Synthetic Smart Meter Data Generation

Final Report

Ahmed Abubakr

Madison Perkins

April 28, 2024

Sponsored by Dr. Abdul Ofoli and Dr. Nur Sisworahardjo

ENEE 4850 Interdisciplinary Design Project II

Instructor: Dr. Abdul Ofoli

Summary

This report goes into the details surrounding the Synthetic Smart Meter Data Generation project. The problem being addressed by this project is the need for more efficient tools to load forecast as the grid becomes more technologically advanced as well as ensuring there is a way for smart meter data to be widely available without violating consumer privacy. To ensure an adequate understanding of this project and to decide which path to take with machine learning for our purposes, we did background research on other universities and professionals who have attempted similar projects in the past. Our project uses Generative Adversarial Networks which consist of two main neural networks: a generator and a discriminator. These two neural networks train each other to ensure the data is as authentic and close to realistic as possible. The ethics and safety considerations used in this project are investigated as well as the codes and standards used as guidelines for our design. The engineering specifications are also identified and discussed to ensure verification in our design, and specifications that were not met are analyzed to ensure other groups could achieve them. Finally, an instruction manual for using the GAN system and a troubleshooting guide is available for end users to ensure easy use of the software.

Table of Contents

Introduction

Problem Statement

The project addresses the growing need for readily accessible smart meter data in the utilities business. Smart meter data has the potential to greatly enhance power system efficiency and green efforts. The approach includes generating synthetic smart meter data from existing datasets using machine learning, notably Generative Adversarial Networks (GANs). The main challenges lie in generating realistic smart meter data, ensuring it mirrors real consumer behavior, and preserving privacy concerns. The project aims for successful GAN training, reliable data generation, and a repeatable framework for similar datasets.

Background and Previous Studies:

Previous research by academics has demonstrated the utility of GANs in generating power consumption statistics. These studies, especially in the residential and commercial sectors, have paved the way for data collection. However, data volume restrictions and privacy issues exist.

The study of smart meter power measurement is a broad and established field in the electrical engineering community. However, only recently have the abilities of machine learning, neural networks, and specifically generative adversarial network (GAN) models been applied to the client data management of the power electronics industry.

In 2019, one of the first applications, by Dr. Yuxuan Gu, of GAN models are published in the development power electronics internet of things (IOE), “GAN-based Model for Residential Load Generation Considering Typical Consumption Patterns.” As society modernizes its power network, there is a greater introduction in smart metering, or power measurement devices that measure the power consumption and are connected to the internet. Additionally, as the largest sector of the power grid are private residential spaces, it is increasingly difficult to monitor and utilize such data that is recorded from smart meters. Studying Smart Metering in a power grid becomes increasingly difficult because of the privacy of individuals and customers. This paper tries to address this concern by creating a GAN model to perform data generation. They use a convolutional GAN model, with the traditional generator and discriminator configuration to perform all data generation. Their model uses a three-layer generator network, one fully connected, and two transposed convolution layers and a six-layer discriminator model with two convolution layers and four fully connected layers. This paper provides a beginning basis for how to create this learning model. However, they address that there is still a problem of creating a larger sum of data.

In April of 2022, Dr. Qin at the School of Electrical Engineering in China, published "GAN-based Residential Load Data Generation Model Considering Users' Privacy." With a similar goal in mind as Dr. Gu, Dr. Qin speaks to the difficulties in obtaining residential power data due to the privacy of customers. In this paper the authors use a GAN model known as the Wasserstein Conditional Generative Adversarial Network. To prevent repetition, the introduction will not be repeated in the description of this paper as the authors use similar background description of what a GAN model is and how it may be adapted to this specific case. Dr. Qin introduces Wasserstein distance, which is the decreases the amount of GAN loss. They reference common problems in GANs such as "Jensen-Shannon (JS) divergence and Kullback-Leiber (KL) divergence" [2]. This new introduction alleviates the stress of common error margins created by these divergences.

