Topology Identification in Power Distribution Systems with Autoencoder Neural Network Model

Group 4:
Yifu Li, Maeshal Hijazi, and Jinshun Su

Agenda

Introduction

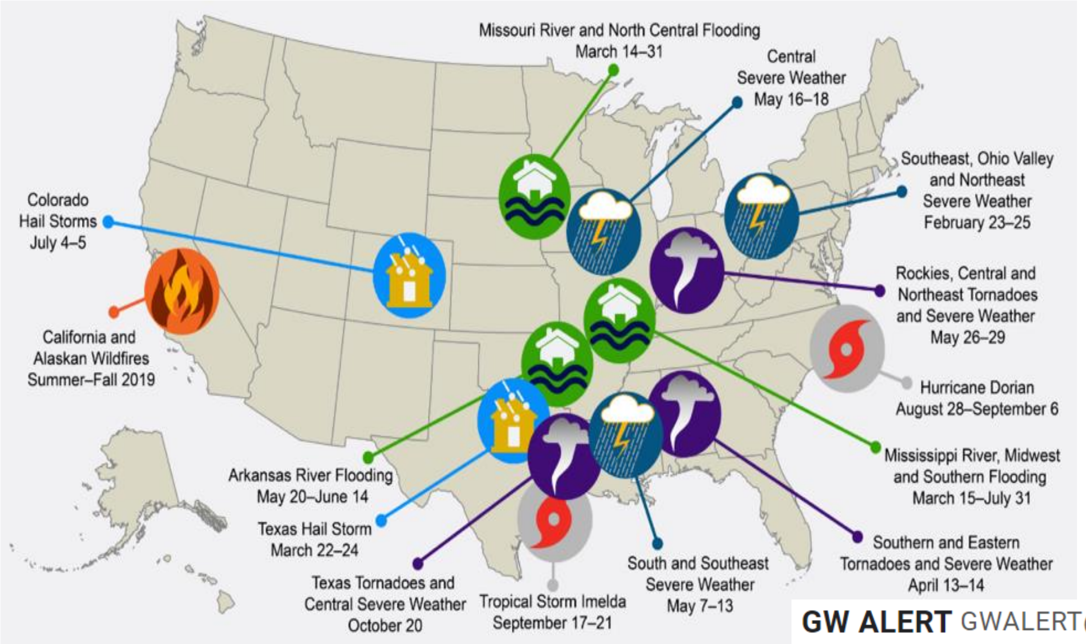
Maria Hurricane

- September 20, 2017
- Puerto Rico
- Entire island without electricity
- $91.61 billion economic losses

California Wildfires

- December 21, 2018
- California, USA
- Affected 4 million people
- $3.5 Billion economic losses

Introduction



Extreme US Natural Disasters in 2019

GW ALERT GWALERT@gwu.... May 23, 2019, 4:38 PM

to GW

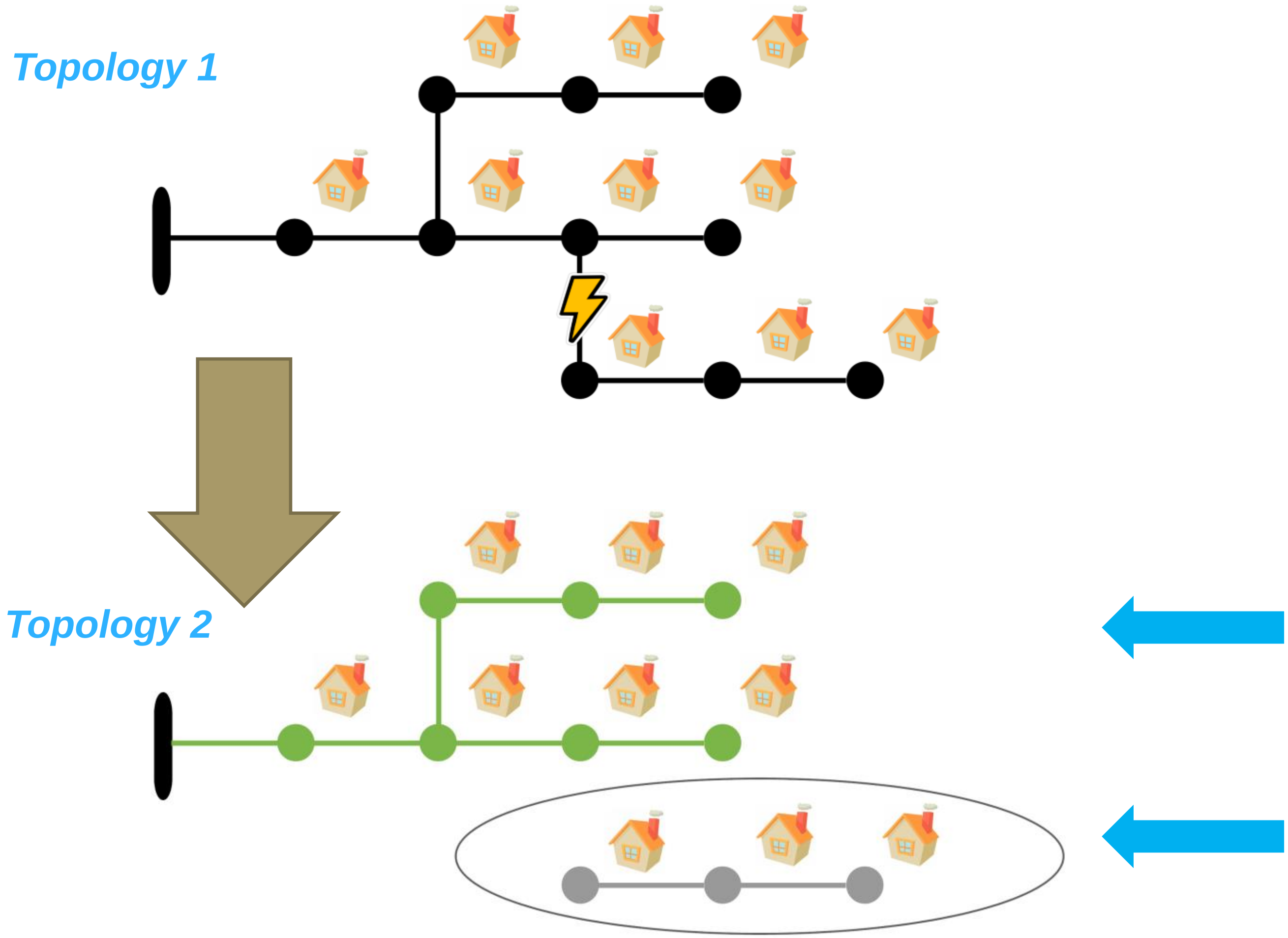
May 23, 2019, Classes have been cancelled at the Arlington Va location tonight as a result of severe weather a power outage . Dominion Power is aware of the outage and are working to restore power to the area as soon as possible. There is no timeframe for restoration at this time.

We will continue to provide updates as they become available.

For preparedness tips and safety information, refer to GW Emergency Response Handbook and download GW PAL.

Introduction

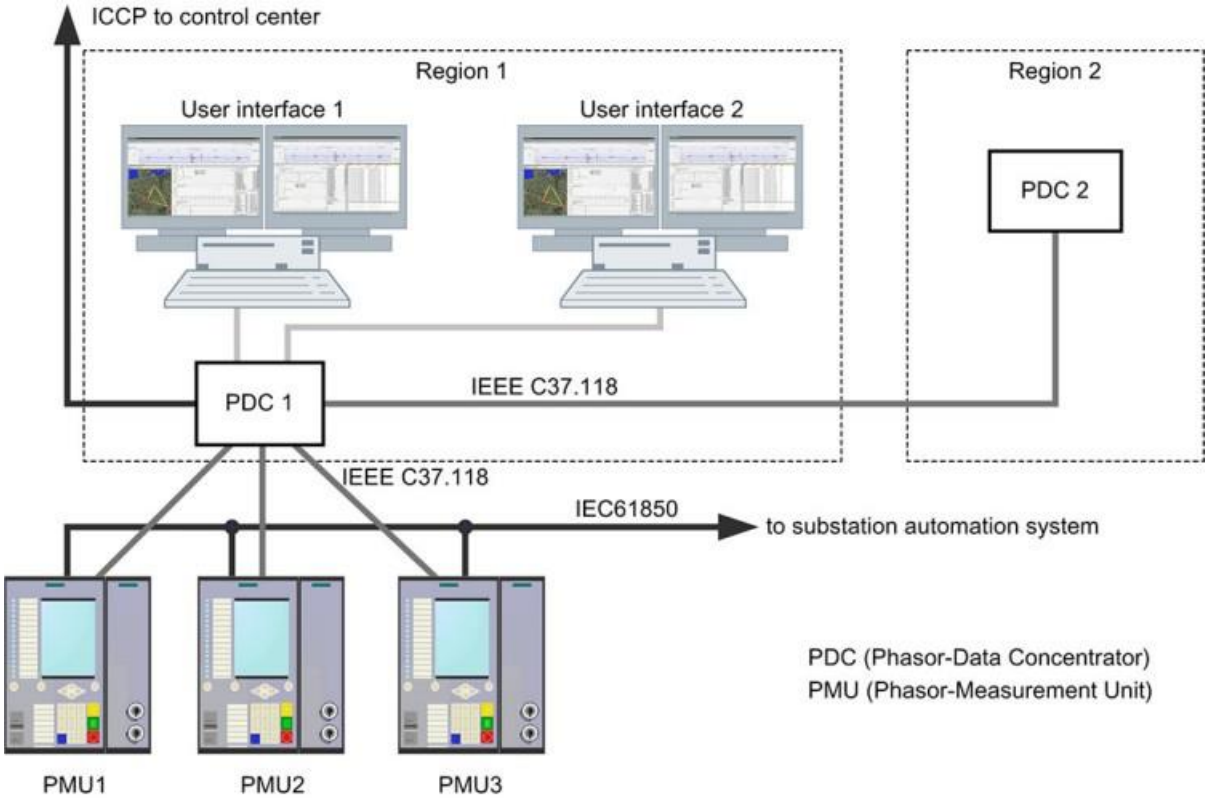
It is essential to identify and estimate the real-time distribution system topology following extreme disasters, which can contribute to ensure continuous, secure, and reliable supply of electricity to end-use consumers.

