Date-Driven Topology Identification in Power Distribution Systems with Machine Learning

by Yifu Li

Bachelor of Engineering in Electrical Engineering and Its Automation,
June 2016, Changzhou University

A Thesis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Master of Science

August 31, 2020

Thesis directed by

Payman Dehghanian
Assistant Professor of Electrical and Computer Engineering

Dedication

This is dedicated to my parents, Weimin Pan and Jun Li, who give birth to me, love me, believe in me, inspire me and have supported me in every step of my life. Without their words, I could never have walked this far...

Also, I'd like to dedicate this thesis to all my relatives and friends. Thank you for all your support and help during this tough period.

Acknowledgments

First of all, I wish to thank Dr. Payman Dehghanian, my academic advisor, for introducing me to the fantastic world of electrical power systems engineering, continually guiding me and giving me confidence and knowledge on this road.

I sincerely thank all of the other wonderful members of the GW SmartGrid Lab for their consistent help during my two-year master's degree period. Specifically, my thanks go to Shiyuan Wang who illuminated the darkness of the current study with his special intelligence, and Li Li who patiently used his expertise to help me within the neural networks field.

Finally, my greatest appreciation to all of my friends, especially Jinshun Su, Dingwei Wang and Fei Teng. When time gets tough, I know that we have each other.

Abstract

Date-Driven Topology Identification in Power Distribution Systems with Machine Learning

With the increase in demand for quality electricity and the number of end-use consumers, the operation and control of power grids have become more and more complex and challenging. Ensuring acceptable reliability and quality of the electricity supply has become particularly important to every aspect of our electrified economy. Due to the growing deployment of Micro-Phasor Measurement Units (μPMUs) in power distribution grids, an abundance of high-resolution measurements is available that can be harnessed for smarter operation and fault analyses in power distribution networks. Traditional models have revealed limitations on the network topology identification which may occupy manpower and material resources with no guaranty to effectively restore power in a short time period when facing faults and other disruptions. This thesis suggests and implements a machine learning framework that uses the μPMU measurements as inputs and provides a full observation of the network topology in real-time. Specifically, the proposed framework employs a Convolutional Neural Network (CNN) to identify the physical state of the power network at all times. The framework was evaluated on the IEEE 34-Node Test Feeder, where the experiments show that the proposed CNN can achieve a promising performance with high accuracy even when the μPMU measurements contain noises and/or missing entries.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

μPMU Micro-Phasor Measurement Unit 2, 8, 11, 16, 18, 22–25, 28, 32, 33, 38–40, 42–45, 47–51, 53–55

AC Alternating current 7

AE Autoencoder 13, 56

AI Artificial Intelligence 8, 12

ANN Artificial Neural Network 8

CapsNet Capsule Neural Network 56

CNN Convolutional Neural Network 8–11, 13–15, 18, 22, 25, 40–43, 45–48, 55, 56

Conv Convolutional 15, 18

CPU Central Processing Unit 45

DER Distributed Energy Resources 12, 15, 18

DL Deep Learning 8, 14

DSSM Deep Structured Semantic Model 13

FACTS Flexible Alternating Current Transmission System 3

FC Fully connected 11, 14, 18

FTU Feeder Terminal Unit 1, 2

GAN Generative Adversarial Network 14

GPS Global Positioning System 7, 12

MATLAB Matrix Laboratory 11, 20, 23, 25, 28, 40, 44

ML Machine Learning 8

MLP Multilayer Perceptron 13

NADE Neural Autoregressive Distribution Estimation 14

PMU Phasor Measurement Unit 1, 6–8, 11, 12, 16, 17, 56

RBM Restricted Boltzmann Machine 13

ReLU Rectified Linear Unit 10, 18, 42

RNN Recurrent Neural Network 13

RTU Remote Terminal Unit 1, 2

SIANN Shift Invariant or Space Invariant Artificial Neural Network 9

SNR Signal-to-Noise Ratio 46–48

SVM Support Vector Machine 56

SW Switch or breaker 23, 53

TTU Transformer Terminal Unit 1, 2

Chapter 1: Introduction

1.1 Background

With the development of modern electrical systems for half a century, there are more and more opportunities to enhance the safety, economics, reliability, sensitivity, and flexibility of the power system. For a long time, the complicated and challenging electrical structure of the power distribution grid, insufficient measurement configuration and utilization, information and measurement sparsity, and other problems have spread throughout the network layout, where an improvement in the monitoring and controllability may affect the reliability of the power supply, power quality, and the economics of the system operation. In order to generate and dispatch the electrical power efficiently and operate the power grid stably and safely, system operators need to be informed of the electrical network topology and demand profiles across the network at all times. The observability and controllability of the electrical network are essential to ensure its safe and economic operation. The existing distribution network Feeder Terminal Unit (FTU), distribution Transformer Supervisory Terminal Unit (TTU), Remote Terminal Unit (RTU) and some other measuring equipment do not provide synchronized measurements; and the uploaded measurements are minutely, which makes it difficult to meet the intelligent fast-speed operation and control of the power distribution systems [7–12]. Hence, with the growing complexity in the power grid structure reinforced with heterogeneous resources and the increasing demand for electricity needed for an electrified economy, Phasor Measurement Units (PMUs) have been introduced and widely deployed to observe the dynamic performance of the power grid with

synchronized measurements [13–23]. Accordingly, an abundance of high-resolution data is now becoming available for further analysis and informed decision making.

As synchrophasor data becomes more available, there is an increasing need to be able to effectively process and analyze the data to ultimately improve the ways power systems are operated. In power distribution systems, Micro-PMUs (or μPMUs) are used [24]. Combined with the topological processing of the visual images, based on the measurements provided by μPMU, one can better handle the abnormal conditions in the grid (i.e., fault location, fault detection, etc.), thereby reducing the enormous economic consequences and improving the power grid resilience. Compared to the traditional event detection schemes and infrastructure such as FTU, TTU, and RTU, Micro-PMUs (μPMUs) in power distribution grids offer yet-untapped potential for online situational awareness, i.e., event detection, classification, and high-fidelity high-resolution measurements.

1.2 Smart Grid Resilience

Smart grids, where cyber-infrastructure is used to make power distribution more dependable and efficient, are prime examples of modern infrastructure systems. The cyber-infrastructure provides monitoring and decision support intended to increase the dependability and efficiency of the system. However, this comes at the cost of vulnerability to accidental failures and malicious attacks, due to the greater extent of virtual and physical interconnection. Any failure can propagate more quickly and extensively, and as such, the net result could be a compromised reliability and security [25, 26].

Reliability and **Resilience** are two different concepts in power grids.

The overall reliability of a cyber-physical power grid is a function of the respective reliability of its elements, including both physical components, e.g., generators and transmission lines, and cyber components, e.g., control software, communication links, FACTS devices, and sensors. Reliability quantifies the likelihood of a system to function (or fail) as specified, under given conditions, over a given duration [27]. It takes a binary view of a system, were the only states possible are "functional" and "failed." It also includes the maintainability of the systems and its constituent components over time [28–49]. As such, this metric is of limited use in evaluating the system after a failure occurs [50, 51].

A quantitative metric and concept useful to this end is "Resilience", defined as the ability of a system to bounce back from a failure [52]. Recovery does not imply the perfect restoration of the system's functionality; but instead implies that the system has returned to a state where it is considered functional [25]. Reference Figure 1.1 shows different operating states a power system may experience [53–55].

- **Normal State**: In this state, the system parameters such as voltage, frequency, current, etc. are within the normal and desired range of operation and the energy supply meets the demand. Event a component fails, the system will be able to meet the demand with all operational variables within the desirable thresholds.

- **Alert State**: In this state, all system parameters are within the acceptable range but very close to their limits. In case of a failure in one system element, the generation and demand will be still in balance; however, some operational variables may be violated.

- **Emergency State**: In this state, some system parameters are outside

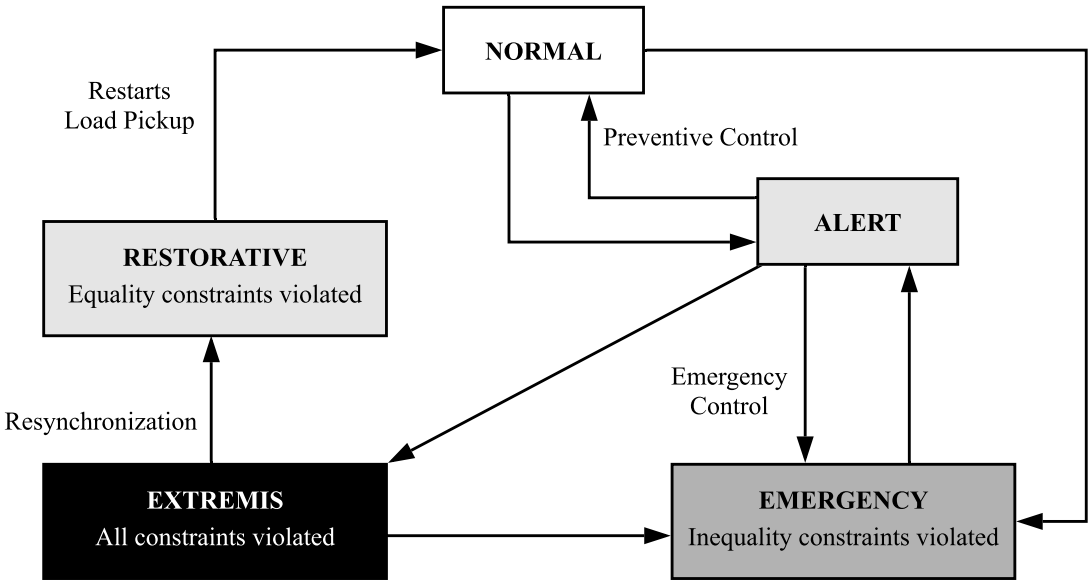


Figure 1.1: Power System Operating States [1]

their acceptable range. This may lead to system disintegration if a failure occurs in the system.

- **Extremis State**: In this state, partial or system-wide black-out may occur. That is the generation and demand are not in balance.

- **Restorative State**: In this state, the system goes into a process of restoration by reconnecting system elements and re-synchronizing generators to achieve the normal operating state.

Over the past couple of years, the GW Laboratory researchers have studied the resileince challenges that the power grid faces to a wide range of threats and have proposed solutions for detection, verification, and mitigation in response. Interested readers can refer to [26, 56–76].

1.3 Challenges and Opportunities

Data analytics can improve resiliency in the dynamic grid [77]. Different from the traditional power grids, modern grids are facing a high penetration of distributed renewables more than ever before and a lot of research has been devoted to capture the uncertainties in the power grid (both transmission and distribution) to harness their full potential in the operation and control paradigms [54, 55, 78–83]. Meanwhile, as the number of consumers connected to the grid increases, the loads become more active and controllable, and the storage devices also need to keep the pace with such evolution. Additionally, new technologies like energy storage resources and electric vehicles have been proliferated in recent years in power grids, making it a heterogeneous ecosystem of physical and cyber infrastructures [72, 84–92].

Access to high-fidelity measurements in power distribution systems is particularly critical, and at the same time challenging due to the following reasons [93]:

(i) The length of the power distribution lines is usually between 5 to 10 kilometers, resulting in the phase angle difference between the two ends of the line to be commonly small (sometimes even lower than 0.1°).

(ii) The proliferation and rushing arrival of renewables have increased the complexity in the grid structure and the way electricity flows in the network. A three-phase unbalance architectures are commonly seen in power distribution systems, which could result in more than 30% inter-harmonics and under 60dB noise conditions.

(iii) The fast switching characteristics of power electronic devices lead to more electrical transients, further mandating the higher efficiency

and dynamic tracking capability of the event detectors in the power distribution sector.

While PMU measurements can be shared over communication networks in real-time and collected at a centralized platform, called Phasor Data Concentrators (PDC) [94] for further processing, the underlying network models are mostly unavailable or incomplete. In most cases, PMU data is hardly available in full (but limited) for research through the non-disclosure agreements (NDAs) [95]. This makes it challenging to get the most out of the synchronized measurements for fault detection and localization applications when the real-time network topology is unknown or not accurate. The problem of estimating the state of the power grid is usually divided into two interrelated phases: the first is the state estimation in which the estimated value is the voltage at all buses across the network, and the second is topology processing and topology error detection, in which the breaker status is used to track the current topology of the grid, and to detect and correct the errors in the calculated topology. These two stages iterate, and the combined process is known as a generalized state estimation [96]. With the measurements received from the μPMUs and when judiciously integrated with the topological processing of the visual images, the abnormal conditions in the distribution network (e.g., fault location, fault detection, etc.) can be better handled, further improving the network reliability, reducing the economic losses, and mitigating the electrical safety concerns.

1.4 PMU Operating Principle

A Phasor Measurement Unit (PMU) is a device used to measure the magnitude and phase angle of an electrical waveform, i.e., phasor quantity (such as voltage or current) in the electricity grid using a common time

source for synchronization [97]. Due to the fact that the time is synchronized by the Global Positioning System (GPS), PMUs are able to capture real-time electrical phasor quantities from multiple remote points on the power grid, thereby providing a real-time snapshot of the entire grid making it possible to approach wide-area monitoring, protection and control.

Additionally, PMUs are also used to measure the frequency in power grids. A typical commercial PMU could report measurements with a high temporal resolution in the order of 30-60 measurements per second. This helps engineers in analyzing dynamic events in the grid which is not possible with traditional SCADA measurements that generate one measurement every 2 or 4 seconds [98]. Hence, the synchronization ability makes PMU have a critical role in protecting the electric systems from power outages, because they could reduce the grid's stress caused by imbalances in power supply and demand.

A PMU could measure 50/60 Hz AC waveforms (voltages and currents) typically at a rate of 48 samples per cycle making them effective at detecting fluctuations in voltage or current at less than one cycle. Phasor measurements from PMUs are constructed from cosine waves, that follow the structure below [99]:

$$Acos(\omega t + \theta) \tag{1.1}$$

Wherein, A is the scalar value, and usually stands for voltage or current magnitude; θ is the phase angle offset from some defined starting position; ω is the angular frequency of the waveform, that is most often described as a constant of $2\pi 50$ Hz or $2\pi 60$ Hz. It is worth noting that when the frequency does not oscillate around or near 50/60 Hz, PMUs are not able to accurately reconstruct these waveforms, because the PMU is unable to fit the waveform exactly when it is non-sinusoidal [99].

Historically, only small numbers of PMUs have been used to monitor transmission lines with acceptable errors of around 1%. While PMUs are generally used on transmission systems, new research is being done on the effectiveness of μPMUs for power distribution systems. That is because the PMU can not only be a dedicated stand-alone sensor, but also its functionality could be incorporated into a protective relay or other intelligent electronics devices [100]. μPMUs can help decrease the error in the phase angle measurements of the distribution line from $\pm 1^{\circ}$ to $\pm 0.05^{\circ}$, giving a better representation of the true phase angle value [24].

1.5 Convolutional Neural Network

First of all, the differences among Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) need to be known. As Figure 1.2 shows below, Artificial Intelligence is a science field that aims at finding solutions to complex problems like humans do. A decision mechanism that is similar to a real human decision mechanism is tried to be modeled with some algorithms. Machine learning is a sub-domain of artificial intelligence. Machine learning uses mathematical and statistical ways to extract information from data, and with that information, ML tries to guess the unknown. Deep learning is a sub-domain of ML and tries to learn the data with the artificial neural network approach [101, 102].

