Generative Adversarial Networks for Time-Based Data

In collaboration with Dow Chemical

DowGAN team: Arty T., Daniel K., Emily N., & Emily M.-S.

What can a GAN model do for its users?

USE CASE 1

Using GAN model to generate augmented data

User input: Time series data (ex: experimentally derived data-points)

User receives: Generated time series data set (based on patterns from experimentally derived data-point)

Purpose: Generate data for processes that are very costly to collect experimentally. Generate artificial data for cross-company collaboration.

USE CASE 2

Using GAN model with new multi-featured data

User input: Time series dataset with multiple features (ex: data from various chemical processes, pressures, temperatures)

User receives: Generated multi-feature time series data set via a model learning each feature trend and their correlations

Purpose: Expanding the model to encompass data with multiple features

USE CASE 3

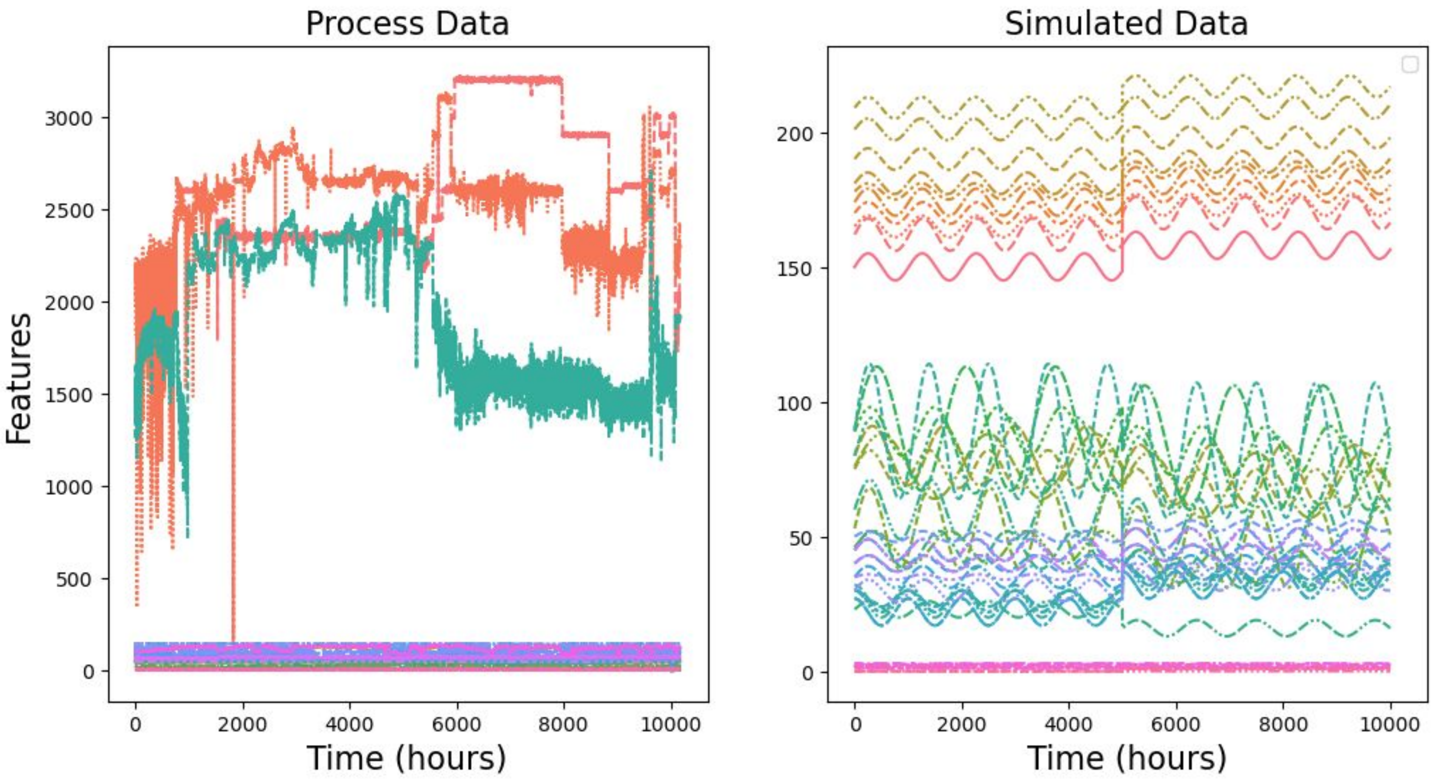
Using GAN model to extrapolate data beyond given time range

User input: Time series data until t = x seconds

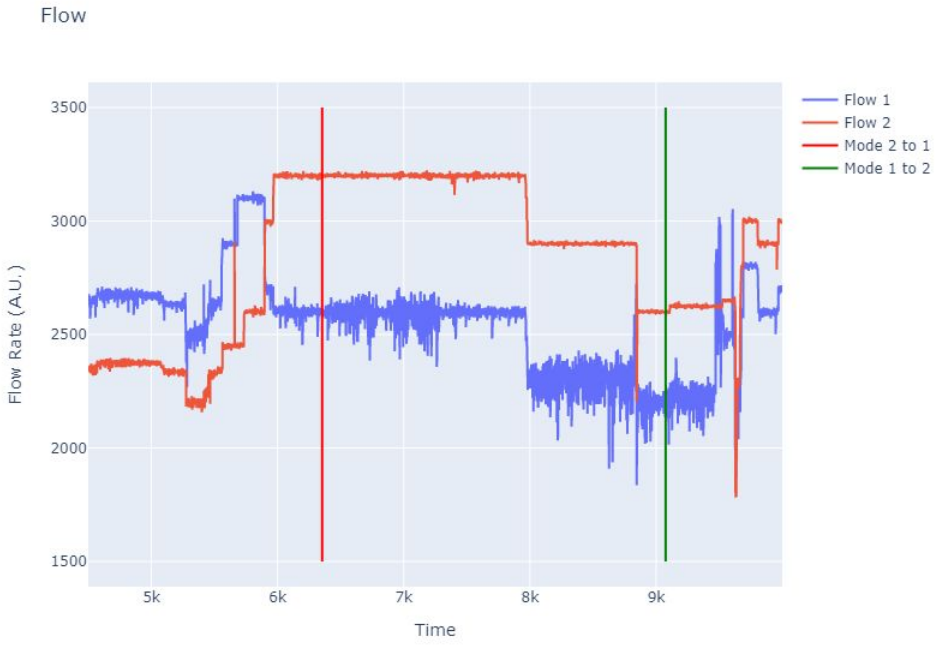
User receives: Time series data past the point of experimentally derived data

