A Conditional Generative adversarial Network for energy use in multiple buildings using scarce data

Gaby Baasch, Guillaume Rousseau, Ralph Evins

September 2021

Energy and AI 5 (2021) 100087

Contents lists available at ScienceDirect

Energy and AI

journal homepage: www.elsevier.com/locate/egyai

Perspective

A Conditional Generative adversarial Network for energy use in multiple buildings using scarce data

Gaby Baasch*, Guillaume Rousseau, Ralph Evins

Energy and Cities Group, Department of Civil Engineering, University of Victoria, Canada

HIGHLIGHTS

- Creating multivariate time series where each building represents a variable significantly eases strict data requirements for building GANS while maintaining per-building load characteristics.
- Conditioning on mean monthly outdoor temperature improves GAN performance.
- The GAN modelled the residential data with high fidelity but could not completely capture the complex temporal behaviour in the commercial data.
- Metrics that are most commonly used to evaluate GANs in the building domain do not sufficiently capture temporal behaviour.

ARTICLE INFO

Article history:
Received 30 January 2021
Received in revised form 20 April 2021
Accepted 8 May 2021
Available online 15 May 2021

Keywords:
Generative adversarial network
Building load profile
Machine learning
Data scarcity

ABSTRACT

Building consumption data is integral to numerous applications including retrofit analysis, Smart Grid integration and optimization, and load forecasting. Still, due to technical limitations, privacy concerns and the proprietary nature of the industry, usable data is often unavailable for research and development. Generative adversarial networks (GANs) - which generate synthetic instances that resemble those from an original training dataset - have been proposed to help address this issue. Previous studies use GANs to generate building sequence data, but the models are not typically designed for time series problems, they often require relatively large amounts of input data (at least 20,000 sequences) and it is unclear whether they correctly capture the temporal behaviour of the buildings. In this work we implement a conditional temporal GAN that addresses these issues, and we show that it exhibits state-of-the-art performance on small datasets. 22 different experiments that vary according to their data inputs are benchmarked using Jensen-Shannon divergence (JSD) and predictive forecasting validation error. Of these, the best performing is also evaluated using a curated set of metrics that extends those of previous work to include PCA, deep-learning based forecasting and measurements of trend and seasonality. Two case studies are included: one for residential and one for commercial buildings. The model achieves a JSD of 0.012 on the former data and 0.037 on the latter, using only 396 and 156 original load sequences, respectively.

1. Introduction

The information revolution carries the promise of innovation and high-impact industry disruption. Indeed, the potential for machine learning and big-data to assist building decarbonization is tremendous. The Smart Grid is optimized using detailed supply and demand information [1], high-resolution metered data supports city-wide retrofit strategies [2] and forecasting models trained on big-data are used for energy management [3]. In fact, a review by Hong et al. found over 9579 studies on machine learning in buildings. However, of the 153 studies that they selected for in-depth analysis none had been adopted broadly by the building industry [4]. This is largely because, in practice, building data collection is prevented by technical, regulatory and economic challenges. For example, smart meters that measure temporal electricity consumption are becoming increasingly pervasive (with over 1 billion devices installed as of 2020 [5]), but concerns over privacy prevent utility companies from disclosing this information [6,7].

The lack of building data availability has lead to a keen research interest in its generation. One approach to generative modelling is to use detailed physical models to simulate temporal building behaviour [6]. In the emerging field of urban building energy modelling, for example, city-wide building consumption is simulated based on a smaller subset building archetypes [8]. The availability of representative building reference models makes this approach desirable, but it is subject to modelling assumptions, the references may not closely represent the

Abbreviations: GAN, Generative Adversarial Network; JSD, Jensen-Shannon Divergence; KLD, Kullback-Leibler Divergence; PCA, Principle Component Analysis; MAE, Mean Absolute Error; MAPE, Mean Absolute Percentage Error; RMSE, Root Mean Squared Error; DTW, Dynamic Time Warping; MMD, Maximum Mean Discrepancy; PRD, Precision and Recall for Distributions; SSIM, Structural Similarity; TOVO, Trained on Original, Validated on Original; TGVO, Trained on Generated, Validated on Original.

* Corresponding author.
E-mail addresses: gbaasch@uvic.ca (G. Baasch), guillaumer@uvic.ca (G. Rousseau), revins@uvic.ca (R. Evins).

https://doi.org/10.1016/j.egyai.2021.100087

Table 1
Summary of the GAN studies reviewed over the course of this work. The metrics column contains the key indicators that were used to evaluate the results for the given study. We do not guarantee that the review is exhaustive, but we believe this is still a good representation of existing work. Studies that used GANs in buildings for different applications than load profile generation (such as [19,20] and [21]) are excluded from this table.

	metrics	# input sequences	period	temporal generator?
Data augmentation for forecasting				
[3] (2019)	forecasting: MAE, MAPE, correlation	building 1: 280 building 2: 81	daily	no
[15] (2019)	forecasting: MAPE, RMSE, DTW	(# unclear)	unclear	no
[16] (2020)	PCA, forecasting: MAPE, RMSE, MAE	3 case studies (# unclear)	unclear	no
[17] (2020)	forecasting: MAPE, MAE,	2 case studies (# unclear)	unclear	yes R-GAN
General generation: use case unspecified (same as this work)				
[18] (2018)	MMD, clustering forecasting: MAPE	36,500 (type unclear)	daily	no
[9] (2019)	JSD, RMSE, auto-correlation, mean and standard variance	33,760 (residential)	weekly	no
[6] (2020)	KLD of 5 key parameters, mean and standard deviation	56,957 (commercial)	daily	no
[10] (2020)	JSD, RMSE, PRD, SSIM, mean and standard variance	20,000 (residential)	weekly	no

actual buildings, and there is an established performance gap between simulated and real buildings [6]. Data-driven methods offer a different approach that helps to overcome these limitations, but they typically either require detailed end-user data or they fail to model the full diversity, accuracy and complexity of the original [9,10]. Fortunately, generative adversarial networks (GANs) offer a promising alternative.

Initially introduced by Goodfellow et al. in 2014, a GAN consists of two neural networks that are in competition with one another: a generator that acts as a counterfeiter and a discriminator that acts as a detective [11]. They have been applied across various domains to produce realistic human faces [12], to compose music [13] and to create paintings [14] with almost uncanny fidelity. In buildings GANs may offer the solution to the information shortage that prevents the wide-spread adoption of data-driven building decarbonization techniques, while also offering privacy guarantees.

1.1. Literature review

The buildings research community has acknowledged the potential of GANs and several papers have been published, but at the time of writing there has not yet been an overview of the state of the research. Table 1 therefore provides a succinct summary of the key contributions of the previous works that were reviewed for this study. The works in the table can be classified into two categories: those that focus on forecasting and those that are use-case agnostic. This work falls into the latter category so those will be the focus here. Several research gaps were identified. First, even though all of the reviewed works generated time-series sequences, none of the use-case agnostic studies used an underlying network architecture that directly modeled temporal dynamics, and none evaluated the temporal components of the generated data such as trend and seasonality. Second, the use-case agnostic works use large pre-existing datasets to train their generative models, but these are not always available in practice.[1] Finally, most works only test their approach on a single dataset so it is unclear how well they will generalize, especially between residential and commercial buildings.

1.2. Key contributions

In this work we address the aforementioned issues. We apply a recently developed, time series GAN (TimeGAN) that achieves state-of-the-art performance for temporal generation [23], and add a novel extension so that it works in a conditional setting. We demonstrate that, based on generate load distribution, the performance of our model is competitive with existing building GANs, even using a data size that is only 1–2% the size, on both residential and commercial buildings. We augment the metrics used by previous studies to include PCA and time-series specific evaluation to evaluate our results, and determine that the standard metrics that are currently used for evaluation in existing works miss important shortcomings of generative models.

The remainder of this paper is organized as follows: Section 2 presents the theoretical overview of GANs (2.1) and TimeGAN (2.2); Section 3 overviews the experimental methodology, including the GAN implementation (3.1), the residential and commercial case studies (3.2), the 22 different modelling experiments (3.3) and the metrics (3.4); Section 4 presents the results (4.1 for all 22 experiments, 4.2 for the residential case study and 4.3 for the commercial case study); Section 5 presents the discussion.

2. Conditional time series generative adversarial network

TimeGAN, developed by Yoon et al. in [23], is a logical extension of the original GAN architecture by Goodfellow et al., so this section of the paper will start by overviewing the latter (Fig. 1). Next, TimeGAN and C-TimeGAN (Fig. 2) will be described. For more information on the original TimeGAN implementation, the reader is referred to [23].[2]

2.1. Original GAN

A generic GAN consists of two neural networks: a generator (g) and a discriminator (d), who have learnable parameters θ_g and θ_d, respectively. g accepts random noise vectors $Z \in \mathcal{Z}$, where $\mathcal{Z}$ is a vector

[1] This highlights a more general circularity problem with GANs: as with other deep learning techniques, their training benefits from large data. For example, the initial GAN developed by Goodfellow et al. was trained on the MNIST dataset [22] which has 60,000 training examples.

[2] The TimeGAN formulation presented in [23] includes both static and temporal inputs, but the implementation provided by the authors does not yet include the static component. This is one of the reasons that we introduce C-TimeGAN.