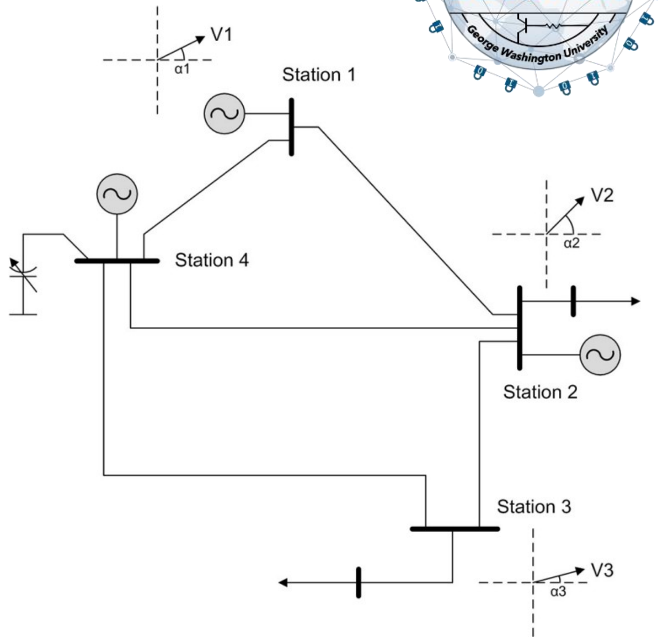


Reliable Supply

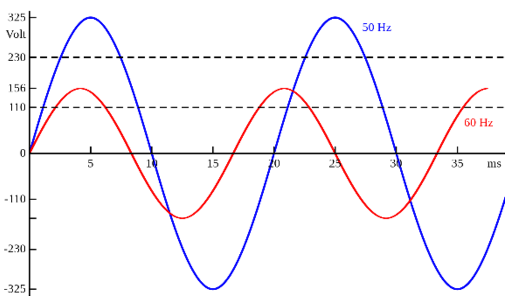
Restorative Service

Introduction





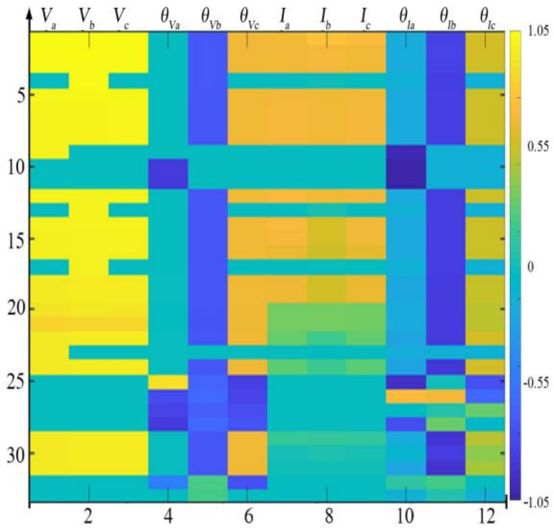
PDC (Phasor-Data Concentrator)
PMU (Phasor-Measurement Unit)



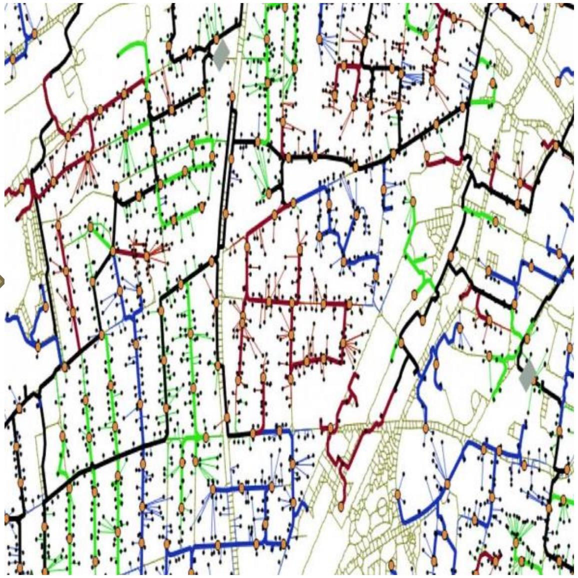
Phasor Measurement Unit (PMU):
- Magnitudes of voltage and current;
- Phasor angles of voltage and current;

Introduction

Our project investigates the effective utilization of the Autoencoder (AE) neural network model for online identification of the distribution system (DS) topology based on a large amount of real-time phasor measurement units (PMUs) detection data.



Autoencoder Model



PMUs Data

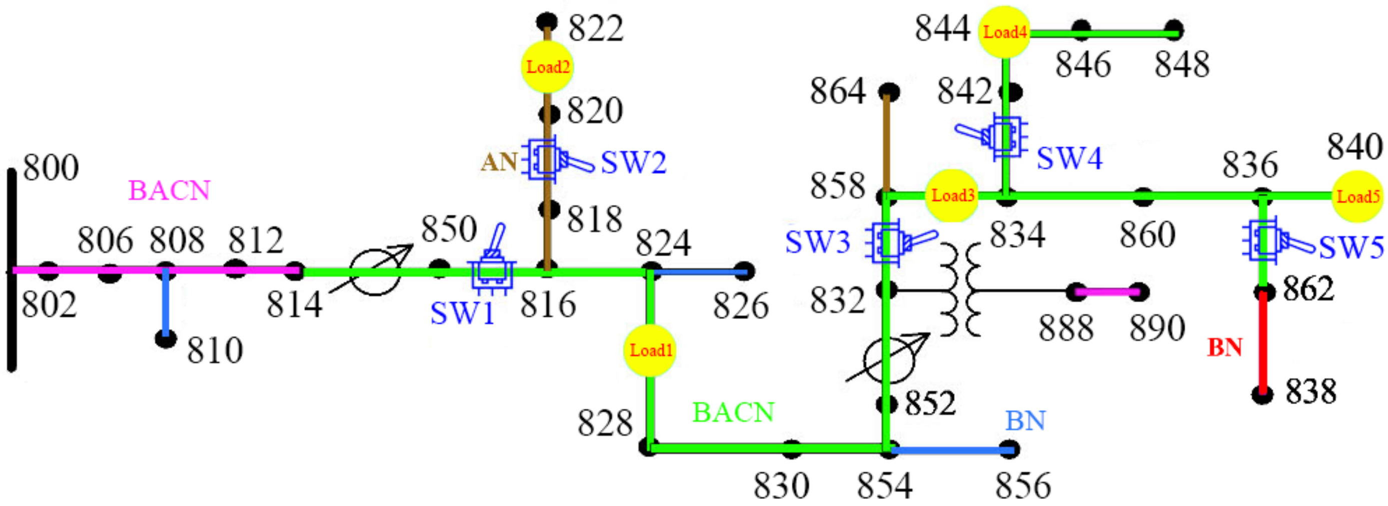
Recognize

DS Topology

Agenda

Data Mining

❑ Dataset Description



This radial power distribution system is an actual feeder located in Arizona, the feeder's nominal voltage is 24.9 kV and is characterized by:

1)Very long and lightly loaded overhead distribution lines

2)Two in-line regulators required to maintain a good voltage profile across the network