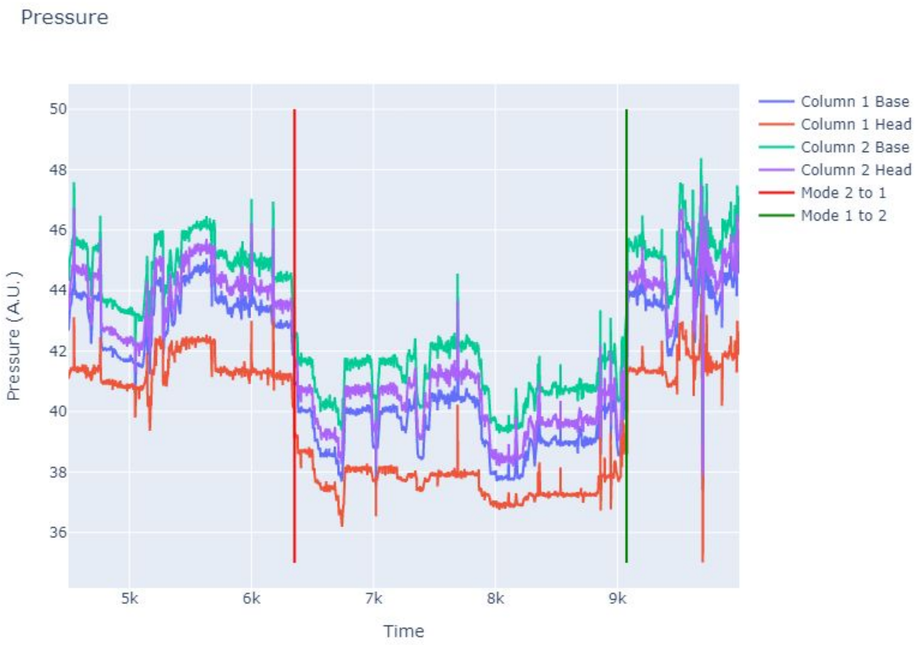
Purpose: Generate time series data for costly or time intensive experiments to leverage in decision making.

Data we are currently working with: testing & training



Data has two operating conditions

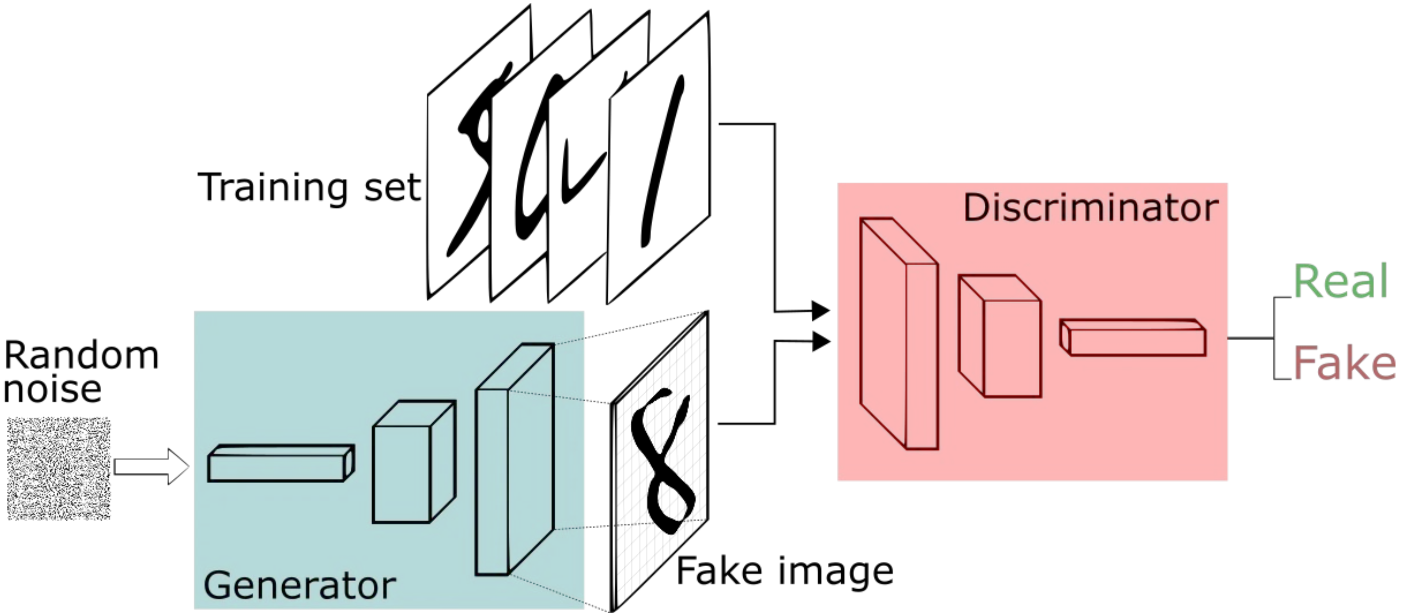




Manual operating conditions for some features

Condition is based on column pressure

How do GANs work?



