A Data-Driven Approach for Generating Synthetic Load Patterns and Usage Habits

Samer El Kababji, *Student member*, IEEE, Pirathayini Srikantha, *Member*, IEEE

***Abstract*—Today's electricity grid is rapidly evolving to become highly connected and automated. These advancements have been mainly attributed to the ubiquitous communication/computational capabilities in the grid and the *internet of things* paradigm that is steadily permeating modern society. Another trend is the recent resurgence of machine learning which is especially timely for smart grid applications. However, a major deterrent in effectively utilizing machine learning algorithms is the lack of *labelled training data*. We overcome this issue in the specific context of smart meter data by proposing a flexible framework for generating synthetic labelled load (e.g. appliance) patterns and usage habits via a non-intrusive novel data-driven approach. We leverage on recent developments in generative adversarial networks (GAN) and kernel density estimators (KDE) to eliminate model-based assumptions that otherwise result in biases. The ensuing synthetic datasets resemble real datasets and lend to rich and diverse training/testing platforms for developing effective machine learning algorithms pertaining to consumer-side energy applications. Theoretical and practical studies presented in this paper highlight the viability and superior performance of the proposed framework.**

***Index Terms*—Smart grids, Machine learning algorithms, Demand-side management, Statistical learning**

I. Introduction

The proliferation of advanced sensors, communication systems and automation processes in today's grid is offering tremendous opportunities for efficient, sustainable and resilient system operations [1]. As such, in the consumer-side, three main trends can be observed: 1) Widespread adoption of the advanced metering infrastructure (AMI) by electric power utility companies (EPUs) for billing and demand-response purposes; 2) Increased deployment of home energy management systems (HEMS) by consumers for automating local demands; and 3) Internet of things (IoT) paradigm enabling seamless connectivity among all types of day-to-day devices. One outcome of this changing landscape is the continuous generation of energy datasets. For example, an EPU managing 1 million consumers has an annual data intake rate of 1000 TB [2]. Although vast volumes of datasets are produced continuously in the grid, in order to obtain any meaningful insights, more granular and detailed information is necessary.

For instance, consider a smart meter which has the capability to record cumulative power usage by the corresponding household every one minute to one hour intervals. More granular labelled information such as that indicating which load was active at each recording period will enable HEMS and EPUs to utilize advanced machine learning algorithms to make passive/active recommendations that allow consumers to modify local demands in a manner that is conducive for reducing carbon footprint and costs. Additional sensors such as circuit meters can be installed at each load to obtain these labelled datasets. However, this is intrusive and not scalable for a large number of households. Another option will be to utilize load profiles made available in public datasets (e.g. [3], [4]). Available public datasets are limited in terms of dataset size, location, usage modes, season, and sampling frequencies.

In this paper, we focus on overcoming these limitations by proposing a novel framework for generating labelled synthetic datasets that capture power consumption patterns and usage habits for individual loads (e.g. appliances). Datasets provided in references [5] and [6] are used for training the framework presented in this paper. Our framework will learn the underlying distributions of load operations and ultimately generate synthetic samples from these learned distributions. Once trained, there will be no limit on the number of samples that can be drawn from the trained framework. Every "type" of load is associated with a label (or alternately referred to as *condition*) which allows for differentiated training of loads representative of various attributes (e.g. location, etc.).

Existing literature can be divided into two classes: *model-based* and *data-driven* approaches. With the model-based approach, the operational behaviour of a load is captured by a set of mathematical equations that are derived from knowledge of the physics of the load and its electrical attributes. For instance, reference [7] models select loads using MATLAB to generate their power demand profiles. This approach requires extensive knowledge of each load's physical characteristics and this greatly limits the flexibility of modelling various types of loads. Furthermore, the usage habits of these loads are usually set by the user. Another approach is to model loads using pre-set power demand curves and probability of operation [8]. In reference [9], machine learning constructs such as recurrent neural networks are utilized to learn the power-voltage relationships of various loads. As such, reference [10] provides a comprehensive overview of various approaches utilized to model loads in the literature. Biases are inherent in these model-based approaches mainly due to the assumptions made regarding the operation of loads.

With data-driven approaches, no prior assumptions are made regarding the operational characteristics of loads which adds inherent flexibility to the proposed frameworks. Generative methods that include Gaussian Mixture and Markov Models have been utilized in references [11] and [12]. Other work that utilize machine learning techniques include references

S. El Kababji is with the Department of Electrical and Computer Engineering, Western University, London, ON, Canada and P. Srikantha is with the Department of Electrical Engineering and Computer Science at York University, Toronto, ON, email: selkaba@uwo.ca and psrikan@yorku.ca.

[13] - [15]. More recently, reference [16] utilized Generative Adversarial Networks (GAN) to generate synthetic power profiles for renewable energy resources. On the demand side, GAN has been leveraged in references [17] - [20]. However, these proposals focus on either generating aggregate power consumption patterns at the household level or higher (e.g. neighbourhood, etc.) - not at individual load levels. These datasets synthesize measurements typically recorded by smart meters. While these models will be useful in load flow and planning studies, they cannot be utilized for applications that require operational characteristics of individual loads.

Hence, our work differs from prior work as we utilize GAN and Kernel Density Estimator (KDE) for generating patterns and habits for individual loads in a household. The synthetic profiles generated by our proposed framework can be readily applied to a wide range of consumer-based smart grid applications that include demand response, energy management and non-intrusive load monitoring applications. Furthermore, our proposal offers a bottom-up approach where low-level load profiles can be aggregated over households, neighbourhoods and so on to aid with studies that utilize aggregate datasets (e.g. load flow/planning studies). As per our knowledge, this is the first time the GAN construct is utilized to generate synthetic profiles at the individual load level.

As such, the major contributions of this paper are four-fold: 1) We propose a flexible framework to generate synthetic load *patterns* and usage *habits* for individual loads that can be tailored to specific households with no model-based assumptions; 2) We automate pre-processing of training data using signal processing techniques (e.g. matched filter); 3) We present practical and theoretical studies of the performance of the proposed synthetic data generation framework; and 4) We perform comparative studies for generating usage habits. Specifically, the two key constructs leveraged in the proposed framework to enable the generation of realistic labelled datasets are: 1) Generative adversarial network (GAN) and 2) Kernel density estimator (KDE). We have optimized various components in these constructs so that stability in the learning process can be guaranteed. Furthermore, we have identified unique evaluation metrics to gauge the "performance" or discernibility of the synthetic datasets from real datasets.

The remainder of this paper is organized as follows. In Sec. II, the general framework for producing labelled synthetic patterns and usage habits for individual loads is introduced. Then, in Sec. III, the automated pre-processing of datasets for the training of the proposed framework is presented. Sec. IV and V introduce the proposed load pattern and usage habit synthesis systems respectively. Comparative, practical and theoretical studies of these systems are also presented in these sections. Finally, we conclude in Sec. VI.

II. Proposed Framework

The proposed framework aims to address the lack of labelled training data pertaining to the operational patterns and usage habits of individual loads. As such, consider Fig. 1.

The left subfigure outlines the power consumption profile of a clothes dryer (CDE) during a specific operation mode which

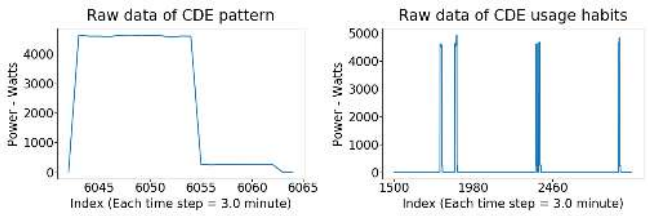


Fig. 1: Sample CDE pattern and usage habits.

we refer to be the load pattern for the CDE in this paper. This has been manually extracted from the raw measurements recorded by circuit meter readings of the same CDE over a three-day period and this is illustrated in the right subfigure. From this time-series data, it is also possible to discern all time instances at which the CDE was activated. This will provide insights into the usage habits of the load. These raw datasets have been obtained from the publicly available The Almanac of Minutely Power (AMP) dataset [5], [21] and The Rainforest Automation Energy (RAE) dataset [6]. The AMP dataset contains power consumption readings recorded by 12 circuit meters where each connects to one load (i.e. total of 12 loads) residing in a single household. These measurements have been obtained over a sampling period of 1 minute across 2 years. As for RAE dataset, the sampling interval is 1 second.

