# Generating Synthetic Time-Series Data For Smart-Building Knowledge Graphs Using Generative Adversarial Networks

Jesse van Haaster[2742175], Victor de Boer

De Boelelaan 1105, 1081HV Amsterdam, The Netherlands
j.w.van.haaster2@student.vu.nl

**Abstract.** Knowledge Graphs represent data in triples to link connected data points to each other. This form of knowledge representation has several applications, such as querying and finding information, or making data inferences. However, in some domains, such as medical records or smart house devices, these knowledge graphs are hard to make publicly available on a large scale due to privacy-concerns. Therefore it would be beneficial to have a way to generate synthetic knowledge graph data from the original data to use on a large scale. The goal of this paper is to find out to what extent it is possible to create meaningful synthetic time-series data for knowledge graphs. To try and find a solution for creating data with similar features to the original data, two existing generative adversarial networks (GAN) will be tested, CTGAN and TimeGAN. The results from the experiment show that both models manage to capture some important features from the dataset, but neither model possesses all of the features from the original data. Further investigation needs to be done to find a solution that satisfies the requirements for a meaningful synthetic knowledge graph.

## 1 Introduction

According to Feng[6], knowledge graphs (KGs) provide useful interpretations of information, by connecting nodes and edges to each other. The nodes stand for entities in the graph and the edges show the relationship two nodes have to one another. This way knowledge can be represented explicitly, making it easier to use. However in some domains, such as Smart Home data, not all of these knowledge graphs can be made publicly available, because it contains personal and privacy-sensitive data. Because of this it would be useful to be able to generate a synthetic knowledge that has the same features as the original dataset, but does not run the risk of exposing any personal data. According to Lantzaki [9] these sets could also be used for benchmarking certain data management tasks like querying, storage, updating, comparing or integrating. It could also be used as a way of testing knowledge graph inference algorithms according to Feng[6]. In the knowledge graph used for this research called OfficeGraph [12], measurements have been made for a year tracking various data, some of which are temperature, $CO_2$ value, humidity, etc. using a variety of smart house

sensors. The data in OfficeGraph comes from an office building in Eindhoven and the dataset contains approximately a years' worth of measurements. These same types of devices also hang inside people's homes. However, this data cannot be freely used due to privacy concerns. Therefore it would be beneficial to train a model based on the original data, to be able to generate synthetic knowledge graphs. These graphs would contain values based on the values from the original knowledge graph, but would be freely usable in the public domain, as the contents no longer contain sensitive personal data and are completely anonymised. Van der Weerdt[12] already presents a couple of important data analytics tasks that could be done with the recorded data, such as checking whether thermostats are turned off when windows are opened, or finding the influence of people on the $CO_2$ values in the office. He also points to the possibility of performing machine learning tasks on the dataset. An important feature of this data is that the values inside the knowledge are dependent on one another. Measurements from a device that were taken close together are related to each other in the way that the values they measure are quite similar, as temperatures change gradually over time instead of spontaneously. The data can therefore also be described as time-series data. It is important that this feature is taken into account when creating synthetic data. In this paper, two approaches for creating synthetic KG data are proposed and evaluated. CTGAN [13] and TimeGAN [14] are both models that could prove to be useful for the task of generating synthetic data. CTGAN is a Generative Adversarial Network (GAN) that was made specifically to generate synthetic tabular data. TimeGAN is a GAN model that was created to be able to generate synthetic time-series data. According to Goodfellow [7], a GAN is a model that employs unsupervised learning, by letting two models 'compete' against one another, a generator and a discriminator. The generator model tries to create values that resemble the distribution of the original data. The distribution is however unknown to the generator, and thus it has to 'guess' what values would be correct. At the same time, the discriminator tries to guess whether the values it receives as input are either real (taken from the original dataset) or fake (created by the generator model). Because the models compete against each other, they force each other to improve. Eventually, in the best case, this causes the model to perform so well, that the samples it creates are indistinguishable from the original data. The contribution of this paper lies in answering the question: To what extent can meaningful synthetic time-series data be generated for knowledge graphs?

## 2 Related Work

Multiple research projects have already been published regarding synthetic KG generation. Feng et al. [6] created GDDx (Graph Differential Dependencies), which generates a synthetic knowledge graph by first describing some of the desired constraints for the knowledge graph, and then generating a knowledge graph by adding these dependencies and the data schema into the generation algorithm. Melo and Paulheim[10] create synthetic knowledge graphs by first

