SynTiSeD – Synthetic Time Series Data Generator

Michael Meiser
Agents and Simulated Reality
DKFI, Saarland Informatics Campus
Saarbrücken, Germany
michael.meiser@dfki.de

Benjamin Duppe
Agents and Simulated Reality
DFKI, Saarland Informatics Campus
Saarbrücken, Germany
benjamin.duppe@dfki.de

Ingo Zinnikus
Agents and Simulated Reality
DFKI, Saarland Informatics Campus
Saarbrücken, Germany
ingo.zinnikus@dfki.de

***Abstract*—Recently, an increasing number of Artificial Intelligence services have been developed for a variety of domains. Machine Learning and especially Deep Learning services require a large amount of data to provide their functionality. Since data collection is typically complex and difficult, there is often not enough data available. Machine learning services such as anomaly detection or disaggregation algorithms are also being developed in the smart living domain. In practice, however, only a few energy datasets are publicly available, as the collection of such data is expensive and time-consuming due to the equipment required. One way to generate more smart meter data is to use a simulation. Developing such a simulation that is capable of generating meaningful data is a complex task. Therefore, in this paper, we present the Synthetic Time Series Data Generator (SynTiSeD), a multi-agent-based simulation tool that generates meaningful synthetic energy data based on real-world data. Furthermore, SynTiSeD allows generating data of critical situations, which are important for the development of such services, but which cannot be provoked in the real world. For transferability, we demonstrate that Nonintrusive Load Monitoring algorithms trained on synthetic data generated by SynTiSeD provide meaningful results that are even better than those of models trained on real data.**

***Index Terms*—Smart Living, Smart Meter, Synthetic Sensor Data, Energy Data, Simulation, Agents, NILM, Seq2Point, windowGRU, Machine Learning.**

I. Introduction

A growing number of Artificial Intelligence (AI) and in particular Machine Learning (ML) services have recently been developed in various contexts. In order to work as intended and become robust, these services require a vast amount of meaningful data, especially ground truth data. In most cases, collecting this data is complex, expensive, and time-consuming, hence there is often not enough data available. The lack of meaningful data is a serious obstacle in the development of powerful AI services.

Although more and more data is being collected around the world to address this problem, several other issues remain. Especially in the area of smart living, various services based on smart meter data are being developed to improve the comfort and safety of future homes, such as anomaly detection or Nonintrusive Load Monitoring (NILM) algorithms. However, data collection is particularly difficult and complex due to the hardware required and the many privacy regulations. A further problem with the limited number of publicly available energy datasets is that the ground truth data of the appliances that are part of a smart meter is not always complete or available, or the data is not sufficiently labeled, although this data is essential for training ML services.

One obvious approach to generate more labeled smart meter data is to use a simulation. For this reason, we introduce the *Synthetic Time Series Data Generator* (SynTiSeD), a simulation tool that allows generating meaningful energy data. *SynTiSeD* acts like a virtual incubator, using various concepts and methods to multiply collected real data to create new, meaningful synthetic data. In this context, ”real data” means energy data recorded in existing households, including all electricity consumers. We developed *SynTiSeD* as a scalable, multi-agent-based framework that is capable of swiftly generating large amounts of labeled data even for a large number of households and residents with different daily routines. Our simulation also allows modeling and generating data for critical situations that cannot or should not be provoked in reality. As our evaluation demonstrates, for transferability *SynTiSeD* is capable of generating data to meaningfully train and even improve the results of NILM algorithms. In addition, the multi-agent structure of *SynTiSeD* allows simulating not only single households with single residents, but also a large number of households with a flexible number of residents with different daily schedules and thus also whole housing complexes or agglomerations. The resulting data enables the creation of large-scale energy consumption forecasts, which are essential for adaptive energy supply systems.