In machine learning, artificial neural networks (ANNs) are abstractions of biological neurons that can be trained to perform useful functions like human brain [103]. A Convolutional Neural Network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery since the 1980s [104]. They are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on their shared-

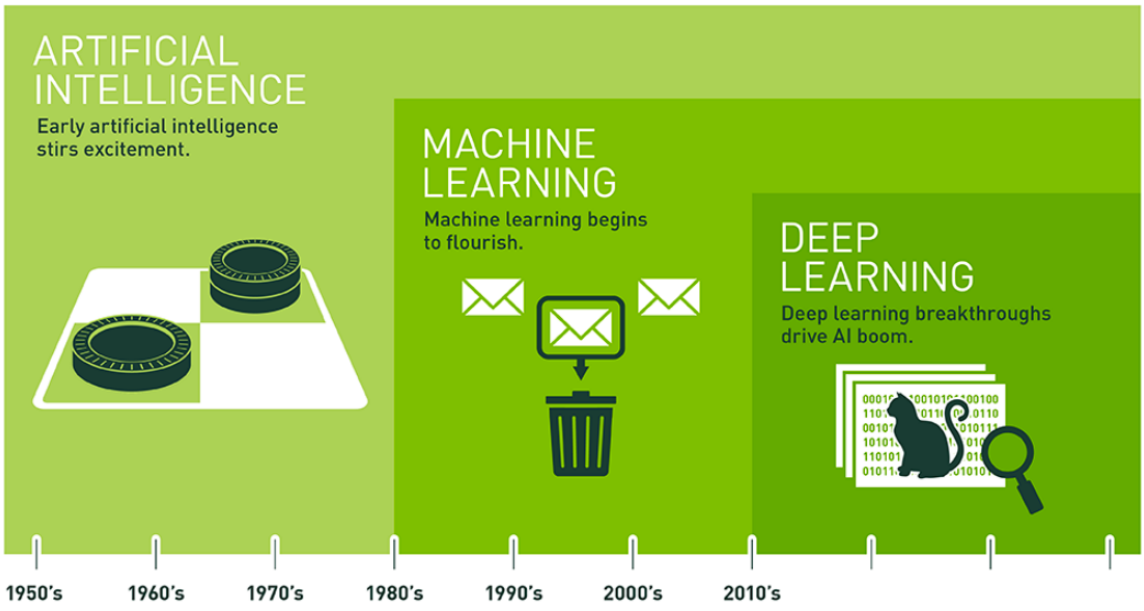


Figure 1.2: Evolution of Artificial Intelligence [2]

weights architecture and translation invariance characteristics [105, 106].

CNNs are regularized versions of multilayer perceptrons. CNNs are a derivative of standard neural networks which are made up of neurons with learnable weights and biases. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. But instead of using fully connected hidden layers in the regular neural network, the CNN introduces a special network structure, which consists of convolution layers and pooling layers, to address the challenges in the computer vision [107].

CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extremity [108]. Each neuron receives several inputs, takes a weighted sum over them, passes it

through an activation function and responds with an output. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. Figure 1.3 shows an architecture of CNN. *Convolution* has the nice property of being translational invariant. Intuitively, this means that each convolution filter represents a feature of interest (e.g pixels in letters) and the CNN algorithm learns which features comprise the resulting reference (i.e. alphabet) [107].

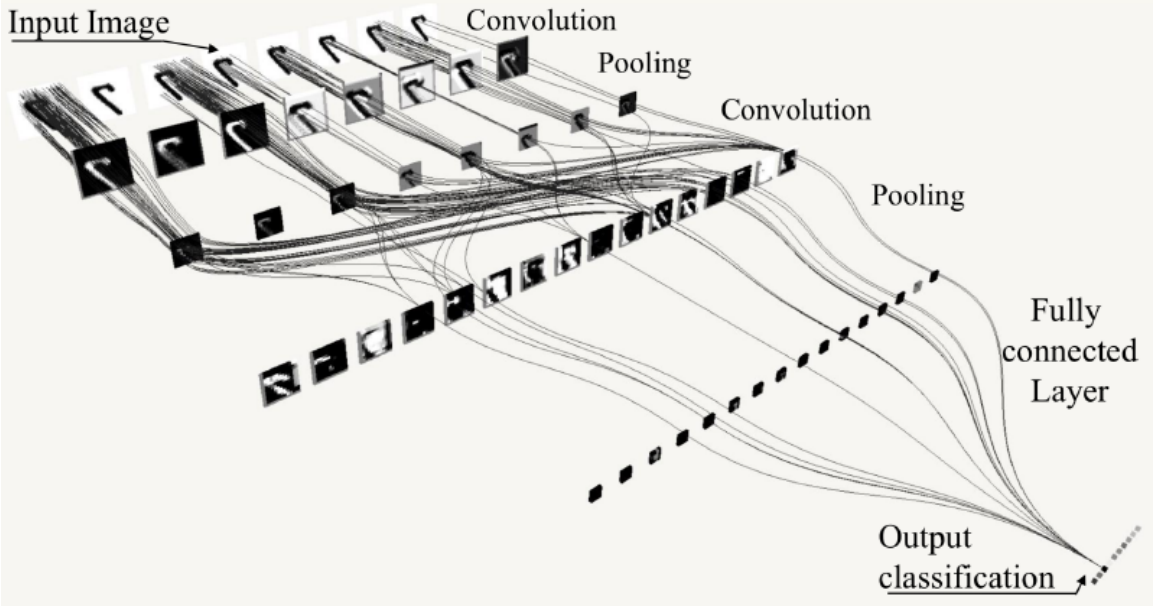


Figure 1.3: Convolutional Neural Network Architecture [3]

After convolution, the next step is to add an *activation* function. Usually, it could be Rectified Linear Unit (ReLU). ReLU transform function only activates a node if the input is above a certain quantity; the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship with the dependent variable. And the *Pooling* Layer is to shrink the image stack into a smaller size. If the size of data is still huge, the convolution and pooling steps are repeated. Finally, the *Fully-Connected* Layer (FC) is to flatten all the shrunken layers and stack up

them into one, normally through the Softmax function [107]. At this point in time, the training part for neural network is done, and further steps are to predict and check the working of the classifier.

1.6 Thesis Outline

To overcome the limitations of the traditional mathematical models, this thesis proposes a machine learning framework for online identification of the distribution network topology. The neural network is trained using μPMU measurements across the network—voltage, current magnitudes, and their phase angles—and achieves the real-time network topology with high accuracy even under noise and missing entries in PMU measurements. The measured data are rearranged into 2-D matrices (heatmaps), where the suggested CNN [69] takes them as the input. The performance of the proposed algorithm is tested and verified in a radial three-phase unbalanced distribution network.

The rest of the thesis is structured as follows: Chapter 2 provides a literature review on the topic and discusses the design of the proposed convolutional neural network framework. Chapter 3 implements the suggested approach on the IEEE 34-Node Test Feeder in MATLAB/Simulink platform and generates μPMU heatmaps under different system operating scenarios. Chapter 4 introduces the proposed CNN framework, and how it is implemented through Python. Section 5 presents the numerical studies and the network topology identification results in power distribution systems with noisy and missing measurements. Finally, the thesis is concluded in Chapter 6.

Chapter 2: Literature Review

2.1 Introduction

With the increasing growth of distributed energy resources in the power grid, more observability and controllability will be needed to accurately monitor the power flows and the unfolding conditions. In 1893, Charles Proteus Steinmetz presented a paper on simplified mathematical description of the waveforms of alternating current electricity. Steinmetz called his representation a phasor [109]. With the invention of PMU in 1988 by Dr. Arun G. Phadke and Dr. James S. Thorp at Virginia Tech, Steinmetz's technique of phasor calculation evolved into the calculation of real-time phasor measurements that are synchronized to an absolute time reference provided by the Global Positioning System (GPS). People therefore refer to synchronized phasor measurements as synchrophasors. Early prototypes of the PMU were built at Virginia Tech, and Macrodyne built the first PMU (model 1690) in 1992 [110].

In order to enhance the power grid resilience, grid operators need reliable and continuous monitoring of distributed energy resources (DERs). The concept of distribution automation was proposed to be used as the business-to-people (B2P) intelligent control between the power generation and consumer terminals, and help automatically realizing a sustainable grid operation [111].

On the other hand, as Figure 1.2 shows in Chapter 1, deep learning is a subset of AI and machine learning that uses multi-layered artificial neural networks to deliver state-of-the-art accuracy in tasks such as object detection, speech recognition, language translation and others. Deep

learning differs from traditional machine learning techniques in that they can automatically learn representations from data such as images, video or text, without introducing hand-coded rules or human domain knowledge. Their highly flexible architectures can learn directly from raw data and can increase their predictive accuracy when provided with more data.

Figure 2.1 is a two-dimension scheme for classification of deep learning based on recommender system; the left part illustrates the first dimension, and the right part illustrates the second dimension [4].

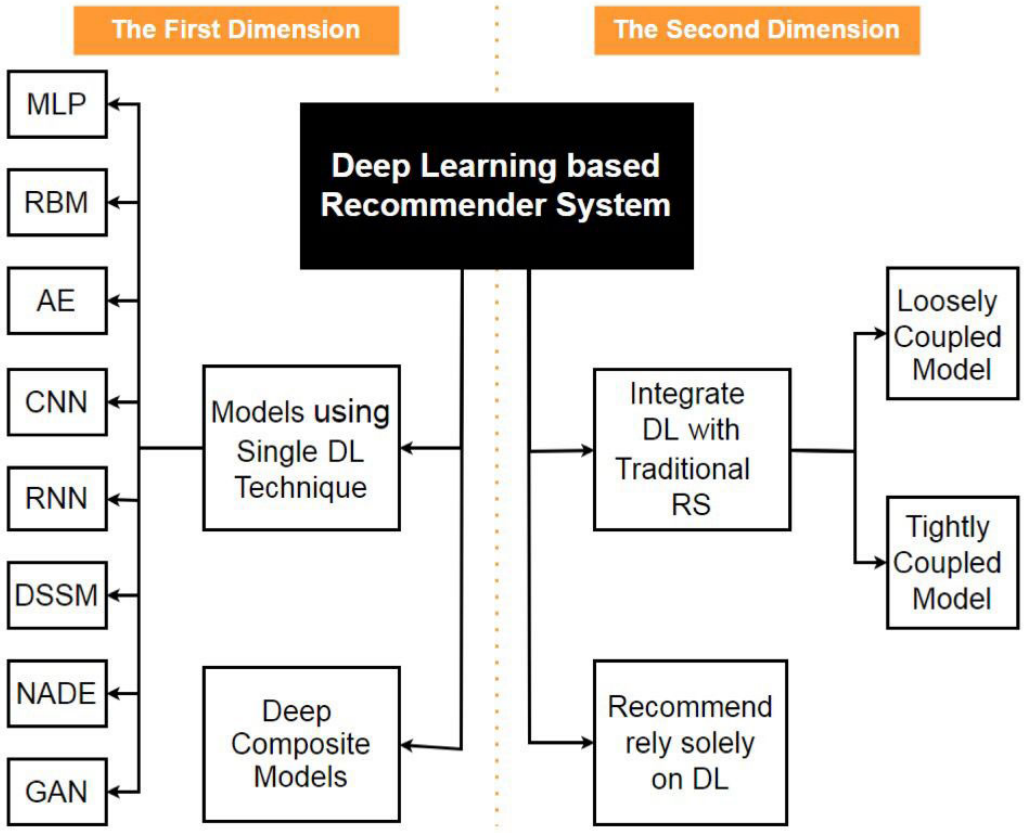


Figure 2.1: Two-dimension Deep Learning Framework based on Recommender System [4]

It is obvious that in the first dimension, Multilayer Perceptron (MLP), Restricted Boltzmann Machines (RBM), Autoencoder (AE), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Deep Structured Semantic Model (DSSM), Neural Autoregressive Distribution Estimation

(NADE), and Generative Adversarial Network (GAN) all belong to Deep Learning (DL) techniques.

Within the family of neural networks, and to train the model with a grid-like topology such as images, deep CNN has been one of the greatest breakthroughs [112, 113]. As it shows in Figure 2.2, the CNN structure consists of a convolutional layer, a pooling layer, and fully connected layers (FCs). When applied to single-label (multi-class) image classification, CNN can handle well-aligned images very well [114].

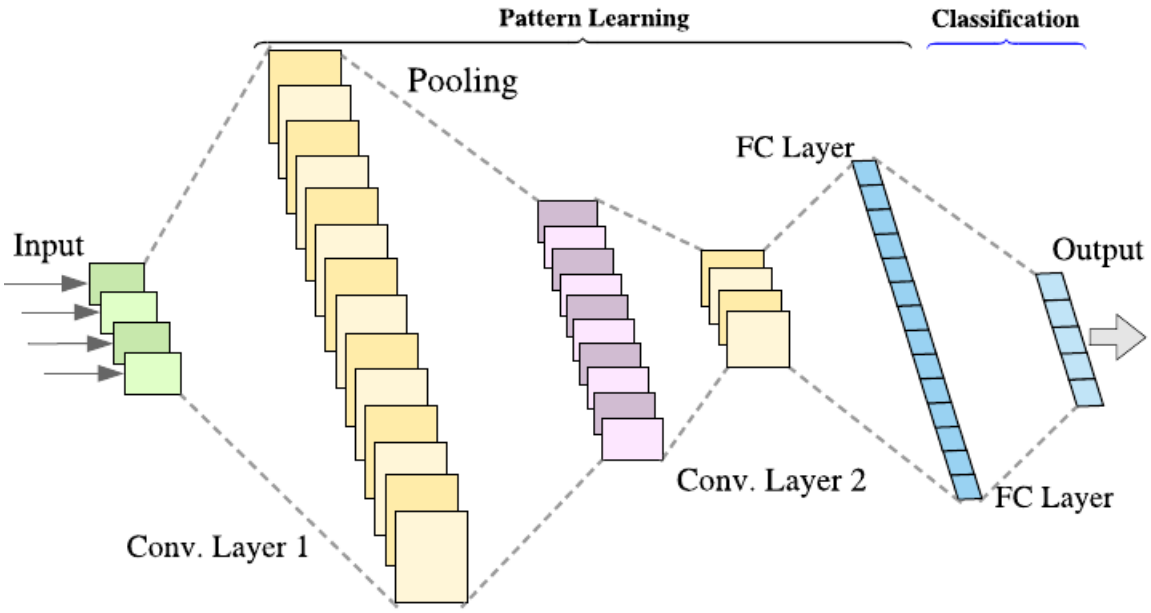


Figure 2.2: A Two-Convolution-Layer CNN Structure

By definition, CNNs are simply neural networks that use convolution in place of the FC layer in that least one of their layers [115]. In general, the implementation of the convolution is actually the cross-correlation assessment and defined by

$$s^p(m,n) = \sum_u \sum_v \sum_w I^u(m+v, n+w) K^p(v,w), \tag{2.1}$$

where $s^p(m,n)$ is the output of the convolutional layer at position (m,n) and p-th channel, $\boldsymbol{I}^u$ is the u-th channel of the data volume, and $\boldsymbol{K}^p$ is the p-th convolutional kernel. A nonlinear activation function is applied to the output of the convolution output, and the final activations of neurons in a convolutional layer are

$$\boldsymbol{I}_l = \sigma(\boldsymbol{s}), \tag{2.2}$$

where $\boldsymbol{I}_l$ represents the output volume of the l-th layer, and $\sigma(\cdot)$ represents the non-linearity of the neurons. By stacking the convolutional layers, the abstraction capacity of the network increases [116].

The representations (outputs) of the last convolutional (Conv) layer are expanded to vectors and processed by the general fully-connected layers, which transform the representations with more nonlinearities and into spaces with different (higher or lower) dimensions. The final layer of a CNN usually reduces the dimensionality of the representations to the number of the classes; cross-entropy [117] is then employed to measure the "goodness" of the classification (Kullback-Leibler divergence between the predicted distribution and the target distribution). Finally, gradients of the cross-entropy loss function with respect to the parameters in the CNN are used to train the CNN by back-propagation.