manually modelling the distributions of the original knowledge graph, and then using the structure and data distribution of the initial graph to create a synthetic graph. In contrast to this, I will use the original data to train a model to learn the distribution which will generate synthetic data for the new graph. The synthetic data will then be converted back into knowledge graph form using the structure from the original graph. Hubert et al. [8] proposed PyGraft, which can generate both knowledge graphs and schemas. In contrast to this and other work, PyGraft does not require any form of KG or schema as input, instead, the algorithm solely needs user input for parameters the synthetic graph should use. From these parameters the algorithm creates a schema, the schema being the total construct of classes, relations and individuals present in the knowledge graph. This schema is then checked for any possible inconsistencies. Finally, the algorithm generates the knowledge graph. A big difference between these works and mine is that OfficeGraph contains time-series data, which has important temporal dependencies that a model should take into account. In the field of generating synthetic time-series data a lot of research has been done. Yu et al.[15] create SeqGAN, which can generate sequences by incorporating a reward system based on Monte Carlo into the generator, which rewards proper sequences. Alzantot [1] et al. created SenseGen, which is built upon an LSTM (Long Short Term Memory) model. They use this model to try and create meaningful sensor data that can both maintain the statistical properties of the data, and the realism of the data. Dahmen and Cook[2] created SynSys, which uses HMMs (Hidden Markov Models) to generate synthetic smart home sensor data. These papers have the same goal as this paper, to create synthetic time-series data. The work of Dahmen and Cook comes closest to my work. However in this paper the focus lies in doing this for knowledge graph data specifically, which means the data should also be converted back into knowledge graph form after generating the data.

## 3 Research Method

### 3.1 Explaining the Knowledge Graph

The goal of this paper is to find whether meaningful synthetic data can be created for knowledge graphs. First however, the necessary data needs to be extracted from the OfficeGraph knowledge graph so that it can be used to train a data generation model. Before proper extraction can be done, first the knowledge graph should be properly analysed. OfficeGraph [12] is a knowledge graphs that captured data inside an office building over the span of eleven months. It contains measurements taken by 444 different IoT devices, by 19 different models of devices, with a total of 11 different possible properties it can capture. (It differs per device model which properties are captured.) Every device has a file containing its own subset of the OfficeGraph knowledge graph inside the knowledge graph containing all of the measurements taken over those eleven months.

Officegraph is built upon the SAREF(Smart Applications Reference) ontology [3] and is part of the InterConnect project. Siebes[11] states that the appli-

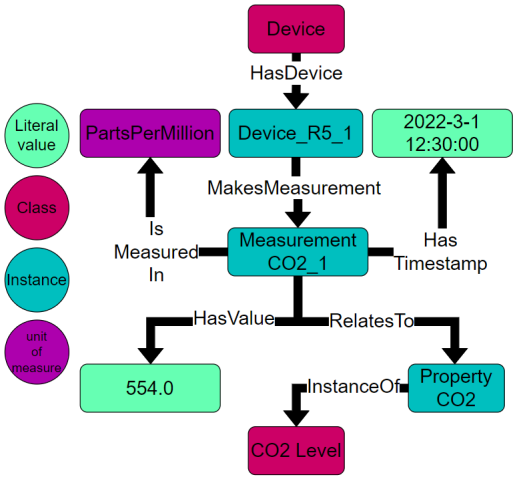



**Fig. 1.** A simplified representation of the knowledge graph structure.

cations for these devices lies in energy management, home safety and assisted living. However due to poor interoperability between different devices it isn't being used to its full potential. SAREF was built to be able to connect data from different smart devices to each other using shared concepts. For the experiment, only the Airwits R5 subset from OfficeGraph is used. With the code I made, the data from every R5 device ontology can be collected for use. With some additions to the code it could be made universal for all devices from OfficeGraph, but for this project, a focus on the R5 devices is plenty. The R5 devices contain continuous data points for the entire year in which the experiment was done, starting in March 2022 and ending in January 2023. Approximately every 30 minutes, the devices made a measurement of the $CO_2$ level, temperature, and humidity. These values were stored in 3 different entities inside the KG, one for every type of measurement. Every measurement got a number assigned to it, which makes it possible to retrace which entities came from what timestamp. This also makes it easier to find which entities belonged to the same measurement.

### 3.2 Extracting the data

To collect the data from the ontology, the algorithm below was used. It was created to be able to iterate over the triples in the KG data, and collect only the entities containing either the temperature, CO2, humidity or timestamp.

First a dictionary is made. Then, for every triple in the KG, the ID assigned to the measurement is taken, and used to check whether said ID is already in the dictionary. If the ID is not yet in the dictionary, a new list is initialised in the dictionary with the ID as its label. The measurement is then checked to see what type of measurement it is ($CO_2$, temperature, humidity, or time). The measurement is then placed into the correct spot in the list. After running

this algorithm on the ontology you get a dictionary of measurements containing the timestamp, $CO_2$ value, temperature and humidity of a measurement. The dictionary is then transformed into a dataframe for easier usage. The reason it is first created as a dictionary is to make sure all the measurements end up in the correct place in the list, as otherwise $CO_2$, temperature and humidity might get placed in the incorrect column.