II. Related Work

Due to the lack of meaningful data, several energy datasets have already been collected in real households. Examples are the GeLaP dataset [1], the REFIT dataset [2], the Eco dataset [3] or the UK-DALE dataset [4]. On some of them, the performance of NILM algorithms has already been tested [3], [5]. Besides real world data sets, there are also synthetically generated data sets like SHED [6] or SynD [7]). However, the real datasets as well as these synthetic datasets share the limitation of providing only partially labeled ground truth data for appliances, which is necessary for training most ML services. Also the datasets cannot be easily varied or additional data cannot be generated if needed (the two frameworks used to create the datasets are not freely available (for SHED [6] and SynD [7])).

There exist smart home simulation tools which can be divided in two categories, namely *model-based* and *interactive approaches* [8]. Both approaches have in common that activities are executed by a resident, which then ultimately generate data. Some interesting approaches are presented in more detail below, though we focus on smart home simulation tools that are capable of generating energy data.

Model-based approaches use predefined activity models to generate synthetic data. These models specify the sequence of occurrences, the probability, and the duration of each activity. This approach facilitates the generation of large datasets in a short time, but the limitation is that complicated interactions or unexpected accidents, which often occur in real houses, are difficult to model. In the following, some selected examples of *model-based approaches* are presented in more detail:

SmartSim [9] is a publicly available tool [1] that can generate synthetic energy data. It creates energy consumption traces for appliances by combining an appliance model, which captures the pattern of energy consumption in the active state, with an appliance usage model, which specifies the frequency, duration, and timing of an activity. *SmartSim* then generates aggregate energy data for a simulated home by combining data from individual appliances. However, the techniques used require a great deal of manual effort and can only be developed by experts, making it difficult to use the tool and to develop new appliance models as well as generate new data.

The *Automated Model Builder for Appliance Loads* (AMBAL) [10] can also generate synthetic power data. This is done by connecting individual consumption patterns a sufficient number of times and optimizing the resulting segments with a *mean absolute percentage error* (MAPE). However, to the best of our knowledge, *AMBAL* is not freely publicly available.

Unlike the *model-based approach*, in theory the *interactive approach* can better capture fine nuances and interesting interactions. The *interactive approach* relies on an avatar that can be controlled by a user or a simulated participant. This avatar moves around the virtual apartment and interacts with its environment, which is equipped with virtual sensors and/or actuators; these interactions can be passive or active. However, the drawback of the interactive approach is that it is time-consuming to generate sufficient data sets, since all interactions must be captured in real time and performed by a user. Although there are some tools that can generate different smart home datasets (e.g. motion data) [11], [12], to the best of our knowledge, no frameworks have been published that can generate high amounts of synthetic energy data while following a strictly interactive approach. However, since the goal of this work is to generate large amounts of synthetic energy data, this approach is not considered in detail here.

There are solutions that combine both model-based and interactive approaches, such as SMACH [13]. The multi-agent based simulator generates synthetic time use surveys from real time use surveys. These surveys are provided for each time sample and consist of a list of all appliances in the household and an indicator of whether these appliances are activated and consuming energy. Then, real energy consumption patterns collected from their own data (27 french households, but not published) are used in combination with the time use surveys to generate synthetic data. However, to the best of our knowledge, SMACH is not publicly available.

In general, the model-based approach allows generating large datasets in a short simulation time, but it struggles to capture the granularity of realistic interactions. On the other hand the interactive approach captures these realistic interactions, it requires a long simulation time, so the resulting datasets are typically smaller than those generated by the model-based approach. Several problems facing smart home simulation research can be identified [8]. One of the main difficulties relates to the fact that many of the available simulation tools focus on testing applications that identify and visualize correlations, rather than focusing on generating representative data sets. Another challenge includes having the flexibility and scalability to add new or customized types of smart appliances, change their generated output, change their location within the smart home, and so on. Another limitation of currently available tools is that they are not able to simulate multiple residents, as this feature is difficult to implement.