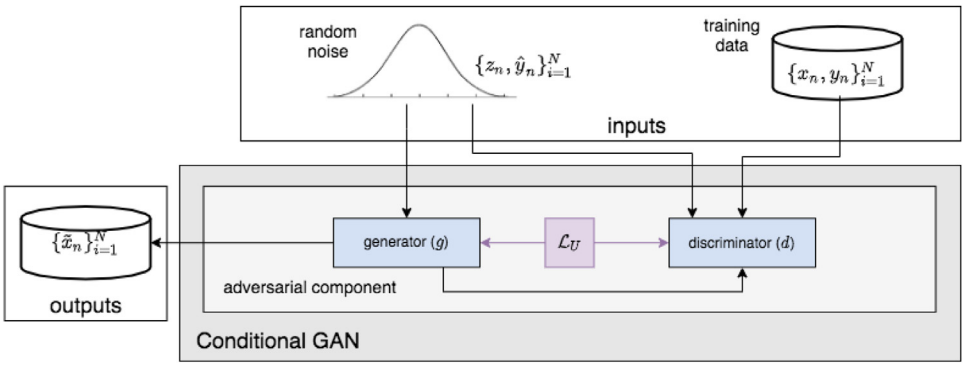


Fig. 1. The architecture of the GAN described in Section 2.1. The input values y and $\hat{y}$ make the GAN conditional, as described in Section 2.1.1.

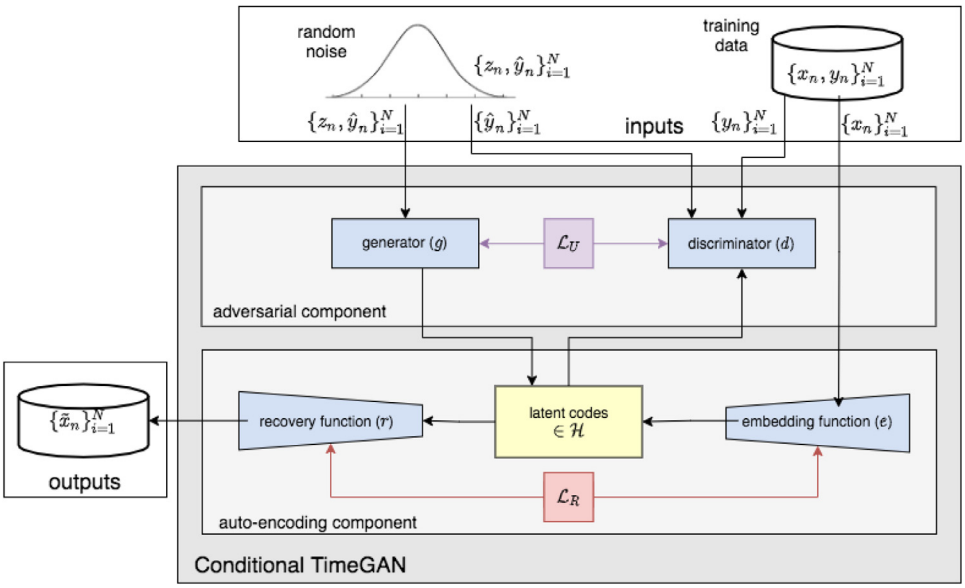


Fig. 2. The architecture of the TimeGAN described in Section 2.1. The input values y and $\hat{y}$ make the GAN conditional, as described in Section 2.2.1.

space of known distributions (e.g Gaussian distributions), and maps it to learned distribution $\hat{p}$. Its goal is to learn a density $\hat{p}(X)$ that best approximates the original distribution $p(X)$, for $X \in \mathcal{X}$. The generated samples are input into d, which also accepts samples from a training dataset $\{x_n\}_{i=1}^N$. d's objective is to output the probability that example x came from $p(X)$, as opposed to $\hat{p}(X)$, i.e. to distinguish which samples ccme from the original data and which were generated. g and d play a two-player minimax game with value function $\mathcal{L}_U$:

$$\min_{\theta_g} \max_{\theta_d} \mathcal{L}_U = \mathbb{E}_{x\sim p}[\log d(x)] + \mathbb{E}_{z\sim \hat{p}}[\log(1 - d(g(z)))] \tag{1}$$

where d aims to *maximize* the probability of assigning the correct label to the original and generated samples $[\log d(x)]$, and g aims to *minimize* the probability that d correctly classifies samples $[\log(1 - d(g(z)))]$.

2.1.1. Conditional extension

A conditional extension to the original GAN formulation was proposed in [24]. g and d are conditioned on some extra information y, for $Y \in \mathcal{Y}$ by adding an additional input layer to the neural networks, resulting in the following extension to Eq. 1:

$$\min_{\theta_g} \max_{\theta_d} \mathcal{L}_U = \mathbb{E}_{x\sim p}[\log d(x|y)] + \mathbb{E}_{z\sim \hat{p}}[\log(1 - d(g(z|y)))] \tag{2}$$

2.2. TimeGAN

TimeGAN consists of the same adversarial component and loss ($\mathcal{L}_U$) as the original GAN formulation, but with two major extensions: an autoencoding component and a supervised autoregressive (AR) learning objective. These are described below.

Autoencoding Component: An autoencoder is a dimensionality reduction technique where embedding (e) and recovery (r) functions learn stochastic mappings $p_e(h|x)$ and $p_r(\tilde{x}|h)$ [25]. h is an instance of $H \in \mathcal{H}$, where $\mathcal{H}$ is the latent vector space corresponding to $\mathcal{X}$.[3] Simply put, e transforms the data into its latent code h, and r undoes the transformation. In TimeGAN, e and r are neural networks that are trained using the recovery loss function:

$$\mathcal{L}_R = \mathbb{E}_{x_{1:T}\sim p}\left[\sum_t ||x_t - r(e(x_t))||_2\right] \tag{3}$$

In TimeGAN, g and d operate in the latent space as follow: (1) g transforms random noise vectors into latent codes and e transforms training samples into latent codes, (2) d classifies whether the latent codes come from the generated or original data, and (3) r transforms data from the latent space back into its original form.

AR Learning Objective: Generating temporal data can be a difficult learning problem for a GAN, especially if the input data is comprised of long sequences. To introduce temporal relationships directly into the learning architecture, TimeGAN uses a supervised loss function based

[3] In statistics, latent - or *unobservable* - variables are variables (typically low-dimensional) that retain important features of original, multidimensional data [26]. The motivation behind their inclusion in TimeGAN is that they retain important temporal dynamics that are otherwise lost in GAN training.

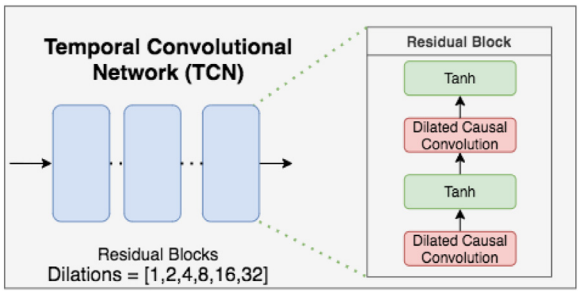


Fig. 3. The architecture of the TCN.

on AR decomposition, so that the GAN can specifically learn the conditional temporal probabilities.[4] For simplicity, this supervised loss is not included in Fig. 2, but it is used to train both g and e.

2.2.1. Conditional extension (C-TimeGAN)

In this work we suggest a novel extension to TimeGAN that is similar to the extension to the generic GAN, so g and d are conditioned on some additional information y, and adversarial objective is the same as in Eq. 2. Networks e and r do not change between TimeGAN and C-TimeGAN.

3. Methodology

3.1. GAN Implementation

Two variants of C-TimeGAN were implemented, a single channel and a multi channel model. The single channel model accepts inputs of shape [672x1] i.e. a single variate time series. The multi channel model instead accepts inputs of shape [672x12], where each of the 12 variables is a distinct building. For simplicity, the this section will only report the network shapes from the multi channel 672 time step models.

g, d, e and r are all implemented in Tensorflow[5] using Temporal Convolutional Networks (TCN) that share the same architecture[6]. Each TCN consists of 4 stacks of residual blocks, with each residual block containing 5 hidden layers and each layer using an increasing number of dilations. This architecture can be seen in Fig. 3.

Both non-conditional and conditional data inputs were used in this work (see Section 3.3 for details). Both architectures are described in Fig. 4. For the non-conditional case g, d, e and r all follow the same structure. Samples of shape [672x12] are passed through the input layer and into the TCN. The TCN's output is then fed to a dense layer to transform it into the final output. g, e and r all contain a sigmoid activation function in their output layer while d does not.

For the conditional models, only g, d are modified. A second input layer is added to accept the conditional input, the conditional input is then passed into a dense layer to transform so that it can be concatenated to the original input. This concatenated input is then passed into the TCN in the same process described beforehand.

g, d, e and r are all trained using Adam optimization and a batch size of 128. Training is split up into 3 main phases, embedding training, supervised training and joint training. During the embedding training, e and r are trained together using the recovery loss for 50,000 iterations. For the supervised training, g is trained exclusively using the supervised loss for 50,000 iterations. Lastly, the joint training phase consists of training g, d, e and r using their respective loss functions for another 50,000 iterations

3.2. Residential and commercial datasets

Two datasets were used in this work: (1) a residential smart meter dataset and (2) a subset of education and government buildings from the Building Data Genome Project [27]. Each consists of 12 buildings with hourly electric load. For the former, the starting indexes are not aligned by timestep, and the (hourly) weather data was obtained from the Government of Canada historical weather service, for the VANCOUVER HARBOUR CS station between 2015 and 2019[7]. For latter, the starting indexes are aligned by time stamp, and the weather was already included. Other features of the two datasets are listed in Table 2 and sample sequences are plotted in Fig. 5.