Although there exist other datasets that report similar measurements like the AMP and RAE datasets such as those listed in reference [22], these are not suitable to train machine learning algorithms. For instance, some of the datatsets lack the time stamps while others record power measurements at high sampling frequency for short duration. Due to privacy issues and the intrusive nature of deploying circuit meters, these datasets containing power consumption measurements for individual loads are not widely available. We aim to expand these datasets by synthetically producing load patterns (similar to left subfigure in Fig. 1) and usage habits (similar to the start times of cycles in the right subfigure of Fig. 1). As we utilize machine learning techniques to enable this, the proposed framework must be *trained* to enable the generation of synthetic datasets that closely resemble real datasets. After training, this framework must be *flexible* and *easy-to-use*.

A. Training of the Data Synthesis Process

Training of the synthesis process entails three main stages as outlined in Fig. 2: 1) *Pre-processing* of training data; 2) *Training* of the load pattern and usage habit generation modules; and 3) *Evaluation* of how realistic the synthesized dataset is.

The training data will be obtained from two sources: real measurement data (e.g. [5] and [6]) and input from stakeholders (e.g. feedback from consumers, manufacturers, etc.). Real measurement data is typically supplied in raw form for each load and must be pre-processed for uniformity. Then, feature engineering is applied to extract important attributes from this training data for the realistic synthesis of labelled datasets as discussed in detail in Sec. III. Inputs from stakeholders (e.g. consumers, manufacturers, etc.) corresponding to load patterns and usage habits also form training datasets. This

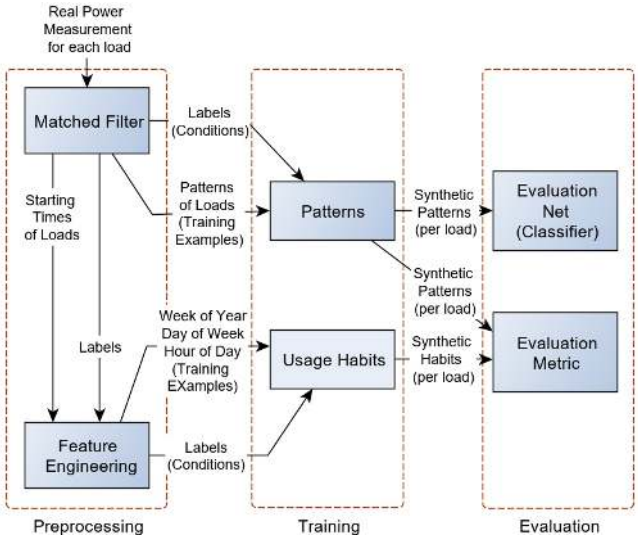


Fig. 2: Proposed framework training process.

information can be readily obtained from mobile applications such as Trickl from London Hydro [23].

Next, in the training stage, these pre-processed datasets are utilized to improve the weight parameters of the neural networks which are used in load pattern and usage habits synthesis modules. Appropriate loss functions will be selected in the design of these modules so that these can be trained in a stable manner to generate realistic datasets. As there is no direct measure for gauging how realistic the synthetic dataset is, we use two different metrics for this purpose in the evaluation stage: 1) We utilize a distance measure that computes similarities between synthetic and real datasets; and 2) We design a neural network based system called the Evaluation Net to mimic visual inspection by humans to discern realistic from unrealistic data.

B. Input and Outputs of Trained Generator

As outlined in Fig. 3, once the synthesis framework is trained, it is utilized to produce labelled datasets for specific loads. The input into the framework is the name of the

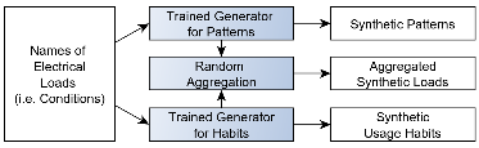


Fig. 3: Engaging with the trained framework.

electrical load to be synthesized. Outputs from the modules will produce labelled synthetic samples pertaining to the corresponding load's patterns and usage habits. These can then be combined by an aggregator to generate cumulative load profiles. Thus, the framework can flexibly produce synthetic datasets both at individual load and aggregate levels.

III. Pre-processing Training Datasets

In this section, we focus on the automatic pre-processing of inputs to extract pertinent features that will effectively train the proposed synthesis modules. Manually extracting features from training samples will be cumbersome and become intractable when processing raw measurement inputs collected over long time periods from numerous circuit meters (e.g. AMP dataset). For data obtained via mobile applications like Trickl, there exist application programming interfaces (APIs) that allow for the necessary rendering of collected data. Thus, we focus on pre-processing raw measurements collected from circuit meter data in this section. In the following, we first present pertinent features that we have selected for training each one of the synthesis modules. Then, our approach for automating the extraction of these features is detailed.

A. Features Selected

For the training of the load pattern synthesis module, raw time-series datasets similar to the right sub-figure in Fig. 1 recorded for various loads must be processed in order to obtain specific load patterns for different operational modes. The left sub-figure in Fig. 1 is one example of such a pattern extracted from the time-series data generated over a long time interval for CDE. We use a construct called the matched filter that automates the process of extracting load usage patterns from individual circuit meter data obtained for loads under study. The matched filter utilizes a template to extract load usage patterns that are similar (not necessarily identical) to the template. This template can be defined by the data engineer where he or she selects power measurements corresponding to one active period of load operation in the training dataset. This approach allows for greater flexibility in comparison to rule-based approaches that require extensive prior knowledge about the operation of various types of loads and utilize assumptions which can introduce implicit biases (e.g. Gaussian noise/error).

Algorithm 1 Computation of T_w and T_l

Input:
Unified granularity interval in seconds (T_g)
Dataset sampling interval in seconds (T_s)
Template array Arr_l for loads $l \in 1 \dots N$
Output:
Master window (T_w)
Operation cycle for load l (T_l) $\forall l = 1...N$
for l=1 to N **do**
  Ceiling($T_l \leftarrow length(Arr_l) * T_s/T_g$)
end for
$T_w \leftarrow \max\{T_1, ..., T_l, ..., T_N\}$

Our framework allows for training *conditional* GAN using multiple datasets which may utilize different sampling frequencies to record power data. We define a unified granularity interval T_g in seconds to accommodate various sampling frequencies present in these datasets. Typically, T_g is larger than or equal to the largest sampling interval utilized in the training datasets. Hence, datasets are down-sampled to T_g. For illustrative purposes, consider the case where the training dataset is composed of time-series power measurements for N types of loads that are sampled every T_s seconds. Each load l will be associated with a manually selected template

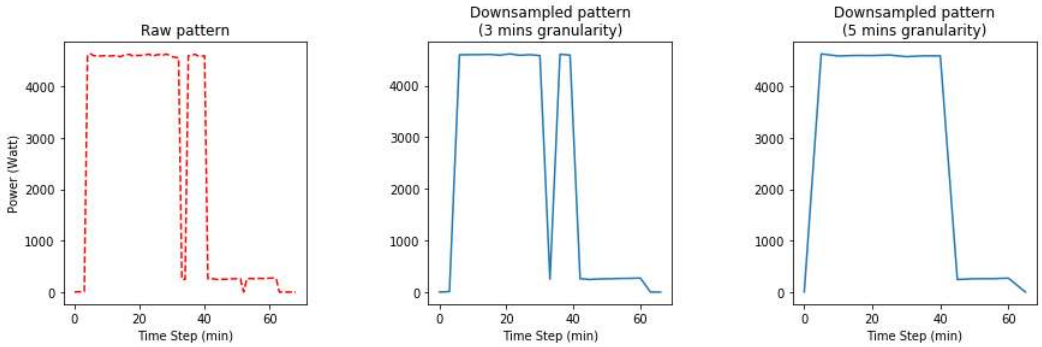


Fig. 4: Impact of reducing granularity (T_g) on training samples for cloth dryer.