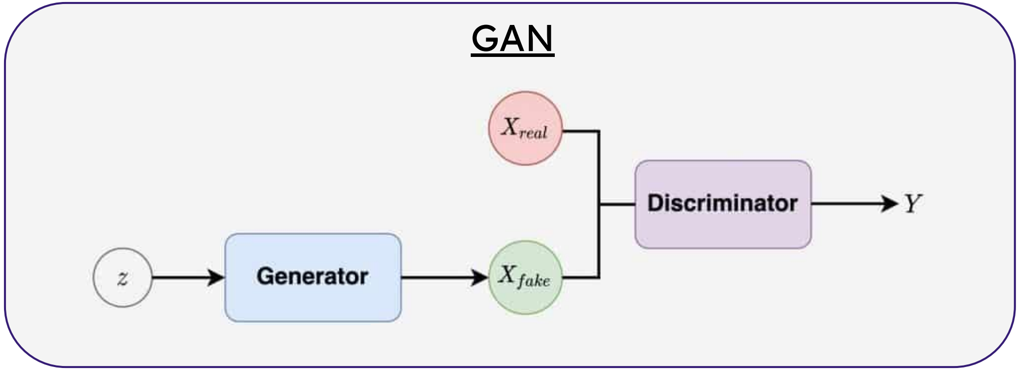
Conditional GANs vs. GANs

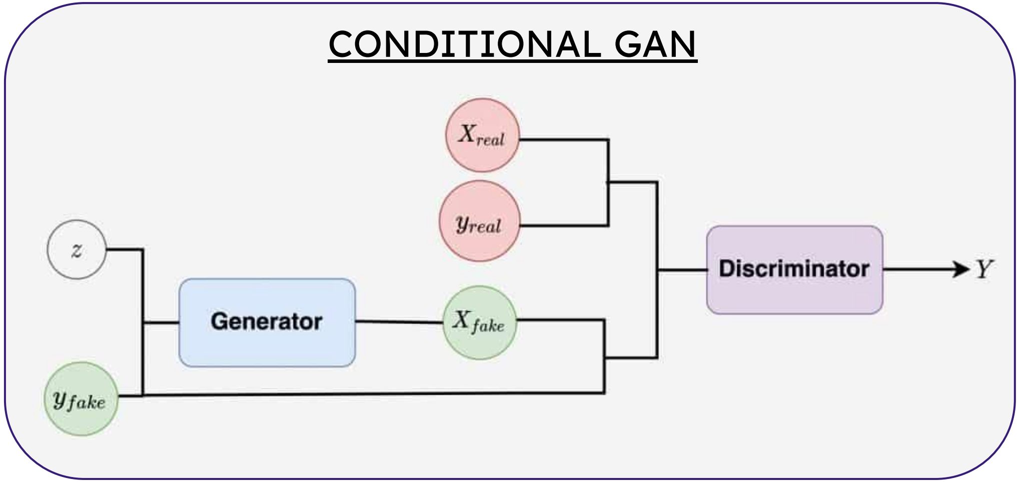
Typical GANs — “unsupervised”

- No control on modes of the data being generated

Conditional GAN (CGAN) — “semi-supervised”

- Generator learns to generate a fake sample **with a specific condition or characteristics** (such as a label associated with an image or more detailed tag)





Initial CGAN model architecture: basic convolutions

Number of datapoints: 10,000

Number of samples: 59

Number of features per sample: 45

Number of datapoints per sample: 168

Process Condition: 1 or 2

DATA PRE-PROCESSING

- Target = samples of time-series data, condition/label = process condition
- Min-max scaling of the target
- Batching the targets and conditions for input

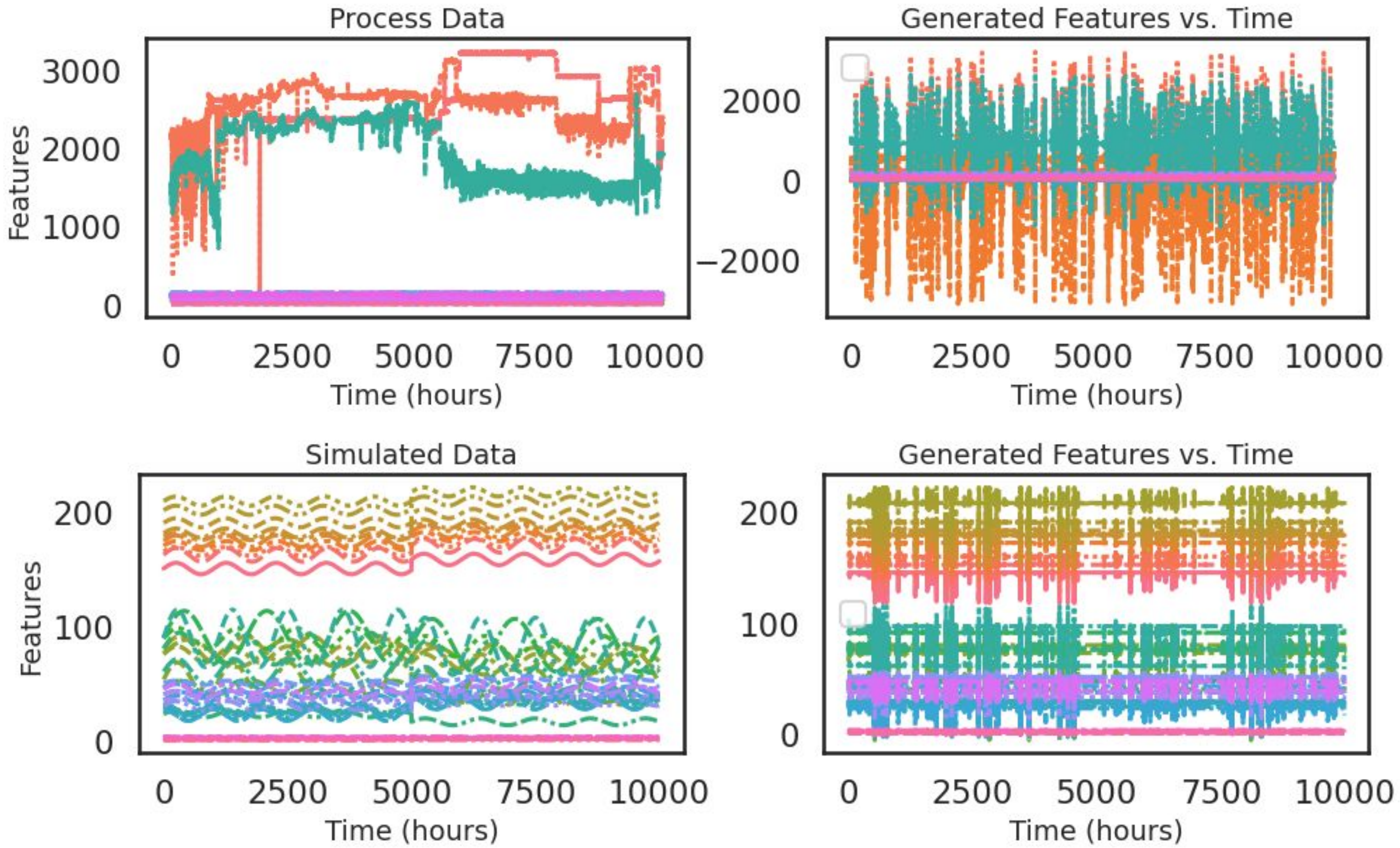
DISCRIMINATOR

- Concatenate time-series matrix + conditions for each time-series point
- Passes through a series of convolutional/max pool layers
- Activation function: Sigmoid
- Output size: [10, 1]

