ng AY. Rectifier nonlinearities improve neural network acoustic models. In: Proc Icml. Vol 30. Citeseer; 2013. p. 3.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. *Adv Neural Inf Process Syst*. 2014;27.

Andreini P, Bonechi S, Bianchini M, Mecocci A, Scarselli F. Image generation by GAN and style transfer for agar plate image segmentation. *Comput Methods Programs Biomed*. 2020;184:105268.

Zhang B, Gu S, Zhang B, Bao J, Chen D, Wen F, et al. Styleswin: Transformer-based gan for high-resolution image generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022:11304-14.

Bao J, Chen D, Wen F, Li H, Hua G. CVAE-GAN: fine-grained image generation through asymmetric training. In: Proceedings of the IEEE International Conference on Computer Vision. 2017:2745-54.

Islam J, Zhang Y. GAN-based synthetic brain PET image generation. *Brain Inf.* 2020;7:1-12.

Han C, Hayashi H, Rundo L, Araki R, Shimoda W, Muramatsu S, et al. GAN-based synthetic brain MR image generation. In: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018). IEEE; 2018:734-8.

Rubner Y, Tomasi C, Guibas LJ. The earth mover's distance as a metric for image retrieval. *Int J Comput Vis.* 2000;40:99-121.

Arjovsky M, Chintala S, Bottou L. Wasserstein generative adversarial networks. In: International Conference on Machine Learning. PMLR; 2017:214-23.

Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC. Improved training of Wasserstein GANs. *Adv Neural Inf Process Syst.* 2017;30.

Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput.* 1997;9(8):1735-80.

Martin JH. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Pearson/Prentice Hall; 2009.

Chen T, Guestrin C. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016:785-94.

Batra N, Kukunuri R, Pandey A, Malakar R, Kumar R, Krystalakos O, et al. Towards reproducible state-of-the-art energy disaggregation. In: 6th ACM Int Conf on Systems for Energy-Efficient Buildings, Cities, and Transportation. 2019:193-202.

D'Incecco M, Squartini S, Zhong M. Transfer learning for non-intrusive load monitoring. *IEEE Trans Smart Grid.* 2019;11(2):1419-29.

Zhang C, Zhong M, Wang Z, Goddard N, Sutton C. Sequence-to-point learning with neural networks for non-intrusive load monitoring. In: AAAI Conf on Artificial Intelligence. Vol 32. 2018.

Kelly J, Batra N, Parson O, Dutta H, Knottenbelt W, Rogers A, et al. Nilmtk v0. 2: a non-intrusive load monitoring toolkit for large scale data sets. In: 1st ACM Conf on Embedded Systems for Energy-Efficient Buildings. 2014:182-3.

Krystalakos O, Nalmpantis C, Vrakas D. Sliding window approach for online energy disaggregation using artificial neural networks. In: 10th Hellenic Conf on Artificial Intelligence. 2018:1-6.

Reinhardt A, Klemenjak C. How does load disaggregation performance depend on data characteristics? Insights from a benchmarking study. In: 11th ACM Int Conf On Fut Energy Sys. 2020:167-77.

Klemenjak C, Faustine A, Makonin S, Elmenreich W. On metrics to assess the transferability of machine learning models in non-intrusive load monitoring. 2019. arXiv preprint arXiv:1912.06200.

Yue Z, Witzig CR, Jorde D, Jacobsen H-A. Bert4nilm: A bidirectional transformer model for non-intrusive load monitoring. In: Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring. 2020:89-93.

Chen K, Wang Q, He Z, Chen K, Hu J, He J. Convolutional sequence to sequence non-intrusive load monitoring. *J Eng.* 2018;2018(17):1860-4.

Chen K, Zhang Y, Wang Q, Hu J, Fan H, He J. Scale-and contextaware convolutional non-intrusive load monitoring. *IEEE Trans Power Syst.* 2019;35(3):2362-73.

Athanasiadis C, Doukas D, Papadopoulos T, Chrysopoulos A. A scalable real-time non-intrusive load monitoring system for the estimation of household appliance power consumption. *Energies*. 2021;14(3):767.

Luo J, Liu S, Cai Z, Xiong C, Tu G. A multi-task learning model for nonintrusive load monitoring based on discrete wavelet transform. *J Supercomput*. 2023;79(8):9021-46.

Cochran W. *Sampling Techniques*. 2nd ed. John Wiley & Sons; 1963.