Lastly, after the extensive research conducted by the past two publications on residential power consumption GAN models, the authors Yushan Liu, Zhouchi Liang and Xiao Li write on the application of GANs in the commercial sector. In 2023 Dr. Liu, publishes "Enhancing Short-Term Power Load Forecasting for Industrial and Commercial Building: A Hybrid Approach Using TimeGAN, CNN and LSTM." Dr. Liu created a methodology using a 'hybrid' data feeding in the training of his neural network. This combination of the two methodologies timeseries generation adversarial network (TimeGAN) and convolutional neural network (CNN) solve the issue of a lack of long-term data [3]. Additionally, these methodologies differ in their application of data loss. TimeGANs use "gradual supervision loss" which is more accurate than the traditional "unsupervised adversarial loss" used by "traditional GANs" [3]. This is important because it increases the accuracy and decreases the number of epochs required to create an efficient model.

In general, the electrical engineering community is aware of the necessity of the data generation of smart meters. This is particularly prevalent in China, as in the last three years the University of Beijing has published multiple papers on the introduction of GANs into the industry. While there are a few works under way on a similar subject, it is important to note that there is still an evident lack of research into the introduction of GANs in smart meter data generation.

In this project, like the first two works, the Synthetic Smart meter data generation team from the University of Tennessee at Chattanooga (UTC) uses provided data from the residential sector of the power grid. However, unlike the first two, smart meter team used residential data recorded from smart meters at communal transformers. In other words, the data provided to the team includes that of several households. The team used the works published by the University of Beijing as a background and drew inspiration from their designs, which are traditional convolutional for GANs.

Additionally, the third work referenced, authored by Dr. Liu, will is similar in its application of GANs to the data generation of the power grid; however, did not rely on data from the commercial or industry sector as Dr. Liu did. His adaptation of the hybrid GAN model is be noted, as the team used a small amount of data over a brief period.

Description of Design

A type a GAN system called DoppelGANger was used for the design of the model. DoppelGANger is a GAN that generates multivariate time series and associated information. This information refers to permanent qualities of data that are not part of the time series but nonetheless provide useful information about it. DoppelGANger was created with the goal of producing synthetic networked system data for data sharing, but it also serves as a basic framework for creating a variety of synthetic time series data. DoppelGANger's architecture differs from that of typical GANs in a few key areas to address difficulties in a time series scenario [8].

DoppelGANger and other time series GANs are based on recurrent neural networks (RNNs) that can detect temporal correlations. DoppelGANger employs a common form of RNN known as long short-term memory (LSTM). An RNN is a form of artificial neural network that processes sequential or time series data. These deep learning techniques are frequently employed for ordinal or temporal issues, such as language translation, natural language processing (NLP), audio recognition, and picture captioning. The RNN iterates over a sequence several times to generate the whole series. This approach may be used to multivariate time series. A typical issue with RNN generators is that lengthy sequences need a high number of RNN runs, making it difficult to capture longer-term correlations. DoppelGANger's designers employ a technique known as "batch generation" to address this issue. DoppelGANger creates several steps ahead, reducing the number of passes required [8].

The model employs an unusual kind of normalizing for time series with a large range of values across samples and series. For example, in a given sample, one series has a range of 0 to 100, whereas another has a range of -5 to 5. Rather than normalizing by the global lowest and maximum for all series, each series is normalized by its own minimum and maximum in that sample, and these values are saved as data for that series. DoppelGANger learns these variables and creates new minimum and maximum values for rescaling each produced series. Experimentation has shown that this can assist in preventing mode collapse in many circumstances where ranges differ significantly between series [8].