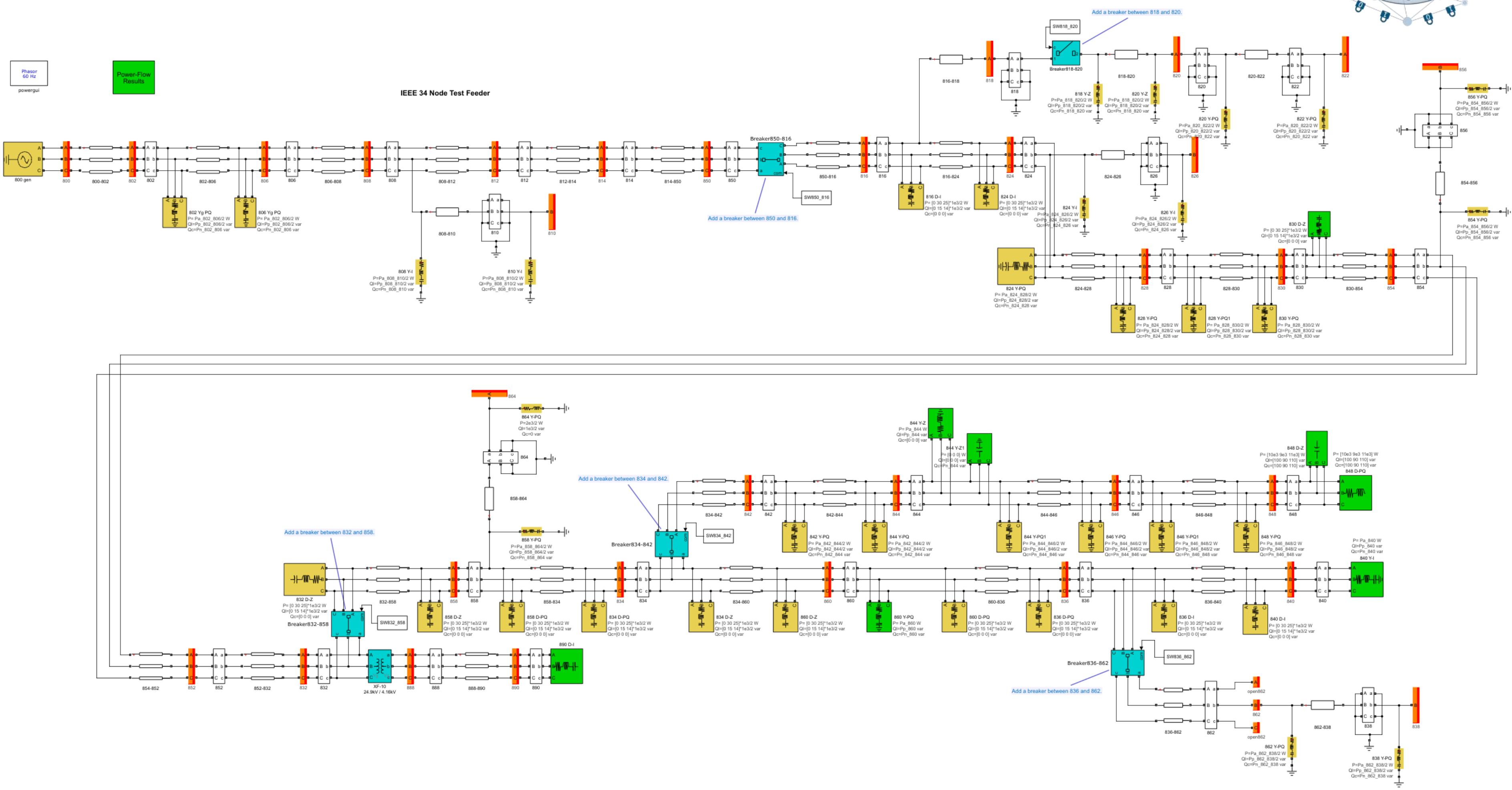
3)A wye-wye grounded transformer reducing the voltage to 4.16 kV for a short section of the feeder

4)24 unbalanced loading with both "spot" and "distributed" loads. Distributed loads are assumed to be evenly distributed on the distribution line.