Based on the categorization previously discussed, *SynTiSeD* can be assigned to both approaches. It follows the model-based approach, since it can specify the sequence of events, the probability of their occurrence, and the duration of each activity. Additionally, it follows *interactive approaches for generating simulated datasets*, as it uses agents that interact with the apartment, although they are not controlled by a user alone. Also, *SynTiSeD* does not use a three-dimensional environment because it is complicated to create and does not improve the data generation. To the best of our knowledge, there are no applications in the smart living domain that are comparable to the framework we have developed, namely an extendable, multi-agent-based simulation tool for generating synthetic energy data based on real data. Although *SynTiSeD* is agent-based, it allows generating large amounts of data efficiently. Scalability with multiple households, residents, and appliances was also ensured.

III. SynTiSeD

The current *SynTiSeD* is a new version of the simulation tool presented in [14]. Since then, *SynTiSeD* has evolved from an interactive three-dimensional framework to a model-based Python application, resulting in significant improvements in both usability and performance when generating labeled synthetic data. In the following, the structure, concepts, and components of the most recent version of *SynTiSeD*, which is available on Github [2], are explained in more detail.

A. Structure

SynTiSeD is schematically structured as illustrated in figure 1. Households can be created, to which different *residents*

[1]SmartSim: https://github.com/klemenjak/smartsim

[2]SynTiSeD: https://github.com/mmeism/SynTiSeD_research

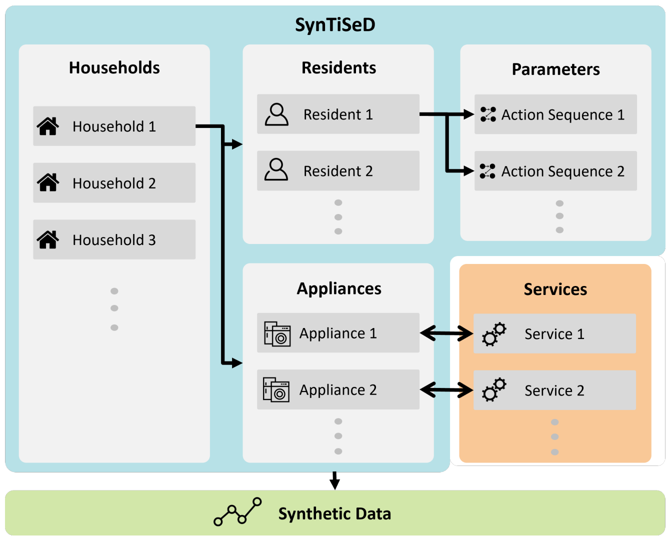

Fig. 1. Schematic *SynTiSeD* architecture

can be added and to which *appliances* can be attached. A resident agent - or for short resident in the following - is an entity which acts and interacts with appliances in a household using an *action sequence*. An action sequence is a sequence of actions, which in turn basically consists of three components: the name of the appliance, the time at which it is switched on and the probability of it being switched on. One or more action sequences can be assigned to the resident agent, where one action sequence represents the resident's behavior for one day. How exactly an action sequence is structured and how it can be created is described in more detail in section III-C1. Appliances are connected to services which return energy data that is then used by the simulation (see section III-D).

Furthermore, *SynTiSeD* provides several other parameters that can be adjusted by the user, such as simulation speed, variance, number of days to be simulated or the start date of the simulated days. With the simulation speed parameter it is possible to regulate how fast the simulation is executed. So it is possible to generate synthetic data in real time and use the simulation as a full replacement for a real household as well as to simulate a large amount of data in a very short time (depending on the computing power the energy data as well as the ground truth data of all uses appliances of one whole day can be generated in less than one second). The variance parameter allows a user to specify with how much variance each action is executed. The variance v spans an interval starting at zero, where a value is selected to be added or subtracted to the point in time when the resident agent is supposed to perform the action. The user can also specify the date of the generated days and the number of consecutive days to be generated in each instance of the simulation.

B. Power Consumption Sequencer

The *Power Consumption Sequencer* (Sequencer) selects power consumption patterns (PCPs) from the ground truth data of an appliance (e.g. recorded by a smart plug). As input, the *Sequencer* requires time series data that indicate the amount of energy an appliance has consumed at any given timestamp. The *Sequencer* is able to handle input data with any sample rate, as long as the data is consistent and has no gaps.