2.2 State-of-the-Art Research

The main challenge in DER monitoring at all times is that it is difficult to obtain the real-time grid topology. A variety of research methods have been proposed to identify the power system topology from synchrophasor measurements, and several methods of external network modeling were discussed to implement online security analyses [118]. Mathematically,

grid topology identification could be realized through voltage estimation across the grid [119], but the accuracy goes low when this method is applied on the topology of a frequency changing distribution grid or with limited μPMU sensors [120–123]. In [124], a Jacobian-based equivalent approach is used for detecting the electrical network topology changes in the external system. An approach which is a hybrid of power flow and state estimation is discussed in [125]. A method to capture the network topology changes based on an extended Ward equivalent is discussed in [126]. H. Singh, et al. There are also distribution network specific methods based on various assumption topologies, [127] introduced a technique that estimates the status of a suspect lines as part of the state estimation process. Focused on the transmission systems with inaccurate parameters, an offline REI (radial, equivalent, independent) equivalent [128] is suggested to be built from a base-case condition and to be updated using online data [129]. In order to improve the accuracy of the topology detection process, the problem of using telemetry data to correct and adjust the transmission parameters are considered in [130]. The network topology estimation accuracy could vary greatly depending on the information injections at the non-PMU buses. If the injections at the non-PMU buses are zero, the estimates will be the true equivalent at the PMU buses. In [131], a least-square "model-free" approach is proposed to estimate the equivalent power system topology by calculating the load variations with limited observation at each bus. In [132], a method for visualizing PMU data by reducing the system to an equivalent model at the PMU buses is discussed, which assumes the electrical network topology is known; an equivalent procedure is performed to reduce the network to a Ward type equivalent at the PMU buses. Another useful visualization technique has been done by biplots introduced in [133]. S. V. Wiel, et al. [96]

developed a greedy search algorithm to estimate the current topology of a power grid from phasor measurements. It studies the PMU placement at strategic points in a distribution system [134] to achieve a promising sensitivity to single-line outages. G. Cavraro, et al. [135] proposed a novel method for topology detection in distribution networks called the Time-Series Signature Verification for Topology Detection (TSV-Top). This approach relies on measurement time series from PMUs and performs the projection of actual voltage phasor patterns onto a library of signals associated with possible topology transitions of a given distribution network [136].

The above literature review revealed that most of the power grid topology identification and estimation tools are based on mathematical models, the majority of them assuming an electrical network topology first and then measure the collected data to compare the features and determine the accuracy of the previously assumed network topology. Such strategies are time-consuming, less accurate, and with practical limitations. On the contrary, there are more recent strategies leveraging machine learning advancements. Instead of accurately modeling the system, recent works have focused on training artificial neural networks to automatically recognize the electrical network topology and solve complex problems. In [137] and [138], two learning algorithms based on nodal voltage graphical models are introduced which can estimate the network topology under varying topological restrictions. D. Deka, et al. [139] developed a learning framework to reconstruct the radial operational structure of the distribution grid from synchronized voltage measurements across the network subject to the exogenous fluctuations in nodal power consumption. For the economic purposes, P. K. Ghosh, et al. [140] proposed a novel approach for complete system and fault observability using a minimum number of strategically-placed

PMUs. Reference [141] proposed a data-driven approach based on sensor measurements that identifies DERs' connectivity by converting topology identification problems to probability distance minimization problems via the Kullback-Leibler (KL) divergence metric.

2.3 Proposed Framework

In this thesis, the proposed CNN for the heatmap classification in the IEEE 34-Node Test Feeder has the following architecture: Input(33×12)–Conv(32, 5×3)–Conv(32, 3×3)–FC(100)–FC(9). Note that the axes of the input heatmap are with different units; Therefore, narrow kernel has been chosen in the first Conv layer which could cover the 3-phase data in each group, and the stride of the convolution operation in the first layer is (3, 2)—other Conv layers' strides are (1, 1). This design processes the data in each group first, then combines the information of each group in the second Conv layer and the FC layer. Batch normalization [142] is used in each Conv layer. Dropout [143] is adopted in the last Conv layer and the FC layer to prevent over-fitting. ReLU was chosen as the nonlinearities in the neural net.

The proposed framework for online power system topology identification is illustrated in Figure 2.3 . The μPMUs data is first used for offline training of the pre-built CNN model. The trained model is then used for online identification of the power distribution network topology.

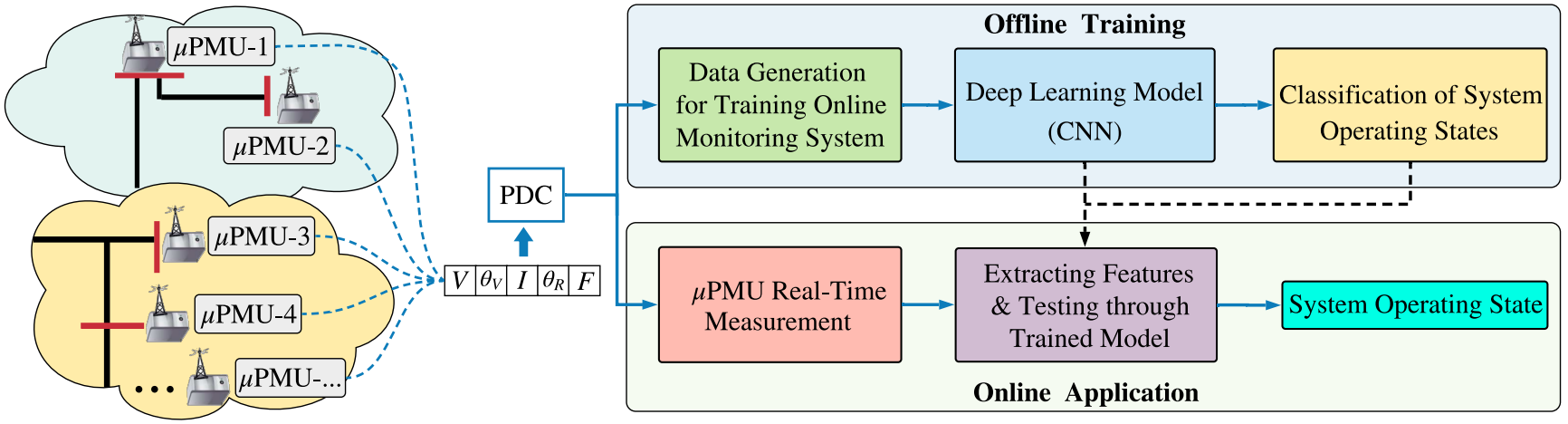


Figure 2.3: The proposed framework for power system online topology identification

Chapter 3: Simulation on MATLAB

3.1 Background

MATLAB is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages [144]. Simulink is a MATLAB-based graphical programming environment for modeling, simulating and analyzing multi-domain dynamical systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. It offers tight integration with the rest of the MATLAB environment and can either drive MATLAB or be scripted from it. Simulink is widely used in automatic control and digital signal processing for multi-domain simulations and model-based designs [145–147].

3.2 Scenario Generation

To run a model in MATLAB Simulink, we select the IEEE 34-Bus System as an example. This radial power distribution system is an actual feeder located in Arizona, the detailed information of which is provided in the Appendix A and its structure is illustrated in Figure A.1. 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

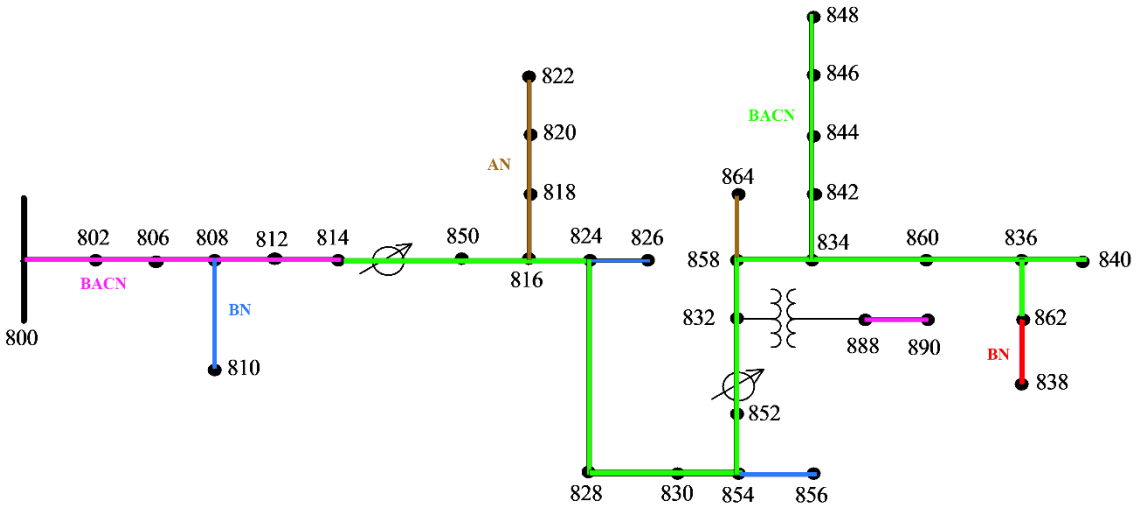


Figure 3.1: IEEE 34-Node Test Feeder with Colored Phasing.

(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

Except for the power generator at node 800, we can see there is one transformer located between node 832 and node 888. And there are two regulators located between node 814 to node 850, and between node 852 to node 832, respectively. From the information list above, this radial feeder contains unbalanced phases. In order to show it more intuitively, the next step is to mark branches with different status in different colors, as shown in Figure 3.1.

Different colors are here used to mark the phasing status, e.g., pink is used to mark the line 800-812 as BACN, meaning that it is a three-phase

four-wire segment in the distribution grid, while line 808-810 is one-phase two-wire segment. Finally, to generate several scenarios, I added some details into the structure of the network as shown in Figure 3.2.

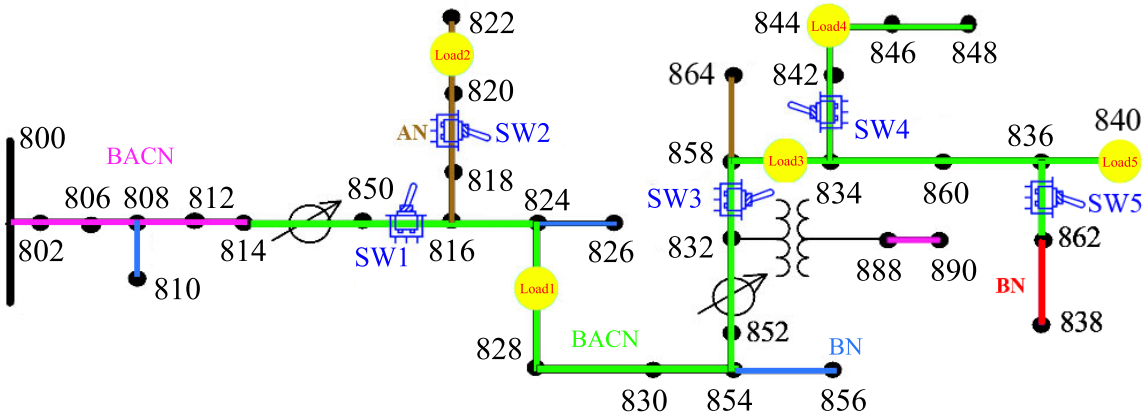


Figure 3.2: IEEE 34-Node Test Feeder Scenarios.

To gain a full observation of the IEEE 34-node test feeder as shown in Figure 3.2, I set 33 μPMUs on each node expect node 800 (substation bus). In order to make the scenario more clear, I use the Table 3.2 here to mark the specific parameters involved. Table 3.1 shows the node a μPMU is connected to and the measurement captured at the measurement points.

Table 3.1: Full observation μPMU Components

μPMU	802	820	832	848
(**Voltage** & angle, **Current** & angle)	806	822	888	860
(3×2, 3×2) if three-phase	808	824	890	836
(6, 6) if three-phase	810	826	858	840
	812	828	864	862
	814	830	834	838
	850	854	842	
	816	856	844	
	818	852	846	

As shown in Table 3.2, in order to generate different electrical network topologies for the CNN training dataset, I also added 5 breakers to be able

to change the structure of the feeder by switching the breaker on and off. Additionally, to generate more scenarios under one topology, here I marked 5 loads, so the μPMU data could be varying under different realizations of the load demand.

Table 3.2: Network Model Specifications and Variables

Breaker Name	**SW*1**	**SW2**	**SW3**	**SW4**	**SW5**
Breaker Location	850-816	818-820	832-858	834-842	836-862
Load Name	**Load1†**	**Load2†**	**Load3†**	**Load4**	**Load5**
Location	824-828	820-822	858-834	844	840

* SW: Breaker
† : Distributed load

In this table, Load 1, 824-828 means it is a distributed load; Load 4, 844 means it is a spot load. It is known that I can generate different μPMU data through adjusting the impedance of the loads. But in MATLAB Simulink, the block parameter of Three-Phase Series RLC Load does not own the "impedance" parameter label, the panel is shown below in Figure 3.3. Accordingly, I turned to adjust active power, inductive reactive power, and capacitive reactive power of the load instead.

Table 3.3 shows the result of the final generated scenarios. Take Topology 2 as an example, where SW 1 was 1, SW 2 and 3 were 0, which means when Breaker 850-816 was closed (status=1), Breaker 818-820 and Breaker 832-858 were opened (status=0), and the entire branch behind 858 was off-line. The red 0 stands for assuming all the rest be opened for reducing the unnecessary calculations. Because all four loads expect Load 1 were off-line, the marked 0 means it is meaningless to discuss the load change. Due to the fact that the network Topology 5 is too similar with Topology 4

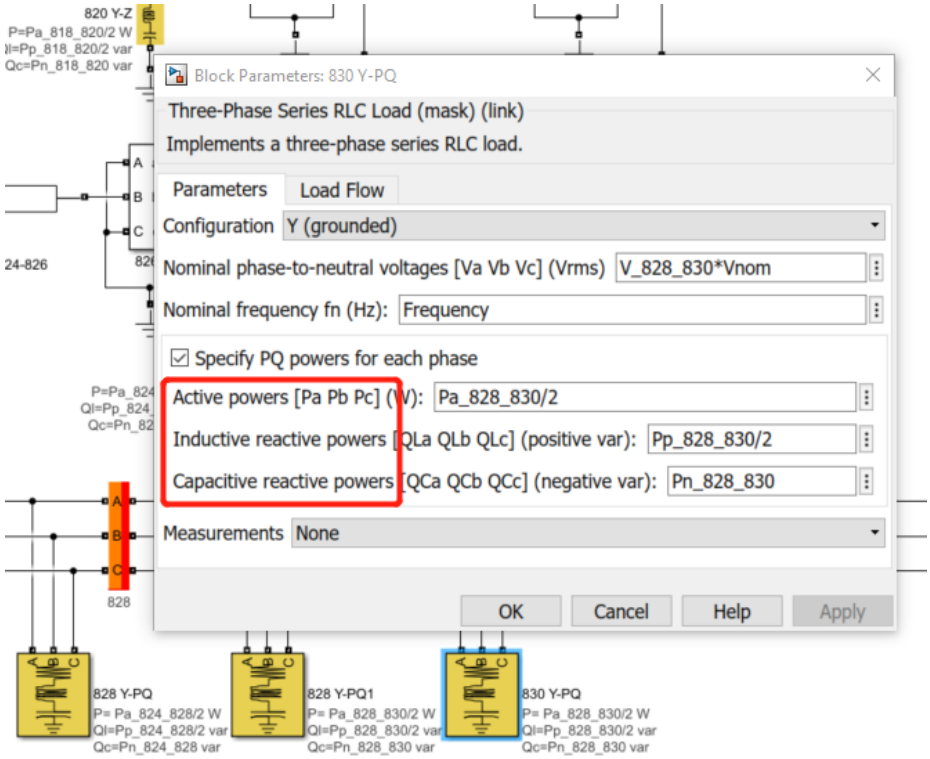


Figure 3.3: Three-Phase Series RLC Load Block Parameter Panel.