5)Shunt capacitors

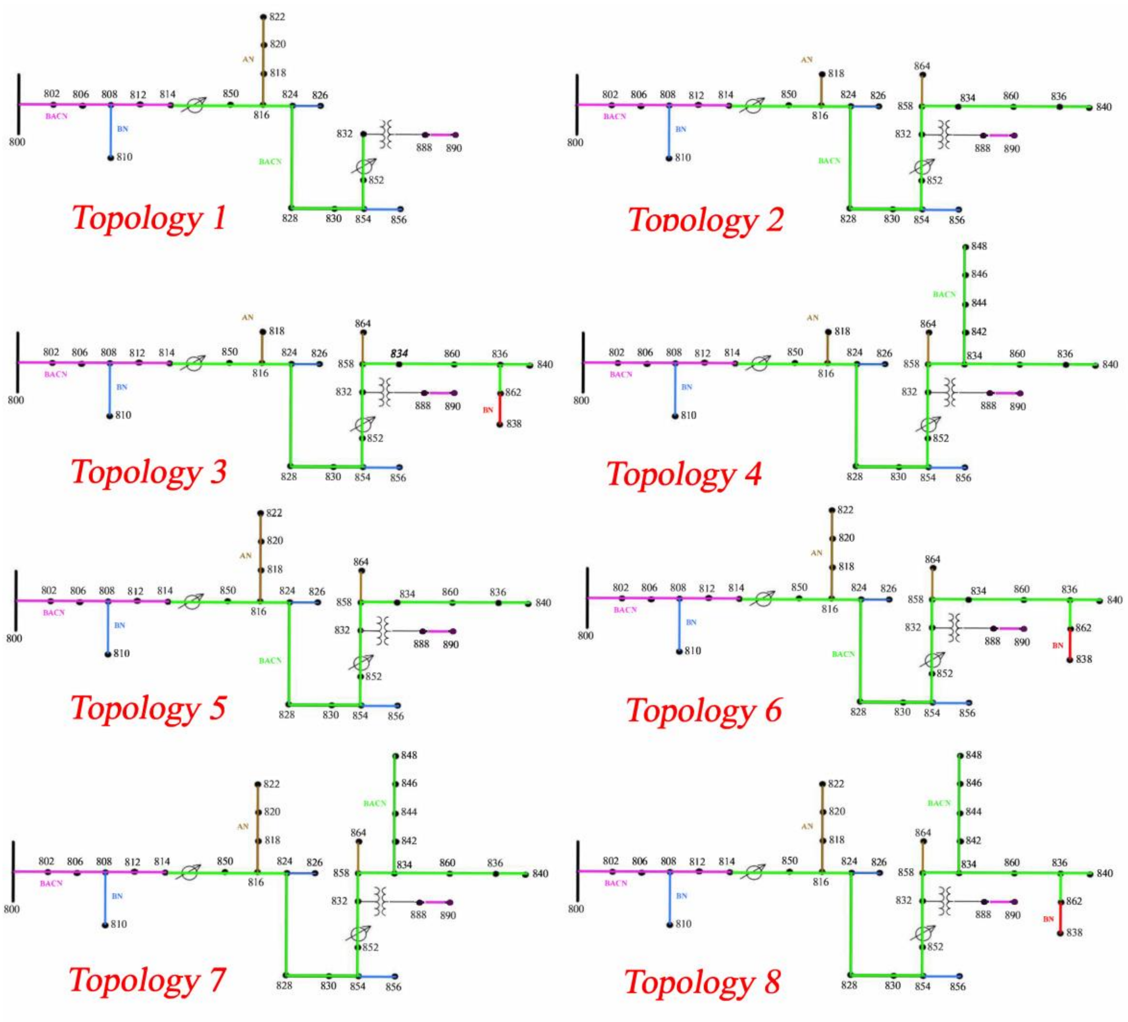
Data Mining

❑ Dataset Description



Data Mining

❑ Dataset Description

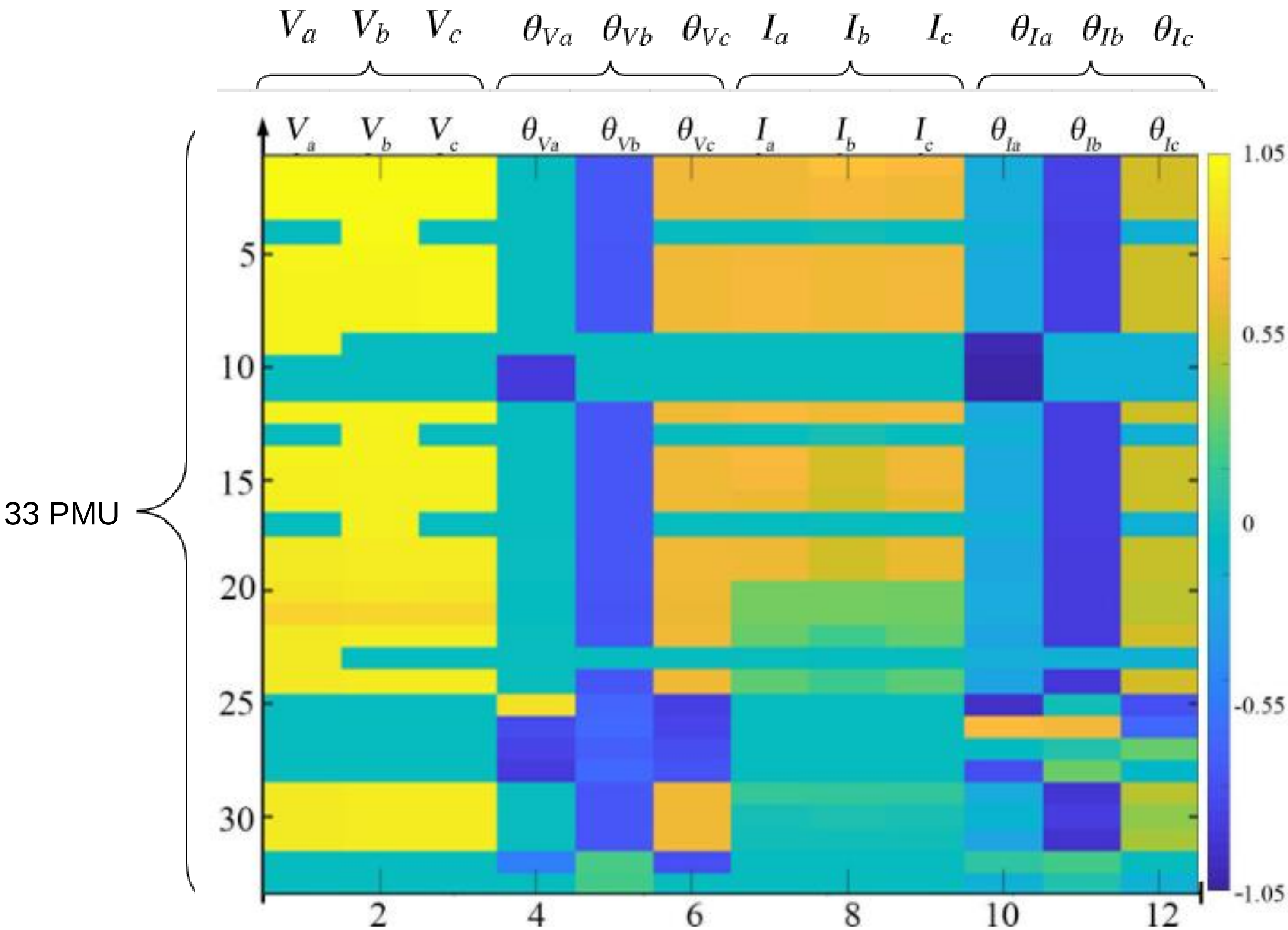


Topology	SW1	SW2	SW3	SW4	SW5	Number of Scenarios
1	1	1	0	0	0	1600 (40^2)
2	1	0	1	0	0	2197 (13^3)
3	1	0	1	0	1	2197 (13^3)
4	1	0	1	1	1	2401 (7^4)
5	1	1	1	0	0	2401 (7^4)
6	1	1	1	0	1	2401 (7^4)
7	1	1	1	1	0	3125 (5^5)
8	1	1	1	1	1	3125 (5^5)

95% ~ 105%

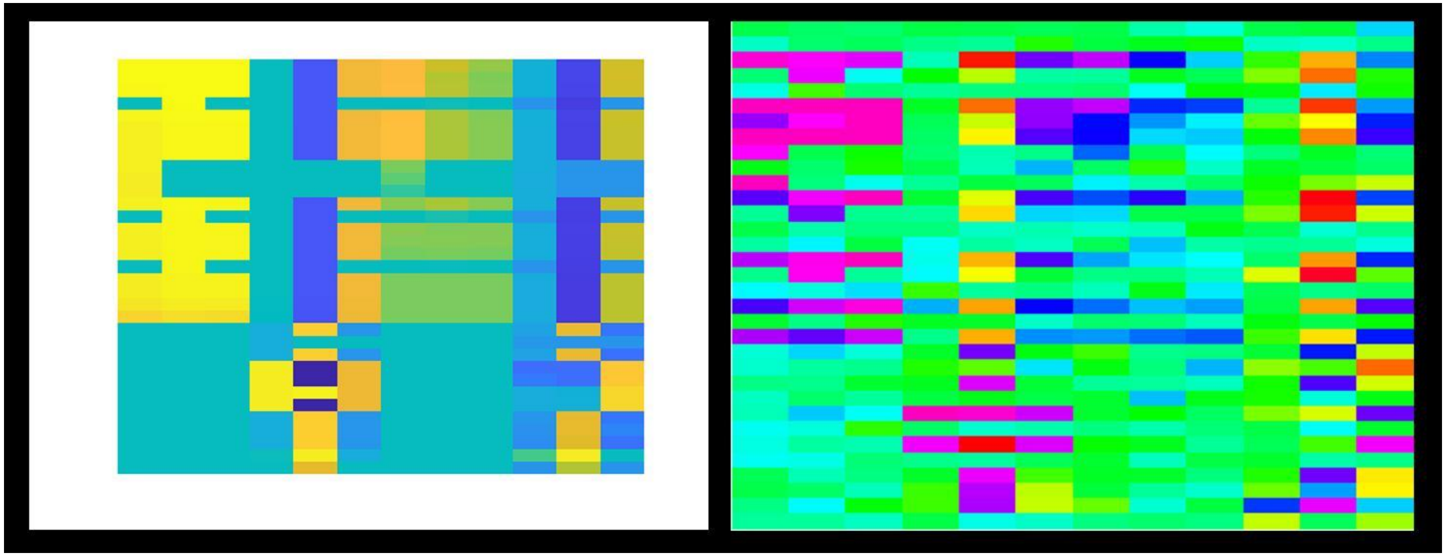
Data Mining

❑ Dataset Description



Data Mining

❑ Dataset Preprocessing



“parula” heatmap from MATLAB

Weakness:
•Very wide margin in the image
•Enlarge the size of input data
•Affect the prediction accuracy

“gist_rainbow” colormap in Python

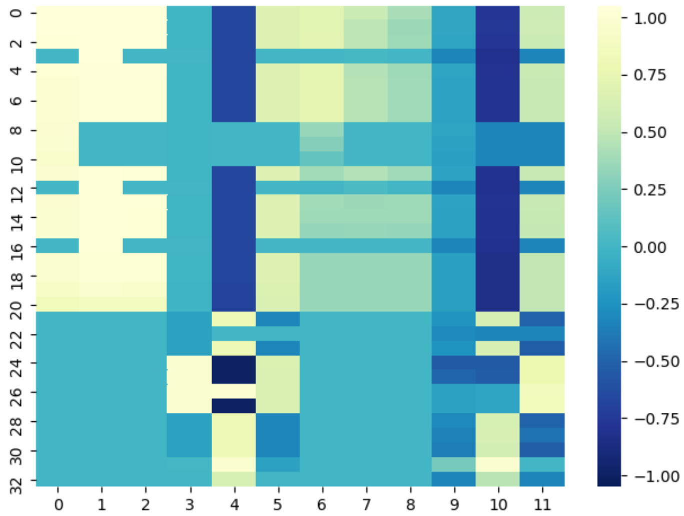
Strength:
•Enough compressibility
•Distinguished color

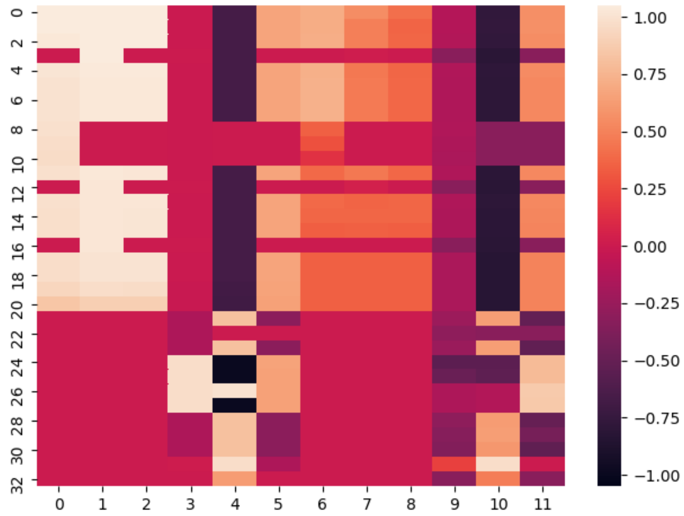
In Python, we used "seaborn.heatmap" library to generate the heatmap. In general format of heatmaps, the single color depth various changes cannot distinguish the data boundary well, so we use a colormap format as "gist_rainbow" in generating images as the input of the neural network.