GENERATOR

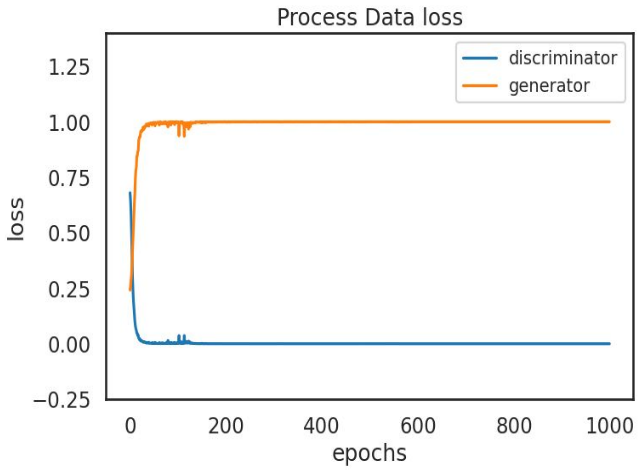
- Concatenate time-series matrix + conditions for each time-series point
- Pass through convolutional/batch norm layers
- Activation function: Tanh
- Output size: [10, 168, 45]

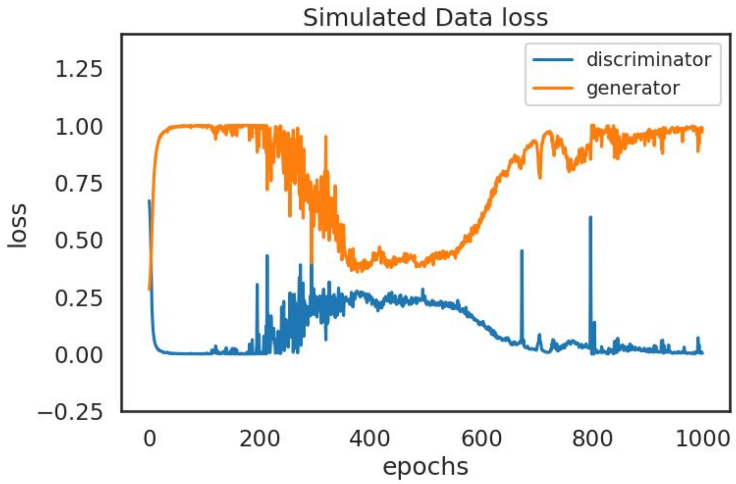
Results



Process Data
Generated Features vs. Time
Simulated Data
Generated Features vs. Time
Features
Time (hours)
3000
2000
1000
0
−2000
200
100
2500
5000
7500
10000

Models are not learning





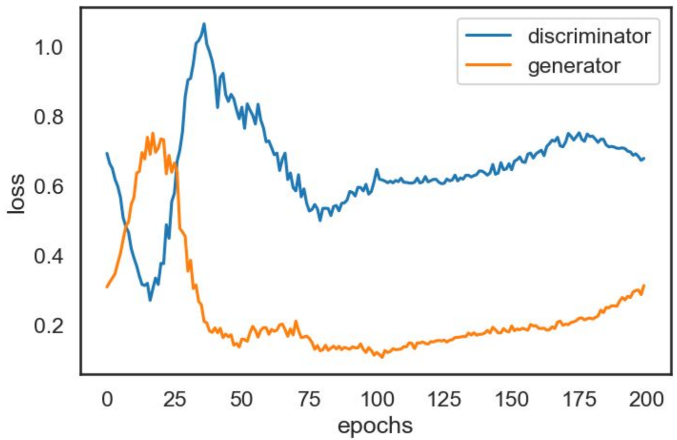
Data preprocessing:
Dropping NaN values
MinMaxScaler (0,1)

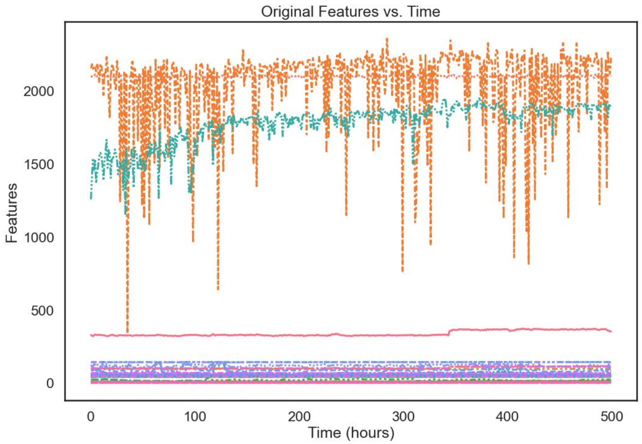
D. parameters / G. parameters:
Epochs: 1000
Activation function: Sigmoid()
Learning rate: 0.0005
Optimizer: Adam
G. Loss function: MSE Loss
D. Loss function: BCE Loss

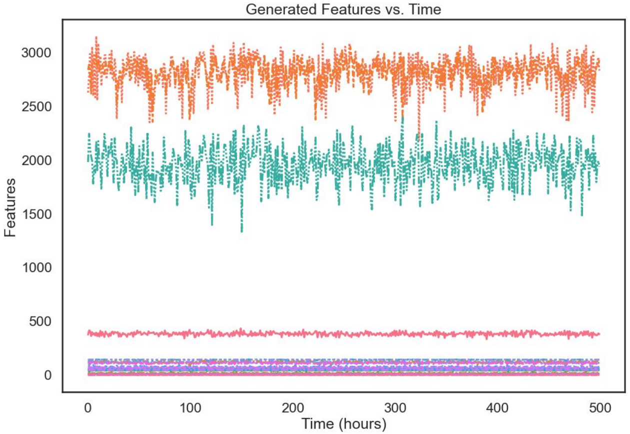
Vanishing gradient problem - discriminator is not providing enough information for generator to make progress, weights are not updated

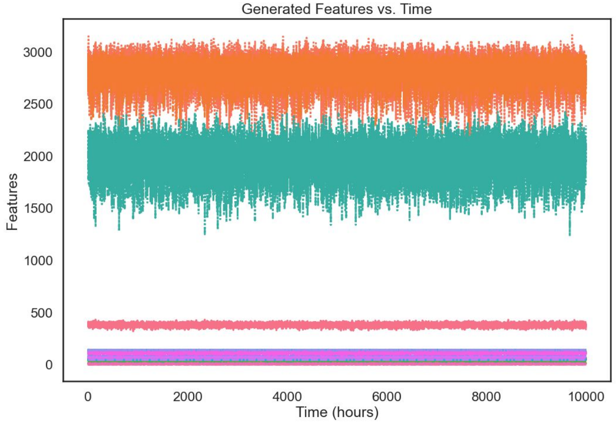
Example Optimization

D. parameters / G. parameters:
Epochs: 200
Activation function: Sigmoid()
Learning rate: 0.0002
Optimizer: Adam
G. Loss function: MSE Loss
D. Loss function: BCE Loss