Arr_l. The template is an array of values representing one set of power measurements when the load is in active operation. These templates have time intervals that can differ from load to load. We introduce T_l and T_w for loads $l \in 1...N$ which are computed according to Alg. 1. T_l and T_w are unit-less integer values. T_l represents the number of samples in a load's template after the downsampling operation. T_l is used in the construction of the matched filter as detailed in Sec. III-B. T_w determines the input dimension of the GAN used to generate patterns. When there are multiple datasets, the value of T_w is set to be the maximum value computed across the datasets. Similarly, T_l is also selected to be the maximum value computed across the datasets.

If $T_l < T_w$, then zero padding is applied for the remaining $T_w - T_l$ samples. The template for load l can be selected to be any one instance that occurs in the dataset via manual inspection by the data engineer. In other words, T_g is the only value that is defined for feature extraction from the training datasets. Other parameters (e.g. T_s and Arr_l) are obtained from these datasets. When T_g takes a smaller value, greater will be the number of samples in each template. This, however, leads to higher dimensionality in the training dataset. On the other hand, when T_g is larger, distinguishing features pertaining to the load's operation will be lost as illustrated in Fig. 4. In this figure, it is clear that when T_g is increased from 3 minutes to 5 minutes, the second peak in the operational cycle of the cloth dryer is lost. Thus, a balance must be struck in the selection of the value T_g. For the datasets utilized in this paper, T_g is empirically selected to be 180 seconds. Table I lists the parameters T_g, T_l and T_w corresponding to the four appliances considered in this paper.

As these patterns are extracted, the corresponding start-time of operation is also captured as this provides insights into the usage habits of that particular load. The inputs into the usage habits synthesis module are: *the week of year, day of week and hour of day* information extracted from the time stamp corresponding to the beginning of each load pattern.

B. Automated Load Pattern Extraction

Circuit meters record the raw power consumption readings for each load (similar to the right subfigure of Fig. 1) and these are typically accompanied with measurement noise. Furthermore, when the sensor is deployed over a long time span (e.g. two years), there will be multiple instances at which the load has been active. Thus, to automate the extraction of these load patterns from noise-ridden time-series data, we utilize the signal processing concept known as matched filter. This filter maximizes the signal to noise ratio when additive stochastic noise is present in measurement signals [24]. Although matched filter is traditionally utilized in radar applications, it efficiently performs pattern extraction as pertaining to our application.

General characteristics of the power consumption patterns associated with a load are similar for various modes of operation. For each load $l \in \mathcal{L}$, we define a down-sampled "template" pattern $s_l[n]$ depicting one operational cycle of a load where n is a time step. This can be manually extracted from down-sampled datasets or constructed on our own. For a load l with an operational cycle of T_l, the matched filter $h_l[n]$ is defined to be a shifted and time-reversed version of this template (i.e. $h_l[n] = s_l[T_l - n]$). This filter is then convolved with the time-series down-sampled dataset $p_l[n]$ that corresponds to the same load l to result in $y_l[n]$ as follows:

$$y_l[n] = \sum_{k=-\infty}^{\infty} p_l[k] h_l[n-k] \tag{1}$$

As the filter moves through the time-series dataset from the beginning to the end, it encounters load patterns that are similar to the template. This is detected by examining the output $y_l[n]$ of the convolution process (listed in Eq. 1) which evaluates to a value larger than a pre-set threshold λ_l. When this occurs, it can be inferred that a load pattern has been encountered. This pattern will then be extracted along with the corresponding timestamp marking the beginning of the detected pattern. These form training inputs for the synthesis modules that are collected from circuit meter readings.

The following is a set of guidelines that we have utilized for selecting λ_l. Consider the discrete time series $y_l[n]$ resulting from the convolution of the matched filter $h_l[n]$ with the time series of physical power measurements $p_l[n]$. We first identify the largest value resulting from the convolution, i.e. $\max\{y_l\}$. This occurs when the matched filter detects power measurements that are similar to the load's template. To allow for the detection of patterns with greater variability for that load, λ_l is set to be a fraction r_l of the maximum value identified earlier (i.e. $\lambda_l = r_l \times max\{y_l\}$). When r_l is higher, the patterns detected will be very similar to the load template.

On the other hand, when r_l is smaller, random patterns that do not correspond to the actual operation of the load will be detected. Thus, a balance between these two tradeoffs must be struck when selecting r_l. Table I lists r_l that have been empirically selected for each load l.

Load	Cloth Dryer	Dishwasher	Fridge	Heat Pump
T_l	26	42	32	19
r_{th}	0.4	0.6	0.4	0.75
T_g (sec)	180			
T_w	42			

TABLE I: Pre-processing parameters for various loads.

IV. Load Patterns Synthesis

Equipped with labelled training data, in this section, we present the design, training and evaluation processes associated with the proposed load pattern synthesis module. The machine learning concept called *conditional GAN* is utilized to construct the load pattern synthesis module. GAN was originally proposed in reference [25] to generate synthetic images. We differ considerably from the traditional GAN application as our intent is to synthesize load patterns which are one-dimensional time-series data over a fixed window T_w instead of a two-dimensional image. Thus, the architecture and training methods utilized for synthesizing images cannot be directly transferred to our application. As such, we provide an overview of GAN in the following, prior to delving into the details of the module design.

A. GAN Overview

There are two main components in GAN: the *generator* and the *discriminator* as illustrated in Fig. 5.

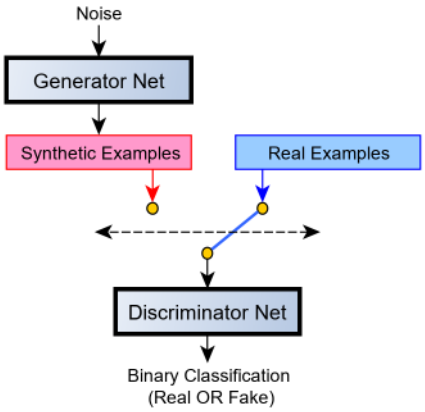


Fig. 5: Summary of a GAN system.

The generator and discriminator are two different neural networks. In a typical GAN, the generator takes in as input random noise z of n_z dimensions (i.e. $z \in \mathbb{R}^{n_z}$) sampled from the probability distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$. z is utilized by the generator to trigger the generation of a synthetic pattern. The input x into the discriminator is either the synthetic pattern produced by the generator or a real pattern obtained from the training data. x belongs to the n_x dimensional real space (i.e. $\mathbb{R}^{n_x}$). The task of the discriminator is to determine the probability of input x being real (i.e. not synthetically produced by the generator).

The generator and discriminator are both trained simultaneously. However, the objectives of these entities are opposite to one another. The generator aims to *minimize* the probability of the discriminator correctly identifying its output as synthetic. Ultimately, the generator is trained to render its output indistinguishable from the real training samples. The discriminator, on the other hand, aims to *maximize* the probability of correctly identifying the source of the input samples (i.e. synthetic or real). The penalties/costs imposed for deviating from the goals set by each component are utilized to optimize the internal neural network parameters until desired performance is attained. These opposing goals lead to a min-max game and this alludes to the "adversarial" component of GAN.