3.2.1. Data preprocessing

For both datasets, each building was individually standardized to the range [0,1] using Min-Max normalization

$$\hat{x} = \frac{x - min(x)}{max(x) - min(x)} \tag{4}$$

where x is the vector of all load values from one of the buildings in the dataset. For the single channel models, the datasets were instead normalized over every building, therefore x would represent the vector of all load values from every building in this case. Missing values were replaced by values sampled from that building's distribution. Data is then broken up into smaller sequences of 672 or 168 time steps depending on the model being trained. To create more data for training, a sliding window with a lag of 1 time step was also applied during the sequencing step.

3.3. Data input experiments

The residential dataset was used to test case for a large array of different experiments whose parameters are summarized in Table 3. GANs have a high computational cost so only 12C with L and LM were also tested using the commercial data. Many of the residential experiments were not able to generate data that represented the original, but to save the time of other researchers we believe that it is important to publish negative results. Therefore, the results of all unsuccessful experiments will still be presented,[8] but only the best performing will be evaluated in more detail.

The clustering (LC) case was included because all of [9,10], and [6] used clustering. It was done using a k-means model trained using the set of real load curves. A different k-means model was trained for both the 168 and 672 time step sets. In order to find the optimal number of clusters k, two common clustering metrics are used. The first is Davie-Boulin Index (DBI), which is used to quantify the cluster scatter and separation. The second is the Error Sum of Squares (SSE), which quantifies the difference between samples in each individual cluster. For both sets of load curves the number of clusters tested ranged from 2 to 18. The optimal number of clusters was 6 and 7 for the 672 and the 168 time step set respectively

3.4. Metrics

The metrics that are used in this study were chosen (1) to provide a numerical comparison with the results from previous works, and (2) to describe the distribution and temporal behaviour of the real vs. generated data. JSD was chosen as the key numerical metric because it has

[4] An AR model is one where the probability of observing a value at a specific time t is conditional on values from the previous $t-1$ timesteps.

[5] https://www.tensorflow.org/

[6] The TimeGAN repository is available at https://github.com/jsyoon0823/TimeGAN

[7] https://climate.weather.gc.ca/historical_data/search_historic_data_e.html

[8] For brevity, only the JSD and the forecaster errors described in Section 3.4.1 will be reported in the main article. Refer to Appendix A for all the distribution plots.

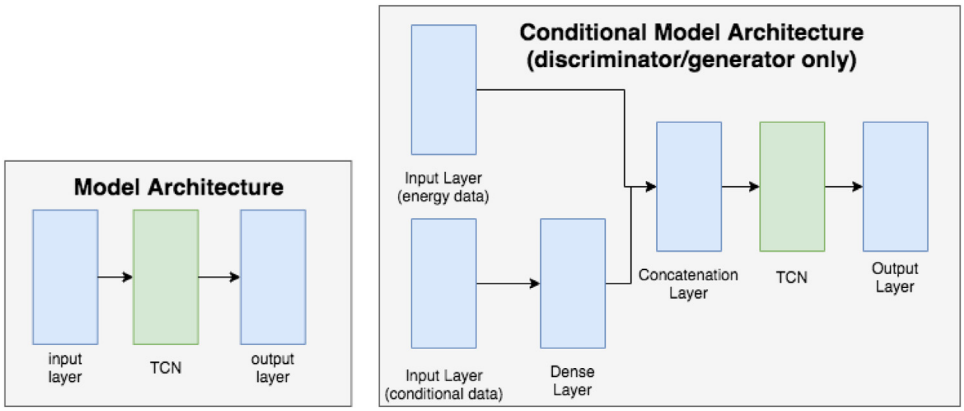
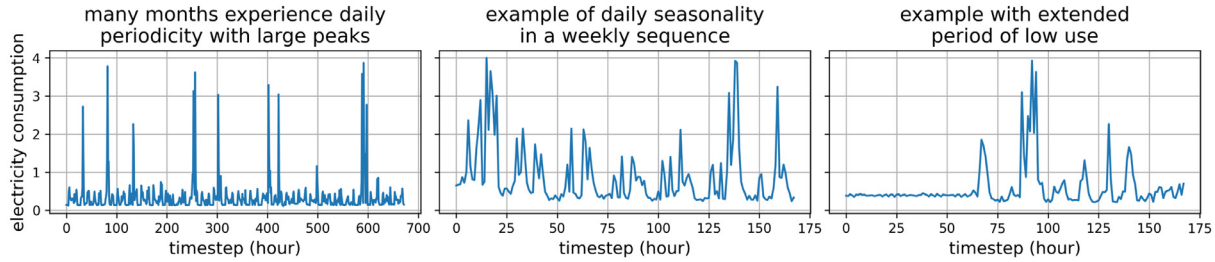


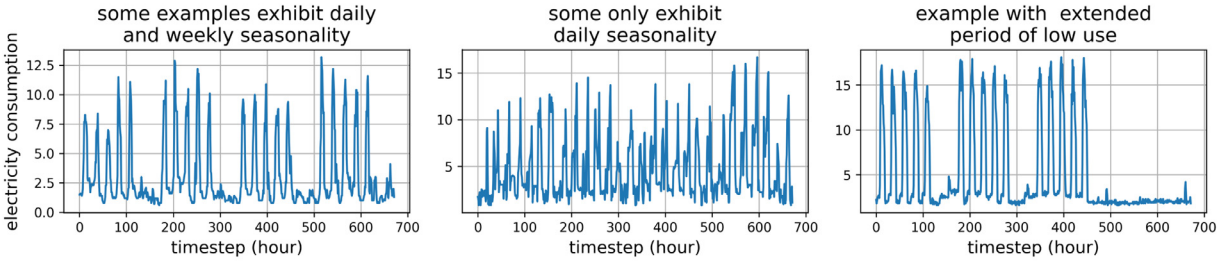
Fig. 4. The model architectures for the unconditional and conditional model. In the conditional case, e and r still use the unconditional architecture.

Table 2
The relevant features of the two datasets used to generate stochastic load profiles. The names specify whether the residential or commercial dataset was used, as well as the length of the input sequences. 168 is a weekly sequences and 672 is 28 days.

Name	Location	Timespan	Input Sequences (sliding window)	Input Sequences (raw)	Output Sequences
res_{168}	Vancouver	3 years	269,544	1608	1608
res_{672}	Vancouver	3 years	263,496	396	396
$comm_{672}$	London	1 year	97,056	156	156



(a) Residential.



(b) Commercial.

Fig. 5. Example sequences from that illustrate the types of features found in the residential and commercial data.

been used in comprehensive GAN studies that cover a multitude of different modelling approaches (see Table 1). Forecasting validation error is also applied to indicate the fidelity of the generated sequences, and to offer a sanity check on JSD. Of the distribution metrics, normalized load histograms, standard variance and mean load are well used in this domain, PCA was implemented by the creators of TimeGAN, and, to the best of our knowledge, we are the first to introduce the strength of the seasonality and trend. All of these metrics are described in the following sections.

3.4.1. Numerical scores

Jensen Shannon Divergence (JSD): JSD was applied to quantify the difference between the original and generated load distributions, using a process similar to that in [10]. Before taking the JSD, the original and generated data were (1) log transformed, (2) normalized using Eq. 4 and (3) transformed into the probability distributions P_o and P_g. For (3), each of the datasets were divided into K segments (in this work $K = 100$), so that the range of the kth interval is $[\frac{k-1}{K}, \frac{k}{K}]$:

$$P_o = \left[\frac{N_{o,1}}{N}, \frac{N_{o,2}}{N}, \ldots, \frac{N_o}{N}\right] \qquad P_g = \left[\frac{N_{g,1}}{N}, \frac{N_{g,2}}{N}, \ldots, \frac{N_g}{N}\right] \tag{5}$$

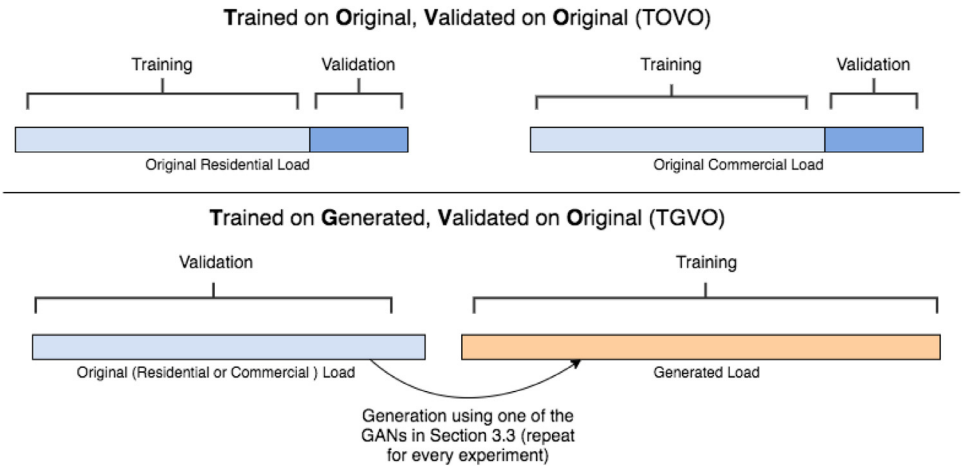


Fig. 6. The two validation schemes (TOVO, and TGVO) used to evaluate the time series forecaster.