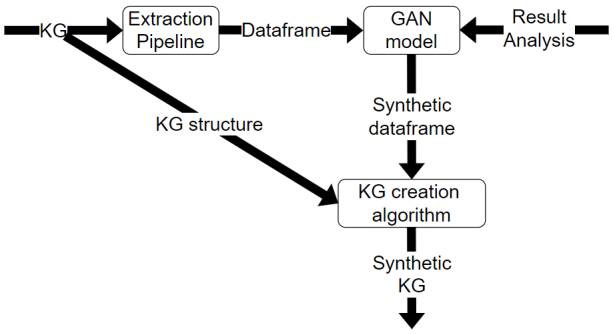



**Fig. 2.** A simplified version of the pipeline used in this thesis.

### 3.3 CTGAN

With the now collected data, CTGAN can be used. CTGAN is a model based on GAN that specialises in creating synthetic tabular data. As the data extracted from the knowledge graph is in the form of a table, it seemed useful to use CTGAN for the synthetic data generation. According to Xu[13], CTGAN uses mode-specific normalisation to model data that does not follow a Gaussian distribution. From their research, it followed that CTGAN performed better than Bayesian networks in at least 87.5 percent of datasets. CTGAN learns the joint distribution of all the columns. This important feature in CTGAN makes sure that the generated data is reliant on each other, making the synthetic data more reliable. The CTGAN model is made with a learning rate of $2e^{-3}$, does not enforce the minimum and maximum values of the original dataset, and is run for 1000 epochs. The reason for not enforcing minimum and maximum values from the original data is that this would cause the model to create a lot of time values at the edge of the original timespan, and because it was not allowed to go further back/forward in time, it creates a lot of duplicate values for the minimum and maximum value of time. It is also more realistic in the sense that in rare cases, such outliers could happen, and therefore should be accounted for. After creating a model using CTGAN, the model can be sampled to create synthetic data. The sampled synthetic data is also in the form of a dataframe. Now that the synthetic data has been created, it needs to be converted back into knowledge graph form. To do this, I analyse the original KG structure and copy the structure of the initial ontology, to then iterate over the synthetic dataframe. For every row in

the dataframe, three new URIs are created, one for each value in the row. Onto these URIs I add all the same attributes that the original dataframe uses. In the end, a synthetic KG will have been created with the same structure as the original KG.

### 3.4 TimeGAN

TimeGAN is a model created by Yoon et al.[14] for the purpose of generating synthetic time-series data. The model is made specifically to maintain temporal dynamics between rows of data. This could prove to be beneficial for the OfficeGraph dataset, as adjacent rows are dependent on each other. TimeGAN simultaneously learns to encode features, generate representations and iterate over time. TimeGAN takes a 3 dimensional array as input, therefore the collected data first needs to be transformed before it can be used. First, I convert the time values to numerical timestamps, so that they are readable for the model. Then the tabular data is put through a MinMax scaler, so all of the data is in the same format. Now the data has been transformed into a 2 dimensional array. To make timeGAN learn the temporal dynamics, each row is added together with the 3 previous rows. This changes the 2 dimensional array into a 3 dimensional array. The data is then also mixed, so the created model does not overfit the data. This process also causes the dataset to become 4 times as large, as every data point is recorded 4 times. The data is now completely processed, and ready to be used for the TimeGAN model. After timeGAN has generated synthetic data, the data needs to be converted back into tabular data, so that the data can be added into a knowledge graph. Because all the data is used 4 times to capture the temporal dynamics of the original data, the synthetic set becomes 4 times as large as the original. Therefore the synthetic set is sampled for 25 percent of its size to be able to compare it to the original dataset. The same process used to convert the data into an array is then used in reverse to revert back to tabular data, inverse transforming the MinMaxed data to get real values again and then creating a new table with all of the array values. The data can now be put into a new knowledge graph.

### 3.5 Evaluation Setup

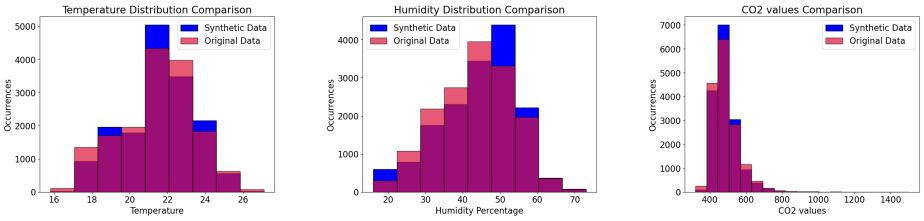
To evaluate the performance of both models, both quantitative and qualitative methods are used. According to Eigenschink et al. [5], human evaluations play an important part in evaluating the realism of a synthetic dataset. In this paper, this is done by creating graphs of the synthetic data and comparing it to the original data. First, the data distribution of the synthetic model is compared to the original distribution. It is important that the distributions are similar to one another, as the synthetic data should be a representation of reality. A T-test is also performed to test whether the distributions are independent. Second, the average distance between adjacent data points is measured(Euclidean Distance). Donahue et al. [4] used the mean Euclidean distance to calculate the difference between the real dataset and a synthetic dataset. This metric can be applied here

too to compare the original difference to the synthetic difference, however instead of measuring the distance between the original data and the synthetic data, I test the difference between neighbouring datapoints in both the original data and the synthetic data. This way it can be tested whether adjacent datapoints have similar fluctuations as the original data, which is important for the realness of the synthetic data. A T-test is again performed to check for independency of the average distances. Lastly a visual comparison is done of the distribution of all the Euclidean distances of the original and synthetic dataset. This is done to make sure that the average Euclidean distance is an actual representation of the data, and not by outliers. In summary, according to my requirements, the synthetic data should follow the same distribution as the original data and the same distances between data points to qualify as meaningful synthetic time-series data.