Table 3.3: Generated Network Topology Scenarios

Topology	SW1	SW2	SW3	SW4	SW5	Load1	Load2	Load3	Load4	Load5
1	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	1	0	0	0	0
3	1	1	0	0	0	1	1	0	0	0
4	1	0	1	0	0	1	0	1	0	1
5	1	0	1	0	1	1	0	1	0	1
6	1	0	1	1	0	1	0	1	1	1
7	1	0	1	1	1	1	0	1	1	1
8	1	1	1	0	0	1	1	1	0	1
9	1	1	1	0	1	1	1	1	0	1
10	1	1	1	1	0	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1

in terms of μPMU data, and Topology 1 and 2 could produce too little data which violates the balance in machine learning training data [148], keen considerations were taken in generating different load scenarios. Here we focus on 8 different topology scenarios that marked with gray blocks.

Table 3.4 shows the general operating status configuration of the 8

simulated network topologies, and the scenario number respectively.

Table 3.4: Network Topology Realizations with the Corresponding Number of Generated Scenarios

Topology	**SW1**	**SW2**	**SW3**	**SW4**	**SW5**	**Number of Scenarios**
1	1	1	0	0	0	1600
2	1	0	1	0	0	2197
3	1	0	1	1	0	2401
4	1	0	1	1	1	2401
5	1	1	1	0	0	2401
6	1	1	1	0	1	2401
7	1	1	1	1	0	3125
8	1	1	1	1	1	3125

Let the load change amplitude be uniformly distributed between 95% to 105% of the rated demand at each load point. In Topology 1, only Load 1 and Load 2 are served through the connected distribution line and it is not necessary to adjust the remaining three load points for scenario generation. Assuming each load has 40 possible amplitudes in the constrained range above, the total number of scenarios is found 40^2 in this case, i.e., 1600. Under the network Topology 8, all five loads are being served in the distribution grid, and as each one is characterized with 5 possible amplitudes for the training process, there are 5^5, i.e., 3125, number of scenarios generated. The total number of generated scenarios that contribute to the training dataset is found 19651.

Figure 3.4 shows all topologies that I used to to train the CNN. To summarize, I generated 19651 scenarios within 8 topologies, and there are 33 μPMUs in the whole test feeder. The Figure 3.5 is an example for a 33 by 12 matrix that was generated under one scenario; The program I used to generate μPMU data through MATLAB is attached in Appendix B.

As the screenshot of EXCEL shows above, columns A, B, and C stand

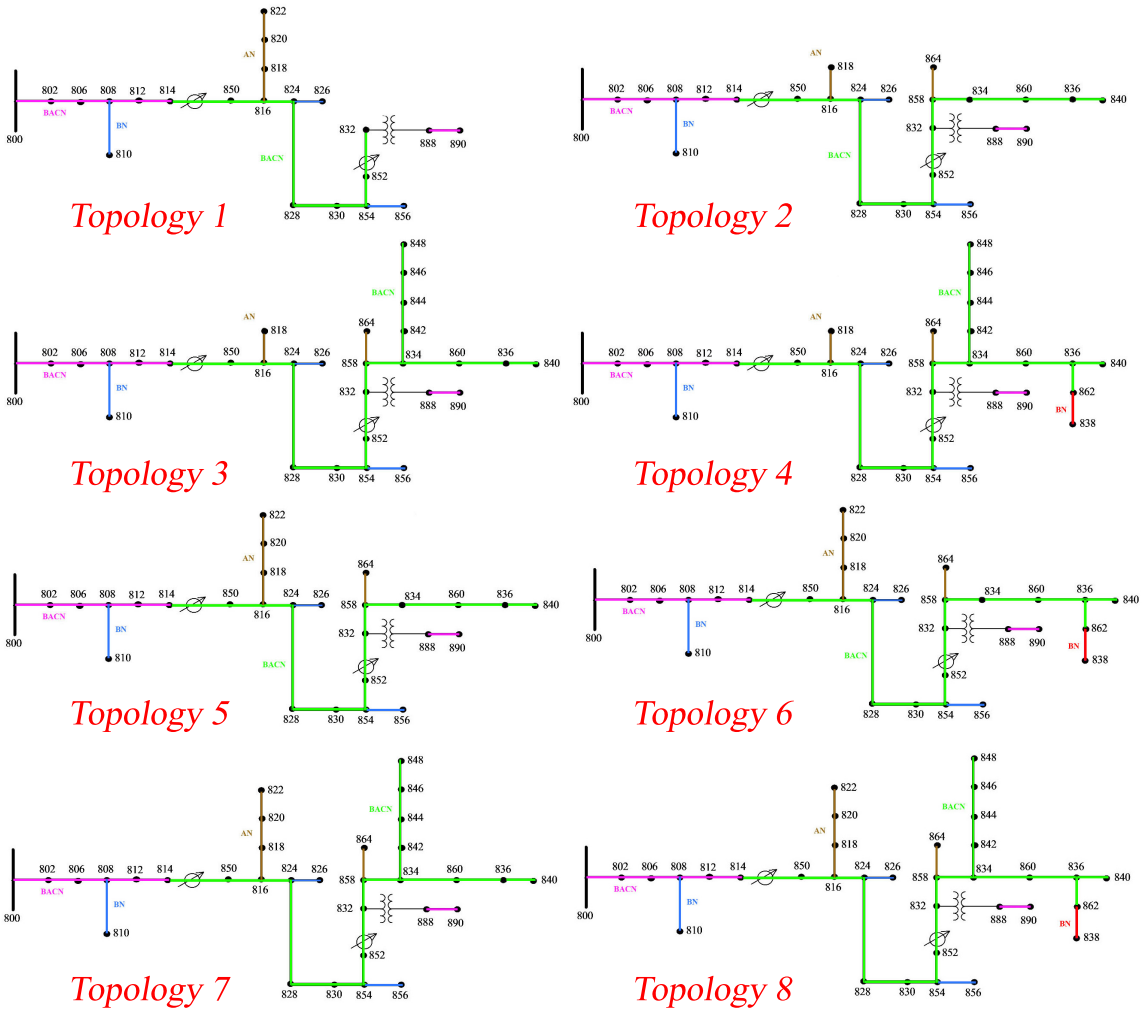
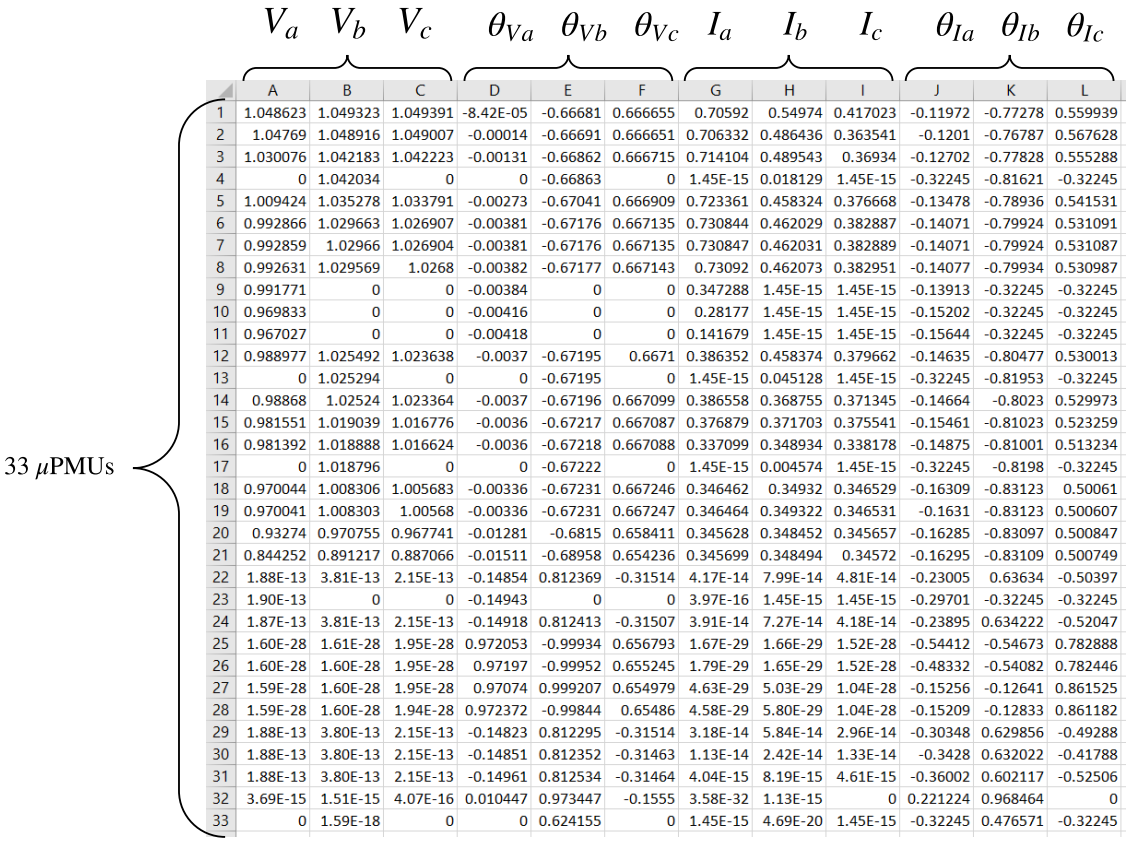


Figure 3.4: All Studied Network Topologies



	A	B	C	D	E	F	G	H	I	J	K	L
1	1.048623	1.049323	1.049391	-8.42E-05	-0.66681	0.666655	0.70592	0.54974	0.417023	-0.11972	-0.77278	0.559939
2	1.04769	1.048916	1.049007	-0.00014	-0.66691	0.666651	0.706332	0.486436	0.363541	-0.1201	-0.76787	0.567628
3	1.030076	1.042183	1.042223	-0.00131	-0.66862	0.666715	0.714104	0.489543	0.36934	-0.12702	-0.77828	0.555288
4	0	1.042034	0	0	-0.66863	0	1.45E-15	0.018129	1.45E-15	-0.32245	-0.81621	-0.32245
5	1.009424	1.035278	1.033791	-0.00273	-0.67041	0.666909	0.723361	0.458324	0.376668	-0.13478	-0.78936	0.541531
6	0.992866	1.029663	1.026907	-0.00381	-0.67176	0.667135	0.730844	0.462029	0.382887	-0.14071	-0.79924	0.531091
7	0.992859	1.02966	1.026904	-0.00381	-0.67176	0.667135	0.730847	0.462031	0.382889	-0.14071	-0.79924	0.531087
8	0.992631	1.029569	1.0268	-0.00382	-0.67177	0.667143	0.73092	0.462073	0.382951	-0.14077	-0.79934	0.530987
9	0.991771	0	0	-0.00384	0	0	0.347288	1.45E-15	1.45E-15	-0.13913	-0.32245	-0.32245
10	0.969833	0	0	-0.00416	0	0	0.28177	1.45E-15	1.45E-15	-0.15202	-0.32245	-0.32245
11	0.967027	0	0	-0.00418	0	0	0.141679	1.45E-15	1.45E-15	-0.15644	-0.32245	-0.32245
12	0.988977	1.025492	1.023638	-0.0037	-0.67195	0.6671	0.386352	0.458374	0.379662	-0.14635	-0.80477	0.530013
13	0	1.025294	0	0	-0.67195	0	1.45E-15	0.045128	1.45E-15	-0.32245	-0.81953	-0.32245
14	0.98868	1.02524	1.023364	-0.0037	-0.67196	0.667099	0.386558	0.368755	0.371345	-0.14664	-0.8023	0.529973
15	0.981551	1.019039	1.016776	-0.0036	-0.67217	0.667087	0.376879	0.371703	0.375541	-0.15461	-0.81023	0.523259
16	0.981392	1.018888	1.016624	-0.0036	-0.67218	0.667088	0.337099	0.348934	0.338178	-0.14875	-0.81001	0.513234
17	0	1.018796	0	0	-0.67222	0	1.45E-15	0.004574	1.45E-15	-0.32245	-0.8198	-0.32245
18	0.970044	1.008306	1.005683	-0.00336	-0.67231	0.667246	0.346462	0.34932	0.346529	-0.16309	-0.83123	0.50061
19	0.970041	1.008303	1.00568	-0.00336	-0.67231	0.667247	0.346464	0.349322	0.346531	-0.1631	-0.83123	0.500607
20	0.93274	0.970755	0.967741	-0.01281	-0.6815	0.658411	0.345628	0.348452	0.345657	-0.16285	-0.83097	0.500847
21	0.844252	0.891217	0.887066	-0.01511	-0.68958	0.654236	0.345699	0.348494	0.34572	-0.16295	-0.83109	0.500749
22	1.88E-13	3.81E-13	2.15E-13	-0.14854	0.812369	-0.31514	4.17E-14	7.99E-14	4.81E-14	-0.23005	0.63634	-0.50397
23	1.90E-13	0	0	-0.14943	0	0	3.97E-16	1.45E-15	1.45E-15	-0.29701	-0.32245	-0.32245
24	1.87E-13	3.81E-13	2.15E-13	-0.14918	0.812413	-0.31507	3.91E-14	7.27E-14	4.18E-14	-0.23895	0.634222	-0.52047
25	1.60E-28	1.61E-28	1.95E-28	0.972053	-0.99934	0.656793	1.67E-29	1.66E-29	1.52E-28	-0.54412	-0.54673	0.782888
26	1.60E-28	1.60E-28	1.95E-28	0.97197	-0.99952	0.655245	1.79E-29	1.65E-29	1.52E-28	-0.48332	-0.54082	0.782446
27	1.59E-28	1.60E-28	1.95E-28	0.97074	0.999207	0.654979	4.63E-29	5.03E-29	1.04E-28	-0.15256	-0.12641	0.861525
28	1.59E-28	1.60E-28	1.94E-28	0.972372	-0.99844	0.65486	4.58E-29	5.80E-29	1.04E-28	-0.15209	-0.12833	0.861182
29	1.88E-13	3.80E-13	2.15E-13	-0.14823	0.812295	-0.31514	3.18E-14	5.84E-14	2.96E-14	-0.30348	0.629856	-0.49288
30	1.88E-13	3.80E-13	2.15E-13	-0.14851	0.812352	-0.31463	1.13E-14	2.42E-14	1.33E-14	-0.3428	0.632022	-0.41788
31	1.88E-13	3.80E-13	2.15E-13	-0.14961	0.812534	-0.31464	4.04E-15	8.19E-15	4.61E-15	-0.36002	0.602117	-0.52506
32	3.69E-15	1.51E-15	4.07E-16	0.010447	0.973447	-0.1555	3.58E-32	1.13E-15	0	0.221224	0.968464	0
33	0	1.59E-18	0	0	0.624155	0	1.45E-15	4.69E-20	1.45E-15	-0.32245	0.476571	-0.32245

Figure 3.5: Example of a μPMU Data.

for the three-phase normalized voltage values (divided by nominal voltage), columns D, E, and F stand for the voltage phase angle radian values divided by π. And the last 6 columns G to L are the same as the previous 6 columns but the voltage variables are replaced with the current. The rows 1 to 33 represent the 33 μPMUs. So all numbers in this matrix is ranged between 1.05 to -1.05, which are suitable for convolution calculations.

3.3 Simulink

To run the entire test feeder in MATLAB, the most important step is to model it in Simulink. Figure 3.6 shows the entire structure of the 34-Node Test Feeder.