B. Cost Function and Module Training

In this paper, we consider the neural networks pertaining to both the generator and discriminator to be multi-layer perceptrons (MLPs). We do not utilize convolutional neural networks which are primarily leveraged in the GAN literature as we are not dealing with images. The cost function employed to train the GAN plays an important role in the stable fine-tuning of MLP parameters to synthesize realistic outputs. The original cost function $V(D,G)$ proposed in reference [25] is:

$$V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{d}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \quad (2)$$

where $\mathbb{E}$ is the expectation with respect to the random variable specified in the subscript, D is the output of the discriminator (i.e. probability of whether the input is real), G is the output of the generator (i.e. the synthetic pattern), $p_{\mathrm{d}}(\boldsymbol{x})$ is the probability distribution of the real data x, and z is the noise drawn from the probability distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$. The generator aims to minimize $V(D,G)$ and the discriminator aims to maximize $V(D,G)$. It is well-known in the literature that the GAN system is notoriously hard to train [26]–[29]. In the specific case of our application, this cost function causes the training process to diverge as the penalties imposed by $V(D,G)$ result in saturating the gradients that are used to optimize the MLP parameters. To overcome this issue, we define a new cost function $L(D,G)$ that enables stable training behaviour with desirable outputs.

$$L(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{d}}(\boldsymbol{x})}[\log(1 - D(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log D(G(\boldsymbol{z}))] \quad (3)$$

$L(D,G)$ and $V(D,G)$ are equivalent from the perspective of the discriminator but not from the point of view of the generator as the minimization and maximization takes place over G and D respectively for both loss functions. We show in Theorem 1 that this cost function allows the generator to learn the probability distribution of the original dataset at optimality with no assumptions made regarding the distributions of the real or synthetic datasets. Proof for Theorem 1 is based on an approach that is similar to that listed in reference [25].

Theorem 1: *Performing* $\min_G \max_D$ *operations on the proposed value function* $L(D,G)$ *results in a globally optimal solution*

that requires the probability distributions of real data and synthetic data to be identical.

Proof: The first observation leading to the afore-mentioned theorem is that the optimal output of the discriminator is:

$$D^*(\boldsymbol{x}) = \frac{p_{\mathrm{g}}(\boldsymbol{x})}{p_{\mathrm{g}}(\boldsymbol{x}) + p_d(\boldsymbol{x})} \tag{4}$$

where $p_{\mathrm{g}}(\boldsymbol{x})$ and $p_d(\boldsymbol{x})$ are the distributions of the synthetic and real datasets respectively and $\boldsymbol{x}$ represents the random variable resulting from sampling these distributions. To derive this result, consider the value function L in its extended form:

$$L(D,G) = \int_{\boldsymbol{x}} p_{\mathrm{d}}(\boldsymbol{x}) \log\big(1 - D(\boldsymbol{x})\big)\mathrm{d}\boldsymbol{x} + \int_{\boldsymbol{z}} p_{\boldsymbol{z}}(\boldsymbol{z}) \log D(G(\boldsymbol{z}))\mathrm{d}\boldsymbol{z} \tag{5}$$

Knowing that the distribution of data synthesized by the generator is $p_{\mathrm{g}}(\boldsymbol{x})$, the second integral in the above expression can be replaced with $\int_{\boldsymbol{x}} p_{\mathrm{g}}(\boldsymbol{x}) \log D(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$. This results in the following expression:

$$L(D,G) = \int_{\boldsymbol{x}} \Big[p_{\mathrm{d}}(\boldsymbol{x}) \log\big(1 - D(\boldsymbol{x})\big) + p_{\mathrm{g}}(\boldsymbol{x}) \log\big(D(\boldsymbol{x})\big)\Big]\mathrm{d}\boldsymbol{x} \tag{6}$$

Using the property that the value of y that maximizes the expression $m\log(y) + n\log(1-y)$ for any $(m,n) \in [\mathbb{R}^2 \backslash \{0,0\}]$ is $\frac{m}{m+n}$, we identify that the maximum of L occurs at $D^*(\boldsymbol{x}) = \frac{p_{\mathrm{g}}(\boldsymbol{x})}{p_{\mathrm{g}}(\boldsymbol{x}) + p_d(\boldsymbol{x})}$. This expression is utilized to derive the objective function pertaining to the generator:

$$P(G) = \max_D L(D,G) = L(D^*,G) \tag{7}$$

$$= \int_{\boldsymbol{x}} \Big[p_{\mathrm{d}}(\boldsymbol{x}) \log\big(1 - \frac{p_{\mathrm{g}}(\boldsymbol{x})}{p_{\mathrm{g}}(\boldsymbol{x}) + p_d(\boldsymbol{x})}\big) + p_{\mathrm{g}}(\boldsymbol{x}) \log\big(\frac{p_{\mathrm{g}}(\boldsymbol{x})}{p_{\mathrm{g}}(\boldsymbol{x}) + p_d(\boldsymbol{x})}\big)\Big]\mathrm{d}\boldsymbol{x} \tag{8}$$

$$= \int_{\boldsymbol{x}} \Big[p_{\mathrm{d}}(\boldsymbol{x}) \log\big(\frac{p_{\mathrm{d}}(\boldsymbol{x})}{p_{\mathrm{g}}(\boldsymbol{x}) + p_d(\boldsymbol{x})}\big) + p_{\mathrm{g}}(\boldsymbol{x}) \log\big(\frac{p_{\mathrm{g}}(\boldsymbol{x})}{p_{\mathrm{g}}(\boldsymbol{x}) + p_d(\boldsymbol{x})}\big)\Big]\mathrm{d}\boldsymbol{x} \tag{9}$$

With a GAN system, the main goal for the generator is to learn the distribution of the real dataset at optimality. Hence, let us assume that this is indeed the case (i.e. $p_{\mathrm{g}} = p_{\mathrm{d}}$) and substitute this into D^*. This results in $D^* = \frac{1}{2}$ which then implies that $P(G) = -\log(4)$ from the above expression. Subtracting $-\log(4)$ from both sides of the above expression results in:

$$\begin{aligned} P(G) = -\log(4) &+ \int_{\boldsymbol{x}} \Big[p_{\mathrm{d}}(\boldsymbol{x}) \log\big(\frac{p_{\mathrm{d}}(\boldsymbol{x})}{p_{\mathrm{g}}(\boldsymbol{x}) + p_d(\boldsymbol{x})/2}\big)\Big]\mathrm{d}\boldsymbol{x} \\ &+ \int_{\boldsymbol{x}} \Big[p_{\mathrm{g}}(\boldsymbol{x}) \log\big(\frac{p_{\mathrm{g}}(\boldsymbol{x})}{p_{\mathrm{g}}(\boldsymbol{x}) + p_d(\boldsymbol{x})/2}\big)\Big]\mathrm{d}\boldsymbol{x} \end{aligned} \tag{10}$$

Each integral term represents Kullback-Leibler divergence which is defined as: $D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x)\log(\frac{p(x)}{q(x)})dx$ and $P(G)$ is expressed using this measure as follows:

$$P(G) = -\log(4) + D_{KL}(p_{\mathrm{d}}||\frac{p_{\mathrm{d}} + p_{\mathrm{g}}}{2}) + D_{KL}(p_{\mathrm{g}}||\frac{p_{\mathrm{d}} + p_{\mathrm{g}}}{2}) \tag{11}$$

The summation of the D_{KL} terms results in the Jensen-Shannon divergence which is defined as: $JSD(P||Q) = \frac{1}{2}\big(D_{KL}(P||R) + D_{KL}(Q||R)\big)$ where $R = (P+Q)/2$. This is applied to $P(G)$:

$$P(G) = -\log(4) + JSD(p_{\mathrm{d}}||p_{\mathrm{g}}) \tag{12}$$

The JSD term is non-negative and is 0 when both distributions are identical. Hence, the minimum of $P(G)$ is achieved when both the synthetic and real distributions are identical. The resulting optimal value of $P(G)$ is $-\log(4)$. □

While training the GAN using $L(D,G)$, we noticed that at the initial stages, the probability distributions of the real and synthetic datasets are not close to one another. When this occurs, the cost function saturates especially when the log terms tend to $-\infty$. In order to prevent this saturation, we utilize the cost $-\log(1 - D(G(z)))$ to train the generator in a stable manner without any loss of generality. This change in the generator cost has been applied in the original GAN paper [25] as well to avert these saturation issues.

C. Evaluation Metrics

Once the GAN is trained, it is necessary to evaluate its performance. However, the main challenge lies in how to evaluate the "performance" of the synthetic patterns. In the traditional GAN, the images generated are visually inspected by humans to evaluate whether these look real or not. Results of this assessment are then utilized to gauge the performance of the system. This is an arduous task that is qualitative in nature and therefore subject to bias. We consider two other approaches for performance evaluation.