Table 3
The nomenclature for the data inputs for the experiments.

Channels	
single channel	1C
12 channel	12C
Load + Conditional Input	
load only	L
load + cluster label	LC
load + mean outtemp.	LM
load + time stepped, hourly outtemp.	LH
load + month label	LL
load + cluster label + mean outtemp.	LCM

where $N_{o,k}$ and $N_{g,k}$ are the number of load values within the kth interval of the original or the generated data. $JSD(P_o||P_g)$ was then calculated by taking the square of the JS distance.[9,10]

Forecasting Validation Error (MAE): Time series forecasting can be used to test whether the generated data is a good representation of the original. The original and synthetic data-sets are both broken up into sequences of 24 time steps in length to create training and testing data for the forecaster. The goal of the forecaster is to take one of these sequences as input and output a prediction for the 25th time step of that sequence. A simple single layer Temporal Convolutional Network (TCN) was used for all forecasters. each forecaster was trained for 2000 iterations with a batch size of 128 and an Adam optimizer. To quantify how well the generated data represented the original, two cross-validation processes are used, and the validation scores between them are compared (see Fig. 6):

1. **T**rained on **O**riginal, **V**alidated on **O**riginal (TOVO)
2. **T**rained on **G**enerated, **V**alidated on **O**riginal (TGVO)

A different TOVO forecaster was trained for the two datasets. For TGVO, a different model is trained on every experiment that is specified in Table 3. The Mean Absolute Error (MAE) is the validation metric, and a lower score signifying better performance. The goal is to achieve a TGVO score that is as close to the TOVO score as possible.

[9] https://scipy.github.io/devdocs/generated/scipy.spatial.distance.jensenshannon.html. See [10] for the full equation.

[10] The equation for JSD uses logarithms, and the bounds on the results change depending on the base of the logs. For log base 2 it is bounded by 0 and 1; for log base e it is bounded by 0 and ln(2). In both cases, a score of 0 means that the distributions were identical. In this work we use log base 2 because it is intuitive to think of a range between 0 and 1.

3.4.2. Distributions

All of the following distributions were calculated between the generated data and the original sequences that existed before applying the sliding window.

1. *Normalized Load Histograms:* Histograms of the log normalized loads are used to show how distributions of the original and generated data differ.[11]
2. *Principal Component Analysis (PCA):* PCA is used to compress data into n-dimensional space while retaining as much of the original variability as possible [28]. Here, the log normalized data into 2 dimensions. As suggested by [23], this provides a qualitative assessment of the diversity of the original vs. the generated data.
3. *Standard Variance and Mean Load:* The mean and the standard variance are calculated for every sequence in the original and generated datasets and they are plotted against eachother to compare their relationship.
4. *Strength of Seasonality and Trend:* Any time series sequence y_t can be decomposed into its seasonal (S_t), trend (T_t) and residual (R_t) components.[12] The decomposition can be written as $y_t = T_t + S_t + R_t$. The variation of the seasonality and trend can then be measured relative to the variation in the residuals using

$$F_T = max\left(0, 1 - \frac{var(R_t)}{var(T_t + R_t)}\right) \qquad F_S = max\left(0, 1 - \frac{var(R_t)}{var(S_t + R_t)}\right) \tag{6}$$

where F_T and F_S measure the strength of the trend and seasonality, respectively. For both, the range of possible values lies between 0 and 1, where 0 is the lowest possible strength [29].

Numerous methods can be applied to decompose a time series into its components. In this work STL decomposition is used [30].[13] Compared with other methods, STL is more robust to outliers and it can handle many different types of seasonality [29]. Based on domain knowledge about the behaviour of buildings, a daily seasonality is specified in this work.

[11] The log was taken before regularization so that the histograms look more normal and are easier to interpret.

[12] According to [29], a seasonal pattern occurs when the time series is affected by seasonal factors such as the day of the week or the hour of the day, a trend pattern occurs when there is a long-term increase or decrease in the data, and the residual component is whatever is left over.

[13] We use the STL implementation from the statsmodels Python library: https://www.statsmodels.org/stable/examples/notebooks/generated/stl_decomposition.html

Table 4
JS divergences for all experiments (lower is better). For each dataset the best performing case highlighted in bold. Refer to Table 3 for the nomenclature.

		L	LC	LM	LH	LL	LCM
res_{168}	1C	0.309	0.424	0.995	0.491	0.323	0.314
	12C	**0.077**	-	0.099	0.145	0.201	-
res_{672}	1C	0.953	0.408	0.342	0.568	0.507	0.361
	12C	0.15	-	**0.012**	0.172	0.133	-
$comm_{672}$	12C	-	-	**0.037**	-	-	-

Table 5
The validation errors (MAE) for the trained forecasters. Refer to Table 3 and Section 3.4.1, Forecasting Validation Error, for the nomenclature. The best performing cases (i.e., the TGVO scores that are closest to the TOVO scores, and the TGVO score that are the lowest) for each training set are highlighted in bold.

		TGVO						TOVO
		L	LC	LM	LH	LL	LCM	
res_{168}	1C	0.095	0.072	0.119	0.081	0.154	0.096	0.042
	12C	**0.051**	-	0.056	0.078	0.081	-	
res_{672}	1C	0.116	0.101	0.118	0.105	0.079	0.090	
	12C	0.063	-	**0.046**	0.059	0.052	-	
$comm_{672}$	12C	-	-	**0.166**	-	-	-	0.041

4. Results

The results for all 22 experiments are presented in the Section 4.1. The sections that present the results for the residential and commercial case studies (4.2 and 4.3) will focus only on the best performing model (i.e. the 12C monthly model conditioned on mean temperature).

4.1. All experiments

4.1.1. JS Divergence

Table 4 displays the JSDs for all of the conducted experiments. The high JSDs for RES_{168}_1C_LM and RES_{672}_1C_L indicates that the models were not able to converge during training. These will not be considered in the rest of this paper.

Aside from the non-convergent cases, the JSDs on the residential data had a wide range of [0.012, 0.568], and the 12C model out performed the 1C model for every data input case. Overall, the 1C baselines (i.e. L, LC) were not able to generate sequences that represented those in the original dataset. In many cases, the 168 model outperformed the 672 model, but the lowest JSD was from the latter. For RES_{168}, the case with no conditioning had lowest JSD for both the 1C and 12C case. For 12C, conditioning on monthly label also produced a relatively low JSD. RES_{672}, conditioning on mean outdoor temperature (LM) significantly outperformed the other experiments, with a JSD of 0.012, compared with the next lowest of 0.078.

4.1.2. Forecasting

The cross-validation results for TOVO and TGVO defined in Section 3.4.1 are presented in Table 5. Recall that if a GAN generates high-fidelity sequences than the validation error on the original data should be similar regardless of whether the forecasters was trained on generated or original data. On the other hand, if the GAN generates sequences that do not represent the originals, then the forecaster that is trained on the former will not perform well when validated on the latter. There is only one TOVO score for the residential data because the forecasters were trained with sequences of length todo, so whether length of the input sequences was irrelevant.

The models that had the lowest JSDs also had the lowest forecasting errors and, as with the JSDs the 12C model outperformed the 1C model for every type of data input. This shows that there is consistency between the two metrics and further implies that the 672 step, multi-channel model conditioned on mean monthly outdoor temperature was the best performing. This model will therefore be presented in more detail in the following sections. The distributions for all of the other residential experiments are available in Appendix A.[14]

4.2. Case study 1: Residential data

4.2.1. Residential distributions

Fig. 7 illustrates that, in addition to achieving a low JSD and TGVO error, RES_{672}_12C_LM was able to model other important attributes in the time series data. The distributions along the x and y-axis in Fig. 7b show that the amount of variability captured by the principle components is similar between the original and the generated data. Qualitatively, the most noticeable difference was that the original data was more spread out along component 1 than the generated data, which had a higher peak of lower values. This indicates that the generated data contained less more diversity than the original, which is not surprising. Component 1 had a mean and standard deviation of 0.0(±2.8) for the original and -0.2(±2.75) for the generated; component 2 had a mean and standard deviation of 0.0(±0.92) for the original and -0.1(±0.73) for the generated.

The model's ability to capture the relationship between the standard variances and the means of the sequences (Fig. 7c) was quite impressive, since it was able to model clusters of outliers, but the generated sequences tended to have lower variance than the original. This is a similar observation to that which was discovered from the PCA.

The distributions of the F_S and F_T (Fig. 7d) exhibited reasonable overlap between the original and generated data, though less so than the other types of distributions discussed above. For F_S, the mean and standard deviation were 0.22(±0.12) and 0.35(±0.17) for the original and generated data, respectively. The GAN tended to model stronger seasonality than what was present in the original data. F_T had a mean and standard deviation of 0.07(±0.09) for the original and 0.15(±0.19) for the generated data; both the mean and variability in F_T is was notable higher for the latter (by about 10% of the range of possible F_Ts).

4.2.2. Residential forecasting

The forecasting model that was trained and validated on the original residential data (TOVO) achieved a MAE of 0.042. The forecaster was trained on the normalized data, so the total range of values was [0, 1]. Relative to this, an MAE of 0.042 is low (4.2% of the total range). The MAE of the forecaster that was trained on the generated and validated on the original (TGVO) was 0.046. Relative to the range of values in the data, the TGVO validation only increased by 0.4% when compared to TOVO. This shows that forecaster that was trained generated data was useful for predicting on the original.