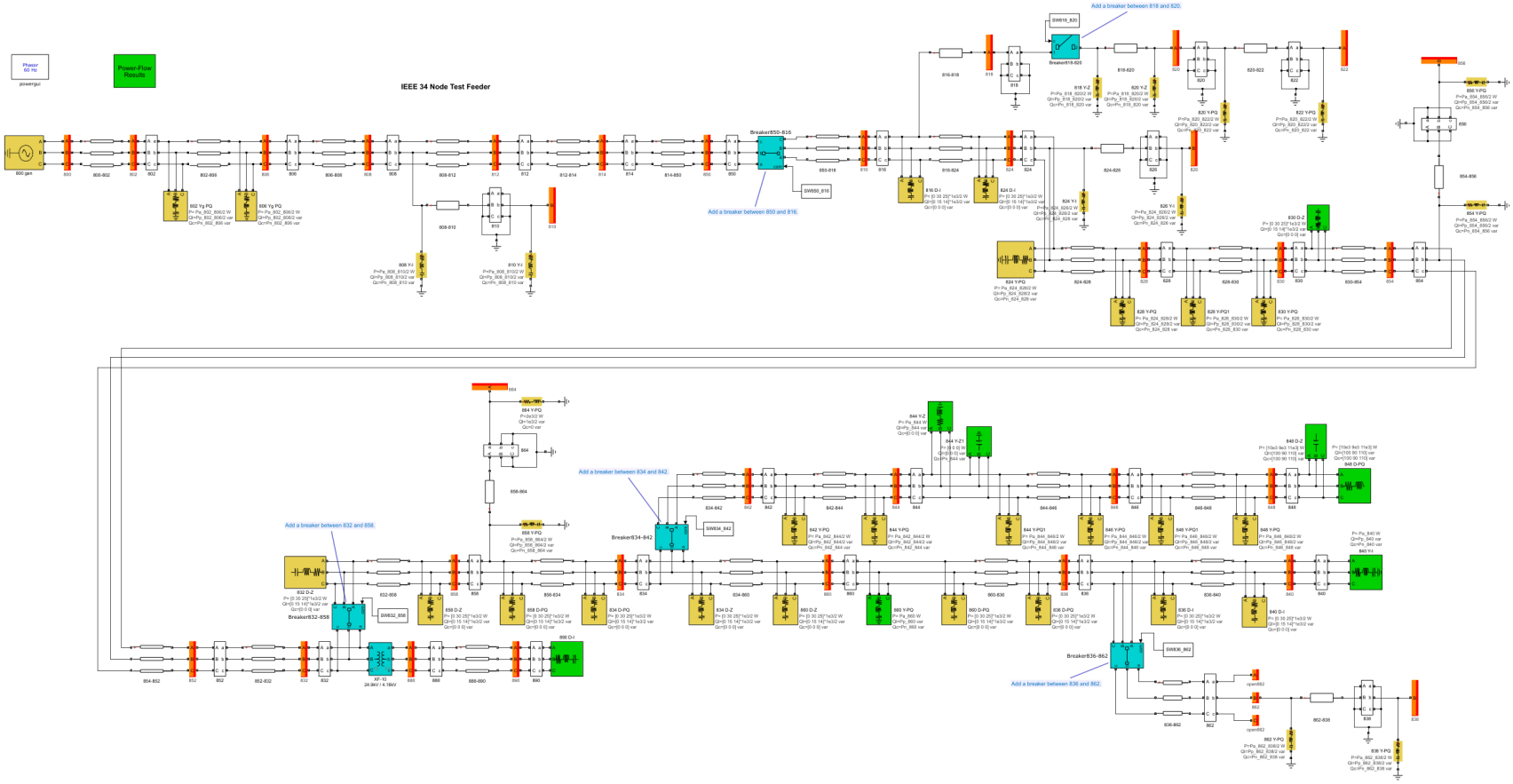


Figure 3.6: The structure of 34-Node Test Feeder in Simulink

3.3.1 Environment Block

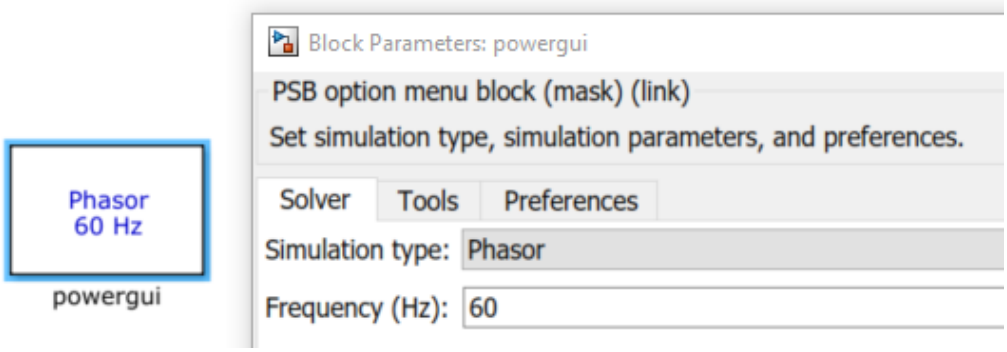


Figure 3.7: PSB option menu block

First, a "PSB option menu" block named "powergui" is added here, as Figure 3.7 shows. This block is an environment block for Simscape Electrical Specialized Power Systems models [149]. We choose the Simulation type as "Phasor", and set the Frequency to 60 Hz.

3.3.2 Power Source Block

From Appendix A, the feeder's nominal voltage is 24.9 kV, which means the phase-to-phase nominal voltage is 24900 V. We then add a Three-Phase Source block from Simscape toolbox as the main voltage source shown below.

We set the configuration as "Yg" which means it is Y-connected to the ground. And select the specified internal voltages for each phase. For line-to-neutral voltage, it should be the phase-to-phase nominal voltage divided by $\sqrt{3}$ and multiplied by an efficiency factor (EF).

$$V_{\text{line-to-neutral}} = \frac{V_{\text{phase-to-phase nominal}}}{\sqrt{3}} \cdot \text{EF} \tag{3.1}$$

For *short* lines, the efficiency factor is 1.05 meaning that there should be 5% used to compensate the voltage drop inside the winding when the

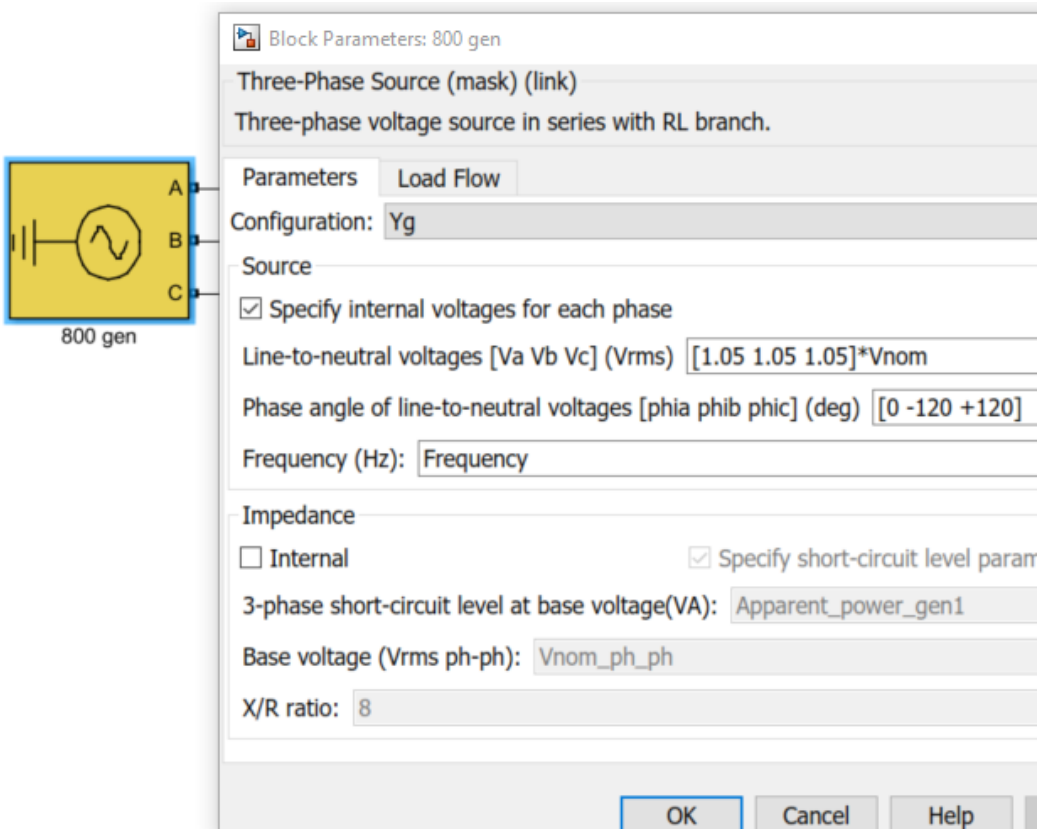


Figure 3.8: The Three-Phase Source block

transformer is fully loaded. And for *long* lines, the first-end voltage should be 10% higher than the rated voltage of the line; that means the efficiency factor is 1.1, because there should be another 5% used to compensate the voltage loss of the long line. The frequency is the same as the "powergui" as 60 Hz. And the phase angles of the line-to-neutral voltages is $0°$, $-120°$, and $120°$ because there is no lagging or leading for this test network. Second, we use the Load Flow Bus blocks to simulate spots, and use Three-Phase VI Measurement blocks to simulate μPMUs.

3.3.3 μPMU Block

As shown in Figure 3.9, μPMU 806 is connected to a three-phase distributed load; so the "connectors" parameter label in the Load Flow Bus block should be "ABC". The "Base voltage" is 24.9 kV/$\sqrt{3}$. And the "Swing bus or PV bus voltage" and the "Swing bus voltage angle" should be according to the reference in Appendix A.4, where the values are shown in Appendix A.4.2.

In the Three-Phase VI Measurement block, the "voltage measurement" should be "phase-to-ground". To measure phase-to-ground voltages in per unit, the block converts the measured voltages based on the peak value of the nominal phase-to-ground voltage as follows:

$$V_{\text{abc}}(\text{pu}) = \frac{V_{\text{phase-to-ground}}}{V_{\text{base}}}, \tag{3.2}$$

where

$$V_{\text{base}} = \frac{V_{\text{nom}}}{\sqrt{3}} \cdot \sqrt{2} \tag{3.3}$$

And I set the "Base power" here to 1.5e6 VA, so that the output could be

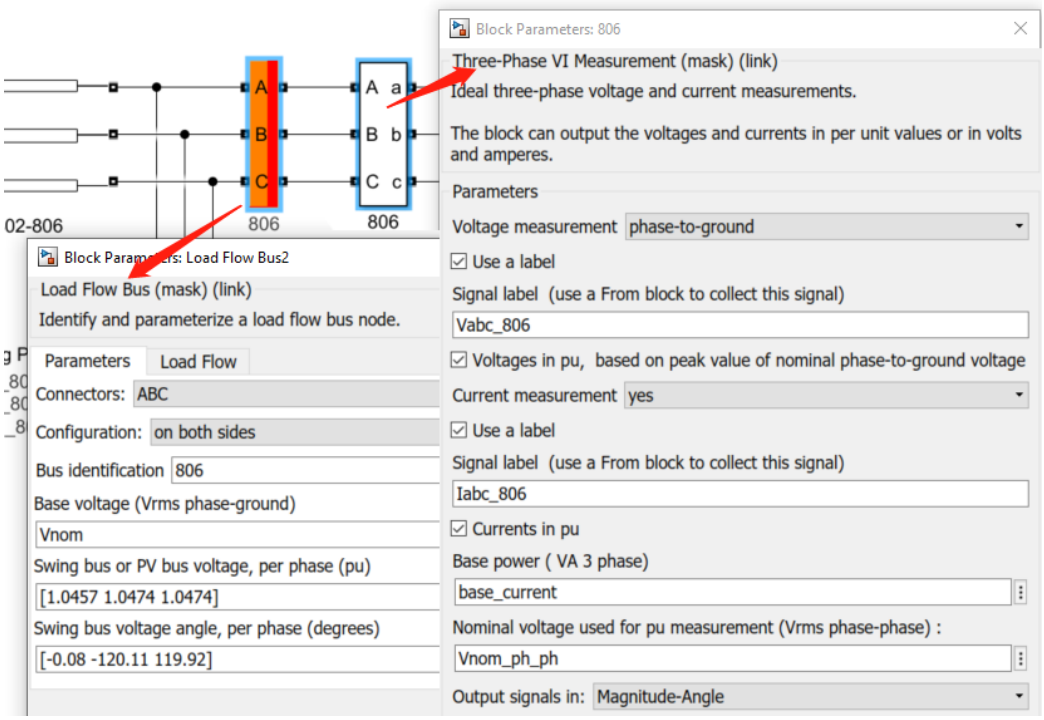


Figure 3.9: The μPMU Block in Simulink

converted into per-unit. The "Nominal voltage used for pu measurement" is 24.9 kV. And finally, I choose the "Output signals" as "Magnitude-Angle" to output the magnitudes and angles of the measured voltages and currents. Here I labeled each μPMU with voltage signal labels and current signal labels, so that all the results could be shown in "Data Inspector" after simulations.

3.3.4 Distribution Line Block

For distribution lines, we use the "Distributed Parameters Line" block from Simscape toolbox, as shown in Figure 3.10, to simulate the resistance of the distribution lines. As it shows, the distribution line 806-808 is a three-phase line, so the "Number of phases" should be 3. The "Frequency" should be the same as the main bus, 60 Hz. As for the "Resistance per unit length", "Inductance per unit length", and "Capacitance per unit length" blanks, they

should be 3 by 3 matrices for this three-phase segment. In "[r1 r0 r0m]", r1 means **positive resistances**, r0 means **zero-sequence resistances**, and r0m means **zero-sequence mutual resistances**. The rest symbols could be the same with "l" standing for inductance and "c" standing for capacitance.

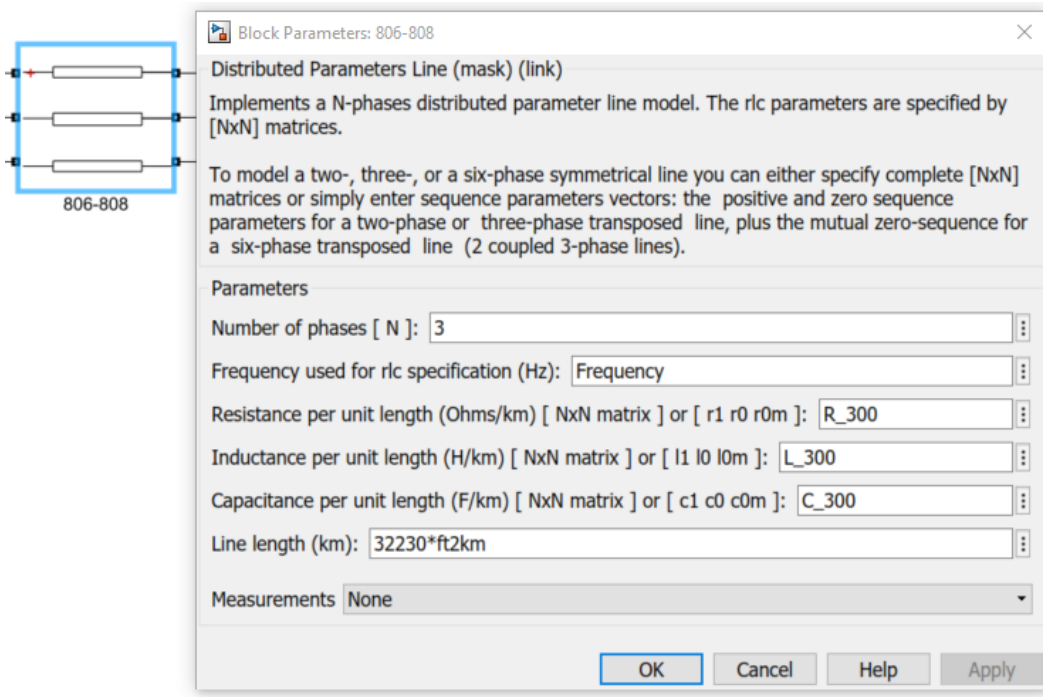


Figure 3.10: Distributed Parameter Line Block in Simulink

In Table A.2, the line 806-808 owns the "Config." of 300, and from Appendix A.3, it is known that for "300 Config." lines:

$$R\_300' = \begin{bmatrix} 1.3368 & 0.2101 & 0.2130 \\ 0.2101 & 1.3238 & 0.2066 \\ 0.2130 & 0.2066 & 1.3294 \end{bmatrix} \tag{3.4}$$

$$X\_300' = \begin{bmatrix} 1.3343 & 0.5779 & 0.5015 \\ 0.5779 & 1.3569 & 0.4591 \\ 0.5015 & 0.4591 & 1.3471 \end{bmatrix} \tag{3.5}$$

$$B\_300' = \begin{bmatrix} 5.3350 & -1.5313 & -0.9943 \\ -1.5313 & 5.0979 & -0.6212 \\ -0.9943 & -0.6212 & 4.8880 \end{bmatrix} \tag{3.6}$$

We set the conversion factors as tabulated in Table 3.5:

Table 3.5: Convert Factors

miles to kilometers	**mi2km**	1.60934
feet to kilometers	**ft2km**	0.0003048
microsiemens to Farads	**ms2F**	$(1e-6)/(2\cdot\pi\cdot 60)$

Finally, the three parameters in Figure 3.10 are converted to US customary units as follows:

$$R\_300 = \frac{R\_300'}{\text{mi2km}}, \tag{3.7}$$

$$L\_300 = \frac{X\_300'}{\text{mi2km}\cdot 2\cdot\pi\cdot 60}, \tag{3.8}$$

$$C\_300 = \frac{B\_300'}{\text{mi2km}\cdot\text{ms2F}} \tag{3.9}$$

3.3.5 Spot Load Block

There are 6 spot loads in IEEE 34-Node Test Feeder according to Table A.4, knowing that the load model for load 848 is "D-PQ", and $V\_848 = [1.0311.0291.031]\cdot\sqrt{3}$. As Figure 3.11 shows, I here use the "Three-Phase Series RLC Load" block to represent the spot loads.