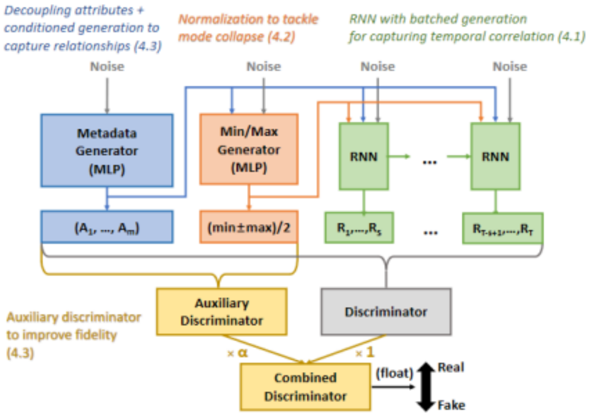

Decoupling attributes + conditioned generation to capture relationships (4.3)
Normalization to tackle mode collapse (4.2)
RNN with batched generation for capturing temporal correlation (4.1)
Noise
Noise
Noise
Noise
Metadata Generator (MLP)
Min/Max Generator (MLP)
RNN
...
RNN
(A_1, ..., A_m)
(min±max)/2
R_1,...,R_S
...
R_T-S+1,...,R_T
Auxiliary discriminator to improve fidelity (4.3)
Auxiliary Discriminator
Discriminator
× α
× 1
Combined Discriminator
(float)
Real
Fake

Figure 1 – DoppelGANger’s model diagram [9]

Ethics and Safety

The biggest ethical consideration discussed by the team in this project was the way in which data would be obtained. Originally, we did not view this as an ethical issue, but as we learned more about smart meter data and the rules surrounding it, we became aware of how obtaining data, in general, should be approached as an ethical dilemma because it can give insight into personal information of individuals. While smart meter data is not explicitly protected by any federal laws, there are a handful of states moving to protect customer data in the grid. This means that accessing smart meter data is not necessarily illegal, but the smart meters contain personally identifiable information (PII). There are many privacy concerns regarding the use of smart meter data, so to be ethically and morally responsible, we wanted to ensure any data we used was given to us consensually by knowledgeable sources. We accomplished this goal by receiving data from 100 meters over a span of 3 months from EPB.

The biggest safety concern corresponds to the biggest ethical concern in this project. With the rise of artificial intelligence and the usefulness of data, there are bad actors that feel no moral or ethical responsibility when it comes to respecting the work, likeness, or data of others. Information is essential to training machine learning, and sometimes taking information that belongs to another individual for the purpose of training an artificial intelligence program can cause harm to an individual themselves or their livelihood. In the case of this project, we ensured that the data and information we used would not harm anyone else or any business. These are safety concerns regarding machine learning and the growth of this technology in general, and if this project were in the hands of a bad actor, there is no way for us to ensure they would follow the same ethical principles we did.

Engineering Specifications

Specifications

1. Quick Runtime – GANs can take a range of time to create images, but usually is in minutes. For our project, we plan for the GAN to generate the data in less than 5 minutes for ease of use.

2. The data will be organized in excel by smart meter number and time of day for simplified training. The generated data will be organized in a corresponding excel sheet with the same organizational principles applied. The generated data should have the date and time (hour and minute) that corresponds to the power measurement.

3. Low computational power – The final program should function using only a laptop or a similar device. The program's function will be tested on a regular laptop after all the testing of the program has been done.

4. Pseudocode – The code should have comments that describe what the code is doing. We will test this by someone outside of the project to verify if the descriptions are clear enough.

5. Repository and documentation - In the final processes of the project, the model will be uploaded to a GitHub repository. Additionally, the team will provide an instruction manual that will provide a detailed walk-through process of how to insert data into the model, train the model, and then produce more data.

6. Time based power consumption - The model may require time-based input correlating with input data to create time-based power consumption and averages. This will be used in the creation of synthetic data when input. This will be tested by comparing the synthetic data with actual data from another source.

7. Automatic trend interface - Upon completion of data generation, the model will provide the user with data trend metrics automatically. These will be provided in the form of labeled trend charts and console printed averages.

8. Safe training data acquisition - Training data used within the model will be taken from safe and reliable sources. All data provided for use in the training of the model will either be taken directly from the client or from consenting sources.

9. Diverse data sets - The GAN model will have the capability to train itself on diverse datasets, and datasets that are significantly different, due to reasons such as season, time, and culture, from the customer provided will still function as planned.