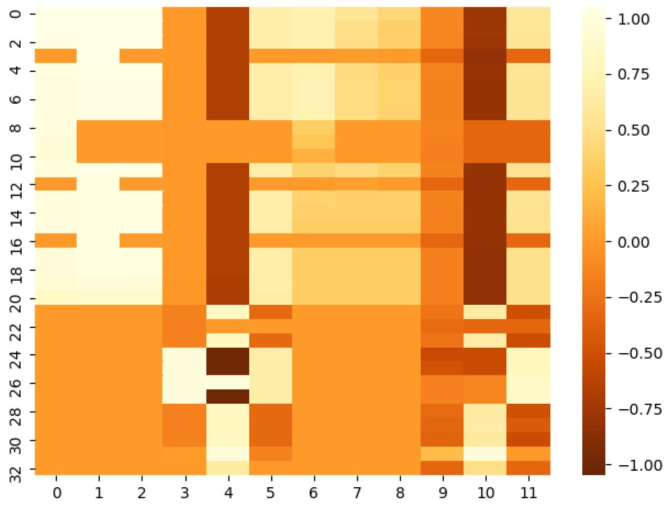
Ref: 1) http://jakevdp.github.io/blog/2014/10/16/how-bad-is-your-colormap/
2) https://seaborn.pydata.org/generated/seaborn.heatmap.html