4.2.3. Residential examples

Fig. 8 displays two generated and two original sequences. There is no inherent relationship between the chosen examples; they were selected to highlight some of the interesting features in the original data that the generator was able to capture.

4.3. Case study 2: Commercial data

4.3.1. Commercial distributions

Fig. 9 displays the distribution plots for the commercial data. Visualizing the log normalized load histograms highlights the differences between the original and generated data that cannot be inferred from the JSD alone. Here we can see that the original data had certain bins with higher counts than the original data (the highest count in the original was almost 1000 times higher than the generated), and that the generated data had a shape that looked was bi-modal than the original.

[14] Over the course of the research, these were also evaluated to determine that the 672 step, 12C_LM model was indeed the best performing.

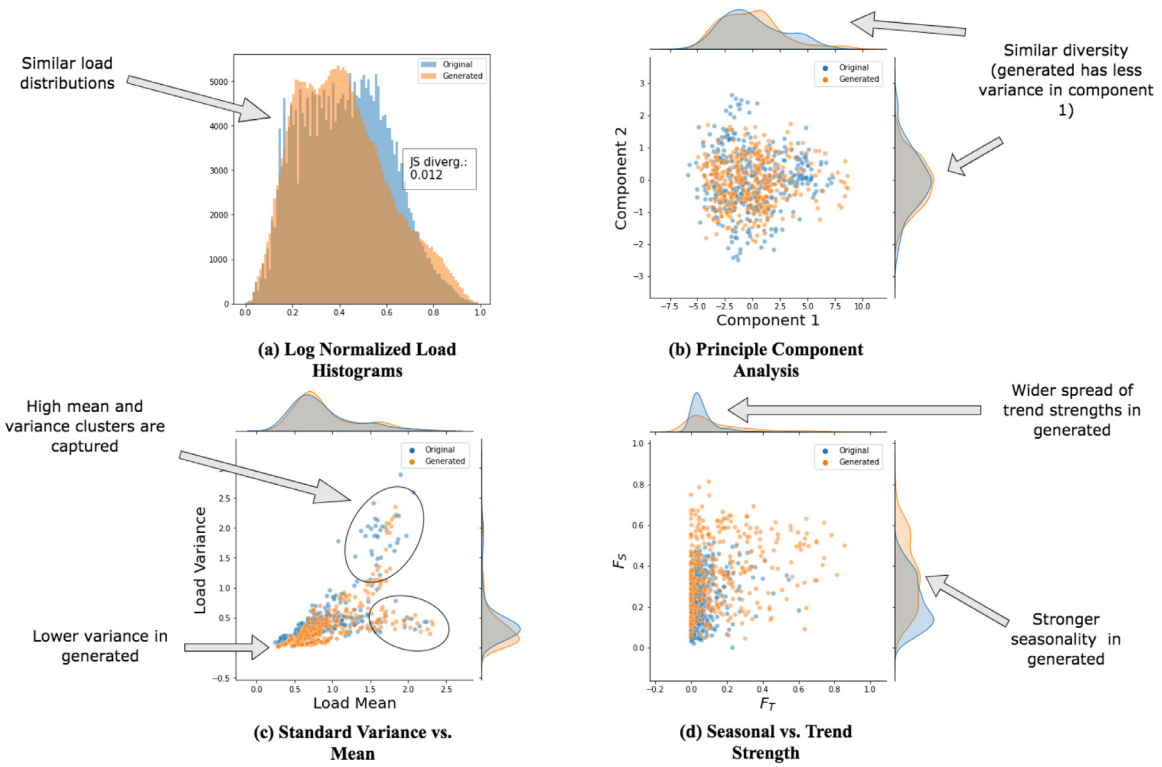


Fig. 7. The distribution results on the residential data.

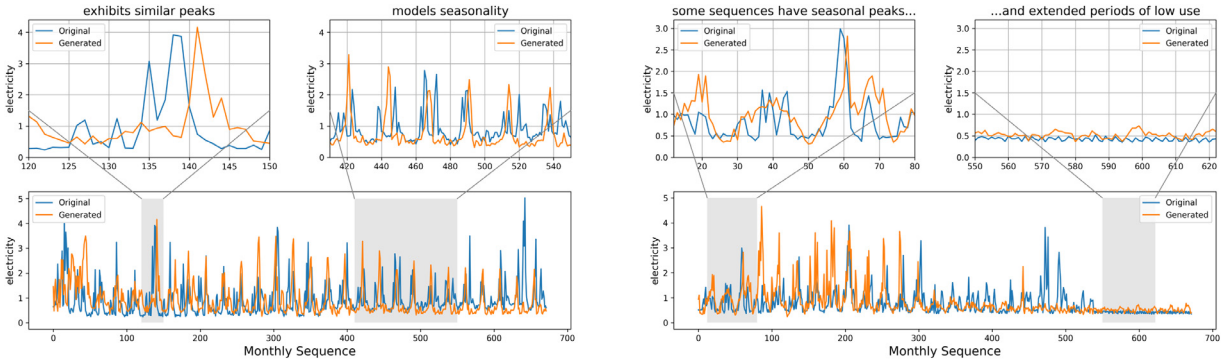


Fig. 8. Example sequences from the original and generated residential data.

The mean and standard deviations from PCA (Fig. 9b) were 0.0(±2.71) on the original and 0.07(±1.44) on the generated data for component 1; and 0.0(±1.66) and 3.08(±1.5) for component 2. The difference in the means of the original and generated distributions for component 2 is notable: the generated data appears to have more samples with higher diversity.

The behaviour of the commercial generator in terms of the variances vs. means was similar to that of the residential, in that clusters of buildings with aggregate statistics were appropriately modelled. The overlap of the distributions on the x and y-axes if the Fig. 9c show that the shapes were almost perfectly modelled. For the time series components (Fig. 9d), however, the commercial model did not perform well. Specifically, it was unable to capture the strength of the trend in the original data.

4.3.2. Commercial forecasting

From Table 5, the TOVO score was 0.041 but the TGVO score was 0.166. The TOVO score was similar to the residential data, but the TGVO score was worse than all of the TGVO models, including the ones that did converge. This is a poor performance provides further evidence that even though the JSD and mean and standard deviation appeared reasonable, the generated commercial data likely lacks fidelity, and it is not *useful* for forecasting.

4.3.3. Commercial examples

As demonstrated by the sample sequences in Fig. 5, the commercial data exhibited a different type of seasonality than the residential data. Specifically, the commercial buildings often had a weekly seasonality with high usage on the week days and low usage on the weekends. This is an expected pattern in commercial data. Fig. 10 shows that, even though the generator is able to produce weekday sequences with high fidelity, it is not able to model the weekly seasonality. This likely explains why the distribution metrics looked good but the forecasting validation error was low. This could also help to explain why the original data in the distribution plots had some values with much higher counts: the

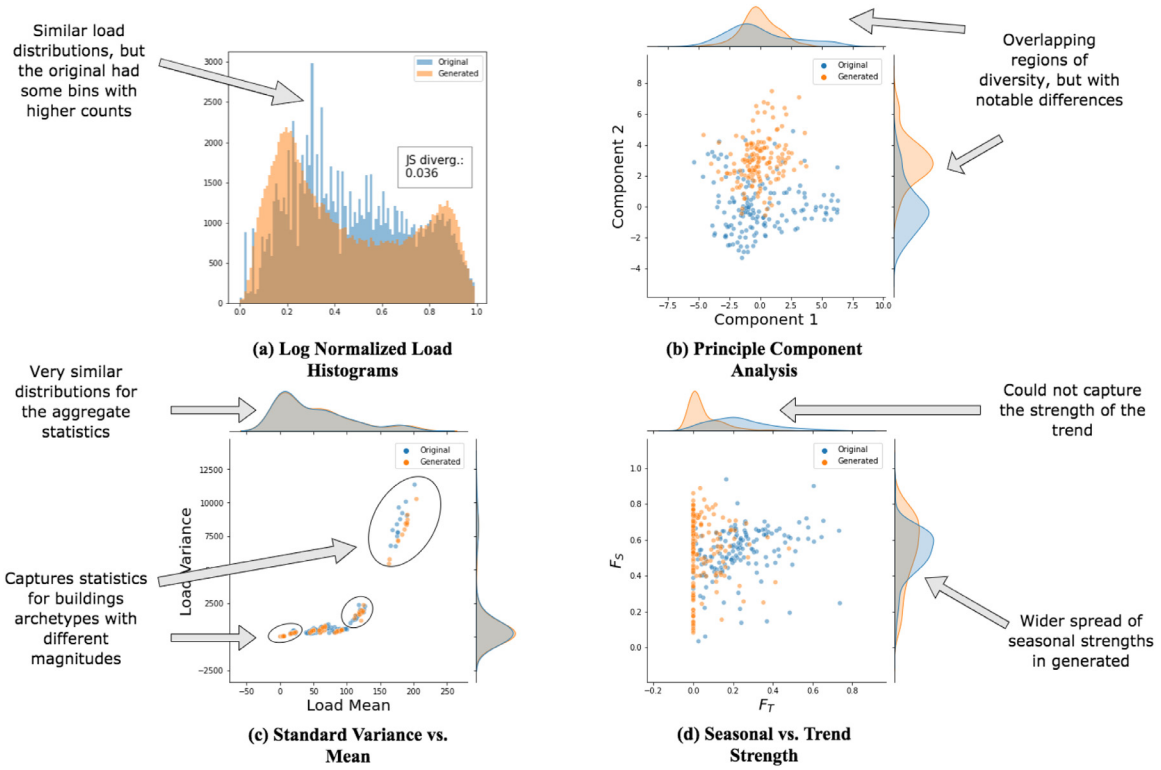


Fig. 9. The distribution results on the commercial data.