10. 50% error in generator and discriminator - The final GAN synthetic data will be produced at a point where both the discriminator and generator have reached points to display 50% error each. This will show that the model has reached the maximum level from all possible training with that specific input dataset.

Verification

1. This specification was met. The model takes a few minutes to run even with a large set of data.

2. Data has been organized before testing and snippets of the data were taken at each of the testing stages.
3. This specification was met. The code was built using the Google Collab platform which has its own cloud-based servers that give the required computational power and thus can be used on virtually any type of device.
4. The pseudocode was verified by Madison's friend William to be comprehensive.
5. This specification was met. The instruction manual as well as means for troubleshooting the GAN system for this project is included in this report in following sections.
6. This specification was met. The data provided by the model is shown on a graph that compares the 'real' data with the 'synthetic'.
7. Not met.
8. This specification was met. The data used in this project was given to us knowingly by EPB to aid in the training process.
9. Not fully met. The code would need to be altered with each different type of data entered.
10. Not fully met. The code needs some more optimization as the results are not within spec of a perfect GAN model.

Codes and Standards

1. PEP 8 - Guide for Python Formatting

   PEP 8 identifies the conventions used in python coding as well as the libraries available. This standard includes instruction on things such as the code layout, how white space should be used, the use of commas, comments, naming conventions, and other programming recommendations. We used PEP 8 to ensure our code follows guidelines for functionality as well as comprehension for others that may be interested in viewing the code. PEP 8 ensured that we used the python programming language correctly and efficiently throughout our project and provided a solid reference for questions of syntax, layout, etc.

2. IEEE 3652.1-2020

   This standard is specifically regarding federated systems, which are systems collectively used and created by different organizations or individuals. This standard sets a framework for privacy and structure regarding machine learning systems that are dependent on data and/or devices from various organizations. While this is not entirely applicable to our project, we used these standards to create a framework for our GAN to be used as a foundation for others. We also made sure that any data we obtained from individuals and companies was kept private so that their data is not released without consent.

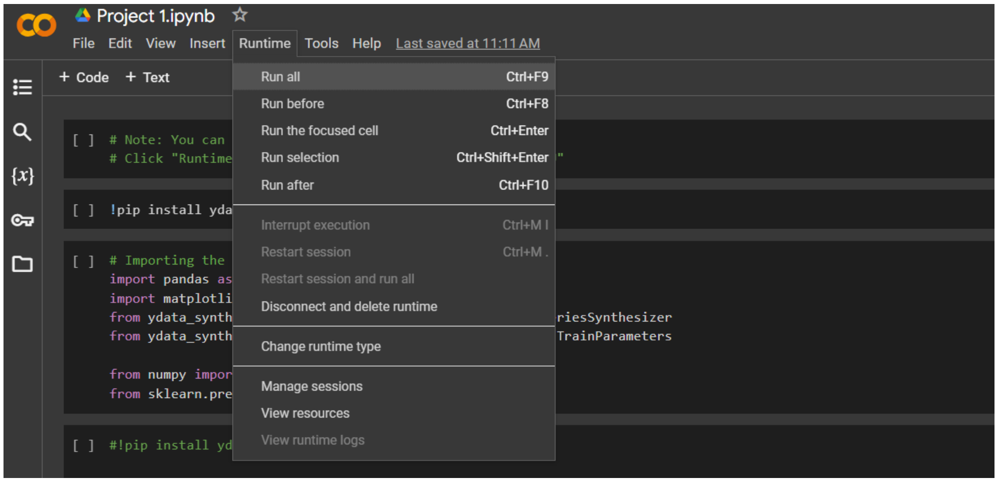
Machine learning and AMI (advanced metering infrastructure) are both still new technologies, so there are not standards that we felt we could apply to this project yet. Existing standards ensure interoperability of machine learning technologies as well as interoperability and communication for AMI. The standards regarding smart meters are more technical than data related. The lack of standards and codes regarding data as well as machine learning within the power grid is concerning since both of those topics are so significant with how the grid has developed. Standards take 2-5 years to develop, though, so it is a slow process to get such things implemented. Madison has a personal goal to contribute to the development of standards regarding the use of machine learning in the smart grid after graduation.