# 4 Results

## 4.1 CTGAN

CTGAN creates a model of the data distribution in the original data, which means that any amount of synthetic data rows can be created. For the simplicity of the results, I sampled the same amount of rows as the original dataset. As stated in the evaluation setup, first a comparison is made of the data distribution of the synthetic data and the original data.




The graphs show the similarity of these two datasets. As can be seen from these graphs, the synthetic data mimics the distribution from the original data. The synthetic distributions seem to accurately depict the original dataset. To test whether CTGAN actually follows a similar distribution to the original, a t-test will be performed. For the t-test a p-value of 0.05 is taken, with the null hypothesis being that there is no statistical difference between the two datasets. The results of the test are as follows.

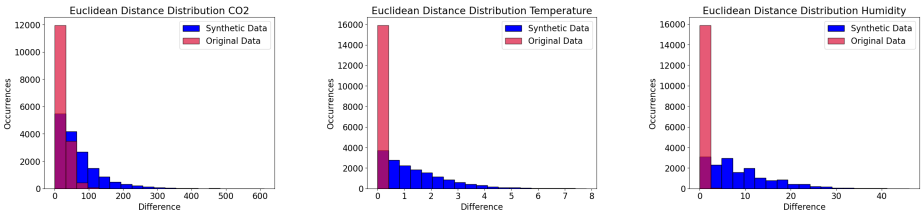
| Feature | t-statistic | p-value |
|---|---|---|
| $CO_2$ | 1.21 | 0.225 |
| Temperature | -4.22 | 0.000 |
| Humidity | -8.82 | 0.000 |

From the t-test we can gather that the synthetic data for temperature and humidity still follow a distribution that is significantly different from the original data, however the values from CO2 are so similar that the null hypothesis cannot be rejected. This shows that CTGAN performs rather well at capturing the data distribution, despite the fact the data is not evenly distributed.

Second, the average Euclidean distance of the original and synthetic data is compared. The results of the calculation are seen in the table below.

| Feature | Real Data | Synthetic Data |
|---|---|---|
| $CO_2$ | 23.19 | 81.03 |
| Temperature | 0.07 | 1.39 |
| Humidity | 0.26 | 8.23 |

As can be seen in the table, the difference between two real datapoints is significantly smaller than between the synthetic points for all 3 data columns. An explanation for this is that CTGAN randomly samples datapoints from its' distribution, and does not take any previous points into account when generating a new row of data. In the original data however, there are only small differences between datapoints, as previous values play an important role in the current value. Because of this, the values in the synthetic dataset fluctuate more than in the original data and are thus further apart. In the graphs below we can see this clearly again. The distance between rows of the synthetic data seems to be dependent purely on the likelihood of the next row being close or far away from the current point.




Again a t-test is performed to test whether the values are in fact significantly different.

| Feature | t-statistic | p-value |
|---|---|---|
| $CO_2$ | -55.68 | 0.000 |
| Temperature | -62.18 | 0.000 |
| Humidity | -75.46 | 0.000 |

From the t-test we can gather that the real data and the synthetic data are in fact significantly different.

When looking at a sample of the synthetic data and comparing it to the real data, the same issue can be seen. Changes in the real data appear gradual, where the differences between the synthetic data fluctuate a lot. As can be seen in the samples below, there is a visible difference between the actual data and the synthetic data when it comes to differences between neighbouring points.

| time | co2_values | temp_values | humidity_values |
|---|---|---|---|
| 2022-03-05 17:58:00 | 452.0 | 22.4 | 23.0 |
| 2022-03-05 19:03:00 | 486.0 | 21.5 | 30.0 |
| 2022-03-05 19:52:00 | 420.0 | 21.4 | 33.0 |
| 2022-03-05 20:14:00 | 517.0 | 22.1 | 19.0 |
| 2022-03-05 20:45:00 | 538.0 | 21.4 | 23.0 |
| 2022-03-05 20:59:00 | 573.0 | 23.6 | 22.0 |
| 2022-03-05 21:29:00 | 464.0 | 21.3 | 23.0 |
| 2022-03-05 21:44:00 | 453.0 | 21.5 | 24.0 |
| 2022-03-05 22:11:00 | 518.0 | 21.5 | 53.0 |
| 2022-03-05 22:26:00 | 467.0 | 21.6 | 27.0 |
| 2022-03-05 22:45:00 | 420.0 | 21.2 | 20.0 |
| 2022-03-05 22:53:00 | 483.0 | 21.3 | 32.0 |
| 2022-03-06 00:03:00 | 463.0 | 21.7 | 22.0 |
| 2022-03-06 01:01:00 | 430.0 | 21.4 | 25.0 |
| 2022-03-06 01:30:00 | 529.0 | 22.3 | 23.0 |
| 2022-03-06 01:53:00 | 457.0 | 21.8 | 21.0 |
| 2022-03-06 02:25:00 | 471.0 | 21.8 | 22.0 |
| 2022-03-06 02:40:00 | 453.0 | 22.6 | 22.0 |
| 2022-03-06 02:46:00 | 518.0 | 21.2 | 25.0 |
| 2022-03-06 03:21:00 | 536.0 | 21.7 | 21.0 |