There are a number of parameters available to specify how the *Sequencer* selects individual PCPs, which are used to avoid unintended artifacts and to choose only the PCPs that should be selected. The sequenced PCPs are then clustered to detect anomalies and unusual PCPs. The user can review the sequenced PCPs and decide whether to keep anomalies in the dataset or discard them. It would also be possible to automate this process, for example, by using a neural network classifier trained to keep only similar PCPs. However, this approach also faces drawbacks. For example, if a new program of a washing machine is sequenced and PCPs are sequenced that the neural network classifier does not have seen them before, it can potentially sort out the PCPs of the new program.

When the sequencer selects PCPs, it also notices the points in time when the appliance consumed energy. These timestamps form action sequences, i.e. routines that SynTiSeD uses to generate energy data.

C. Resident

Resident agents are independent entities that act in a household based on associated action sequences (see figure 1). How such an action sequence is structured and how it can be used is explained in more detail below (section III-C1). A resident executes actions sequentially, without deeper decision making, as resident agents do not communicate with each other. However, since *SynTiSeD* is an agent-based framework, it is possible to implement both decision making and communication between residents (section III-C2).

1) Action Sequence: As already indicated, an action sequence is a series of actions that a resident agent performs, acting and operating in a household. An internal representation of an action sequence consists of a header followed by a variable number of rows, each row representing an action. An action itself consists of the name of the appliance to be activated, a start and end time, a probability with which the action is performed, and a variance.

The *name* of the appliance represents its unique ID, which *SynTiSeD* uses to identify the appliance in the simulation. For this reason, two different appliances located in the same household must have different names. The name of an appliance is the only action parameter that is mandatory to set. The *start time* as well as the *end time* of an action are timestamps and determine when the appliance was switched on or off. If only the start time of an appliance is set, the appliance will be turned on at that time and a PCP from the database will be inserted at that time. If an end time is set, the appliance will be turned off at that time. If both a start time and an end time are set, the appliance will consume power at that time period. Either one or both times must be set, but the end time can only be set if the appliance was previously turned on in the sequence. The *probability* of an action is a float value between 0 and 1 and specifies the probability of the action being executed. The action is executed with 0% probability for a value of 0, and with 100% probability for a value of 1. The *variance* of an action specifies the variance in seconds with which an action is executed. The value of the variance spans an interval with zero in which a value is selected to be added to or subtracted from the start and end time of the action. The variance can be either uniform or normally distributed.

2) Behavior Modeling: An action sequence can be the output of the sequencer, so it is derived from real data and thus represents actual human behavior. However, it is also possible to create them manually. For instance, if a person is new to an apartment and there is no real data about her habits and daily routines, action sequences can be created based on her behaviors. Another option is to create them manually if critical situations are to be simulated that do not occur in the real data. For example, in the real world, you do not want to provoke the situation where a person accidentally slips in the shower and is unable to move. However, having the data about these situations is necessary to be able to observe how ML assistance services react to them. For example, by interrupting an action sequence, *SynTiSeD* allows to comfortably simulate such a situation without endangering anyone.

Furthermore, it would also be possible to dynamically generate an action sequence by a model that represents a certain type of human behavior. Since *SynTiSeD* is designed as an multi-agent framework, the resident agents of a household are able to dynamically react to situations and interact with each other through such a model. An example of a model that is capable of generating action sequences is the *activity prediction model* presented in [14], [15]. In this model, multiple *Long Short-Term Memory models* (LSTMs) are utilized to predict future activities to generate action sequences using a pseudo probabilistic approach.

Another possible model for a resident's decision making is a *partially observable Markov decision process* (POMDP), a generalization of a *Markov decision process* (MDP) [16]. In principle, using a POMDP, an agent computes probability values for the possible states it could be in and uses this to compute the value of its possible actions. This way, the action sequences that a resident agent executes would not have to be fixed from the beginning, but can be created dynamically at simulation runtime.