Manual for End Users

Link to the code:
https://colab.research.google.com/drive/1PQ7YmH8WQY1aeDNtkwtNURv0XJdaGiGh?usp=sharing (also in [10])

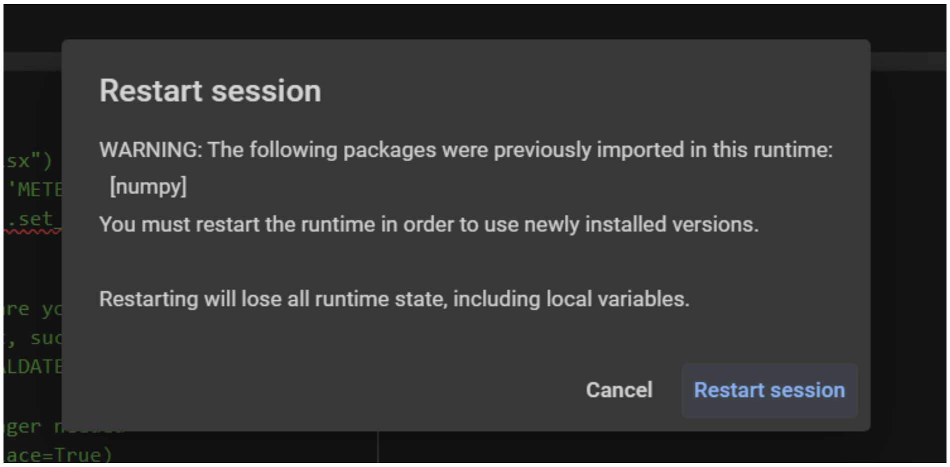
As the code was made in Google Collab, the guide assumes that the end user is going to use the same platform.

1. The code is initiated as follows:



```
[ ]  # Note: You can
     # Click "Runtime                "

[ ]  !pip install yda

[ ]  # Importing the
     import pandas as
     import matplotli
     from ydata_synth                riesSynthesizer
     from ydata_synth                TrainParameters

     from numpy impor
     from sklearn.pre

[ ]  #!pip install yd
```

2. The program will then initiate a pip install within the program itself (no worries about having it install something on your computer as this will only be installed within the cloud's runtime).
3. It will then ask you to restart the session so just click on the button:

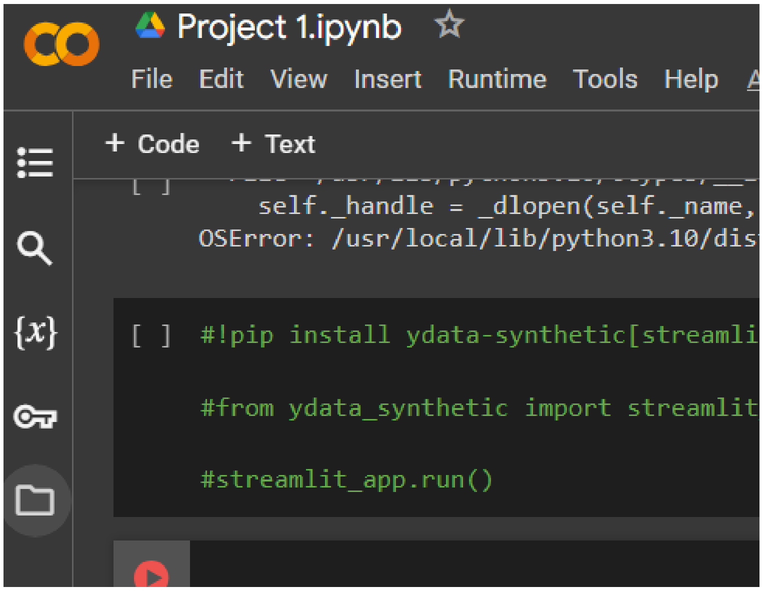


Restart session

WARNING: The following packages were previously imported in this runtime:
[numpy]
You must restart the runtime in order to use newly installed versions.