The first approach is to utilize the maximum mean discrepancy (MMD) measure which identifies the "distance" between two probability distributions. In our case, the two distributions pertain to the synthetic load patterns and the real samples. MMD is defined as follows:

$$\begin{aligned} MMD = \Big(\frac{1}{m(m-1)} \sum_{i=1}^{m}\sum_{j\neq i}^{m} k(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i=1}^{n}\sum_{j\neq i}^{n} k(y_i, y_j) \\ - \frac{2}{mn} \sum_{i=1}^{m}\sum_{j=1}^{n} k(x_i, y_j)\Big)^{\frac{1}{2}} \end{aligned} \tag{13}$$

where x_i is the i^{th} real sample, y_i is the i^{th} synthetic sample, and m and n are the total number of real and synthetic samples respectively. $k(.)$ is a Gaussian Radial Kernel defined as:

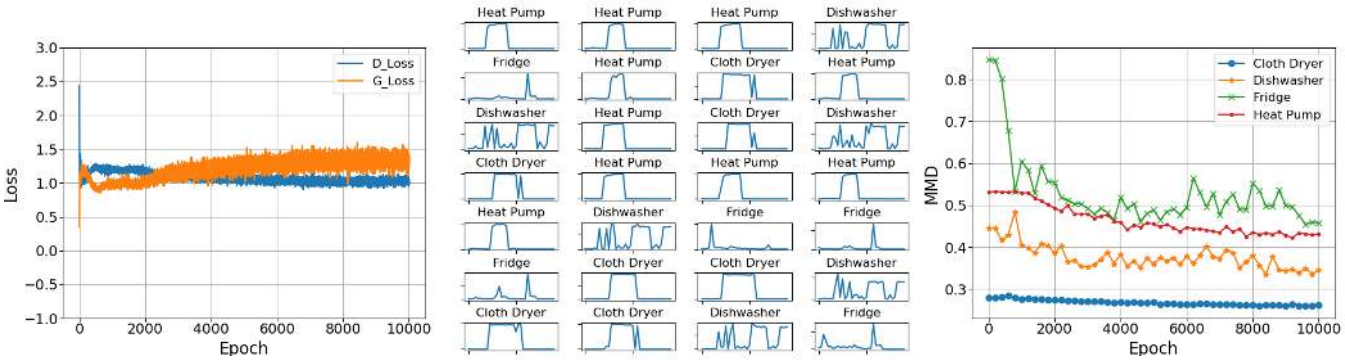
$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right) \tag{14}$$

where σ is a free parameter.

Our second approach is to mimic the visual inspection process by humans and automate this by training a neural network. We refer this classifier to be the Evaluator Net which serves to identify which class (e.g. dishwasher, dryer, etc.) the sampled synthesized load patterns fall under. Thus, if the synthetic samples resemble realistic data, then the Evaluator Net must correctly identify which load the input pattern represents. The actual performance of our model based on the afore-mentioned evaluation metrics is presented in Sec. IV-E.

D. Architecture

The architecture of the generator and discriminator components of the GAN along with the Evaluator Net is presented next. In order to allow for flexibility in the training and implementation processes, we combine the synthesis of



(a) Evolution of L during training.

(b) Synthetic patterns from proposed GAN.

(c) MMD for the four loads.

Fig. 6: Training and synthesis of the GAN system.

patterns for all loads into a single GAN system. For training or synthesizing the pattern of a specific load, labels that are one-hot encoded are appended to the corresponding inputs as discussed in Sec. II.

The set $\mathcal{L}$ represents various types of loads that are supported by the proposed framework. Since loads are typically identified by labels (e.g. dishwasher, fridge, etc.) these should be converted to numerical values in order to be processed as conditions in the proposed modules. Assigning an integer value for each load implies ordinal relationship amongst these. In order to avoid this implicit bias, we utilize one-hot encoding which is a unit vector representation of labels. For example, for a set of four loads, one load is assigned the label of [0 0 0 1], the other load is assigned the label of [0 0 1 0] and so on. Inputs into the pattern synthesis module will be the extracted load pattern of length T_w concatenated with the corresponding one-hot encoded label. In order to distinguish patterns generated for loads based on other external factors such as different locations and seasons, conditions assigned to load patterns can be extended to reflect these attributes.

We utilize the notion of *conditional GAN* [30] where the training of the system and synthesis of patterns are conditioned upon the label appended to the input. This simplifies the system as dedicated GAN will not be necessary for each load. Thus, loads can be added as needed into the synthesizing system in a flexible manner. Table II contains the specifications we have utilized to construct the GAN system to synthesize load patterns for four loads (clothes washer, dishwasher, fridge and heat pump). This system is constructed and trained using Google's TensorFlow Python library.

The input dimensions for the generator is $n_z = 100$ plus the one-hot label of size 4 (to represent all four loads uniquely using one-hot encoding) (i.e. nodes in layer 1 (L1) is 104). Input dimensions of the discriminator is $n_x = T_w$ plus the one-hot label which is also of size 4. Both neural networks are composed of 5 layers including the input layers. The number of nodes per layer (with the exception of the first layer which is the input layer) and the activation functions used at each layer are specified in Table II. These have been selected via empirical experiments as typical in the machine learning literature. Training samples are generated using the preprocessing technique outlined in Sec. II. Training samples and synthesized pattern are constructed over a period of 126 minutes (i.e. $T_w \times T_g$).

Cost Function	$L(D, G)$
Training Datasets	From references [5], [6]
Unified Granularity T_g	180 seconds
Master Window T_w	42
Conditions (loads)	4 (One-hot Encoded)
Generator	5 Layers
Nodes/layer:	L1: 104, L2: 100, L3: 150, L4: 100, L5: 42
Activation/layer	Leaky ReLU, Leaky ReLU, Leaky ReLU, Tanh
Discriminator	5 Layers
Nodes/layer	L1: 46, L2: 100, L3: 150, L4: 100, L5: 1
Activation/layer	Leaky ReLU, Leaky ReLU, Leaky ReLU, Sigm

TABLE II: Load patterns GAN architecture.

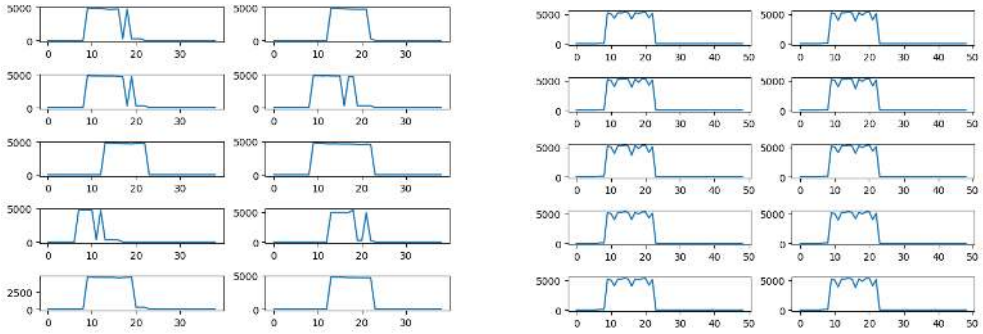
Cost Function	Categorical Cross Entropy
Training Datasets	From references [5], [6]
Features	42
Classes	4 (One-Hot Encoded)
Layers	6
Nodes/layer	L1: 42, L2: 8, L3: 10, L4: 10, L5: 10, L6: 4
Activation/layer	ReLU, ReLU, ReLU, ReLU, Softmax

TABLE III: Evaluator Net architecture.

The architecture of the Evaluator Net is listed in Table III. The loss function utilized to train the six-layer MLP is referred to as categorical cross entropy [31]. The data points are constructed from both references [5] and [6]. They are split into three equal parts for: training, validation and testing in order to avoid overfitting.