For spot load 848, the "Active power" Pa_848 = [20 20 20] kW and "Inductive reactive power" Pp_848 = [16 16 16] kVAr are set. The "Capacitive reactive power"is set to Pn_848 = [150 150 150] kVAr according to Table A.6.

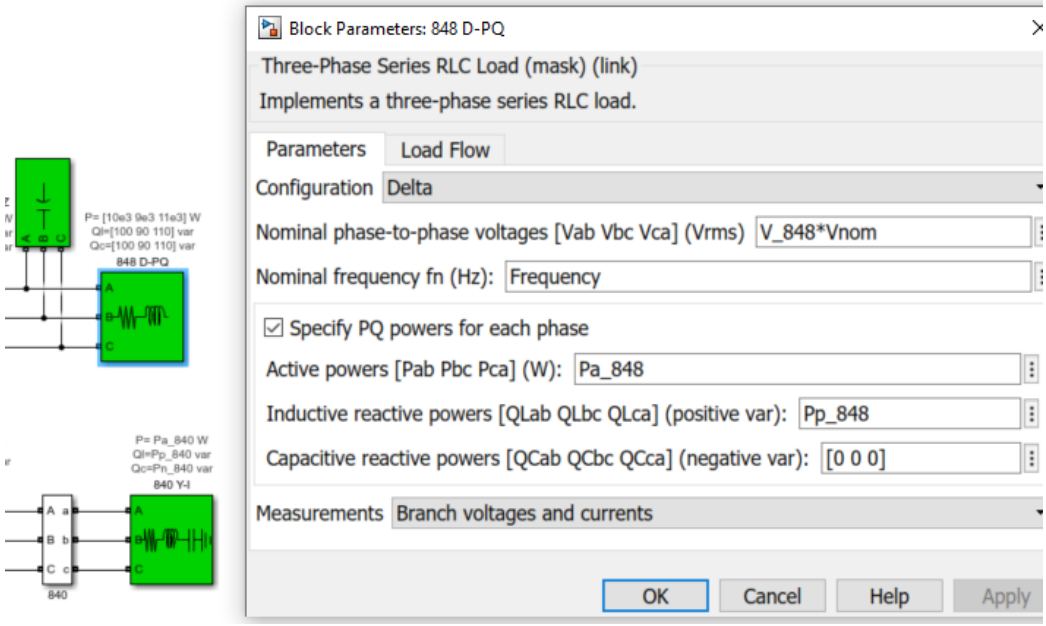


Figure 3.11: Spot Load Block in Simulink

Hence, the "Configuration" should be "Delta", and "Nominal phase-to-phase voltage" should be $V\_848 \cdot \frac{24.9}{\sqrt{3}}$ kV. The "Nominal frequency" remains the same, i.e., 60 Hz. For "Measurements", we select "Branch voltages and currents" to measure the three voltages and the three currents of the Three-Phase Series RLC Load block. And for "Load Flow", we choose "constant PQ", because "D-PQ" means the active power P and reactive power Q are kept constant and equal to the values specified on the Parameters tab of the block dialog box.

Notice that for some spot loads (e.g., load 844 and load 848 according to Table A.6 in IEEE 34-Node Test Feeder), the shunt capacitor is separated, so the parameter "Capacitive reactive power" should be set in another RLC load block with Pa_848 and Pa_848 be [0 0 0], and "Nominal phase-to-phase voltage" be $[111] \cdot \frac{24.9}{\sqrt{3}}$ kV, and "Measurements" be "None", and "Load Flow" be "constant Z".

3.3.6 Distributed Load Block

There are 19 distributed loads in IEEE 34-Node Test Feeder according to Table A.5, knowing that the load model for load 828-830 is "Y-PQ", and $V\_828\text{-}830 = [1.007\ 1.015\ 1.011]$ kV according to Appendix A.4.4. The V_828-830 should be multiplied by $\sqrt{3}$ if it is Delta-connected. As Figure 3.12 shows, we here use the "Three-Phase Series RLC Load" block to represent the distributed loads.

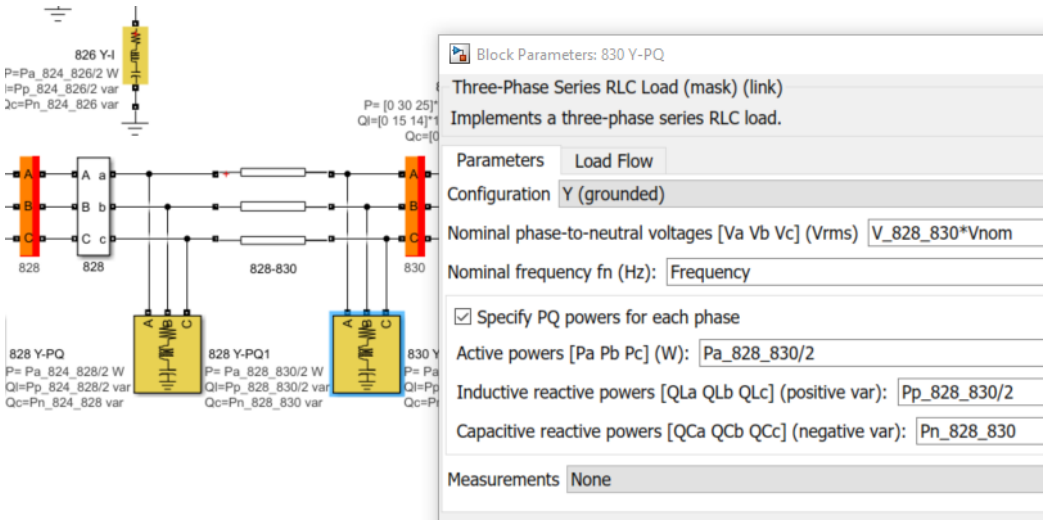


Figure 3.12: Distributed Load Block in Simulink

In order to balance the distributed load, we here let the load 828-830 be divided into 2 equivalent loads and hung on both sides of the distribution line 828-830. For one equivalent load, the "Configuration" is "Y (grounded)" because it is Y-connected. The "Nominal phase-to-phase voltage" is $V\_828\text{-}830 \cdot \frac{24.9}{\sqrt{3}}$ kV. The "Nominal frequency" is the same as 60 Hz. For distributed load 828-830, the "Active power" Pa_828-830 = [7 0 0] kW, "Inductive reactive power" Pp_828-830 = [3 0 0] kVAr, and "Capacitive reactive power" Pn_828-830 = [0 0 0] kVAr are set, so the Pp_828-830 does not need to be divided by 2. But in order to avoid errors in the simulation process, it is better to set all zeros into 1e-6. We set the "Measurements" be

"None", and "Load Flow" be "constant PQ".

3.3.7 Subsystem Block

As shown in Figure 3.6, the green square marked "Power-Flow Results" in the upper left corner collects all the data measured by the μPMUs of this test feeder, that is the Subsystem block.

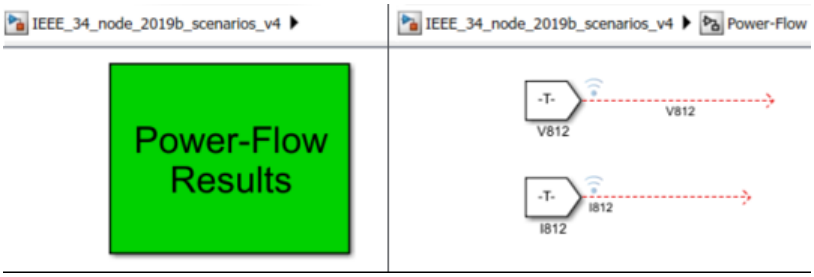


Figure 3.13: Results Subsystem Block in Simulink

The right hand side of Figure 3.13 shows a part of this Subsystem representing which types of μPMU data are collected. For example, for μPMU 812, the needed data are the voltage and currents; so two "From" blocks are linked with "V812" and "I812" where the labels in Figure 3.9 present the voltage and current captured by μPMU 812.

Chapter 4: Machine Learning with Python

4.1 Introduction

For the general loopy power grids, there may be multiple paths between two nodes. If power distribution grids are featured with minimum cycle length greater than three, the nodal voltages are sufficient for efficient topology estimation without additional assumptions on the system parameters. In contrast, the detection of line failures or status change using nodal voltages does not require any structural assumption on the network [137, 138]. As for the case of the IEEE 34-Node Test Feeder, the minimum cycle length in a radial graph is considered to be infinite as it has no cycles by definition. Hence, using nodal μPMU measurements, such as voltage and current phasors, real-time network topology estimation on the IEEE 34-node feeder system is effectively viable.

The problem of detecting the network topology change can recast as a classification problem based on the heatmaps which are obtained by the μPMUs measurements. The conventional classification approaches often involve manually designed features like thresholds and signatures in each scenario. However, these approaches require the human expertise and the type of topology it can detect would be limited. E.g., a threshold may be suitable for a certain topology, but if one node becomes offline in the electrical network, the threshold would not work anymore. This thesis proposed an artificial neural network platform that can learn the features (representations) of the data automatically.

4.2 The Proposed Framework

As shown in Figure 2.3, the μPMUs data collected from each bus in the distribution network is first used for offline training of the pre-built CNN model. The trained model is then used for online identification of the power distribution network topology. A flowchart of the proposed CNN is shown in Figure 4.1, where the first step is to collect μPMU data and normalize them into per-units. Such data with their corresponding topologies are then inputted into the neural network, and the trained network learns to identify distribution grid topology with μPMU measurements. The CNN used cross-entropy as the loss function. Finally, additional μPMU measurements beyond the training set were used to verify the model accuracy.

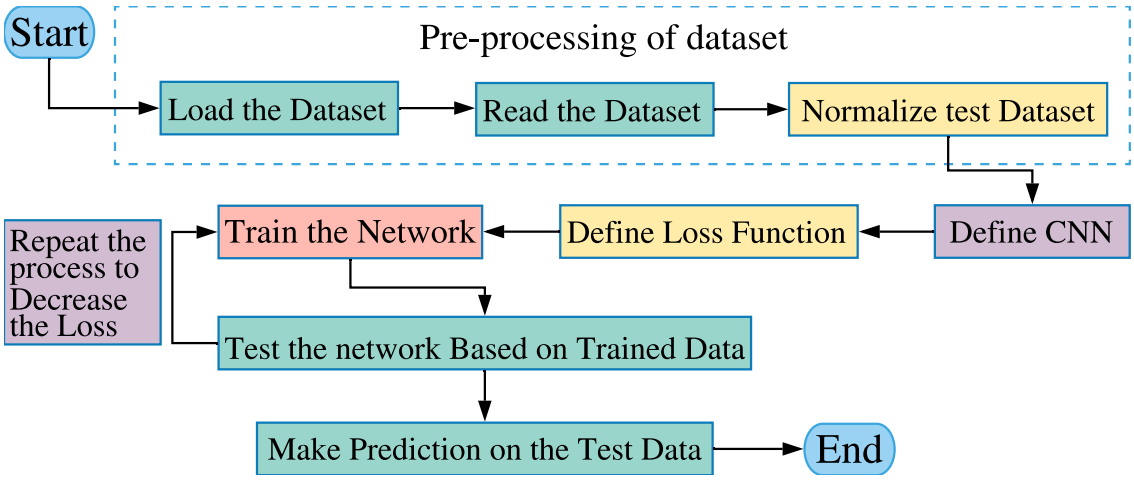


Figure 4.1: Proposed CNN Working Flowchart [5]

This CNN architecture will be used as a building block in the proposed framework that identifies the power distribution network topology in real-time. Upon simulating each scenario in MATLAB Simulink environment, the resulting μPMU data will characterize a 33 by 12 heatmap matrix which contains three-phase voltage, three-phase voltage angle, three-phase current, and three-phase current angles. For the nodal measurements that

contain single-phase or two-phase data, we let the remaining entries be zero. Then we line up the data in a heatmap format into four groups, process the data in each group individually, and finally integrate the information in each group together. A partially connected neural network is dedicated to the processing of these groups of heatmaps. In practice, a partially connected neural network is equivalent to a CNN. I design a CNN by carefully selecting the kernels in the first layer. As mentioned before, simulation of each scenario results in a heatmap (see Figure 4.2 for a heatmap example); these heatmaps are used as the inputs to the proposed CNN.

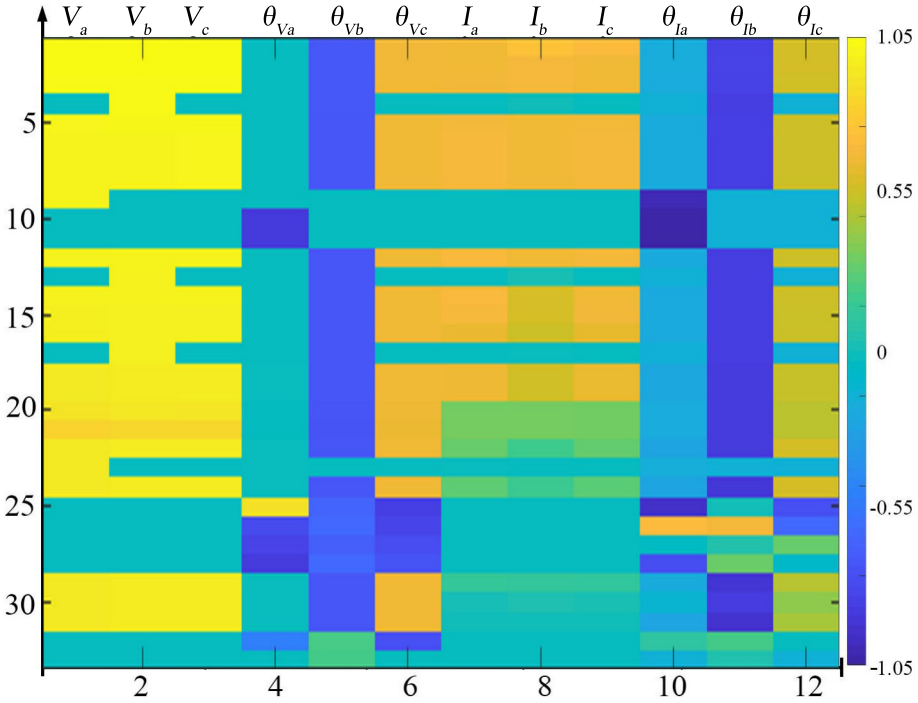


Figure 4.2: A Heatmap Example of the Generated μPMU Measurement Data Sample.