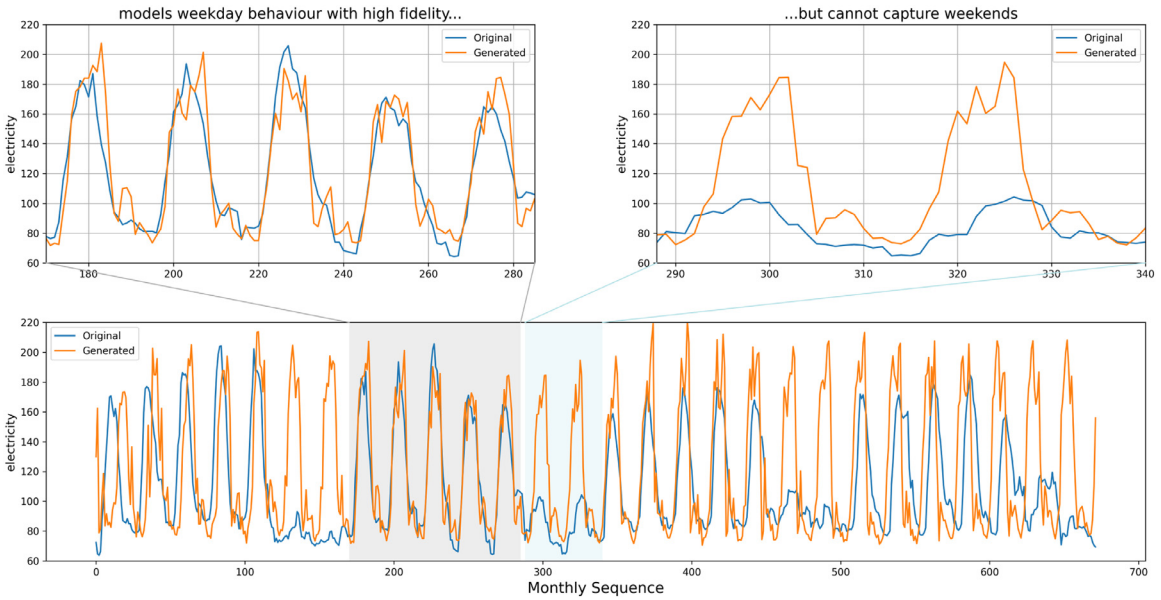


Fig. 10. Example sequences from the original and generated commercial data.

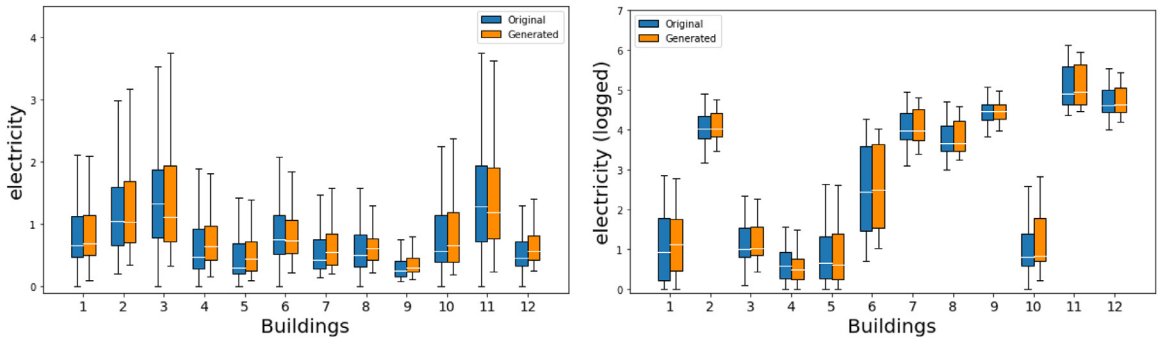
lower usage values on the weekends are not present in the generated sequences.

4.4. Per-building distributions

While Figs. 7a and 9 a plot the regularized load distributions across the entire datasets for the residential and commercial cases, Fig. 11 shows the distributions for each building individually, in the form of boxplots. These illustrate that the models each individual building, even for the commercial case where the buildings have large differences (note that, for visibility, the loads were logged in 11 b).

5. Discussion

In this section we will discuss the performance of our model in relation to other works using metrics that are standard in this field of study (see Table 1). This approach to comparison provides valuable insight into the relative performance of our approach, but it does not account for the use of a different GAN architecture and dataset than previous studies. We acknowledge that a more direct analysis must be conducted to truly establish the performance. This leads to a general observation about this domain; there is a strong opportunity to introduce and enforce



(a) Residential. (b) Commercial.

Fig. 11. Per-building distributions.

standard practices that will help to accelerate the research. Future work should compare and benchmark different approaches more robustly, including different approaches to conditioning.

5.1. Competitive performance with less data

This research successfully developed a load sequence generator that is competitive with existing works, but that can be applied for significantly smaller datasets. Unlike the other use-case agnostic studies in Table 1 we demonstrate our approach for both residential and commercial data. Our best performing model (i.e. the 672 step, multi-channel model, conditioned on mean monthly temperature) achieved a JSD of 0.012 on the residential case study and 0.037 on the commercial case study, using only 396 and 156 original input sequences (before applying the sliding window). Using log e instead of log 2, these values are 0.008 and 0.024. The best performing architecture (ACGAN) in the work by Wang et al., which is the most comprehensive study on building GANs to date, achieved a JSD of 0.0463 using 20,000 weekly sequences [10]. Both our residential and commercial models outperformed [10] using less that 2% of the data. Gu et al. also used ACGAN and achieved a JSD of 0.0045 using 33,760 sequences (log e). Our residential result (0.008) was close to theirs, using only 1% of the data. Our commercial result was slightly worse, but our dataset was 0.5% the size.[15]

In addition to JSD, [9] and [10] also plot the distributions of the real and generated means and standard variances. There is no numerical metric that summarizes these, but based on visual inspection we conclude that our approach performs just as well or better. The reader is invited to compare these plots between the papers. We do not compare our results with [18] and [6] because they use daily sequences, which is a much easier generation problem.[16]

It is worth noting that the computational resources required to run GANs are very high, so hyperparameter exploration for many experiments was unfeasible. It is possible that our results could have been improved had more optimal hyperparameters been selected, however, the original TimeGAN paper [23] suggests that it is fairly stable, so we do not expect this to significantly impact the results.

5.2. Benefit of multi channel and conditional formulations

From the analysis above, we can conclude that our modelling approach was able to reduce data requirements without depreciation in performance. In this section we aim to describe the key contributions of the modelling approach that allowed for this result.

The poor performance of the 1C models (Table 4) shows that our approach did not acheive low JSD *because* of small data, but rather *in spite* of it. The primary insight that led to the improved performance was the creation of a multivariate time series, in which each variable represents a single building. Tables 4 and 5 show that, on the residential data, the 12C input data shape significantly improved model performance for every input case. Intuitively, treating a set of buildings in this way helps the GAN to find temporal associations between the buildings by identify the difference between definite patterns and randomness. Additionally, the multi-channel approach circumvents the need for clustering, which depends on the user intervention to determine the optimal amount of clusters, and which may have different degrees of success on different datasets.[17]

Another novel insight from this work is the conditioning on mean outdoor temperature. In the residential monthly, multi channel model, this conditioning resulted in a lower JSD and validation error than all other cases. None of the other conditioning cases outperformed the load only baseline, which shows that the performance gains from using the weather are noteworthy, and not necessarily easy to replicate. Interestingly, however, the buildings in the original residential data had strong monthly correlations with outdoor temperature that were not retained in the generated data, despite the conditioning. Future work should consider investigating why these correlations were lost.

5.3. Issues with temporal correlations

For the commercial data, the JSD and mean and standard deviation plots were competitive with other works, but the trend was not modeled, the forecasting validation error was high, and the distributions of the principle components had notable differences between the real and generated data. Fig. 10 provides an explanation for this behaviour: the daily sequences in the data were captured with high fidelity, but the weekly patterns were not. We tested this further by taking the Pearson correlation of the daily mean loads between all the buildings in the original dataset. In some cases, for instance buildings that had high weekday and low weekend usage, there was a strong correlation that was not retained in the generated data. This shows that the GAN is able to capture

[15] We had to make several assumptions in this comparison, since neither of the papers report the log base or K value in Eq. 5. We assume that both papers use K=100 to match our work. For [10] we assume they use log 2. Since we use log base 2 we are potentially giving their results an advantage over ours, because the JSD for log e returns smaller JSD in general. If they used log base 10 the comparison is fair, but if they used log e their JSDs will appear smaller. For [9] we first square their results because they report JS distance, not JS divergence. This results in a value that is 2 orders of magnitude lower than [10], even though they are using the same model architecture and dataset. Therefore, we assume that they are using log e.

[16] Daily sequences are easier to generate, but they are not as useful.