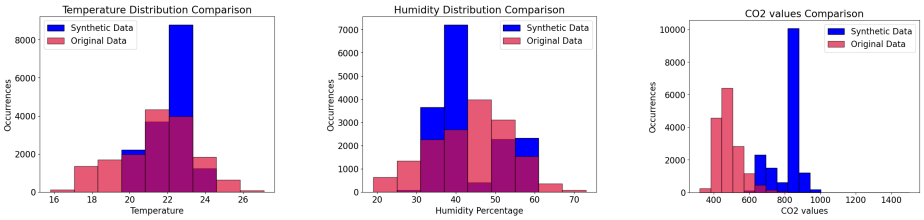
**Fig. 3.** Snippet of CTGAN data

| time | co2_values | temp_values | humidity_values |
|---|---|---|---|
| 2022-03-11 08:06:00 | 405.0 | 21.0 | 23.0 |
| 2022-03-11 08:37:00 | 448.0 | 21.0 | 23.0 |
| 2022-03-11 09:06:00 | 467.0 | 21.1 | 23.0 |
| 2022-03-11 09:36:00 | 480.0 | 21.1 | 23.0 |
| 2022-03-11 10:05:00 | 507.0 | 21.1 | 24.0 |
| 2022-03-11 10:35:00 | 490.0 | 21.2 | 24.0 |
| 2022-03-11 11:06:00 | 549.0 | 21.4 | 24.0 |
| 2022-03-11 11:35:00 | 554.0 | 21.6 | 24.0 |
| 2022-03-11 12:05:00 | 550.0 | 21.7 | 24.0 |
| 2022-03-11 12:35:00 | 556.0 | 22.0 | 24.0 |
| 2022-03-11 13:05:00 | 523.0 | 22.1 | 24.0 |
| 2022-03-11 13:34:00 | 496.0 | 22.2 | 24.0 |
| 2022-03-11 14:05:00 | 536.0 | 22.4 | 24.0 |
| 2022-03-11 14:34:00 | 528.0 | 22.5 | 24.0 |
| 2022-03-11 15:04:00 | 512.0 | 22.5 | 24.0 |
| 2022-03-11 15:34:00 | 530.0 | 22.4 | 24.0 |
| 2022-03-11 16:04:00 | 514.0 | 22.2 | 24.0 |
| 2022-03-11 16:34:00 | 525.0 | 22.1 | 24.0 |
| 2022-03-11 17:03:00 | 502.0 | 22.0 | 24.0 |
| 2022-03-11 17:33:00 | 498.0 | 22.0 | 24.0 |

**Fig. 4.** Snippet of real data

### 4.2 TimeGAN

I ran timeGAN on the GRU module(Gated Recurrent Unit), using 6 hidden dimensions, 3 layers, 3000 iterations and a batch size of 128. The size of the parameters was purposely kept relatively low, as the running of timeGAN takes a lot of time and processing power. It was also found that the model would start converging to one singular measurement per column the higher the values for hidden dimensions, layers or iterations was taken, eventually only outputting one singular measurement for every column. As stated earlier, the size of the TimeGAN output is 4 times as large as that of the original data. For that reason the TimeGAN model is sampled for a quarter of its size, so it can be properly compared to the original data.



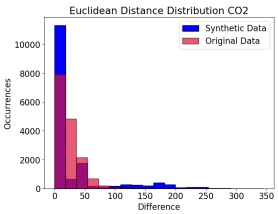

As can be seen in the graphs above, timeGAN does not manage to properly learn the data distribution. For temperature and humidity the generated data is at least within bounds of the original data, but for the $CO_2$ values the most data points are shifted away from the original data, where the original data only has outliers, completely missing the original distribution. Another issue the data distribution has are the big outliers in occurrences of certain values. Because of this the general distributions are also affected. The performed t-test shows the clear difference in values, especially for the $CO_2$ values.