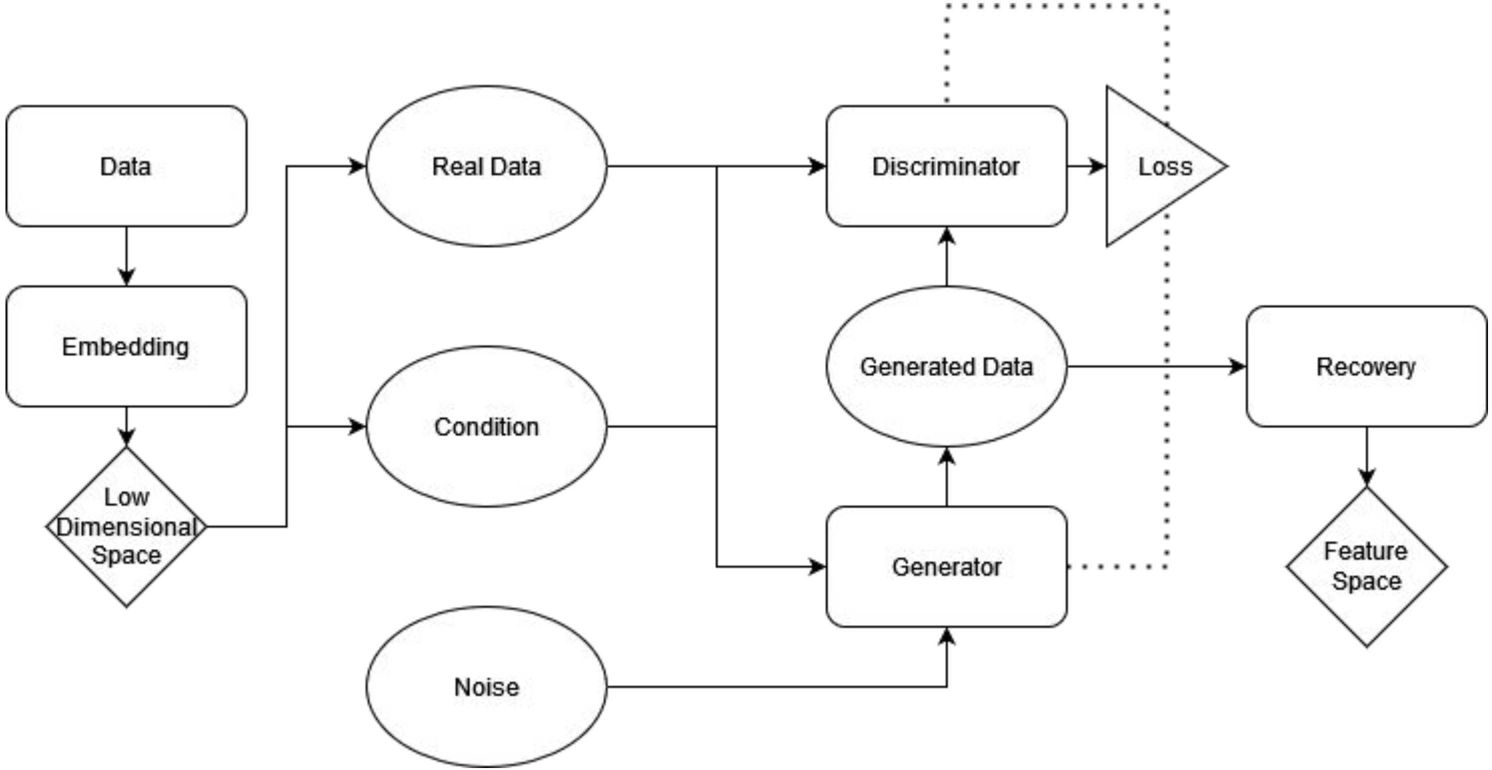








Encoder CGAN Model



Embedding and Recovery Model Architecture

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
               GRU-1  [[-1, 10, 50], [-1, 2, 50]]               0
            Linear-2               [-1, 10, 15]             765
           Sigmoid-3               [-1, 10, 15]               0
================================================================
Total params: 765
Trainable params: 765
Non-trainable params: 0
----------------------------------------------------------------
```

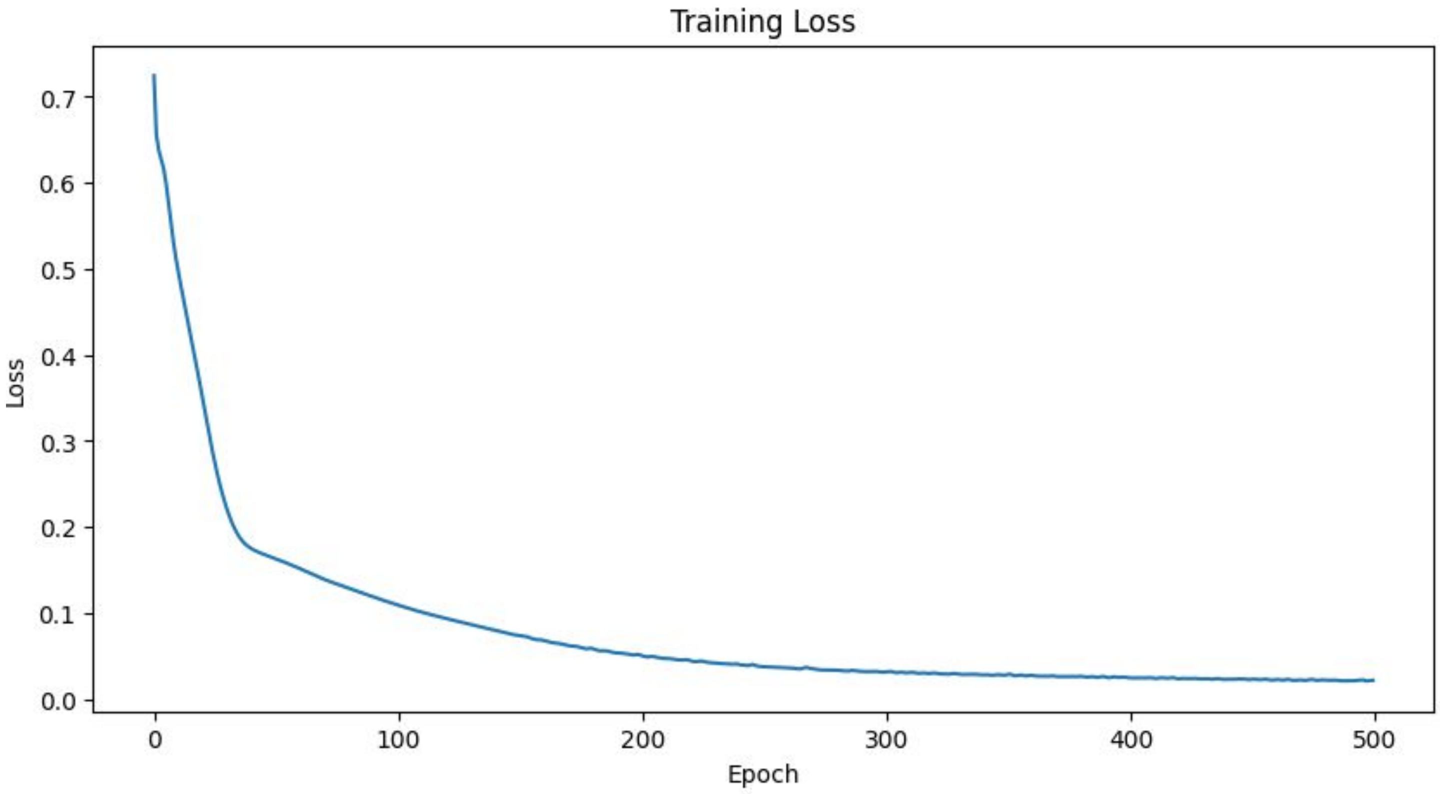
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
               GRU-1  [[-1, 10, 50], [-1, 2, 50]]               0
            Linear-2               [-1, 10, 45]           2,295
================================================================
Total params: 2,295
Trainable params: 2,295
Non-trainable params: 0
----------------------------------------------------------------
```