Restarting will lose all runtime state, including local variables.

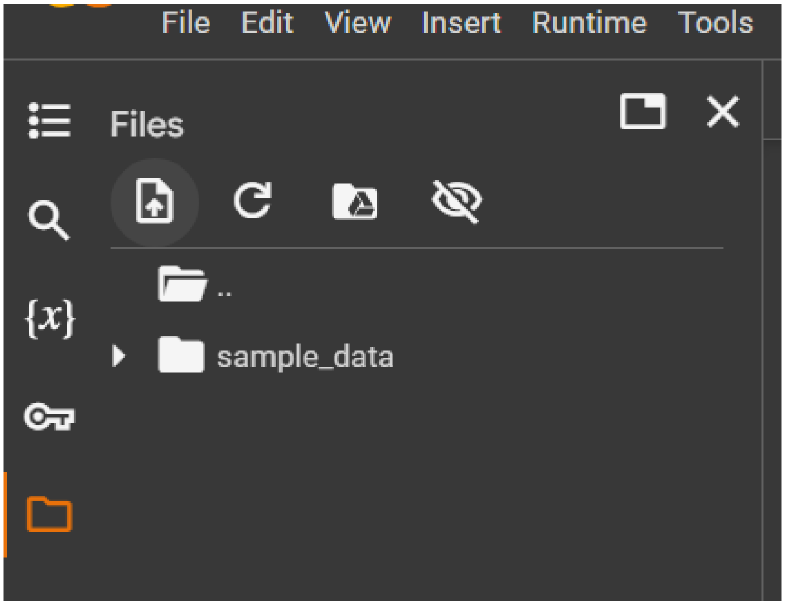
4. Then you need to make sure your dataset is ready as the rest is mostly automatic. First going here on the folder icon bottom left:



```
    self._handle = _dlopen(self._name,
OSError: /usr/local/lib/python3.10/dis
```

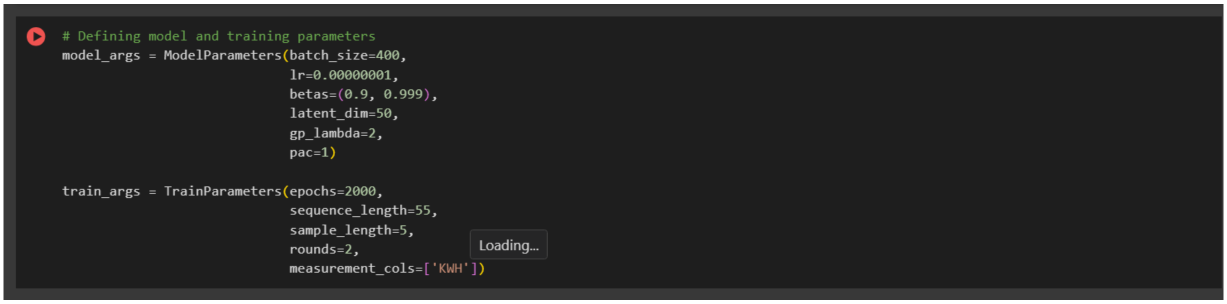
```
[ ] #!pip install ydata-synthetic[streamli

    #from ydata_synthetic import streamlit

    #streamlit_app.run()
```

Then selecting the highlighted icon:



And then choosing the appropriate dataset excel file from your computer. For this guide it should be the provided “test2.csv”.