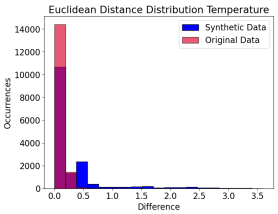
| **Feature** | **t-statistic** | **p-value** |
|---|---|---|
| $CO_2$ | -381.23 | 0.000 |
| Temperature | -34.08 | 0.000 |
| Humidity | 4.57 | 0.000 |

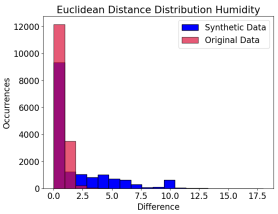
In these figures we see that in the original data, the fluctuations between $CO_2$ values are bigger than the fluctuations in the synthetic data set. We compare the average Euclidean distance from both sets again to check their similarity.

| **Feature** | **Real Data** | **Synthetic Data** |
|---|---|---|
| $CO_2$ | 23.19 | 26.60 |
| Temperature | 0.07 | 0.25 |
| Humidity | 0.26 | 1.9 |

The synthetic data still, although less so than the CTGAN model, fluctuates more than the original dataset. In the graphs below we can see that the differences are closer to those of the original data, but TimeGAN still makes mistakes with bigger differences than those in the original data. It also shows that for $CO_2$ the synthetic model might be too conservative in value differences, as it has a lot more low difference values than the original.




Similar as to the CTGAN data, a t-test will be performed to check for similarity between these datasets. The results of this T-test are seen below.

| **Feature** | **t-statistic** | **p-value** |
|---|---|---|
| $CO_2$ | -11.81 | 0.000 |
| Temperature | -46.15 | 0.000 |
| Humidity | -73.14 | 0.000 |

When looking at a snippet of TimeGAN data, we can see that the adjacent values are indeed more similar to the original than the CTGAN data, but the values for $CO_2$ are not realistic, and the synthetic data seems more static than the original data. On top of that, the data TimeGAN produces is very inconsistent, sometimes producing data that seems somewhat realistic, and at other times producing data that is either completely static or very erratic.

| | time | co2_values | temp_values | humidity_values |
|---|---|---|---|---|
| 0 | 2022-09-03 23:31:20 | 811.5 | 22.6 | 38.5 |
| 1 | 2022-09-03 23:34:06 | 811.2 | 22.6 | 38.7 |
| 2 | 2022-09-04 00:30:37 | 811.6 | 22.6 | 38.5 |
| 3 | 2022-09-04 02:13:09 | 812.2 | 22.9 | 40.9 |
| 4 | 2022-09-04 02:14:30 | 821.1 | 22.8 | 44.1 |
| 5 | 2022-09-04 02:46:09 | 829.1 | 22.1 | 35.9 |
| 6 | 2022-09-04 03:16:29 | 821.2 | 22.8 | 44.1 |
| 7 | 2022-09-04 06:06:41 | 823.6 | 22.8 | 43.7 |
| 8 | 2022-09-04 12:10:26 | 811.3 | 22.8 | 39.7 |
| 9 | 2022-09-04 17:49:15 | 810.8 | 22.6 | 38.8 |
| 10 | 2022-09-04 18:29:01 | 811.4 | 22.8 | 40.2 |
| 11 | 2022-09-04 19:51:04 | 812.6 | 22.8 | 40.0 |
| 12 | 2022-09-04 20:40:16 | 810.8 | 22.6 | 38.7 |
| 13 | 2022-09-04 22:58:25 | 810.8 | 22.6 | 38.8 |
| 14 | 2022-09-05 04:58:12 | 813.9 | 22.9 | 40.8 |
| 15 | 2022-09-05 08:17:02 | 821.9 | 22.8 | 44.2 |
| 16 | 2022-09-05 09:12:18 | 810.0 | 22.7 | 40.4 |
| 17 | 2022-09-05 12:31:55 | 810.2 | 22.8 | 41.4 |
| 18 | 2022-09-05 12:39:30 | 819.0 | 22.4 | 37.2 |
| 19 | 2022-09-05 16:03:01 | 810.5 | 22.6 | 38.9 |

**Fig. 5.** TimeGAN

| | time | co2_values | temp_values | humidity_values |
|---|---|---|---|---|
| 0 | 2022-07-17 15:13:09 | 857.9 | 22.0 | 38.7 |
| 1 | 2022-07-17 15:13:09 | 857.9 | 22.0 | 38.7 |
| 2 | 2022-07-17 15:13:12 | 857.9 | 22.0 | 38.7 |
| 3 | 2022-07-17 15:13:14 | 857.9 | 22.0 | 38.7 |
| 4 | 2022-07-17 15:13:14 | 857.9 | 22.0 | 38.7 |
| 5 | 2022-07-17 15:13:14 | 857.9 | 22.0 | 38.7 |
| 6 | 2022-07-17 15:13:14 | 857.9 | 22.0 | 38.7 |
| 7 | 2022-07-17 15:13:14 | 857.9 | 22.0 | 38.7 |
| 8 | 2022-07-17 15:13:16 | 857.9 | 22.0 | 38.7 |
| 9 | 2022-07-17 15:13:16 | 857.9 | 22.0 | 38.7 |
| 10 | 2022-07-17 15:13:17 | 857.9 | 22.0 | 38.7 |
| 11 | 2022-07-17 15:13:18 | 857.9 | 22.0 | 38.7 |
| 12 | 2022-07-17 15:13:18 | 857.9 | 22.0 | 38.7 |
| 13 | 2022-07-17 15:13:18 | 857.9 | 22.0 | 38.7 |
| 14 | 2022-07-17 15:13:18 | 857.9 | 22.0 | 38.7 |
| 15 | 2022-07-17 15:13:19 | 857.9 | 22.0 | 38.7 |
| 16 | 2022-07-17 15:13:19 | 857.9 | 22.0 | 38.7 |
| 17 | 2022-07-17 15:13:19 | 857.9 | 22.0 | 38.7 |
| 18 | 2022-07-17 15:13:19 | 857.9 | 22.0 | 38.7 |
| 19 | 2022-07-17 15:13:19 | 857.9 | 22.0 | 38.7 |