D. Appliance

A *virtual appliance* can be switched on by a resident agent in the simulation, whereupon it will consume energy. Naturally, a household might contain several different appliances, as well as appliances of the same type (but not the same ID). The total energy consumption of a household (the smart meter) is the summation of the energy consumption of its individual appliances. An appliance can both consume energy, such as a washing machine or electric kettle, as well as produce energy, such as a photovoltaic system. The architecture of *SynTiSeD* allows external services to be connected to appliances, such as a service that generates photovoltaic power output based on weather data [17].

To create a virtual appliance, ground truth data is sequenced (see chapter III-B) and the resulting PCPs are then stored in a database. Each appliance is connected to this database and the PCPs of its type via a service. When the resident agent turns on an appliance during the simulation, a stored PCP is pulled from the database and inserted at the time specified by the action start time and/or end time. Each appliance in a household is an independent entity and knows at any time how much energy it is consuming. Furthermore, each appliance has a unique ID, so that it cannot be duplicated, cannot be switched on more than once, and cannot be used by more than one resident. Of course, there are exceptions for appliances such as the TV, which can be used by multiple residents at the same time.

As already described, the PCPs are obtained from real data by sequencing and clustering. However, it would also be possible to generate these PCPs synthetically. For example, the PCPs of an appliance sequenced from real data can be passed to a neural network that generates new synthetic PCPs out of these real PCPs that are similar to the real PCPs but contain some variance. This way, *SynTiSeD* is able to insert new PCPs whenever it creates new data, thus improving the variance of the resulting synthetic smart meter data. Presumably, this technique is especially helpful when generating large amounts of data, since otherwise real appliance PCPs have to be inserted multiple times.

IV. SYNTISED USAGE

This section explains in more detail how the workflow of *SynTiSeD* was designed (section IV-A) and which opportunities the simulation opens up in practice (section IV-B).

A. Workflow

SynTiSeD generates synthetic data by having each resident of each household perform their assigned action sequences, with residents interacting with connected virtual appliances.

In detail, the workflow for generating a synthetic day is as follows: first, *SynTiSeD* iterates over each timestamp of the day specified by the sample rate, and adds the data from each corresponding timestamp of all permanent virtual appliances connected to the household to the virtual smart meter (for example, a router that always consumes energy regardless of what a resident is doing). At the same time, *SynTiSeD* constantly checks if a resident agent triggers an appliance (a human induced virtual appliance) via the iterated timestamp. If so, *SynTiSeD* queries the stored data of the appliance from a database via a service and inserts it into the smart meter at the appropriate place. A day of the virtual smart meter thus consists of the data of all permanent virtual appliances as well as the data of all human induced virtual appliances used by all residents of the household on that day. In addition to the smart meter, *SynTiSeD* simultaneously generates an individual energy history for each appliance used in the virtual household, providing ground truth data for all appliances used.

In summary, a simulation with *SynTiSeD* runs as follows (see figure 2). First, the real data is preprocessed, i.e. the data is sampled and potential data gaps are filled before it is

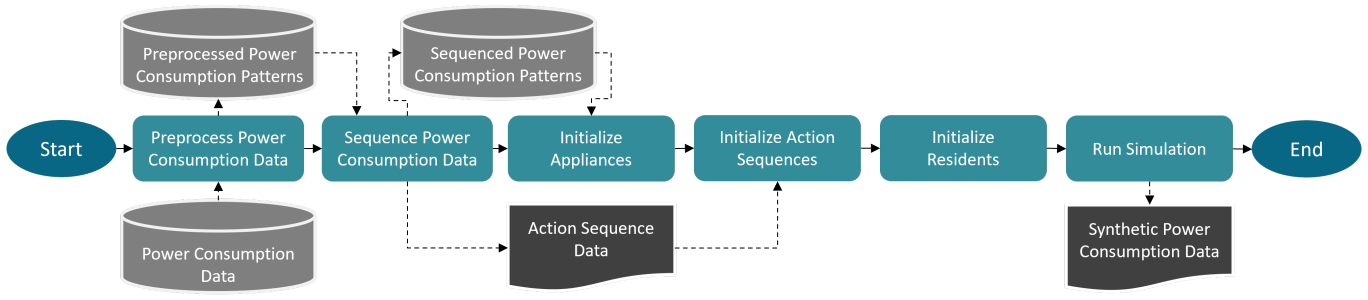
Preprocessed Power Consumption Patterns
Sequenced Power Consumption Patterns
Start
Preprocess Power Consumption Data
Sequence Power Consumption Data
Initialize Appliances
Initialize Action Sequences
Initialize Residents
Run Simulation
End
Power Consumption Data
Action Sequence Data
Synthetic Power Consumption Data