5. Next scroll to this line code and make sure everything is set to the specifications of the data being used (in this case everything should already be set in place):



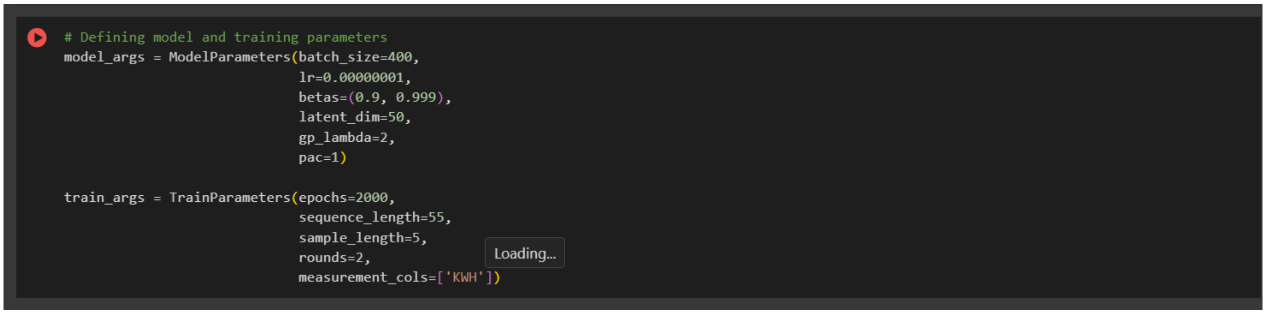
```
# Defining model and training parameters
model_args = ModelParameters(batch_size=400,
                             lr=0.00000001,
                             betas=(0.9, 0.999),
                             latent_dim=50,
                             gp_lambda=2,
                             pac=1)

train_args = TrainParameters(epochs=2000,
                             sequence_length=55,
                             sample_length=5,
                             rounds=2,
                             measurement_cols=['KWH'])
```

This is where you would change several settings according to the data being used if different data is to be set. The settings highlight the model parameters of the GAN and training parameters. If this stage is taking a while the number of epochs can be tuned down to 500 or so, but it should work fine as is with the data provided.

6. Lastly, the code is run again as in step 1 and the results are observed.

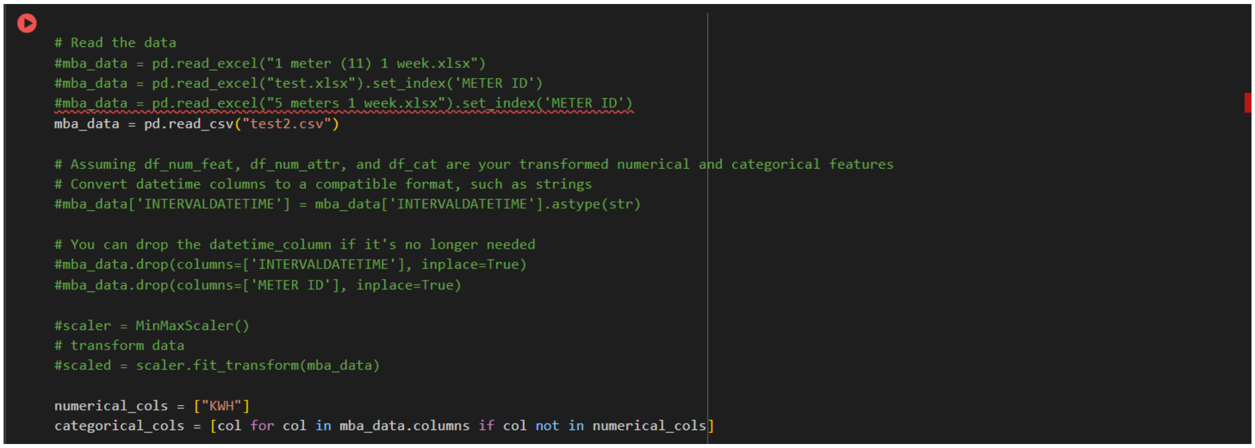
Troubleshooting Guide

As the code is not optimized yet, it would be difficult to try and run it with a different set data. The parameters shown below would need to changed according to the specific dataset used:



```
# Defining model and training parameters
model_args = ModelParameters(batch_size=400,
                             lr=0.00000001,
                             betas=(0.9, 0.999),
                             latent_dim=50,
                             gp_lambda=2,
                             pac=1)

train_args = TrainParameters(epochs=2000,
                             sequence_length=55,
                             sample_length=5,
                             rounds=2,
                             measurement_cols=['KWH'])
```