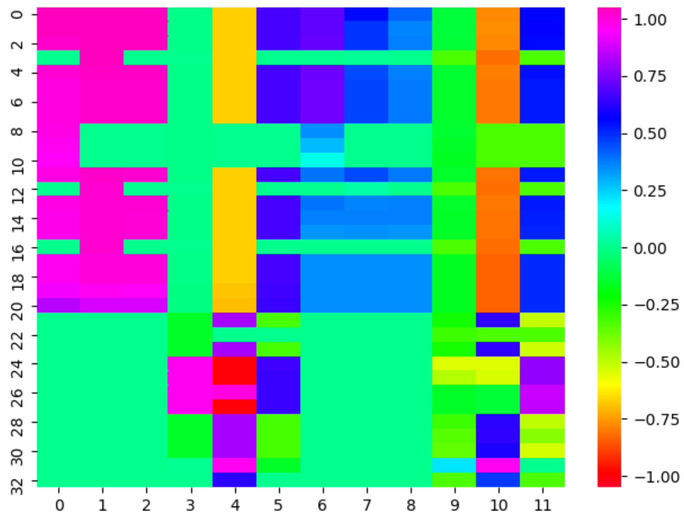
Data Mining

❑ Dataset Preprocessing

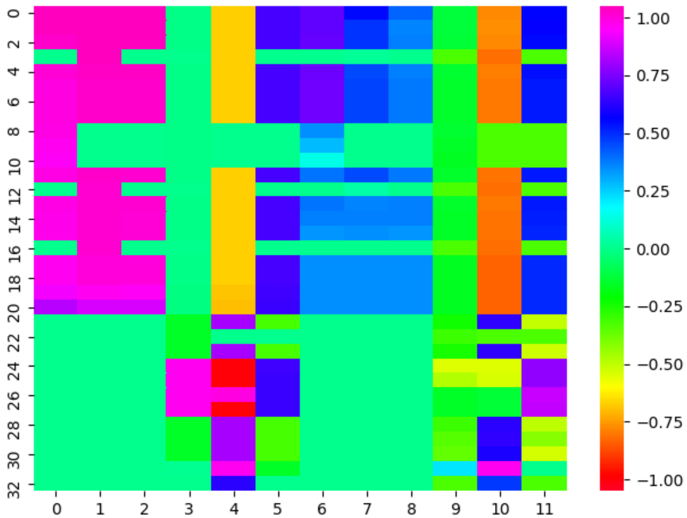








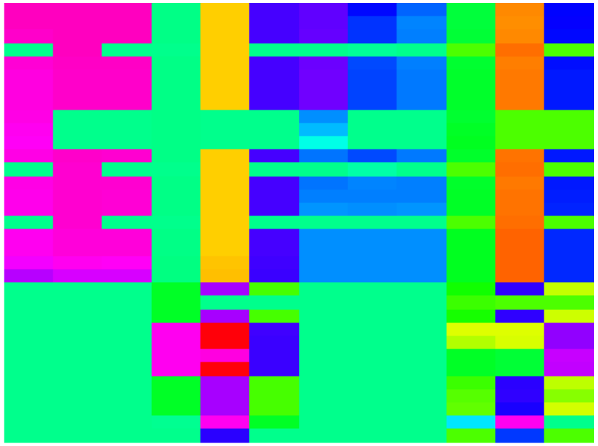
❑ Dataset Preprocessing



497 * 371 → 96 * 96

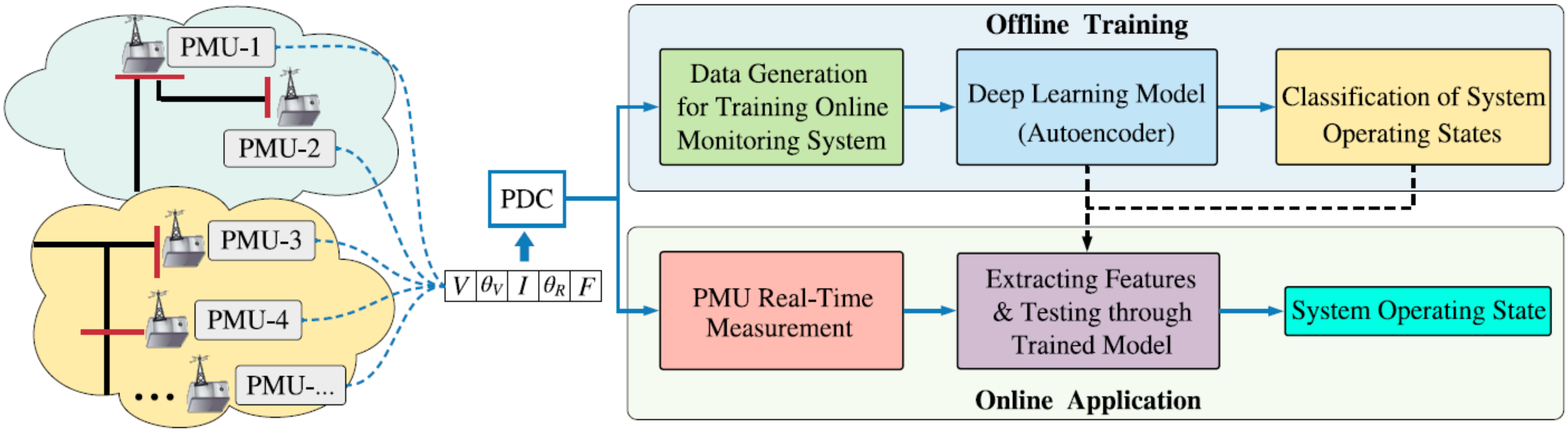
Train: 80%
Test: 10%
Val: 10%

Number: 19447



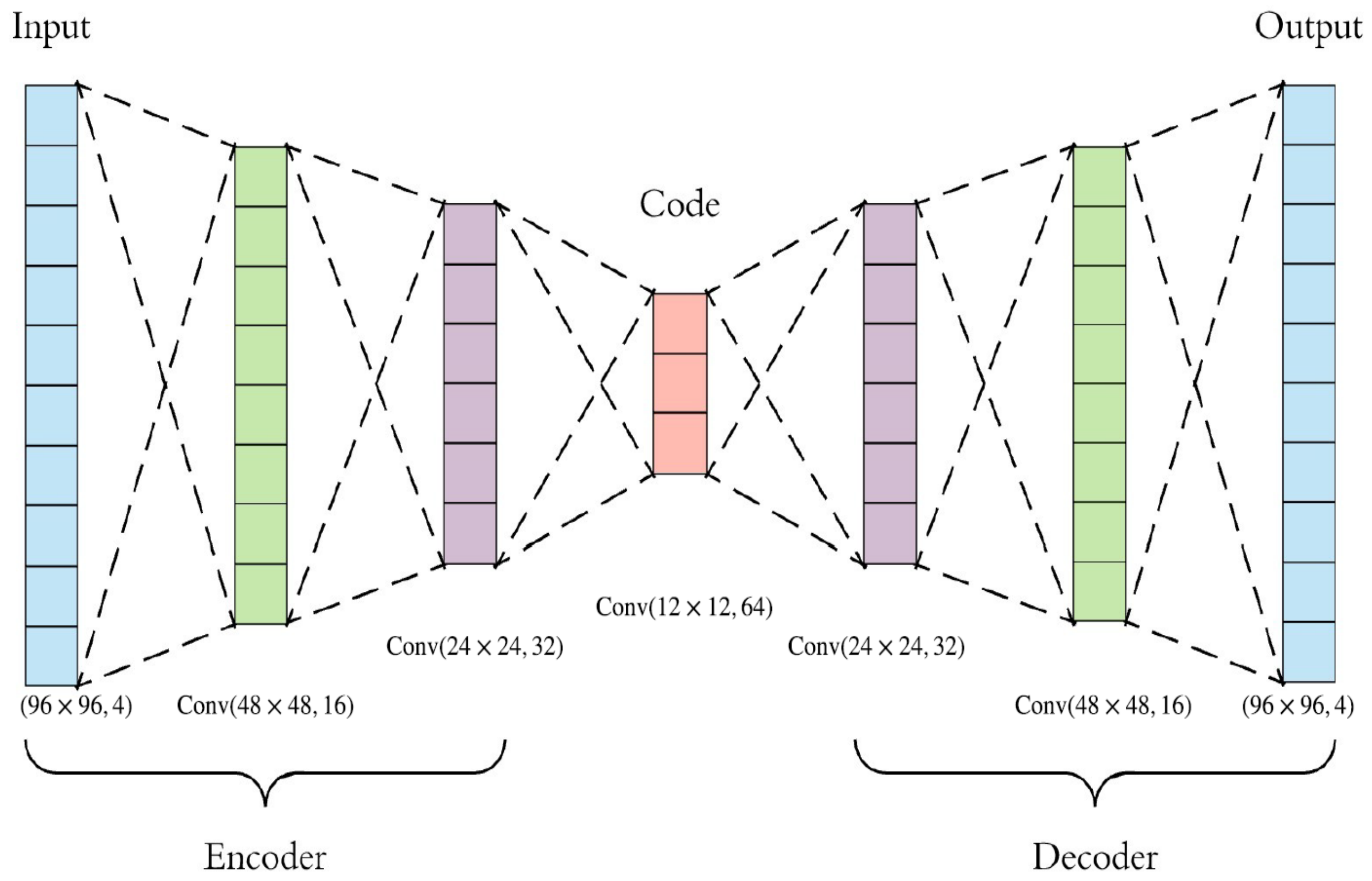
Agenda

Model Description



- The PMUs data collected from each bus in the distribution network is first used for offline training of the pre-built Autoencoder model.
- The trained model is then used for online identification of the power distribution network topology.

Model Description



The flowchart of the proposed autoencoder used in all models

Model Description

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 96, 96, 4)]	0
conv2d (Conv2D)	(None, 96, 96, 16)	592
batch_normalization (BatchNo	(None, 96, 96, 16)	64
max_pooling2d (MaxPooling2D)	(None, 48, 48, 16)	0
conv2d_1 (Conv2D)	(None, 48, 48, 32)	4640
batch_normalization_1 (Batch	(None, 48, 48, 32)	128
max_pooling2d_1 (MaxPooling2	(None, 24, 24, 32)	0
tf_op_layer_Relu (TensorFlow	[(None, 24, 24, 32)]	0
conv2d_2 (Conv2D)	(None, 24, 24, 64)	18496
batch_normalization_2 (Batch	(None, 24, 24, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 64)	0
conv2d_3 (Conv2D)	(None, 12, 12, 64)	36928
batch_normalization_3 (Batch	(None, 12, 12, 64)	256
up_sampling2d (UpSampling2D)	(None, 24, 24, 64)	0
conv2d_4 (Conv2D)	(None, 24, 24, 32)	18464
batch_normalization_4 (Batch	(None, 24, 24, 32)	128
up_sampling2d_1 (UpSampling2	(None, 48, 48, 32)	0
tf_op_layer_Relu_1 (TensorFl	[(None, 48, 48, 32)]	0
conv2d_5 (Conv2D)	(None, 48, 48, 16)	4624
batch_normalization_5 (Batch	(None, 48, 48, 16)	64
up_sampling2d_2 (UpSampling2	(None, 96, 96, 16)	0
conv2d_6 (Conv2D)	(None, 96, 96, 4)	580

Total params: 85,220
Trainable params: 84,772
Non-trainable params: 448

```
######################################################################################
# building the model
input_image = Input(shape=(96, 96, 4))
### Downsampling ---- Encoder
print('-- Encoding --')
z = layers.Conv2D(16, (3,3), padding='same', activation='relu')(input_image) # shape 96 x 96
z = layers.BatchNormalization()(z)
z = layers.MaxPool2D((2,2))(z) # shape 48 x 48

z = layers.Conv2D(32, (3,3), padding='same')(z) # shape 48 x 48
z = layers.BatchNormalization()(z)
z = layers.MaxPool2D((2,2))(z) # shape 24 x 24
z = activations.relu(z)

z = layers.Conv2D(64, (3,3), padding='same', activation='relu')(z) # 24 x 24
z = layers.BatchNormalization()(z)
encoder = layers.MaxPool2D((2,2))(z) # shape 12 x 12

### Upsampling ---- Decoder
print('-- Decoding')
z = layers.Conv2D(64, (3, 3), padding ='same', activation='relu')(encoder) # shape 12 x 12
z = layers.BatchNormalization()(z)
z = layers.UpSampling2D((2,2))(z) # shape 24 x 24

z = layers.Conv2D(32, (3, 3), padding ='same')(z) # shape 24 x 24
z = layers.BatchNormalization()(z)
z = layers.UpSampling2D((2,2))(z) # shape 48 x 48
z = activations.relu(z)

z = layers.Conv2D(16, (3, 3), padding ='same', activation='relu')(z) # shape 96 x 96
z = layers.BatchNormalization()(z)
z = layers.UpSampling2D((2,2))(z) # shape 48 x 48

# 4 channels because we have 4 channels in the input
decoder = layers.Conv2D(4, (3, 3), activation = 'sigmoid', padding = 'same')(z) # shape 48 x 48

# Building the model
autoencoder = Model(input_image, decoder)

# Printing the model summary
print(autoencoder.summary())
######################################################################################
```

Model Description

Ideal Model:

•All 33 PMUs are available
•Each load randomly changed from 95% to 105%
•Has neither missing values nor added noise

Non-Ideal Model:

•One-third of the PMUs are randomly removed from the whole bus system
•Each load randomly changed from 95% to 105%
•First model has 10dB SNR white noise added
•Second model has 10dB SNR white noise added in addition to randomly removed one data point