[17] All of [9], [10] and [6] depend on clustering.

daily but not weekly seasonality, which should be addressed in future work. Further, this provides strong evidence that the standard set of metrics used for evaluation is not sufficient.

5.4. Generating data for multiple buildings

Fig. 11 show that the relative magnitudes and spread of the loads were maintained for the buildings, even for the commercial data where there is a large difference in scale. The ability to retain these distributions is promising for practical applications such as storage and energy system design.

5.5. Limitations and future work

A limitation of our approach is that it generates sets of sequences, but there is no way to know what times the sequences are for, and since we used a rolling window to create more data for the training process, we do not know the start time of any generated sample. This also meant that we could not implement RMSE as a metric, since the daily peaks in the generated data happened at random times the aggregated themselves out. An attempt to overcome this issue was to condition on hourly temperature (LH) and to use that as a proxy for time, but the results did not exhibit strong performance. Future work should explore other ways to overcome this shortcoming.

Importantly, the metrics used to evaluate building load GANs require more discussion. This is particularly apparent based on the evaluation of the behaviour on commercial data. In this paper we introduce PCA and seasonal and trend strength. Our analysis of these was still qualitative and should be quantified, but the results showed that they provide fundamentally valuable information that is otherwise missed [10]. also used PRD and SSIM as scores, but these are image-specific so they cannot be applied here [18]. suggest the use of MMD, which, like JSD, quantifies the difference between two probability distributions [6]. use the KLD of the 5 key parameters for analyzing electric load shape suggested by [31]. These were not applied in this work because they measure daily load profiles and it is less clear how they should be applied for sequences with weekly or monthly seasonality. Both these and MMD should be explored further in future work; a study dedicated to the analysis of different metrics is required.

6. Conclusions

GANs are receiving increasing attention for generating building load sequences, but they often depend on large amounts of data are not always available and the temporal nature of the generated data is not quantitatively evaluated. This study developed an approach that uses substantially smaller datasets than those of previous works and expanded the metrics used for analysis. It was found that a multi-channel TimeGAN, conditioned on mean monthly outdoor temperature generates load sequences that are approximately 1 month in length with high fidelity on the residential case study, but that the commercial case had issues capturing weekly seasonality. In both cases, the numerical results are competitive with other GANs in the domain, even using only 1–2% of the data. Using a multivariate time series where each building represents a neural network input channel and conditioning on outdoor temperature are two novel insights that lead to strong generative performance on data. These are key insights that provide imperative information towards reducing data scarcity in the buildings domain.

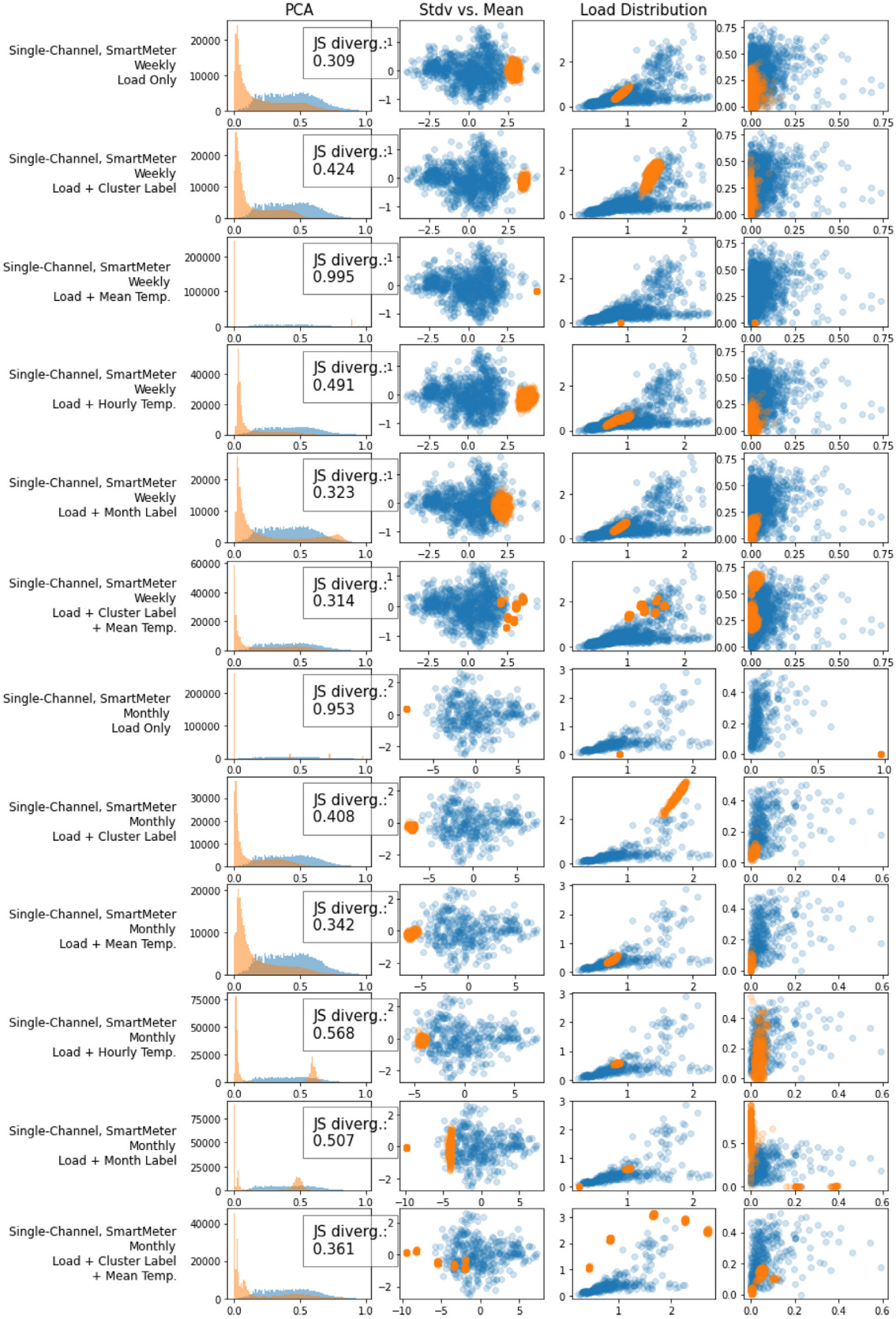
Declaration of Competing Interest

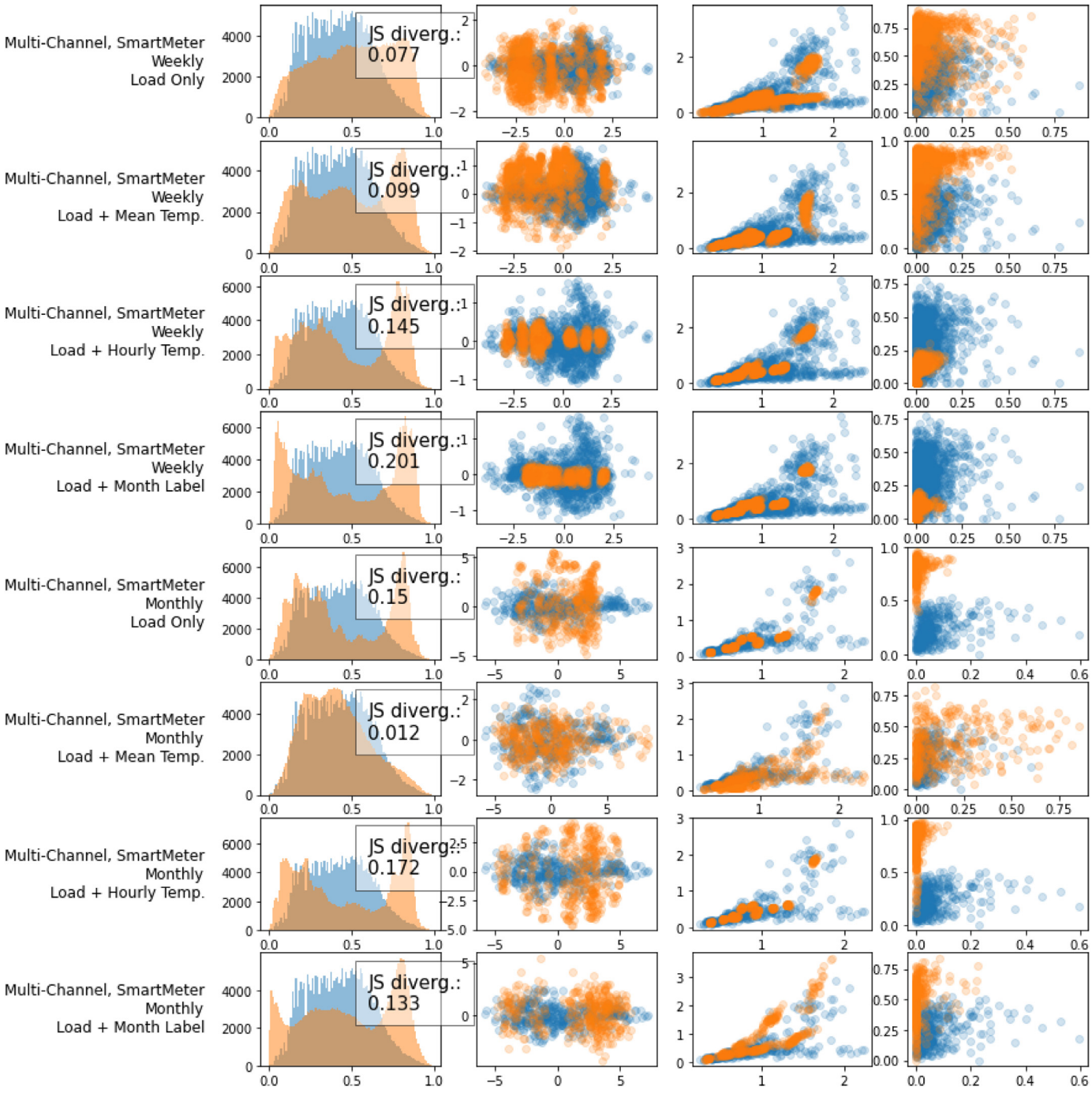
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This project was funded by a Canarie grant. Compute Canada provided the cloud resources used to run train the networks. The lead author was funded via an NSERC British Columbia Graduate Scholarship.