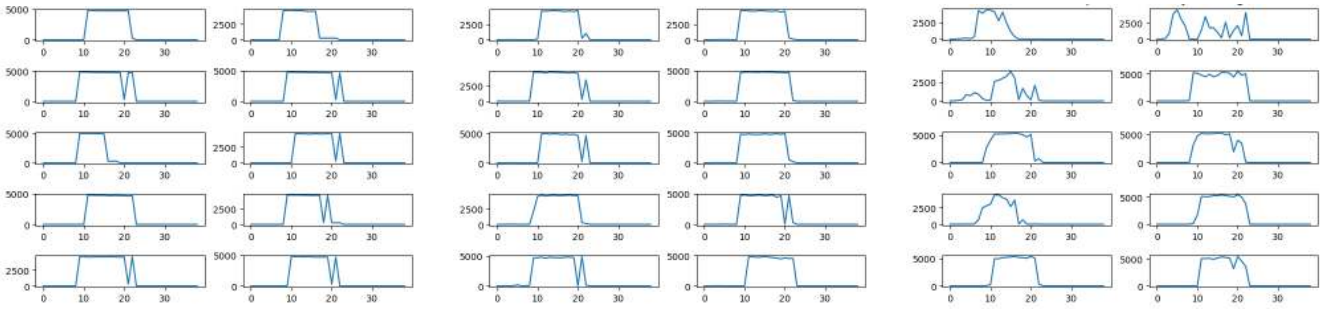
E. Performance

The performance of the proposed GAN system with respect to the two evaluation metrics discussed in Sec. IV-C is presented in the following. First, in Fig. 6a, we examine the evolution of $L(D, G)$ as the GAN system is trained for the generator (i.e. G_{Loss}) and discriminator (i.e. D_{Loss}). It is clear that the system trains without divergence and converges to a steady equilibrium. In Fig. 6b, randomly selected samples of synthetic load patterns generated by the proposed GAN system



(a) Random real patterns. (b) Collapsed synthetic patterns.

Fig. 7: Mode collapse in GAN for patterns (patterns shown for Cloth Dryer).



(a) Real Patterns (b) Synthetic Patterns (Proposed loss) (c) Synthetic Patterns (Wasserstein loss)

Fig. 8: Comparison of real versus synthetic patterns.

for all four loads are presented. The x-axis represents the pattern window with 42 intervals and the y-axis is the power consumption of the load in Watts. By visually inspecting these load profiles, it is clear that these resemble actual patterns of the corresponding loads.

Next, in Fig. 6c, the evolution of the MMD measure for each one of the loads trained by the GAN is recorded during the training period. For all four loads, the distance between the probability distributions of the synthetic and real datasets decrease during the training process.

In traditional supervised learning models, there exists the notion of bias-variance tradeoff where it is shown that with appropriate selection of the hypothesis set, increasing the number of training samples will result in better generalization and elimination of underfitting/overfitting. However, with generative models like GAN, it is shown in recent work like reference [32] that lower number of samples in the training dataset will result in better performance than using a large training dataset. This is the case with our proposed framework where we are looking into expanding a small labelled training dataset into a larger one. However, there exists a phenomenon in generative models that is similar to the notion of overfitting and this is referred to as *mode collapse*. When mode collapse occurs, the generator replicates the same output every time it is queried. We illustrate this in Fig. 7 for our model. In the original paper on GAN (i.e. reference [25]), mode collapse was prevented during the training process where the discriminator was updated over k iterations every time the generator was updated (k is a hyper-parameter that depends on the dataset being synthesized). As our cost function is different from that proposed in reference [25], we utilize a different approach where the generator and discriminator are each updated once after each other (i.e. $k = 1$). This resulted in successfully eliminating the issue of mode collapse.

	Cloth Dryer	Dishwasher	Heat Pump	Fridge
$V(D,G)$	-	-	-	-
$L(D,G)$	0.06	0.14	0.18	0.23
$R(D,G)$	0.08	0.20	0.29	0.69

TABLE IV: MMD for various objective functions.

In Table IV, we assess the impact of the cost function selected to train the GAN system. Specifically, we examine three types of cost functions: the traditional cost function V [25], cost function L proposed in this paper and the regularized cost function R [33]. V and R are utilized widely in the GAN literature and represent the existing literature in the comparison of the proposed cost function L. As mentioned earlier, V results in the divergence of the training process, it is not possible to compute the MMD for this case. The MMD for our cost function is lower than that obtained for R for all four loads. Hence, it is clear that for this application, L results in superior performance in terms of the distance between the probability distributions p_g and p_d.

Finally, the performance of the GAN is assessed by applying the Evaluator Net and the corresponding results are presented in the confusion matrix listed in Fig. 9. The confusion matrix provides a break-down of the proportion of correct and incorrect predictions. As such, if the synthetic data is indiscernible from the real data, the classifier will predict the true class of

the inputs (i.e. represented by the diagonal elements of the confusion matrix). In our case, only 1% of the synthetic data generated by our model for a cloth dryer were incorrectly identified as belonging to a heat pump. The remaining three loads were correctly identified with 100% accuracy. This process mimics the visual inspection of the synthetic dataset. As the testing data (i.e. synthetic patterns) is not used for the training of the Evaluator Net, overfitting does not take place.

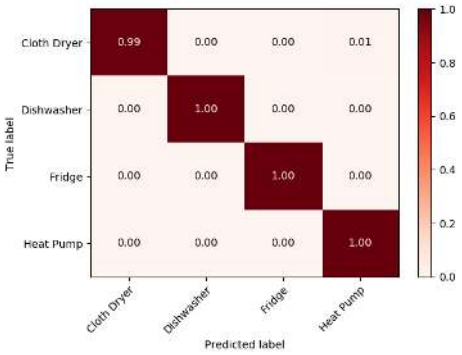


Fig. 9: Normalized confusion matrix for synthetic data.

To further verify the efficacy of the proposed cost function in generating load patterns, the Wasserstein loss function defined in reference [34] is utilized to synthesize load patterns. Wasserstein loss allows for increased gap between synthetic and real images during the training process. For this reason, the patterns generated using the Wasserstein loss do not represent load patterns. Fig. 8 presents randomly selected load patterns generated by our proposed loss function, the Wasserstein loss function and training samples for a cloth dryer. It is clear that there are significant discrepancies in the patterns generated by the Wasserstein method.

V. Habits Generation

In this section, we present the design and evaluation of the proposed load usage habits synthesis module. As discussed in Sec. II, three features which are the *week of year, day of week and hour of day* are extracted from the pre-processed data samples to train this module. In order to produce realistic outputs (i.e. when a load is utilized), the module will have to effectively learn the probability density function associated with the actual usage habits of the corresponding load without making any prior assumptions. We study both conditional GAN and KDE to generate synthetic habits. KDE is not suitable for the load pattern module as it takes in high-dimensional inputs [35]. We compare the performance of the module implemented using KDE and GAN via two evaluation metrics detailed later in this section.

A. KDE Overview

KDE is a *non-parametric* density estimation technique which we leverage to generate usage habits for every load considered in the proposed framework. The start time of operation of various loads can differ from one another significantly. For instance, a dishwasher is usually operated at different times by human operators in comparison to an HVAC which is regulated by environmental factors. KDE is utilized to learn the probability densities of usage habits for every type of load considered in the proposed framework. No assumptions regarding the underlying distribution are made in constructing the KDE for habit generation. Other density estimation techniques such as Gaussian Mixture Models make prior assumptions regarding the underlying distribution of datasets. Next, a brief background of KDE is presented.

The density function can be estimated to be [36]:

$$p_l(\boldsymbol{x}) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{\boldsymbol{x} - \boldsymbol{y}_i}{h}\right) \tag{15}$$

where K is a kernel function that is typically a smooth function, h is the bandwidth parameter and the random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to the d-dimensional real space (i.e. $\mathbb{R}^d$). These kernels are centred around every training point $\boldsymbol{y}_1 \ldots \boldsymbol{y}_N$, summed and normalized to approximate the probability density function for load l that has generated these training points. For our framework, d is 3 and N is the total number of samples extracted from the pre-processing step detailed in Sec. III. We consider 6 types of kernels (i.e. Gaussian, Exponential, Tophat, Epanechnikov, Linear and Cosine). For a specific kernel K, the unknown parameter computed during training is h. When h is large, then the variance of the kernel is larger. This will lead to over-smoothing the density function which will mask the fluctuations in the density functions. When h is small, then the variance of the kernel is smaller and this will capture fluctuations too closely and lead to overfitting issues. Both the kernel function K and bandwidth h are optimized using the k-fold cross validation technique [31] outlined next.