Fig. 2. *SynTiSeD* Flow Chart

stored in a database. The preprocessed data is then passed to the sequencer, which stores both the sequenced PCPs and the corresponding action sequences of the data. Then we are able to initialize households and connect appliances to them. We are also able to initialize residents using the action sequences extracted by the sequencer. After that, the simulation is ready to be started, i.e. the residents execute the action sequences assigned to them in the created households and interact with the linked appliances. This way, virtual power consumption data is created which is then stored. *SynTiSeD* provides a Flask service endpoint [3] that is accessible via RESTful web services and JSON [4] to start the simulation and receive generated data.

B. Use cases

SynTiSeD enables various usage possibilities. For example, we are able to simulate when an apartment is vacant and a new resident is about to move in. If ML services are to be used there, e.g. for activity or anomaly detection, no data about the resident's habits are available because he or she has not yet lived there. The resident's habits can be explored in advance, e.g. in an interview before the resident moves in, and then be recorded in action sequences. These sequences, combined with the already sequenced PCPs, enable *SynTiSeD* to generate synthetic smart meter data. This data in turn allows training, validating and testing ML services so that the resident is ready to use the services as soon as they move in.

SynTiSeD is also capable of augmenting and multiplying a small amount of real data from a household. This way, a few days or weeks of real data can be scaled up to months or even years of synthetic data. In theory, increasing the amount of data will cause the ML models trained on it to become more robust and produce improved results. Further, *SynTiSeD* is suitable for simulating a variety of changes in an existing household. For example, we can simulate a real household as closely as possible, then install a wall box in the resulting virtual household and observe the effects in the energy data.

Since the architecture of *SynTiSeD* also allows simulating a large number of households, we are able to conduct extensive energy forecasting. For example, we simulate a housing complex with a hundred households, each of which is structured differently. We generate single households, family households, or even shared apartments. This means that the individual residents all act according to different action sequences, and the households are also equipped with different sets of appliances. This data enables us to determine how energy consumption might develop in the future. With *SynTiSeD* it is also possible to simulate different scenarios depending on what appliances are available to the households. This allows simulating how much energy the whole housing complex consumes if all the residents return home from work at the same time and then charge their electric car at their wall box, plus possibly cook some food and turn on their washing machine. Or we simulate how much energy an agglomeration consumes if all households had installed a heat pump or an electric ventilation system at the same time. This enables monitoring of whether and when the electrical grid is overloaded, which is important for energy providers who have to ensure that every household is supplied with enough electricity at any time. We are also able to observe how the energy consumption of a housing complex changes if only a certain percentage of the households have installed a heat pump or a wall box. *SynTiSeD* also allows observing how seasonal changes affect the energy consumption, since the heat pumps will consume more energy in winter than in summer. Also, if households are equipped with a solar system, it will generate more electricity when the sun is shining than when it is not.

V. Evaluation

To prove that *SynTiSeD* generates meaningful data that allow ML services to be trained, we synthesized different households of the GeLaP dataset [1] using the procedures and concepts described previously. To create a baseline, we want to reconstruct the real data as closely as possible. Therefore, we sequenced the ground truth data of the available appliances for each household, resulting in PCPs for each appliance and action sequences for the residents. In addition, we subtracted any available appliance data from each household's smart meter data to obtain the unknown consumption data. We used this data as a kind of background energy consumption for the virtual households. Using *SynTiSeD* and the extracted data, the synthetic households are generated.