Embedder

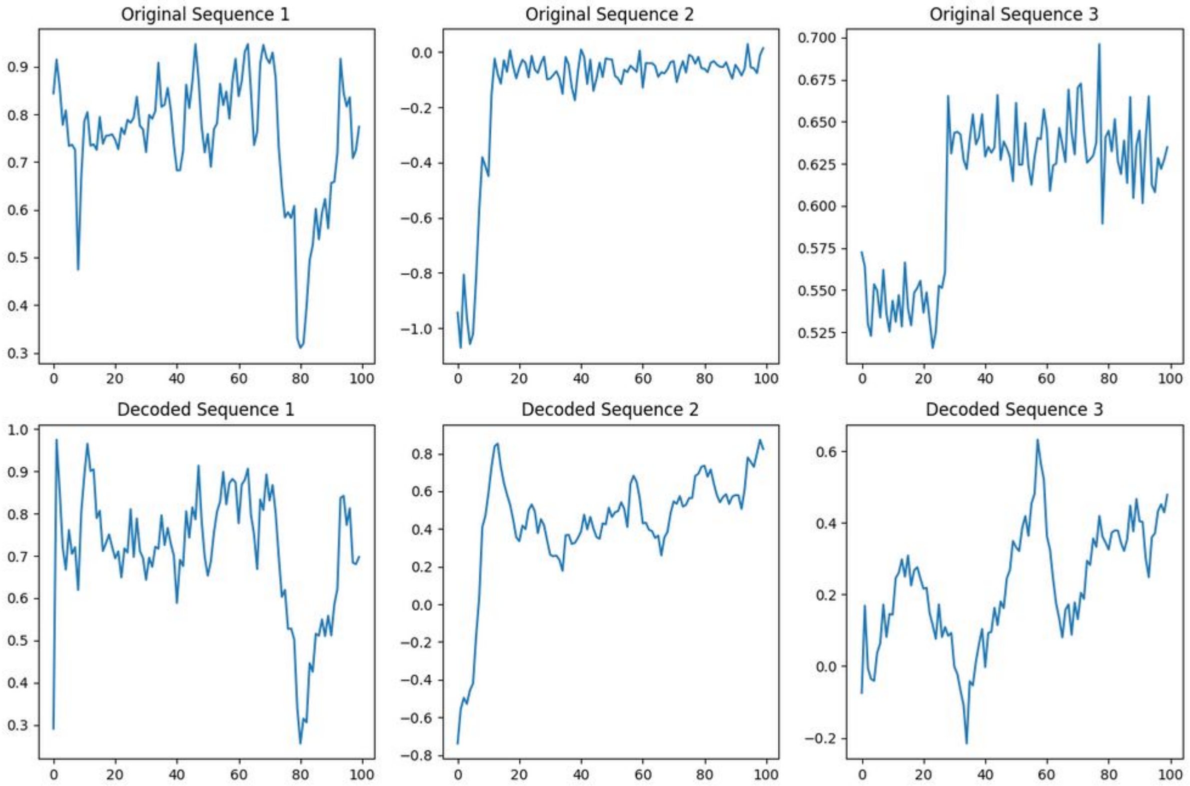
Recovery

Summary by torchsummary
Visual by Netron.app

Embedder Recovery Loss



Recovery of Process Data



CGAN Architecture

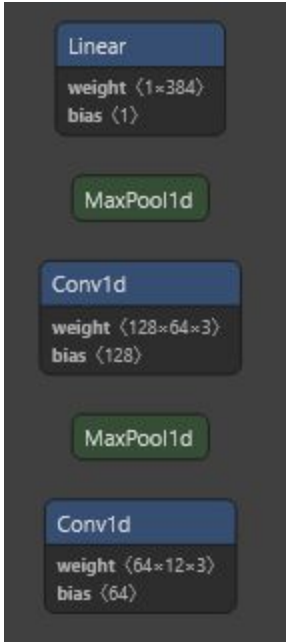
Generator:

- Generates synthetic data by combining noise and condition using a GRU unit and linear layers.

Discriminator:

- Rates the "realness" of data based on condition using convolutional and linear layers.