Code for Proposed AE Model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 96, 96, 4)]       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 96, 96, 16)        592
_________________________________________________________________
batch_normalization_6 (Batch (None, 96, 96, 16)        64
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 48, 48, 16)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 48, 48, 32)        4640
_________________________________________________________________
batch_normalization_7 (Batch (None, 48, 48, 32)        128
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 24, 24, 32)        0
_________________________________________________________________
tf_op_layer_Relu_2 (TensorFl [(None, 24, 24, 32)]      0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
batch_normalization_8 (Batch (None, 24, 24, 64)        256
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dense (Dense)                (None, 128)               1179776
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 1032
=================================================================
Total params: 1,204,984
Trainable params: 1,180,808
Non-trainable params: 24,176
```

*SNR: signal-to-noise ratio
The smaller the SNR is, the bigger the noise is.

Agenda

1 Introduction

2 Data Mining

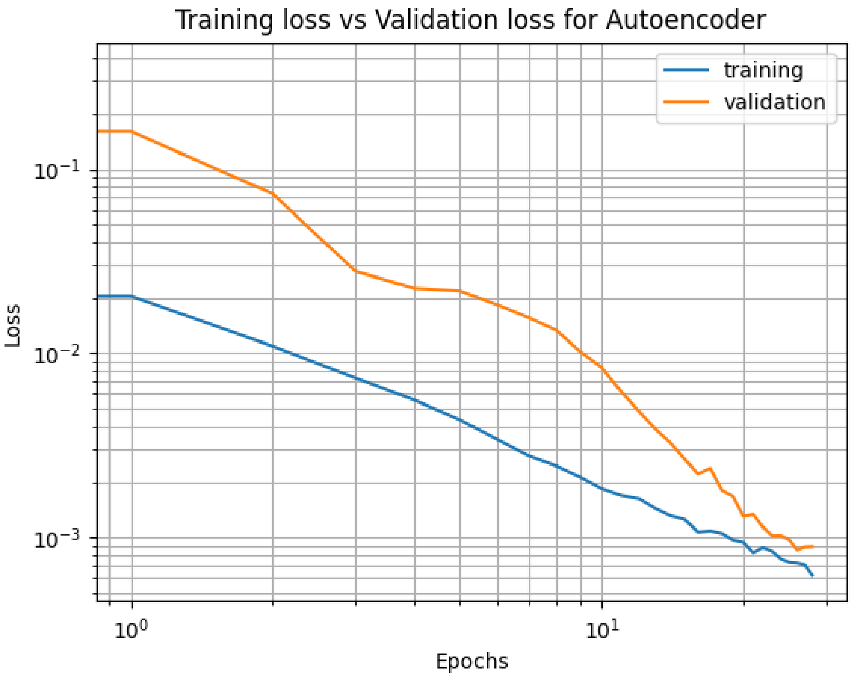
3 Model Description

4 Numerical Results

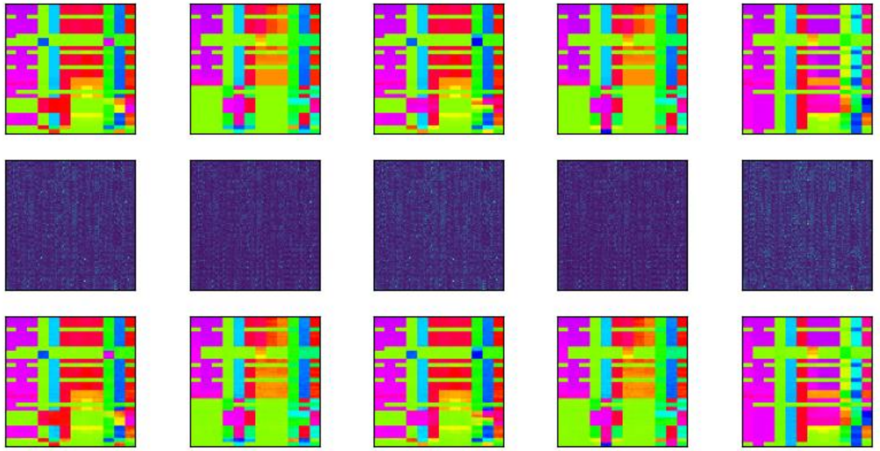
5 Conclusion

Numerical Results

❑ Ideal Model

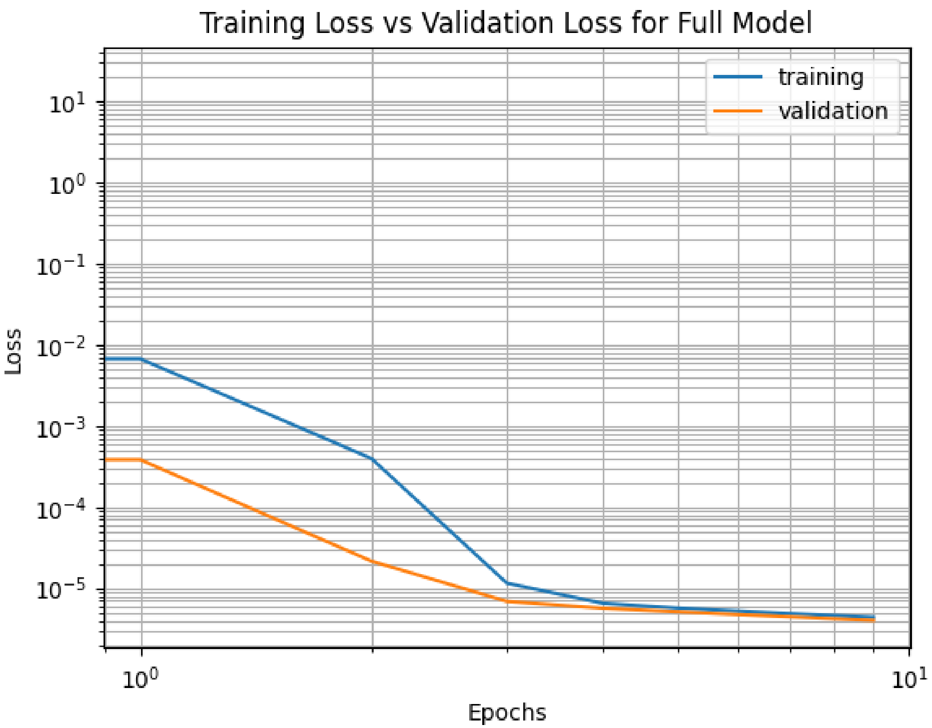


Test Images - Encoded Test Images - Reconstructed Test Images



Numerical Results

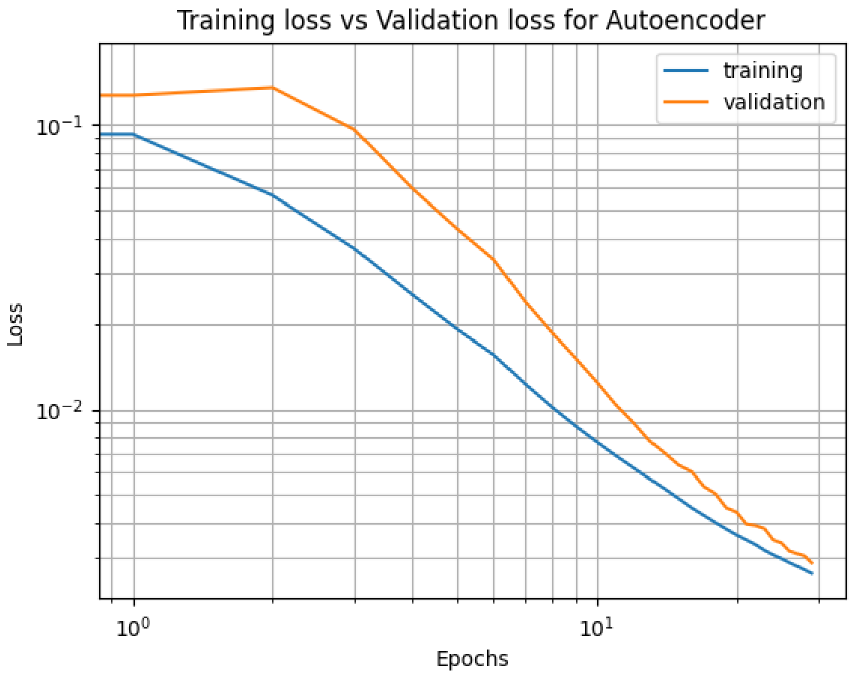
❑ Ideal Model



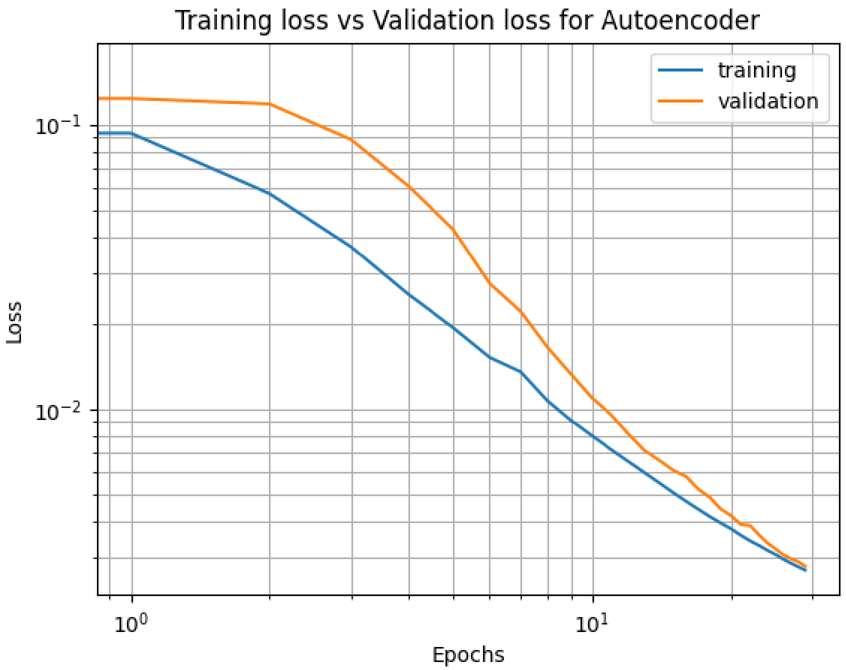