Appendix A. Distribution Plots for All Residential Experiments




Multi-Channel, SmartMeter
Weekly
Load Only
JS diverg.:
0.077
Multi-Channel, SmartMeter
Weekly
Load + Mean Temp.
JS diverg.:
0.099
Multi-Channel, SmartMeter
Weekly
Load + Hourly Temp.
JS diverg.:
0.145
Multi-Channel, SmartMeter
Weekly
Load + Month Label
JS diverg.:
0.201
Multi-Channel, SmartMeter
Monthly
Load Only
JS diverg.:
0.15
Multi-Channel, SmartMeter
Monthly
Load + Mean Temp.
JS diverg.:
0.012
Multi-Channel, SmartMeter
Monthly
Load + Hourly Temp.
JS diverg.:
0.172
Multi-Channel, SmartMeter
Monthly
Load + Month Label
JS diverg.:
0.133

References

[1] Goy S, Sancho-Tomás A. 4 - Load management in buildings. In: Eicker U, editor. Urban Energy Systems for Low-Carbon Cities. Academic Press; 2019. p. 137–79. ISBN 978-0-12-811553-4. doi:10.1016/B978-0-12-811553-4.00004-4. URL http://www.sciencedirect.com/science/article/pii/B9780128115534000044

[2] Pasichnyi O, Levihn F, Shahrokni H, Wallin J, Kordas O. Data-driven strategic planning of building energy retrofitting: the case of stockholm. J. Clean. Prod. 2019;233:546–60. doi:10.1016/j.jclepro.2019.05.373. URL http://www.sciencedirect.com/science/article/pii/S0959652619319158

[3] Tian C, Li C, Zhang G, Lv Y. Data driven parallel prediction of building energy consumption using generative adversarial nets. Energy Build. 2019;186:230–43. doi:10.1016/j.enbuild.2019.01.034. URL http://www.sciencedirect.com/science/article/pii/S0378778818322965

[4] Hong T, Wang Z, Luo X, Zhang W. State-of-the-art on research and applications of machine learning in the building life cycle. Energy Build. 2020;212:109831. doi:10.1016/j.enbuild.2020.109831. URL http://www.sciencedirect.com/science/article/pii/S0378778819337879

[5] Scully P. Smart meter market report. Tech. Rep.. IOT Analytics; 2019.

[6] Wang Z, Hong T. Generating realistic building electrical load profiles through the generative adversarial network (GAN). Energy Build. 2020;224:110299. doi:10.1016/j.enbuild.2020.110299. URL http://www.sciencedirect.com/science/article/pii/S0378778820307234

[7] Hu J, Vasilakos AV. Energy big data analytics and security: challenges and opportunities. IEEE Trans. Smart Grid 2016;7(5):2423–36. doi:10.1109/TSG.2016.2563461. Conference Name: IEEE Transactions on Smart Grid

[8] Roth J, Martin A, Miller C, Jain RK. Syncity: using open data to create a synthetic city of hourly building energy estimates by integrating data-driven and physics-based methods. Appl. Energy 2020;280:115981. doi:10.1016/j.apenergy.2020.115981. URL http://www.sciencedirect.com/science/article/pii/S0306261920314306

[9] Gu Y, Chen Q, Liu K, Xie L, Kang C. GAN-based model for residential load generation considering typical consumption patterns. In: 2019 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT); 2019. p. 1–5. doi:10.1109/ISGT.2019.8791575. ISSN: 2472–8152

[10] Wang Y, Chen Q, Kang C. Residential load data generation. In: Wang Y, Chen Q, Kang C, editors. Smart Meter Data Analytics: Electricity Consumer Behavior Modeling, Aggregation, and Forecasting. Springer; 2020. p. 99–135. ISBN 9789811526244. doi:10.1007/978-981-15-2624-4_5.

[11] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence N, Weinberger KQ, editors. Advances in Neural Information Processing Systems, 27. Curran Associates, Inc.; 2014. p. 2672–80. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[12] Karras T., Laine S., Aila T.. A style-based generator architecture for generative adversarial networks. arXiv:1812.04948.

[13] Dong H-W, Hsiao W-Y, Yang L-C, Yang Y-H. Musegan: multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. Proceedings of the AAAI Conference on Artificial Intelligence 2018;32(1). Number: 1, URL https://ojs.aaai.org/index.php/AAAI/article/view/11312

[14] Xue A. End-to-end chinese landscape painting creation using generative adversarial networks. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision; 2021. p. 3863–71. URL https://openaccess.thecvf.com/content/WACV2021/html/Xue_End-to-End_Chinese_Landscape_Painting_Creation_Using_Generative_Adversarial_Networks_WACV_2021_paper.html

[15] Pang Y, Zhou X, Xu D, Tan Z, Zhang M, Guo N, et al. Generative adversarial learning based commercial building electricity time series prediction. In: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI); 2019. p. 1800–4. doi:10.1109/ICTAI.2019.00271. ISSN: 2375-0197

[16] Moon J, Jung S, Park S, Hwang E. Conditional tabular GAN-based two-stage data generation scheme for short-term load forecasting. IEEE Access 2020;8:205327–39. doi:10.1109/ACCESS.2020.3037063. Conference Name: IEEE Access

[17] Fekri MN, Ghosh AM, Grolinger K. Generating energy data for machine learning with recurrent generative adversarial networks. Energies 2020;13(1):130. doi:10.3390/en13010130. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, URL http://www.mdpi.com/1996-1073/13/1/130

[18] Zhang C, Kuppannagari SR, Kannan R, Prasanna VK. Generative adversarial network for synthetic time series data generation in smart grids. In: 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm); 2018. p. 1–6. doi:10.1109/SmartGridComm.2018.8587464.

[19] Chokwitthaya C, Zhu Y, Mukhopadhyay S, Collier E. Augmenting building performance predictions during design using generative adversarial networks and immersive virtual environments. Autom. Constr. 2020;119:103350. doi:10.1016/j.autcon.2020.103350. URL http://www.sciencedirect.com/science/article/pii/S0926580520309304

[20] Kababji SE, Srikantha P. A data-driven approach for generating synthetic load patterns and usage habits. IEEE Trans. Smart Grid 2020;11(6):4984–95. doi:10.1109/TSG.2020.3007984. Conference Name: IEEE Transactions on Smart Grid

[21] Chen Y, Wang Y, Kirschen D, Zhang B. Model-free renewable scenario generation using generative adversarial networks. IEEE Trans. Power Syst 2018;33(3):3265–75. doi:10.1109/TPWRS.2018.2794541. Conference Name: IEEE Transactions on Power Systems

[22] LeCun Y, Cortes C. MNIST handwritten digit databaseURL http://yann.lecun.com/exdb/mnist/. 2010.

[23] Yoon J, Jarrett D, van der Schaar M. Time-series generative adversarial networks. In: Wallach H, Larochelle H, Beygelzimer A, Alché-Buc Fd, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems, 32. Curran Associates, Inc.; 2019. p. 5508–18. URL https://proceedings.neurips.cc/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf

[24] Mirza M, Osindero S. Conditional generative adversarial nets. arXiv:1411.1784.

[25] Goodfellow IJ, Bengio Y, Courville A. Deep learning. Cambridge, MA, USA: MIT Press; 2016. http://www.deeplearningbook.org

[26] Bartholomew D. Latent variable models and factor analysis. a unified approach. 3 auflage. Chichester: Wiley; 2011.

[27] Miller C, Meggers F. The building data genome project: an open, public data set from non-residential building electrical meters. Energy Procedia 2017;122:439–44. doi:10.1016/j.egypro.2017.07.400. {CISBAT} 2017 International Conference Future Buildings; Districts - Energy Efficiency from Nano to Urban Scale, URL http://www.sciencedirect.com/science/article/pii/S1876610217330047

[28] Bryant FB, Yarnold PR. Principal-components analysis and exploratory and confirmatory factor analysis. In: Reading and understanding multivariate statistics. American Psychological Association; 1995. p. 99–136. ISBN 978-1-55798-273-5.

[29] Hyndman R, Athanasopoulos G. Forecasting: principles and practice. 2nd edition. Melbourne, Australia: OTexts; 2018. OTexts.com/fpp2

[30] Cleveland RB, Cleveland WS, McRae JE, Terpenning I. Stl: a seasonal-trend decomposition procedure based on loess (with discussion). J. Off. Stat. 1990;6:3–73.

[31] Price P. Methods for analyzing electric load shape and its variability. Tech. Rep.. Lawrence Berkeley National Lab(LBNL), Berkeley, CA (United States); 2010.