Generator

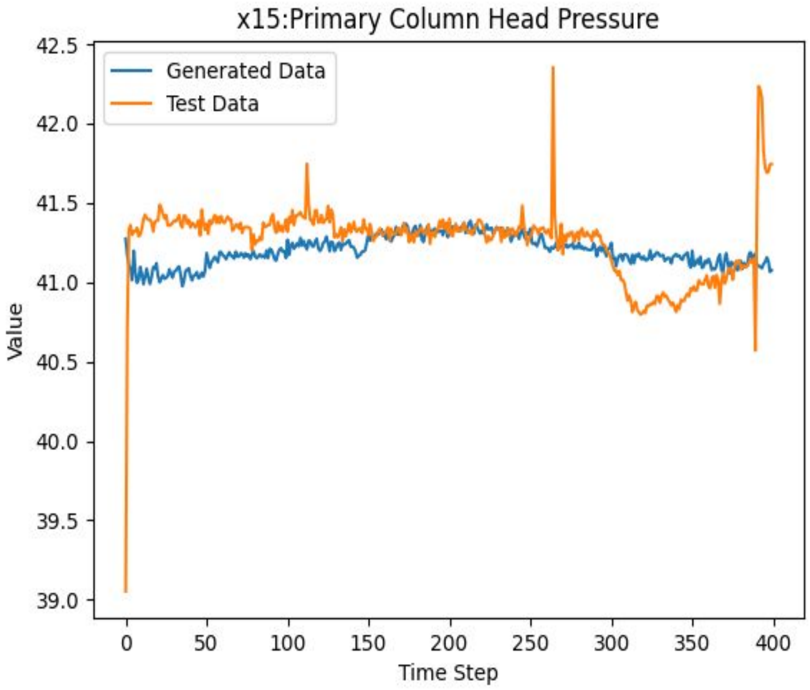


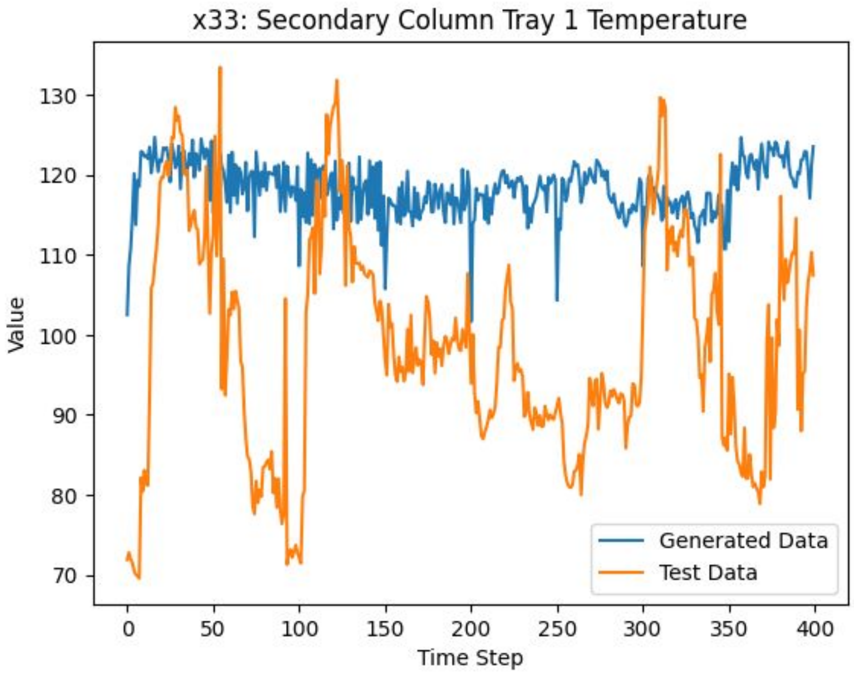
Discriminator

CGAN Loss


Training Loss
Generator Loss
Discriminator Loss
Recovery Loss
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Loss
0
25
50
75
100
125
150
175
200
Epoch