```
Test loss on test set: 4.459287993086036e-06
Test accuracy on test set: 1.0
(3890,) (3890, 8)
Found 3890 correct labels
Found 0 incorrect labels
```

Numerical Results

❑ Non-Ideal Models



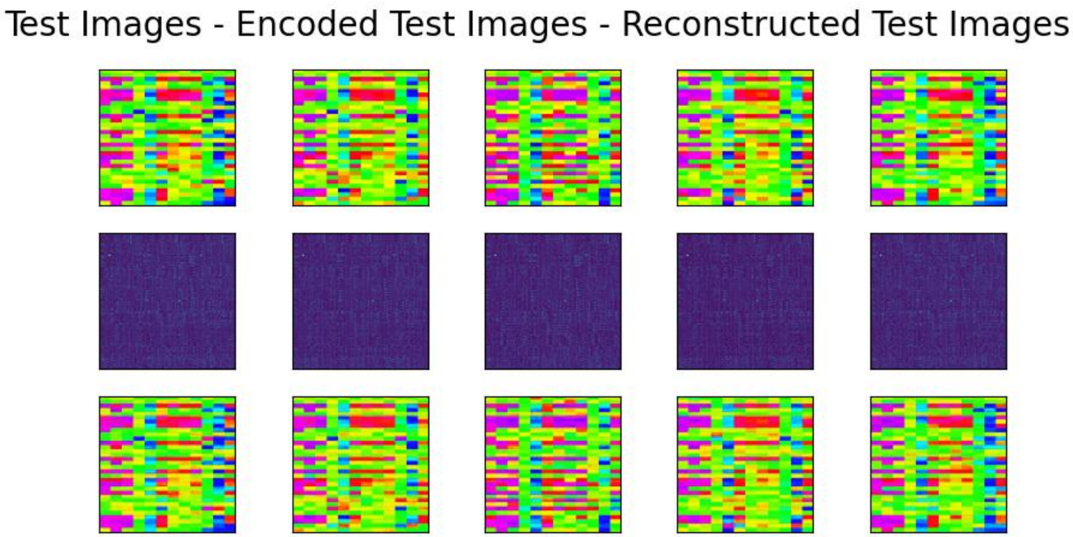
AE 10dB SNR



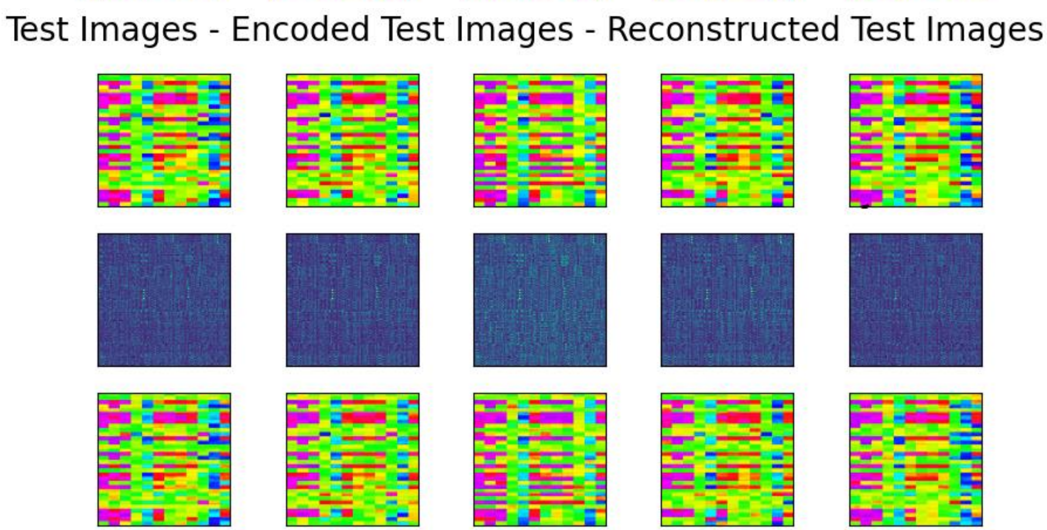
AE missing one data and 10dB SNR

Numerical Results

❑ Non-Ideal Models



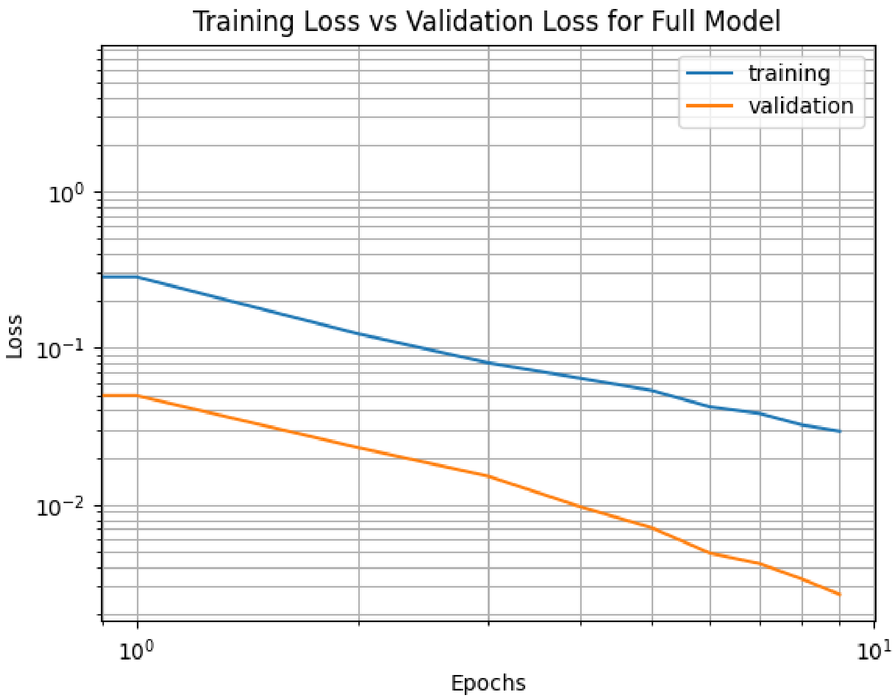
AE 10dB SNR



AE missing one data and 10dB SNR

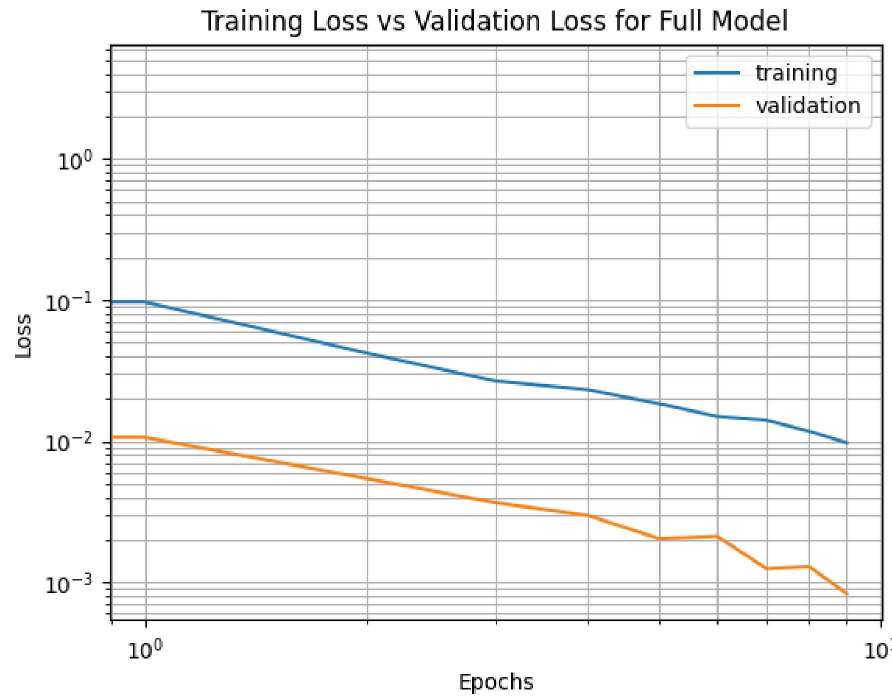
Numerical Results

❑ Non-Ideal Models




```
Test loss on test set: 0.002677673939615488
Test accuracy on test set: 1.0
(1944,) (1944, 8)
Found 1944 correct labels
Found 0 incorrect labels
```

AE 10dB SNR




```
Test loss on test set: 0.0008325269445776939
Test accuracy on test set: 1.0
(1944,) (1944, 8)
Found 1944 correct labels
Found 0 incorrect labels
```

AE missing one data and 10dB SNR

Numerical Results

❑ Model Test on Interfered Datasets

Total 8 topologies

Models \ Interfered Data	40dB SNR	Missing One Data based on 40dB SNR	Missing Two Data based on 40dB SNR
Ideal Model (33 PMU with no Missing and 0dB SNR)	55.357	55.214	55.125
22 PMU with no Missing and 10dB SNR Model	86.571	86.482	86.053
22 PMU with Missing One Data and 10dB SNR Model	77.373	76.998	76.374

7 topologies (Topology 3 be removed)

Models \ Interfered Data	40dB SNR	Missing One Data based on 40dB SNR	Missing Two Data based on 40dB SNR
Ideal Model (33 PMU with no Missing and 0dB SNR)	79.571	79.381	79.285
22 PMU with no Missing and 10dB SNR Model	71.476	71.428	71.428

Agenda

Conclusion

- A deep Learning AE framework is proposed for online detection of distribution system topology
- The proposed AE framework can handle the interfered data (e.g., missing and noise measurements) under unbalanced operating states
- Numerical experiments proved that our trained network can accurately identify the network topology corresponding to the observed data beyond the training dataset

Future Work:

❖The proposed framework could be applied in a larger real-world distribution system (e.g., IEEE-123 node test system)

❖The proposed AE neural network could be optimized for increasing the prediction accuracy of general models for improving universality

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Thank You!

liyifu@gwu.edu
mahijazi@gwu.edu
jsu66@gwu.edu

SMART GRID LABORATORY
George Washington University