Comparing with the Figure 3.5, columns V_a, V_b and V_c stand for the three-phase voltages; columns θ_{V_a}, θ_{V_b} and θ_{V_c} stand for the three-phase voltage angles; columns I_a, I_b and I_c stand for the three-phase currents; columns θ_{I_a}, θ_{I_b} and θ_{I_c} stand for the three-phase current angles.

4.3 Programming in Pycharm

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains [150]. The program used for CNN framework is available in Appendix C. CUDA is a parallel computing platform and application programming interface model created by Nvidia [151]. It allows software developers and software engineers to use a CUDA-enabled graphics processing unit for general purpose processing – an approach termed GPGPU [152]. We here used the "Cuda" function in the main program to accelerate the calculation operations. Due to the fact that the size of the IEEE 34-Node Test Feeder is not that huge, we here used the two layer neural network, as in Figure 2.2, to train the framework (see the program in Appendix C.3). The proposed framework also contains a new function that accounts for loading the heatmap data from the training dataset in Appendix C.4. The program in Appendix C.1 sorts the μPMU data into three folders, "Train", "Test", and "Val", under all the scenarios mentioned in Section 3.2. After calculating the mean and standard deviation through the program in Appendix C.2, the training dataset were loaded into the neural network. The "optimizer" is here selected to be "Adam" [153] and the activation function is ReLU.

Chapter 5: Numerical Case Study

5.1 Experiments Outline

As Figure 2.3 shows, the proposed deep learning framework is consisted of three parts. The first part is gathering μPMU data, the second part is training the CNN offline, and the third part is to identify the power grid topology by inputting real-time data into the trained neural network. In order to verify the practicality of the proposed framework, we here designed 5 experiments to verify the reliability of the framework:

(i) Train a neural network based on the proposed deep learning framework under full network observation, where the training data were collected from each and every node in the grid, and test the accuracy of the power grid topology identification using this network under the scenarios in the presence of interfered data, e.g., noisy data or missing data.

(ii) Train three neural networks based on the proposed deep learning framework under full network observation with interfered training data, test the accuracy of power grid topology estimation using the previously trained networks under the scenarios of inputting abnormal data beyond the training dataset.

(iii) Train several neural networks based on the proposed deep learning framework under partially observable network conditions with fewer number of installed μPMU sensors, in which the input heatmaps in the training dataset would miss one or more rows of data. Then test the accuracy of power grid topology identification using the previously trained networks.

(iv) For randomly missing μPMU measurements, train several neural networks based on the proposed deep learning framework under the sensor-rich scenarios, which is partial observation with two-third sensors, and with interfered training data. Then test the accuracy of power grid topology identification using the previously trained networks.

(v) For economic purposes, train several neural networks based on the proposed deep learning framework under the sensor-less scenario, with partial observability achieved through one-third sensors, in which the missing input data were manually selected. Then test the accuracy of power grid topology identification using the previously trained networks, find the minimum number of the μPMU sensor that could efficiently observe the whole grid.

5.2 Data Generation and Preprocessing

In the case of IEEE 34-Node Test Feeder, the obtained data are 33 by 12 heatmaps which stand for 33 μPMUs data points (rows) and are consisted of the voltage, current, and phase angle information (columns) from the μPMU measurements. These μPMU measurements are obtained under **full observations** of all nodes in the power network as mentioned in Section 3.2 and generated in the MATLAB environment. The entire test feeder was built in Simulink, and all block parameters were set according to the data provided in Appendix A [6].

5.2.1 Parallel Simulation

In order to ease the simulation complexity and computational burden, we here used a technique of parallel simulation operation in Simulink [154],

which could run simulations of multiple scenarios simultaneously. As the progeam shows in Appendix B.2, the "parsim" function is used for parallel operations. The entry "simu_number=5" means that 5 cores of the computer CPU (central processing unit) are assigned as "worker", so that the system could run 5 scenarios in each simulation time.

5.2.2 Data Classification

After the μPMU measurements were captured, all scenarios should be classified into three folders: training dataset, testing dataset, and validation dataset, where each folder contains 8 topologies as presented in Table 3.3. In order to facilitate the CNN computation, the columns of the heatmaps (such as per-unit voltage and current values) were normalized through a zero-mean and unity variance distribution. We randomly separate 80% of the total simulation outputs as the training dataset, 10% for testing dataset, and 10% for validation dataset.

5.3 Results Analysis

5.3.1 Full Network Observation

First, a full observation in the network is studied wehre the data is collected at all 33 μPMU data in the test feeder as shown in Table 3.1. In order to test the performance of the proposed framework, the experiments were conducted that are closer to the realistic situations. In the first group experiments, different interferences were applied in the dataset. The interferences include both the noise and the missing data. The accuracy of the proposed topology identification framework in the conducted experiments is shown in Table 5.1. Wherein, one epoch of training means every sample

in the training dataset is used in the training of the CNN once, and the SNR refers to the Signal-to-Noise Ratio. One can see that *as the number of training epochs increases, the accuracy also increases*.

The "Validation Accuracy" in Table 5.1 corresponds to the condition when the neural network was trained by the *training dataset*, and the accuracy of identifying the electrical network topology is assessed using the *validation dataset* which is included in the training dataset; and the "Prediction Accuracy" refers to the condition when the neural network was trained by the testing dataset, and the accuracy of the network topology identification is assessed using the *testing dataset* which is excluded in the training dataset.

5.3.1.1 Topology Identification Analysis

Table 5.1: The identification accuracy of the interfered dataset under full observation

Test Scenarios	Interference SNR (dB)	Number of Epochs	Best Validation Accuracy (%)	Best Prediction Accuracy (%)
Full Measurement	10	5	98.81	98.72
Full Measurement	10	10	98.81	98.81
Full Measurement	10	20	99.86	99.95
Full Measurement	20	20	100	100
Full Measurement	20-50*	20	100	100
Missing One Data	-	20	100	100
Missing Two Data	-	20	99.91	99.95
Missing One Data	20	20	100	99.95
Missing Two Data	20	20	100	99.95
Missing One Data	10	20	99.95	99.82
Missing Two Data	10	20	98.86	98.99
Missing Two Data	20-50*	20	100	100

*: the intensity of SNR is randomly selected in the range and applied on each data sample.

As one can see, the accuracy of the proposed electrical network topology identification scheme was never found lower than 95%. This is because for one epoch training, all the testing data were included in the training dataset.

For example, for full measurement containing 10dB SNR, and when the number of epochs is 20, the neural network was trained by the training dataset that contained 10dB SNR, leading to an identification accuracy of 99.9%. Hence, the *prediction accuracy* cloud be achieved high as long as the neural network was trained well. In this situation, the prediction is actually called identification, because all scenarios are already known, taking advantage of a full observation. Additionally, when all measurements are available and the SNR is greater then 20dB, the proposed CNN can identify the system topology very accurately (the smaller the SNR, the greater the noise). Moreover, for well trained neural network, *the more missing data in the training dataset, the greater the positive impact on the accuracy of the prediction engine*.

5.3.1.2 Prediction Analysis

In the second group of experiments, the training and testing data are interfered with at different levels. The three "Training Data" in the first row represent three CNNs, which were trained by applying the datasets (i) containing 10dB SNR, (ii) containing one missing data with 10dB SNR, and (iii) containing two missing data with 10dB SNR respectively. The first column contains three situations which means the models are individually tested each with 800 samples (i.e., 100 μPMU data were generated for each network topology) but with 40dB SNR. The electrical network topology identification accuracy is shown in Table 5.2.

Note that in order to conduct the tests closer to the realistic situations, when generating the training dataset, the data are interfered by taking out the missing entries first, then adding noise; On the contrary, when testing the model, the data were added with noise first and then the missing entries

Table 5.2: The identification accuracy (%) by training and testing the CNN in different extents of interferences

Training Data / **Test Data**	10dB SNR	Missing One Data and 10dB SNR	Missing Two Data and 10dB SNR
40dB SNR	71.88	80.75	94.00
Missing One Data 40dB SNR	72.75	80.63	93.62
Missing Two Data 40dB SNR	72.13	82.13	94.50

were studied. The test data were beyond the training dataset, which means that the neural network identifies the system topology by estimating from the unknown inputs. For example, for μPMU data which contained 40dB SNR, using the network which was trained well by the dataset containing 10dB SNR to estimate the system topology will result in an overall identification accuracy of 71.88%.

All these 9 cases achieve the accuracy greater than 70%. One can see that using the same testing data, *if the training data has imperfections* such as missing, and/or outlier values, interferences, but under a certain level, *the trained neural network can provide more accurate results*. This is because the imperfections or complications in the training dataset can make the neural network become more versatile, and thus, the trained network would perform better. In all, the trained CNN under the greatest level of interference achieves a satisfactory topology identification accuracy, implying that the proposed CNN has a very good capacity of generalizing the trained data to unseen inputs.

5.3.2 Missing μPMUs Observation

Table 5.1 shows that the proposed neural network framework could work well under the presence of missing data and noises. Hence, the next step

is to verify if the framework could also work well under missing μPMUs instead of individual data.

5.3.2.1 Missing A Few μPMU Sensors

For missing one μPMU, that means a whole row of data in Figure 3.5 is missing, we here replaced the entries with zero to represent such cases. Table 5.3 shows the identification accuracy under missing different numbers of μPMU.

Table 5.3: The identification accuracy of the interfered dataset under missing several μPMUs observations

Test Scenarios	**Number of Epochs**	**Best Validation Accuracy (%)**	**Best Prediction Accuracy (%)**
32 μPMU Sensors	20	98.13	98.44
31 μPMUs Sensors	20	99.68	99.67
30 μPMUs Sensors	20	99.91	99.86
29 μPMUs Sensors*	20	98.02	97.94
28 μPMUs Sensors*	20	99.32	99.31
27 μPMUs Sensors*	20	99.22	98.99
22 μPMUs Sensors*	20	98.36	98.31

*: the zeroed rows were randomly generated so there may be duplicates.

It can be seen from the first 3 rows of the table above that as the number of missing μPMUs increased, the identification accuracy has also increased, which is in line with the conclusion obtained by Section 5.3.1.

From the case "Missing 4 μPMUs", the zeroed rows were randomly generated; Since it could be duplicate μPMUs, the accuracy observation has a small fluctuation from "Missing 3" to "Missing 4". But the trend from "Missing 4" to "Missing 5" is also rising. The accuracy of the proposed framework under "Missing 6" drops a bit; it may be due to the error in the random training of the neural network or the loss of more than a certain range of μPMUs. Hence, it can be seen that the accuracy of topology identification would not be sacrificed by missing a few μPMU sensors in the grid.

5.3.2.2 Partial Observability with Two-Third of μPMU Sensors

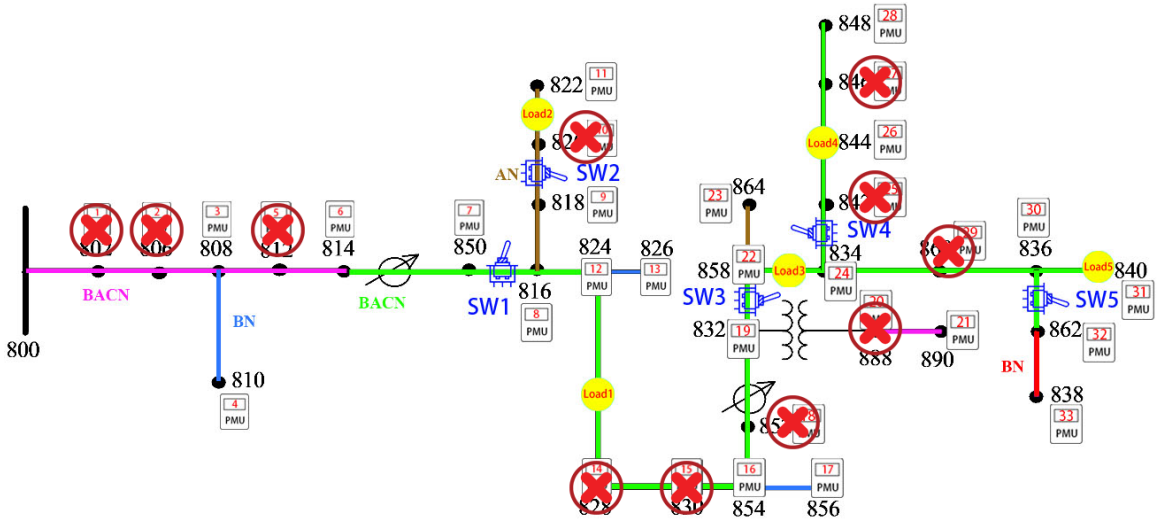


Figure 5.1: The topology observation with 22 μPMU sensors

From Table 5.1, it can be seen that the accuracy of the topology identification and estimation is high under full network observation. Although a comprehensive observation can give a very accurate forecast, it will undoubtedly increase the cost of the electrical equipment (investment costs of PMUs) and maintenance. It is therefore necessary to employ and study the least number of μPMUs to observe the entire electrical network. Since the full observation needs 33 μPMUs across the network (one sensor at each node), we here use 22 μPMUs, which are the two-third of μPMUs in the network, to observe the entire test feeder, and see how accurate the proposed analytics are under such circumstances.

The *locations* of the μPMUs cannot be randomly selected as in Section 5.3.2. The criterion for μPMU removal is by checking whether it is a duplicate μPMU on a same branch with no topology change. The remaining 22 μPMUs are shown in Table 5.4. Comparing with Table 5.1, Table 5.5 below shows the identification accuracy of different cases under the availability

Table 5.4: 22 μPMU Components

μ**PMU**	808	822	890	836
(**Voltage** & angle, **Current** & angle)	810	824	858	840
(3×2, 3×2) if three-phase	814	826	864	862
(6, 6) if three-phase	850	854	834	838
	816	856	844	
	818	832	848	

of two-third of the μPMUs in the network.

Table 5.5: The identification accuracy of the interfered dataset under the two-third μPMUs observation

Test Scenarios	**Interference SNR (dB)**	**Number of Epochs**	**Best Validation Accuracy (%)**	**Best Prediction Accuracy (%)**
Two-Third Measurement	-	20	100	100
Two-Third Measurement	10	20	100	100
Missing One Data	-	20	100	100
Missing Two Data	-	20	100	99.95
Missing One Data	20	20	100	100
Missing Two Data	20	20	100	99.86
Missing One Data	10	20	100	100
Missing Two Data	10	20	99.82	99.73

It is obvious that the validation accuracy and prediction accuracy are both achieved the same high of full observation under the two-third observation, which means the neural network framework could also *work accurately under the two-third observation*. Because the μPMUs are reduced randomly, this experimental conclusion can save one-third of the cost on the premise of ensuring the accuracy of power grid topology identification.

5.3.2.3 Partial Observability with One-Third μPMUs Sensors

Since the two-third observation could work accurately, we here observe the power grid with even fewer number of μPMUs. Figure 5.2 below shows a scenario in which 12 number of μPMUs are used to observe the entire network. Similarly, the locations of the μPMUs are specifically selected.