**Fig. 6.** TimeGAN

| | time | co2_values | temp_values | humidity_values |
|---|---|---|---|---|
| 0 | 2022-03-11 08:06:00 | 405.0 | 21.0 | 23.0 |
| 1 | 2022-03-11 08:37:00 | 448.0 | 21.0 | 23.0 |
| 2 | 2022-03-11 09:06:00 | 467.0 | 21.1 | 23.0 |
| 3 | 2022-03-11 09:36:00 | 480.0 | 21.1 | 23.0 |
| 4 | 2022-03-11 10:05:00 | 507.0 | 21.1 | 24.0 |
| 5 | 2022-03-11 10:35:00 | 490.0 | 21.2 | 24.0 |
| 6 | 2022-03-11 11:06:00 | 549.0 | 21.4 | 24.0 |
| 7 | 2022-03-11 11:35:00 | 554.0 | 21.6 | 24.0 |
| 8 | 2022-03-11 12:05:00 | 550.0 | 21.7 | 24.0 |
| 9 | 2022-03-11 12:35:00 | 556.0 | 22.0 | 24.0 |
| 10 | 2022-03-11 13:05:00 | 523.0 | 22.1 | 24.0 |
| 11 | 2022-03-11 13:34:00 | 496.0 | 22.2 | 24.0 |
| 12 | 2022-03-11 14:05:00 | 536.0 | 22.4 | 24.0 |
| 13 | 2022-03-11 14:34:00 | 528.0 | 22.5 | 24.0 |
| 14 | 2022-03-11 15:04:00 | 512.0 | 22.5 | 24.0 |
| 15 | 2022-03-11 15:34:00 | 530.0 | 22.4 | 24.0 |
| 16 | 2022-03-11 16:04:00 | 514.0 | 22.2 | 24.0 |
| 17 | 2022-03-11 16:34:00 | 525.0 | 22.1 | 24.0 |
| 18 | 2022-03-11 17:03:00 | 502.0 | 22.0 | 24.0 |
| 19 | 2022-03-11 17:33:00 | 498.0 | 22.0 | 24.0 |

**Fig. 7.** Real Data

## 5 Discussion

The results show that both models have their limitations in synthetic data generation. CTGAN performs well at learning the data distribution of the original data, but lacks the ability to capture the temporal dynamics in the original data, which makes consecutive datapoints random, instead of letting previous data influence the new point. Because of this, the synthetic datapoints on their own are an accurate representation of the original data, but the data as a whole does not fully capture the dynamics of the original data. TimeGAN does somewhat capture the temporal dynamics from the original data, but performs worse than CTGAN at capturing the data distribution. It struggles at times to transition to different values, which causes a lot of the data points to be very similar. It also sometimes does the opposite, causing data fluctuations that are too large. Neither of these models can fully satisfy all of the necessities needed for a valid synthetic knowledge graph, however, in case a synthetic knowledge graph is needed where the time-series aspect of the data is not important for the task, CTGAN could be a solution for creating synthetic knowledge graphs, as it created data that was statistically very similar to the original data.

The pipeline I created was focused around the subgraphs of the R5 devices in OfficeGraph. Other insights might have been found if the other devices would have been included too, as they contain different kinds of data from the R5 devices. In the limited time I had, I also only got to test two different models. With more time, other approaches could have been tried, such as LSTM (Long Short Term Memory) models or HMM(Hidden Markov Model) models.

### 5.1 Future Work

In future work, more research could be done into finding a well performing model for time-series data. The pipeline I created for this project could be used again to try other data generation models on the data to see whether they show better results. With some minor adjustments to the code, the pipeline I created could also possibly be used on other knowledge graphs for extracting data. I chose to first extract the data from the knowledge graph to then use for synthetic data generation, however in research such as that of Feng et al. [6], a new synthetic graph was immediately generated from the initial graph. This could be tried on OfficeGraph as well. It could also be studied what caused TimeGAN to have an inconsistent output. Some possible explanations for the results could be that somewhere in the preprocessing of the data for timeGAN a mistake was made. It is also possible that TimeGAN was not made for this type of data, and is therefore showing inconsistencies.