For kettle and washing machine, we trained seq2point and windowGRU disaggregation algorithms (state-of-the-art models from the NILMTK toolkit [18]) on four synthetic

[3]Flask: https://flask.palletsprojects.com/en/2.1.x/
[4]JSON: https://www.json.org/json-de.html

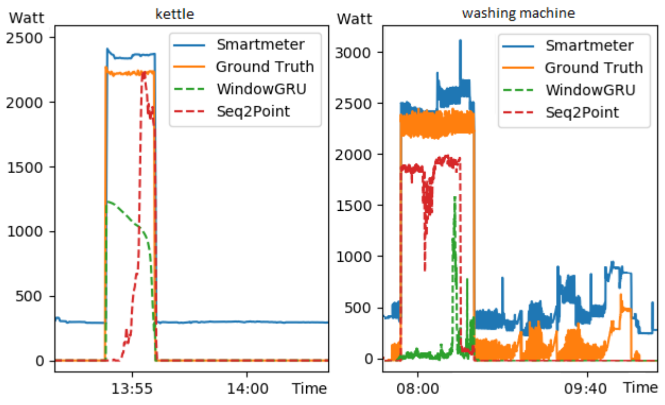

Fig. 3. NILM disaggregation of the synthetic data generated by *SynTiSeD*

TABLE I
BEST MAE RESULTS OF NILM ALGORITHMS TRAINED ON SYNTHETIC DATA COMPARED TO MODELS TRAINED ON REAL DATA

Appliance	**Real Data**		**Synthetic Data**	
	s2p	***wGRU***	***s2p***	***wGRU***
kettle	13.67	8.28	7.88	7.10
washing machine	13.77	9.58	11.86	7.95

households each and, for comparison, on four real households from the GeLaP dataset (each real and synthetic household contains exactly the same number of data points). The transferability of the algorithms was then tested for both the synthetic training data and the real training data on real households that were not part of the training data. For both appliances, both types of algorithms identify the active phases and disaggregate the appliances reasonably well when trained with synthetic data (see figure 3). We observe that the best NILM models trained on synthetic data provide better results in terms of mean absolute error (MAE) than the best models trained on real data (see table I).

VI. Conclusion

Due to the lack of publicly available energy datasets and the various problems in collecting this data, we developed *SynTiSeD*, a multi-agent framework that allows to efficiently and comfortably generate large amounts of meaningful synthetic data based on real-world data. In contrast to real-world datasets, the simulation always provides a stable sampling rate, no data gaps, and reliable ground truth data. Additionally, datasets can be rapidly modified by adding or removing appliances from households or by changing a resident's routine. *SynTiSeD* also allows generating data from critical situations that cannot or should not be provoked in the real world.

However, the major advantage of *SynTiSeD* is that it allows creating an unlimited amount of data with some variance to the real data. The simulation offers the possibility to use different appliances in a household, or residents that follow specified daily routines. In theory, ML services trained on a larger amount of meaningful data are considered more robust than those trained on less data. Furthermore, due to the multi-agent structure of *SynTiSeD*, it is possible to simulate not only a single household with a single resident, but also any number of households with diverse residents, all behaving in different routines. This allows simulating entire housing complexes or agglomerations and to perform comprehensive energy consumption forecasts. Since this synthetic data is generated with the goal of improving the training and validation of ML services, we have demonstrated that state-of-the-art NILM algorithms (seq2point and windowGRU) trained on synthetic data provide meaningful results in terms of transferability that are even better than those of models trained on real data.

References

[1] S. Wilhelm, D. Jakob, J. Kasbauer, and D. Ahrens, "GeLaP: German labeled dataset for power consumption," in *Sixth Int. Congr. on Information and Communication Technology*. Springer, Sep. 2021, pp. 21–33.