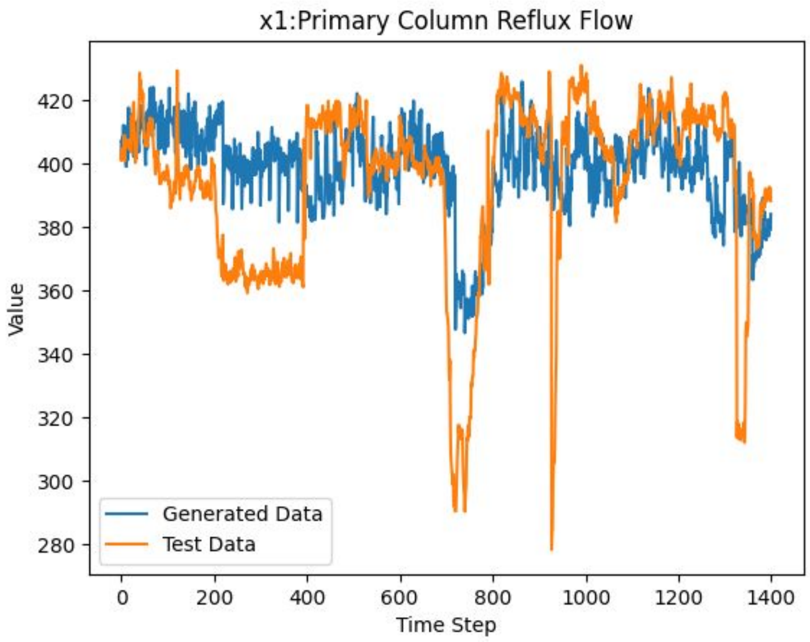
Data Generation Visualization

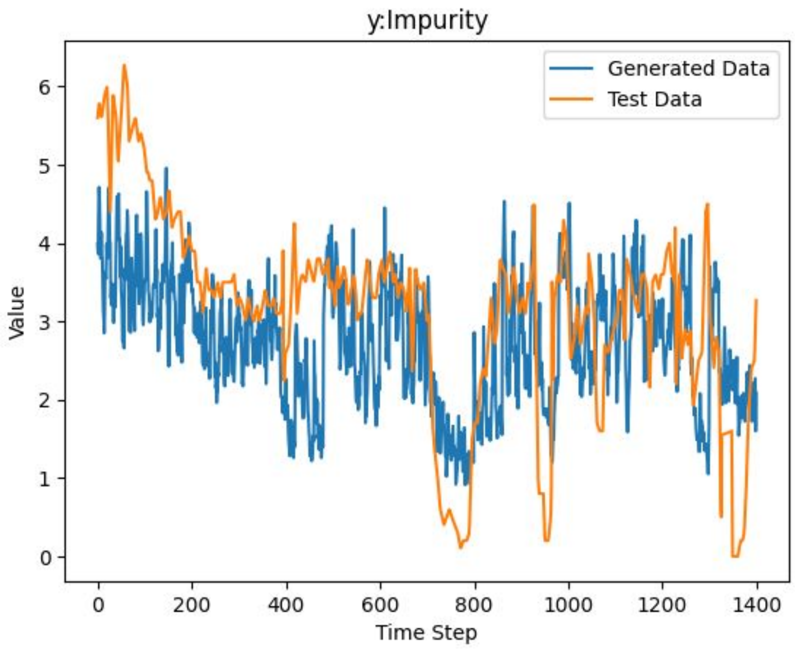




Sequence Length 50

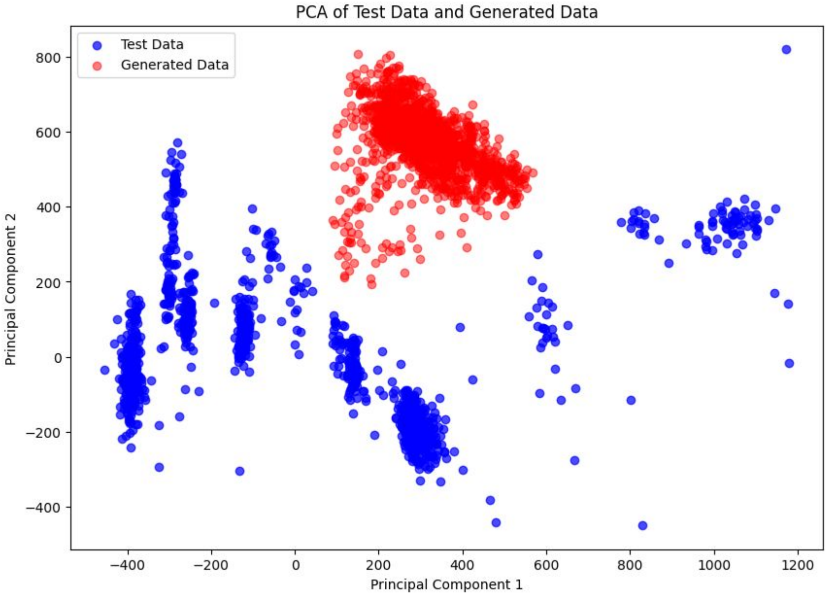
Data Augmentation Visualization



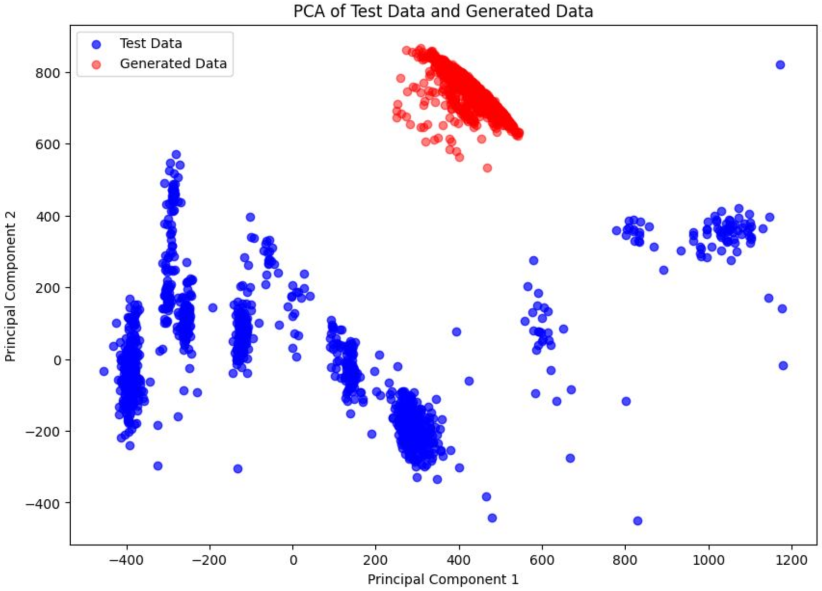


Sequence Length 20

Primary Component Analysis

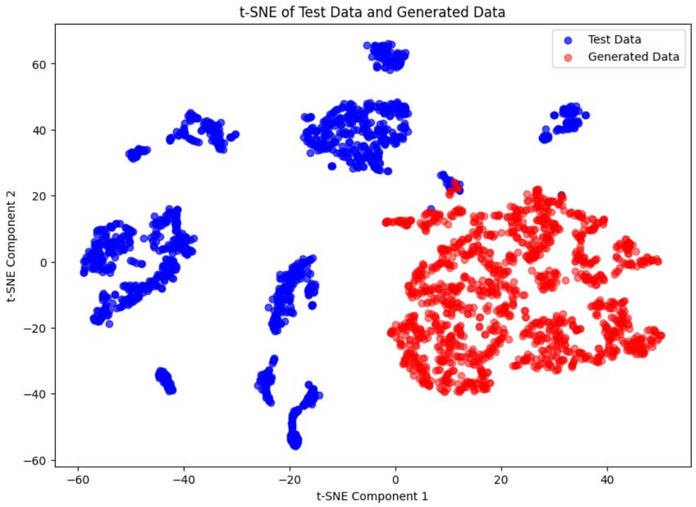


Generated with recurrent condition feed

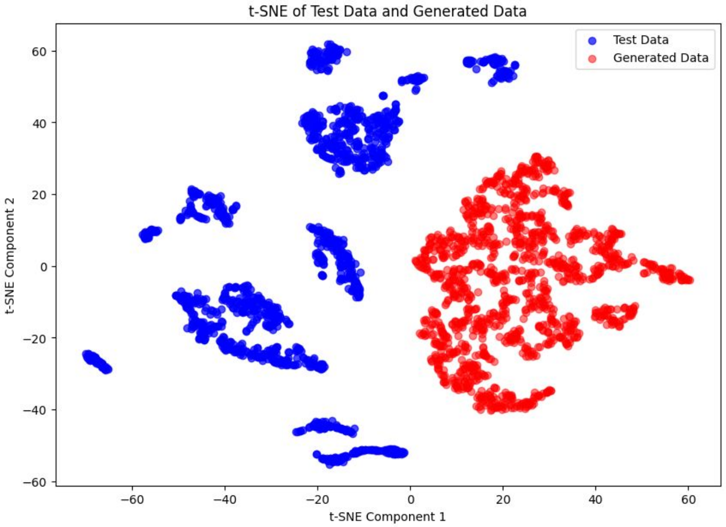


Generated with single condition feed

t-Distributed Stochastic Neighbor Embedding



Generated with recurrent condition feed



Generated with single condition feed

Future Efforts

- Incorporate reconstruction loss into Embedder/ Recovery Model training
- Increase Model Complexity with more layers
- Data preprocessing for steady state condition