B. Kernel and Bandwidth Selection

With the k-fold cross validation technique, the set of available training samples are divided into k partitions (i.e. folds) of equal sizes. The first fold is held-out for validation. Given a specific kernel K, corresponding h is computed using the remaining $k-1$ folds. Then, the log-likelihood score is computed on the validation set and recorded for the resulting h for kernel K. The procedure is repeated k times where a different partition is selected to be the validation set. Then, the k log likelihood scores are averaged. For the habit generation module, we have selected 10-fold cross validation for estimating the KDE parameters. This is justified next.

k	Optimal h	Optimal K	Mean Test Score
2	0.56	Exponential	-4477
3	0.26	Exponential	-2574
4	0.26	Exponential	-1715
5	0.17	Exponential	-1146
6	0.17	Exponential	-871
7	0.17	Gaussian	-713
8	0.17	Gaussian	-594
9	0.17	Gaussian	-569
10	0.17	Gaussian	-447

TABLE V: Optimal parameters for cloth dryer for different k.

Table V lists the best bandwidth and kernel selected for various folds k for the cloth dryer load. The associated mean

log likelihood scores are also recorded. It is clear that as the number of folds increases, the mean test score increases. However, the bandwidth computed for $k > 4$ remains the same. We observe this phenomenon for the dishwasher, fridge, and heat pump as well. With increased number of folds, the computational overhead also increases [37]. Thus, to strike a balance between this overhead and the mean test score, we select $k = 10$ for all loads. Also, $k = 10$ is a common value utilized in the literature for density estimation (e.g. [38]). Table VI lists the optimal parameters obtained for each load using 10-fold cross-validation.

Load	Cloth Dryer	Dishwasher	Fridge	Heat Pump
K	Gaussian	Gaussian	Exponential	Gaussian
h	0.177827941	0.177827941	0.177827941	0.177827941

TABLE VI: Optimized parameters using 10-fold cross-validation.

C. GAN Architecture

To compare the performance of the KDE model with another generative model, we estimate the distribution of the underlying usage habits for each load using a conditional GAN. We construct a 5-layer MLP for both the generator and discriminator. The cost function utilized is the one proposed in this paper (i.e. $L(D, G)$). Specifics of each layer are detailed in Table VII.

Cost Function	$L(D, G)$
Features	3
Conditions (loads)	4
Discriminator	5 Layers
Nodes/Layer	7, 100, 200, 2, 1
Activation/Layer	Leaky ReLU, Leaky ReLU, Leaky ReLU, Sigm
Generator	5 Layers
Nodes/Layer	54, 120, 240, 120, 3
Activation/Layer	Leaky ReLU, Leaky ReLU, Leaky ReLU, Tanh

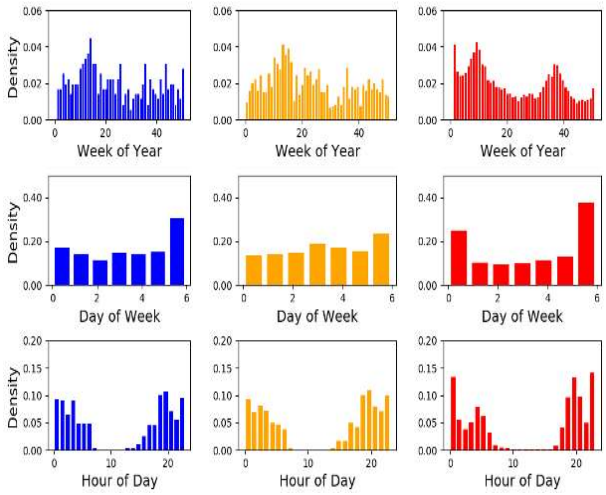
TABLE VII: Load usage habits GAN architecture.

D. Evaluation Metrics

To evaluate the efficacy of utilizing KDE and GAN techniques for synthesizing load usage habits, we leverage on two techniques: one is to construct the histogram of data points that are randomly sampled from the learned density function for visualization purposes and the second is the MMD metric which is utilized to assess the distance between the learned probability and the true probability. With the histogram technique, plots are rendered for each one of the features considered (i.e. 3) as it is difficult to visualize the samples in a d-dimensional space. Data points sampled from the learned probability density function are binned over the appropriate interval regions corresponding to the feature represented by the histogram. The MMD metric introduced in Sec. IV-C is utilized to provide a quantitative measure of the discrepancies between the learned and true density functions.

E. Performance

Next, the performance of the load usage habits synthesis module is examined for both the KDE and GAN based systems using histogram renderings for all three loads and the MMD measure. As such, Fig. 10 illustrates the histograms for all three features. In Fig. 10a, real data points outside of the training set have been organized into the three histograms. It is important to note that the pre-processing of raw circuit meter readings result in data being organized into 3-tuples and thus these features are not originally decoupled from one another. For the ease of visualization, we have separated these tuples into three different histograms. From Fig. 10, it is clear that there are discernible patterns in the frequencies at which the real datapoints are binned together.



(a) Real (b) KDE (c) GAN

Fig. 10: Histograms for load usage habits.

In Fig. 10b, the three-dimensional samples drawn from p learned via the KDE technique are rendered in three histograms. Similarly, the usage habits synthesized by the GAN are illustrated in the last set of histograms in Fig. 10c. It is clear that the underlying trends visible in the real dataset have been captured in both histograms produced using synthetic load usage habits. The histogram plots for the synthetic data will not be identical to the real data as these are drawn from probability distributions. Moreover, as all three features have been decoupled, it is not possible to gauge the correlations amongst the features. Thus, we use the MMD measure to quantify the distances between the learned probabilities and the true probability of the training dataset. These are presented in Table VIII.

	Cloth Dryer	**Dishwasher**	**Fridge**	**Heat Pump**
GAN	0.08	0.13	0.20	0.10
KDE	0.03	0.04	0.06	0.03

TABLE VIII: MMD comparison of usage habit synthesis.

It is clear that the distribution learned using KDE is associated with smaller MMD in comparison to the GAN for all four loads under consideration. This is expected as KDE is a non-parametric machine learning technique that works

well with lower-dimensional feature space. We observed that with a smaller GAN implementation that consists of half the number of nodes in the hidden layers as that of the GAN listed in Table VII, the resulting MMD is very close to that listed in Table VIII. The MMD obtained for dishwasher and fridge is slightly smaller for the smaller GAN in comparison to the larger GAN. For the other two loads, the MMD obtained for smaller GAN is slightly larger than that obtained from the larger GAN. As the KDE implementation results in much smaller MMD in comparison to both GAN implementations, this is more suitable for the synthesis of load usage habits.

Thus, we have designed the load usage habits module using KDE and compared its performance with the GAN based system. The data synthesized by the load pattern module and the usage habits module can now be flexibly utilized to produce labelled datasets for effectively training machine learning algorithms for demand side applications.

VI. Conclusions

In this paper, we have proposed a novel data-driven framework that flexibly synthesizes labelled appliance patterns and usage habits datasets. This framework involves three distinction stages of development: pre-processing of raw circuit meter measurements, design/training of the proposed synthesis modules and evaluation of performance. We have made novel contributions in each one of these stages in this paper. Our approach does not require intrusive installation of sensor devices such as circuit meters for each appliance. Consumers and manufacturers are able to introduce datasets that customize the data synthesis process and this can be easily accommodated by the proposed framework. The resulting synthetic data samples resemble realistic labelled datasets and facilitate the training of complex machine learning algorithms that aid electric power utilities and power consumers with using electricity in an efficient and sustainable manner. As future work, we intend to investigate how these synthetic labelled datasets can be utilized to design intelligent algorithms for HEMS entities and demand response programs.

References

[1] X. Yu and Y. Xue, "Smart grids: A cyber–physical systems perspective." *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1058–1070, Mar. 2016.

[2] "Average data growth for utility with 1m clients." EPRI, 2019.