[2] S. Firth, T. Kane, V. Dimitriou, T. Hassan, F. Fouchal, M. Coleman, and L. Webb, "REFIT Smart Home dataset," 6 2017.

[3] C. Beckel, W. Kleiminger, R. Cicchetti, T. Staake, and S. Santini, "The eco data set and the performance of non-intrusive load monitoring algorithms," in *1st ACM conf. on embedded systems for energy-efficient buildings*, 2014, pp. 80–89.

[4] J. Kelly and W. Knottenbelt, "The uk-dale dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes," *Scientific data*, vol. 2, no. 1, pp. 1–14, 2015.

[5] A. Verma, A. Anwar, M. Mahmud, M. Ahmed, and A. Kouzani, "A comprehensive review on the nilm algorithms for energy disaggregation," *arXiv preprint arXiv:2102.12578*, 2021.

[6] S. Henriet, U. Şimşekli, B. Fuentes, and G. Richard, "A generative model for non-intrusive load monitoring in commercial buildings," *Energy and Buildings*, vol. 177, pp. 268–278, 2018.

[7] C. Klemenjak, C. Kovatsch, M. Herold, and W. Elmenreich, "A synthetic energy dataset for non-intrusive load monitoring in households," *Scientific Data*, vol. 7, no. 1, pp. 1–17, 2020.

[8] J. Synnott, C. Nugent, and P. Jeffers, "Simulation of smart home activity datasets," *Sensors*, vol. 15, no. 6, pp. 14 162–14 179, 2015.

[9] D. Chen, D. Irwin, and P. Shenoy, "Smartsim: A device-accurate smart home simulator for energy analytics," in *2016 IEEE Int. Conf. on Smart Grid Communications (SmartGridComm)*. IEEE, 2016, pp. 686–692.

[10] N. Buneeva and A. Reinhardt, "Ambal: Realistic load signature generation for load disaggregation performance evaluation," in *2017 IEEE int. conf. on smart grid communications*. IEEE, 2017, pp. 443–448.

[11] M. Buchmayr, W. Kurschl, and J. Küng, "A simulator for generating and visualizing sensor data for ambient intelligence environments," *Procedia Computer Science*, vol. 5, pp. 90–97, 2011.

[12] J. Synnott, L. Chen, C. D. Nugent, and G. Moore, "The creation of simulated activity datasets using a graphical intelligent environment simulation tool," in *2014 36th annual int. conf. of the IEEE engineering in medicine and biology society*. IEEE, 2014, pp. 4143–4146.

[13] A. Delfosse, G. Hebrail, and A. Zerroug, "Deep learning applied to nilm: is data augmentation worth for energy disaggregation?" in *ECAI 2020*. IOS Press, 2020, pp. 2972–2977.

[14] S. Alberternst, A. Anisimov, A. Antakli, B. Duppe, H. Hoffmann, M. Meiser, M. Muaz, D. Spieldenner, and I. Zinnikus, "Orchestrating heterogeneous devices and ai services as virtual sensors for secure cloud-based iot applications," *Sensors*, vol. 21, no. 22, p. 7509, 2021.

[15] B. Duppe, M. Meiser, A. Anisimov, A. Antakli, M. Muaz, and I. Zinnikus, "Combining machine learning with inductive logic learning to detect deviations from daily routines in ambient intelligent environments," in *IEEE/WIC/ACM Int. Conf. on Web Intelligence and Intelligent Agent Technology*, 2021, pp. 310–317.

[16] M. T. Spaan, "Partially observable markov decision processes," in *Reinforcement Learning*. Springer, 2012, pp. 387–414.

[17] S. Theocharides, G. Makrides, G. E. Georghiou, and A. Kyprianou, "Machine learning algorithms for photovoltaic system power output prediction," in *2018 IEEE Int. Energy Conf.* IEEE, 2018, pp. 1–6.

[18] J. Kelly, N. Batra, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh, and M. Srivastava, "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, pp. 182–183.