## 6 Conclusion

In this paper, an attempt was made at generating meaningful synthetic time-series data for knowledge graphs. I determined that for synthetic data to be meaningful, it needs to have a similar data distribution as the original data, and needs to have adjacent data rows be dependent on each other. Looking at the CTGAN and TimeGAN model, neither model fulfils all these criteria. The CTGAN model manages to accurately capture the distribution of the dataset, but does not capture any of the temporal dynamics present in the original dataset. Because of this neighbouring data points tend to fluctuate significantly more than they should. The TimeGAN model manages to somewhat capture the temporal dynamics of the data, but the data distribution does not come close to that of the original data. The output data is also not consistent, showing weird patterns across time, sometimes generating completely static data, and sometimes generating very different data, depending on the time linked to the row. Through this paper, I managed to gain a better understanding of the complexity of time-series data, and found a way to create synthetic knowledge graph data for the OfficeGraph dataset. The paper shows that more research needs to be done into synthetic data generation models to generate data for OfficeGraph that satisfies all requirements.

## References

1. Alzantot, M., Chakraborty, S., Srivastava, M.: SenseGen: A deep learning architecture for synthetic sensor data generation. In: 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). pp. 188–193 (Mar 2017). https://doi.org/10.1109/PERCOMW.2017.7917555, https://ieeexplore.ieee.org/abstract/document/7917555?
2. Dahmen, J., Cook, D.: SynSys: A Synthetic Data Generation System for Healthcare Applications. Sensors **19**(5), 1181 (Jan 2019). https://doi.org/10.3390/s19051181, https://www.mdpi.com/1424-8220/19/5/1181, number: 5 Publisher: Multidisciplinary Digital Publishing Institute
3. Daniele, L., den Hartog, F., Roes, J.: Created in Close Interaction with the Industry: The Smart Appliances REFerence (SAREF) Ontology. In: Cuel, R., Young, R. (eds.) Formal Ontologies Meet Industry. pp. 100–112. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-21545-7_9
4. Donahue, C., McAuley, J., Puckette, M.: Adversarial Audio Synthesis (Feb 2019), http://arxiv.org/abs/1802.04208, arXiv:1802.04208 [cs]
5. Eigenschink, P., Reutterer, T., Vamosi, S., Vamosi, R., Sun, C., Kalcher, K.: Deep Generative Models for Synthetic Data: A Survey. IEEE Access **11**, 47304–47320 (2023). https://doi.org/10.1109/ACCESS.2023.3275134, https://ieeexplore.ieee.org/document/10122524/
6. Feng, Z., Mayer, W., He, K., Kwashie, S., Stumptner, M., Grossmann, G., Peng, R., Huang, W.: A Schema-Driven Synthetic Knowledge Graph Generation Approach With Extended Graph Differential Dependencies (GDDxs). IEEE Access **9**, 5609–5639 (2021). https://doi.org/10.1109/ACCESS.2020.3048186, https://ieeexplore.ieee.org/abstract/document/9311121, conference Name: IEEE Access

7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Communications of the ACM **63**(11), 139–144 (Oct 2020). https://doi.org/10.1145/3422622, https://dl.acm.org/doi/10.1145/3422622
8. Hubert, N., Monnin, P., d'Aquin, M., Monticolo, D., Brun, A.: PyGraft: Configurable Generation of Synthetic Schemas and Knowledge Graphs at Your Fingertips (Mar 2024), http://arxiv.org/abs/2309.03685, arXiv:2309.03685 [cs]
9. Lantzaki, C., Yannakis, T., Tzitzikas, Y., Analyti, A.: Generating Synthetic RDF Data with Connected Blank Nodes for Benchmarking. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) The Semantic Web: Trends and Challenges. pp. 192–207. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-07443-6_14
10. Melo, A., Paulheim, H.: Synthesizing Knowledge Graphs for Link and Type Prediction Benchmarking. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) The Semantic Web. pp. 136–151. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-58068-5_9
11. Siebes, R., de Boer, V., Reda, R.: Learning and Reasoning over Smart Home Knowledge Graphs (2022)
12. van der Weerdt, R., de Boer, V., Siebes, R., Groenewold, R., van Harmelen, F.: OfficeGraph: A Knowledge Graph of Office Building IoT Measurements. In: Meroño Peñuela, A., Dimou, A., Troncy, R., Hartig, O., Acosta, M., Alam, M., Paulheim, H., Lisena, P. (eds.) The Semantic Web. pp. 94–109. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-60635-9_6
13. Xu, L., Skoularidou, M., Cuesta-Infante, A., Veeramachaneni, K.: Modeling Tabular data using Conditional GAN. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html
14. Yoon, J., Jarrett, D., van der Schaar, M.: Time-series Generative Adversarial Networks. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://papers.nips.cc/paper{_}files/paper/2019/hash/c9efe5f26cd17ba6216bbe2a7d26d490-Abstract.html
15. Yu, L., Zhang, W., Wang, J., Yu, Y.: SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. Proceedings of the AAAI Conference on Artificial Intelligence **31**(1) (Feb 2017). https://doi.org/10.1609/aaai.v31i1.10804, https://ojs.aaai.org/index.php/AAAI/article/view/10804, number: 1

## 7 Appendix

Links to the code used for this thesis and the data from OfficeGraph:
https://github.com/JaManJesse/SyntheticKnowledgeGraphGeneration
https://github.com/RoderickvanderWeerdt/OfficeGraph