If you get an error denoting that the excel file is not detectable then you need to make sure that the name of the file in the code is the same:



```
# Read the data
#mba_data = pd.read_excel("1 meter (11) 1 week.xlsx")
#mba_data = pd.read_excel("test.xlsx").set_index('METER ID')
#mba_data = pd.read_excel("5 meters 1 week.xlsx").set_index('METER ID')
mba_data = pd.read_csv("test2.csv")

# Assuming df_num_feat, df_num_attr, and df_cat are your transformed numerical and categorical features
# Convert datetime columns to a compatible format, such as strings
#mba_data['INTERVALDATETIME'] = mba_data['INTERVALDATETIME'].astype(str)

# You can drop the datetime_column if it's no longer needed
#mba_data.drop(columns=['INTERVALDATETIME'], inplace=True)
#mba_data.drop(columns=['METER ID'], inplace=True)

#scaler = MinMaxScaler()
# transform data
#scaled = scaler.fit_transform(mba_data)

numerical_cols = ["KWH"]
categorical_cols = [col for col in mba_data.columns if col not in numerical_cols]
```

If for some reason a “.xlsx” is used instead of the “.csv” then ‘pd.read_excel(“name.xlsx”)’ needs to be used instead.

If the runtime is stopped or restarted within Google Collab (the browser was restarted) then the pip install file needs to happen again.

When importing the excel file into the cloud, if you select the required excel file and it does not import properly, import any other file first and then try to import the required one again.

Conclusions and Recommendations

In this project, machine learning was used to generate synthetic smart meter data for the purpose of load forecasting. GAN's were found to be somewhat effective for this task. The final results followed the trend of legitimate smart meter data but had many zero points in it. This is a result of us not having as much training data as we would need to create a complete collection of smart meter data points over a 3-month period. Without the time constraints of the school year, we could have obtained more data to fix these shortcomings. Some next steps for this project would be to obtain that data to do more training, then find a way to get the GAN to generate data for a specific household size based on a user input.

References

[1] P. Qin, X. Wang, Z. Qiao, X. Li, Q. Hu and W. Shu, "GAN-based Residential Load Data Generation Model Considering Users' Privacy," 2022 7th Asia Conference on Power and Electrical Engineering (ACPEE), Hangzhou, China, 2022, pp. 838-843, doi: 10.1109/ACPEE53904.2022.9783676.

[2] Y. Gu, Q. Chen, K. Liu, L. Xie and C. Kang, "GAN-based Model for Residential Load Generation Considering Typical Consumption Patterns," 2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), Washington, DC, USA, 2019, pp. 1-5, doi: 10.1109/ISGT.2019.8791575.

[3] Y. Liu, Z. Liang and X. Li, "Enhancing Short-Term Power Load Forecasting for Industrial and Commercial Buildings: A Hybrid Approach Using TimeGAN, CNN, and LSTM," in IEEE Open Journal of the Industrial Electronics Society, vol. 4, pp. 451-462, 2023, doi: 10.1109/OJIES.2023.3319040.

[4] Bok, Vladimir, and Jakub Langr. GANs in Action. Simon and Schuster, 9 Sept. 2019.

[5] X. Zheng, B. Wang and L. Xie, "Synthetic Dynamic PMU Data Generation: A Generative Adversarial Network Approach," 2019 International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA), College Station, TX, USA, 2019, pp. 1-6, doi: 10.1109/SGSMA.2019.8784681.

[6] https://deepai.org/publication/deep-learning-for-hyperspectral-image-classification-an-overview

[7] https://peps.python.org/pep-0008/

[8] https://ar5iv.labs.arxiv.org/html/2302.10490

[9] https://dl.acm.org/doi/pdf/10.1145/3419394.3423643

[10] https://colab.research.google.com/drive/1PQ7YmH8WQY1aeDNtkwtNURv0XJdaGiGh?usp=sharing