[3] C. Klemenjak *et al.*, "Datasets | NILM wiki," Dec. 2017. [Online]. Available: http://wiki.nilm.eu/datasets.html

[4] O. Parson, G. Fisher, A. Hersey, N. Batra, J. Kelly, A. Singh, W. Knottenbelt, and A. Rogers, "Dataport and nilmtk: A building data set designed for non-intrusive load monitoring," in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Orlando, FL, USA, 2015, pp. 210–214.

[5] S. Makonin, "AMPds2: The Almanac of Minutely Power dataset (Version 2)," 2016. [Online]. Available: https://doi.org/10.7910/DVN/FIE0S4

[6] ——, "RAE: The Rainforest Automation Energy Dataset," 2017. [Online]. Available: https://doi.org/10.7910/DVN/ZJW4LC

[7] J. M. G. López, E. Pouresmaeil, C. A. Cañizares, K. Bhattacharya, A. Mosaddegh, and B. V. Solanki, "Smart residential load simulator for energy management in smart grids," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 2, pp. 1443–1452, Feb. 2019.

[8] R. Stamminger, G. Broil, C. Pakula, H. Jungbecker, M. Braun, I. Rüdenauer, and C. Wendker, "Synergy potential of smart appliances," *Report of the Smart-A project*, pp. 1949–3053, Nov. 2008.

[9] A. Keyhani, W. Lu, and G. T. Heydt, "Composite neural network load models for power system stability analysis," in *IEEE PES Power Systems Conference and Exposition, 2004.*, Oct. 2004, pp. 1159–1163 vol.2.

[10] A. Arif, Z. Wang, J. Wang, B. Mather, H. Bashualdo, and D. Zhao, "Load modeling—a review," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 5986–5999, Nov. 2018.

[11] G. Valverde, A. Saric, and V. Terzija, "Probabilistic load flow with non-gaussian correlated random variables using gaussian mixture models," *IET generation, transmission & distribution*, vol. 6, no. 7, pp. 701–709, Jul. 2012.

[12] W. Labeeuw and G. Deconinck, "Residential electrical load model based on mixture model clustering and markov models," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1561–1569, Jan. 2013.

[13] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in neural information processing systems*, Apr. 2002, pp. 841–848.

[14] A. Al-Wakeel, J. Wu, and N. Jenkins, "K-means based load estimation of domestic smart meter measurements," *Applied energy*, vol. 194, pp. 333–342, May 2017.

[15] K.-J. Park and S.-Y. Son, "A novel load image profile-based electricity load clustering methodology," *IEEE Access*, vol. 7, pp. 59 048–59 058, May 2019.

[16] Y. Chen, Y. Wang, D. Kirschen, and B. Zhang, "Model-free renewable scenario generation using generative adversarial networks," *IEEE Transactions on Power Systems*, vol. 33, no. 3, pp. 3265–3275, May 2018.

[17] C. Zhang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "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 (SmartGridComm)*, Oct. 2018, pp. 1–6.

[18] Y. Gu, Q. Chen, K. Liu, L. Xie, and C. Kang, "Gan-based model for residential load generation considering typical consumption patterns," in *2019 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, Feb. 2019, pp. 1–5.

[19] L. Zhang and B. Zhang, "Scenario forecasting of residential load profiles," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 1, pp. 84–95, Jan. 2020.

[20] M. N. Fekri, A. M. Ghosh, and K. Grolinger, "Generating Energy Data for Machine Learning with Recurrent Generative Adversarial Networks," *Energies*, vol. 13, no. 1, p. 130, Jan. 2020.

[21] S. Makonin, F. Popowich, L. Bartram, B. Gill, and I. V. Bajić, "Ampds: A public dataset for load disaggregation and eco-feedback research," in *2013 IEEE Electrical Power & Energy Conference*, Halifax, NS, Canada, Aug. , pp. 1–6.

[22] J. Gao, S. Giri, E. C. Kara, and M. Bergés, "Plaid: A public dataset of high-resoultion electrical appliance measurements for load identification research: Demo abstract," in *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, New York, NY, USA, Nov. 2014, pp. 198–199.

[23] CFRL, "London hydro unveils trickl app," Mar. 2018. [Online]. Available: https://www.cfrlradio.com/syn/202/71446/london-hydro-unveils-trickl-app/

[24] D. Erdogmus, R. Agrawal, and J. C. Principe, "A mutual information extension to the matched filter," *Signal Processing*, vol. 85, no. 5, pp. 927–935, May 2005.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, Jun. 2014, pp. 2672–2680.

[26] Y. Qin, N. J. Mitra, and P. Wonka, "Do GAN loss functions really matter?" *CoRR*, vol. abs/1811.09567, 2018. [Online]. Available: http://arxiv.org/abs/1811.09567

[27] K. J. Liang, C. Li, G. Wang, and L. Carin, "Generative adversarial network training is a continual learning problem," *CoRR*, vol. abs/1811.11083, 2018. [Online]. Available: http://arxiv.org/abs/1811.11083

[28] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, Conference Track Proceedings*, Apr. 2017.

[29] S. A. Barnett, "Convergence problems with generative adversarial networks (gans)," *CoRR*, vol. abs/1806.11382, 2018. [Online]. Available: http://arxiv.org/abs/1806.11382

[30] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784
[31] C. M. Bishop, *Pattern recognition and machine learning*, 5th ed. Springer, 2007.
[32] F. U. Nuha and Afiahayati, "Training dataset reduction on generative adversarial network," in *INNS Conference on Big Data and Deep Learning 2018, Sanur, Bali, Indonesia*, Apr. 2018, pp. 133–139.
[33] A. M. Oberman and J. Calder, "Lipschitz regularized deep neural networks converge and generalize," *CoRR*, vol. abs/1808.09540, 2018. [Online]. Available: http://arxiv.org/abs/1808.09540
[34] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA*, Dec. 2017, pp. 5767–5777.
[35] H. Liu, J. Lafferty, and L. Wasserman, "Sparse nonparametric density estimation in high dimensions using the rodeo." *Journal of Machine Learning Research - Proceedings Track*, vol. 2, pp. 283–290, Jan. 2007.
[36] E. Alpaydin, *Introduction to machine learning*, 3rd ed. MIT press, 2014.
[37] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statistics Surveys*, vol. 4, no. 0, pp. 40-79, Mar. 2010.
[38] L. Wasserman, "Density Estimation." [Online]. Available: https://www.stat.cmu.edu/∼larry/=sml/densityestimation.pdf. [Accessed: 15-Apr-2020].

Samer El Kababji Samer El Kababji received his B.Sc. Degree in Electrical Engineering and M.Sc. degree in Industrial Engineering from the University of Jordan, Amman, Jordan in 1991 and 1995, respectively. While leading companies in different industries, he accumulated broad knowledge related to smart devices and control of electrical systems. Lately, he founded Smartegrators Ltd., London, ON, a company specialized in offering IoT solutions to the commercial sector. Mr. Kababji earned his M.Eng. degree in Electrical and Computer Engineering from Western University, London, ON in 2018 and he is currently pursuing his PhD degree in the same department. His research interest includes the application of machine learning in Smart Grid. He is a recipient of Ontario Graduate Scholarship for two consecutive years.

Pirathayini Srikantha is currently an Assistant Professor in the Department of Electrical Engineering and Computer Science at York University. She received her B.A.Sc. degree in Systems Design Engineering from the University of Waterloo in 2009 and her M.A.Sc. degree in Electrical and Computer Engineering from the same institute in 2013. She obtained her Ph.D. degree from The Edward S. Rogers Sr. Department of Electrical and Computer Engineering at the University of Toronto in 2017. She is currently serving as an Associate Editor for the IEEE Transactions on Smart Grid journal. She is a certified Professional Engineer (P.Eng.) in Ontario. Her main research interests are in the areas of large-scale optimization and distributed control for enabling adaptive, sustainable and resilient power grid operations. Her work has been published in premier smart grid journal and conference venues. Her research efforts have received recognitions that include the best paper award (IEEE Smart Grid Communications) and runner-up best poster award (ACM Women in Computing).