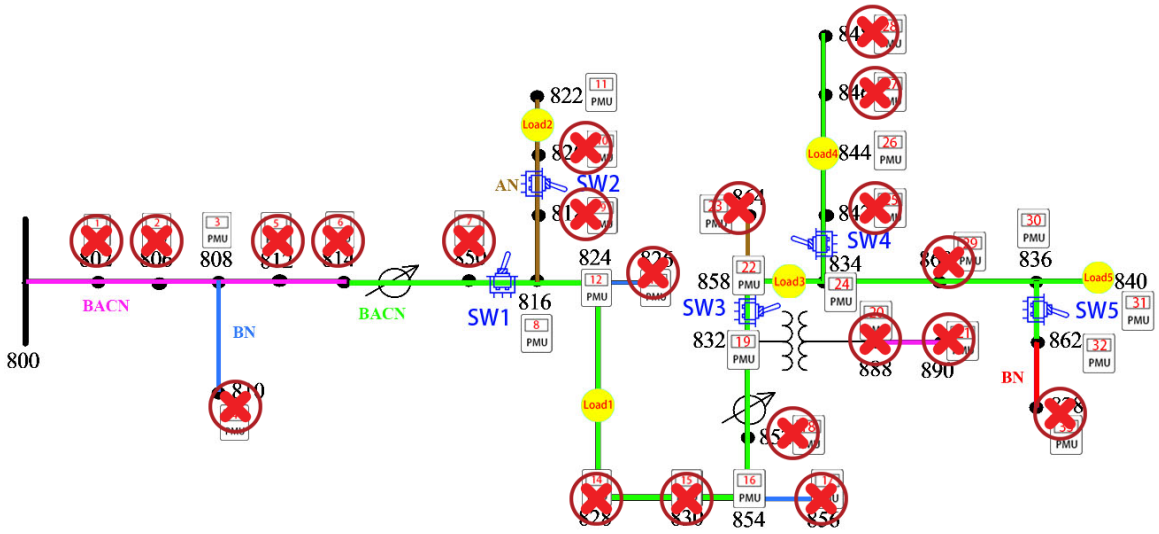


Figure 5.2: The topology observation with 12 μPMU sensors

Table 5.6: 12 μPMU Components

μPMU	808	822	844
(**Voltage** & angle, **Current** & angle)	816	832	836
(3×2, 3×2) if three-phase	822	858	840
(6, 6) if three-phase	824	834	862

Table 5.7: Different numbers of μPMU Observation

Number / μPMU	12	11	10	9	8	7	6	5
808	√	√	√	√	√	√	√	√
816	√	√	√	√	√	√	√	√
822	√	√	√	√	√	√	√	√
824	√	√						
854	√							
832	√	√	√					
858	√	√	√	√	√			
834	√	√	√	√	√	√		
844	√	√	√	√	√	√	√	√
836	√	√	√	√				
840	√	√	√	√	√	√	√	√
862	√	√	√	√	√	√	√	
Best Validation Accuracy (%)	100	100	98.72	94.11	88.82	88.99	98.45	54.52
Best Prediction Accuracy (%)	100	100	98.99	93.96	88.59	89.01	98.63	54.49

√: be picked to observe the power grid.

The accuracy of the proposed topology identification algorithm with 12 μPMUs is found still high and promising. Table 5.7 summarizes several conditions and the corresponding accuracy. The conclusion that can be drawn is under the situation of 5 breakers, placing at least 6 μPMUs in suitable locations can ensure the accuracy of the observation of the entire power grid. The accuracy of case "5 μPMUs" has significantly reduced because of the loss of μPMU 862, which is the only one that could observe the breaking of "SW 5".

The overall observation is that *as the number of μPMUs decreases, the topology identification accuracy continues to decline*. The accuracy of "6 μPMUs" is achieved high which is most likely because the 5 breakers divided the test feeder into 6 zones, and the 6 manually selected μPMU positions are corresponded to 6 zones respectively. So the neural network can easily

distinguish their "0-1" situations. For manual selection of μPMU installation locations, one needs to ensure that there is at least one μPMU sensor on each independent power grid sub-branch, so that the actual *minimum number of μPMUs* installed is equal to the *number of the sub-branch plus one*. Hence, a full observation in the network will be achieved, enabling to harness the measurements for effective topology identification in real-time.

Chapter 6: Conclusion

6.1 Summary of Current Work

This thesis presents a deep learning framework for online detection of power distribution system topology. The proposed framework can handle missing measurements under unbalanced operating states in power distribution systems, and real-time topology identification (for within known database), and estimation (for the beyond known database) of the system topology following disturbances. The proposed framework utilizes phasor measurements from μPMUs at all buses (full observation) and a number of selected buses (partial observation). For *estimation* (prediction), the approach is manually adding several μPMU datasets that contain missing entries or noise or both, and test them with a trained neural network.

The experiments show that the proposed CNN framework not only handles the data with the same level of interference (noise and missing measurements), but also has the capacity of estimating the interfered data which has different distributions from the training examples. Numerical experiments proved that the proposed trained network can almost accurately identify the power network topology corresponding to the observed data beyond the training dataset. For random loss of μPMUs, the availability of μPMUs at two-thirds of the network buses can guarantee around 98% accuracy in the topology identification; For specifically generated scenarios, the number of μPMU that can promise the high identification accuracy should be equal to the number of branches that can be independent.

6.2 Future Research

Future work could be targeted at implementing the proposed framework on a larger real-world power grid, such as the IEEE 123-bus test system, and validating the results accuracy and computational effectiveness during real-time applications. Moreover, the performance of the proposed analytics in power grids with high penetration of renewable energy resources and energy storage technologies should be investigated [155–175]. Moreover, the role and performance of the proposed solutions in an integrated ecosystem of critical infrastructures (e.g., transportation, water, communication, etc) should be analyzed [176].

Additionally, the CNN optimizer was selected as "Adam" [153], which is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update the network weights iteratively based on the training data. There are also many other algorithms that could be tested and analyzed. Each algorithm owned its advantages and shortcomings. Hence, follow-up research could try to use different algorithms and compare their performance in the network topology identification application. Additionally, this thesis mainly used CNN by training the PMU datasets to predict the power system topology, where the Python library in this thesis was Pytorch. There are some other libraries like TensorFlow and Keras that could be explored. Besides CNN, there are also many other ML algorithms, such as Support Vector Machine (SVM), Autoencoder (AE), and Capsule Neural Network (CapsNet), that could be approached in this application and under a variety of operating conditions in the power grid.

Bibliography

[1] M. Fotuhi-Friuzabad, A. Safdarian, M. Moeini-Aghtaie, R. Ghorani, M. Rastegar, and H. Farzin, "Upcoming challenges of future electric power systems: sustainability and resiliency," *Scientia Iranica*, vol. 23, no. 4, pp. 1565–1577, 2016.

[2] "Deep learning|nvidia developer," NVIDIA, [Online] Available at: https://developer.nvidia.com/deep-learning, Accessed:2020.

[3] S. B. Driss, M. Soua, R. Kachouri, and M. Akil, "A comparison study between mlp and convolutional neural network models for character recognition," in *Real-Time Image and Video Processing 2017*, vol. 10223. International Society for Optics and Photonics, 2017, p. 1022306.

[4] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[5] A. Rao, "Convolutional neural network tutorial (cnn) – developing an image classifier in python using tensorflow," edureka.co, [Online] Available at: https://www.edureka.co/blog/convolutional-neural-network, Accessed:2020.

[6] "34-bus feeder case," IEEE PES, [Online] Available at: https://site.ieee.org/pes-testfeeders/resources, Accessed:2010.

[7] A. Razi-Kazemi and P. Dehghanian, "A practical approach to optimal RTU placement in power distribution systems incorporating fuzzy sets theory," *International Journal of Electrical Power and Energy Systems*, vol. 37, no. 1, pp. 31–42, 2012.

[8] P. Dehghanian, A. Razi-Kazemi, and M. Fotuhi-Firuzabad, "Optimal RTU placement in power distribution systems using a novel method based on analytical hierarchical process (AHP)," in *The 10th International IEEE Conference on Environmental and Electrical Engineering (EEEIC)*, 2011, pp. 1–6.

[9] M. Moeini-Aghtaie, P. Dehghanian, and S. H. Hosseini, "Optimal distributed generation placement in a restructured environment via a multi-objective optimization approach," in *16th Conference on Electric Power Distribution Networks (EPDC)*, 2011, pp. 1–6.

[10] A. Razi-Kazemi, P. Dehghanian, and G. Karami, "A probabilistic approach for remote terminal unit placement in power distribution systems," in *The 33rd IEEE International Telecommunications Energy Conference (INTELEC)*, 2011, pp. 1–6.

[11] P. Dehghanian, A. Razi-Kazemi, and G. Karami, "Incorporating experts knowledge in RTU placement procedure using fuzzy sets theory-a practical approach," in *The 33rd IEEE International Telecommunications Energy Conference (INTELEC)*, 2011, pp. 1–6.

[12] M. Shojaei, V. Rastegar-Moghaddam, A. Razi-Kazemi, P. Dehghanian, and G. Karami, "A new look on the automation of medium voltage substations in power distribution systems," in *17th Conference on Electric Power Distribution Networks (EPDC)*, 2012, pp. 1–6.

[13] T. Becejac and P. Dehghanian, "PMU multilevel end-to-end testing to assess synchrophasor measurements during faults," *IEEE Power and Energy Technology Systems Journal*, vol. 6, no. 1, pp. 71–80, March 2019.

[14] S. Wang, P. Dehghanian, and B. Zhang, "A data-driven algorithm for online power grid topology change identification with PMUs," in *IEEE Power and Energy Society (PES) General Meeting*, 2019, pp. 1–5.

[15] S. Wang, P. Dehghanian, and Y. Gu, "A novel multi-resolution wavelet transform for online power grid waveform classification," in *The 1st IEEE International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA)*, 2019, pp. 1–6.

[16] M. Kezunovic, A. Esmaeilian, T. Becejac, P. Dehghanian, and C. Qian, "Life-cycle management tools for synchrophasor systems: Why we need them and what they should entail," in *The 2016 IFAC CIGRE/CIRED Workshop on Control of Transmission and Distribution Smart Grids*. CIGRE, 2016, pp. 1–6.

[17] T. Becejac, P. Dehghanian, and M. Kezunovic, "Analysis of PMU algorithm errors during fault transients and out-of-step disturbances," in *IEEE Power and Energy Society (PES) Transmission & Distribution (T&D) Conference and Exposition Latin America*, 2016, pp. 1–6.

[18] ——, "Probabilistic assessment of PMU integrity for planning of periodic maintenance and testing," in *International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, 2016, pp. 1–6.

[19] ——, "Impact of PMU errors on the synchrophasor-based fault location algorithms," in *48th North American Power Symposium (NAPS)*, 2016, pp. 1–6.

[20] M. Kezunovic, P. Dehghanian, and J. Sztipanovits, "An incremental system-of-systems integration modelling of cyber-physical electric power systems," in *Grid of the Future Symposium, CIGRE US National Committee*. CIGRE, 2016, pp. 1–6.

[21] M. H. Rezaeian Koochi, P. Dehghanian, S. Esmaeili, P. Dehghanian, and S. Wang, "A synchrophasor-based decision tree approach for identification of most coherent generating units," in *The 44th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2018, pp. 1–6.

[22] S. Wang, P. Dehghanian, and L. Li, "Power grid online surveillance through PMU-embedded convolutional neural networks," *IEEE Transactions on Industry Applications*, vol. 56, no. 2, pp. 1146–1155, March/April 2020.

[23] S. Wang, L. Li, and P. Dehghanian, "Power grid online surveillance through PMU-embedded convolutional neural networks," in *IEEE Industry Applications Society (IAS) Annual Meeting*, 2019, pp. 1–7.

[24] A. Von Meier, D. Culler, A. McEachern, and R. Arghandeh, *Micro-synchrophasors for distribution systems*. IEEE, 2014.

[25] M. N. Albasrawi, N. Jarus, K. A. Joshi, and S. S. Sarvestani, "Analysis of reliability and resilience for smart grids," in *2014 IEEE 38th Annual Computer Software and Applications Conference*. IEEE, 2014, pp. 529–534.

[26] T. Nguyen, S. Wang, M. Alhazmi, M. Nazemi, A. Estebsari, and P. Dehghanian, "Electric power grid resilience to cyber adversaries: State of the art," *IEEE Access*, vol. 8, pp. 87 592–87 608, 2020.

[27] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[28] P. Dehghanian, Y. Guan, and M. Kezunovic, "Real-time life-cycle assessment of high voltage circuit breakers for maintenance using online condition monitoring data," *IEEE Transactions on Industry Applications*, vol. 55, no. 2, pp. 1135–1146, 2019.

[29] P. Dehghanian, M. Kezunovic, G. Gurrala, and Y. Guan, "Security-based circuit breaker maintenance management," in *IEEE Power and Energy Society (PES) General Meeting*, 2013, pp. 1–5.

[30] Y. Guan, M. Kezunovic, P. Dehghanian, and G. Gurrala, "Assessing circuit breaker life cycle using condition-based data," in *IEEE Power and Energy Society (PES) General Meeting*, 2013, pp. 1–5.

[31] P. Dehghanian and M. Kezunovic, "Cost/benefit analysis for circuit breaker maintenance planning and scheduling," in *The 45th North American Power Symposium (NAPS)*, 2013, pp. 1–6.

[32] P. Dehghanian, Y. Guan, and M. Kezunovic, "Real-time life-cycle assessment of circuit breakers for maintenance using online condition monitoring data," in *IEEE/IAS 54th Industrial and Commercial Power Systems (I&CPS) Technical Conference*, 2018, pp. 1–8.

[33] P. Dehghanian, T. Popovic, and M. Kezunovic, "Circuit breaker operational health assessment via condition monitoring data," in *The 46th North American Power Symposium*, 2014, pp. 1–6.

[34] P. Dehghanian, M. Fotuhi-Firuzabad, F. Aminifar, and R. Billinton, "A comprehensive scheme for reliability centered maintenance implementation in power distribution systems- part I: Methodology," *IEEE Transactions on Power Delivery*, vol. 28, no. 2, pp. 761–770, 2013.

[35] P. Dehghanian, M. Fotuhi-Firuzabad, S. Bagheri-Shoraki, and A. Razi-Kazemi, "Critical component identification in reliability centered asset management of distribution power systems via fuzzy ahp," *IEEE Systems Journal*, vol. 6, no. 4, pp. 593–602, 2012.

[36] P. Dehghanian, M. Fotuhi-Firuzabad, and A. Razi-Kazemi, "An approach for critical component identification in reliability-centered maintenance of power distribution systems based on analytical hierarchical process," in *The 21st International Conference and Exhibition on Electricity Distribution (CIRED)*, 2011, pp. 1–4.

[37] P. Dehghanian and M. Fotuhi-Firuzabad, "A reliability-oriented outlook on the critical components of power distribution systems," in *The 9th IET International Conference on Advances in Power System Control, Operation, and Management (APSCOM)*, 2011, pp. 1–6.

[38] P. Dehghanian, M. Moeini-Aghtaie, M. Fotuhi-Firuzabad, and R. Billinton, "A practical application of the delphi method in maintenance-targeted resource allocation of distribution utilities," in *The 13th International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, 2014, pp. 1–6.

[39] F. Pourahmadi, M. Fotuhi-Firuzabad, and P. Dehghanian, "Identification of critical components in power systems: A game theory application," in *IEEE Industry Application Society (IAS) Annual Meeting*, 2016, pp. 1–6.

[40] S. Bahrami, M. Rastegar, and P. Dehghanian, "An fbwm-topsis approach to identify critical feeders for reliability centered maintenance in power distribution systems," *IEEE Systems Journal*, pp. 1–10, 2020.

[41] S. Moradi, V. Vahidinasab, M. Kia, and P. Dehghanian, "A mathematical framework for reliability-centered asset management implementation in microgrids," *International Transactions on Electrical Energy Systems*, 2018.

[42] H. Mirsaeedi, A. Fereidunian, S. M. Mohammadi-Hosseininejad, P. Dehghanian, and H. Lesani, "Long-term maintenance scheduling and budgeting in electricity distribution systems equipped with automatic switches," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 1909–1919, 2018.

[43] M. Asghari Gharakheili, M. Fotuhi-Firuzabad, and P. Dehghanian, "A new multi-attribute support tool for identifying critical components in power transmission systems," *IEEE Systems Journal*, vol. 12, no. 1, pp. 316–327, 2018.

[44] F. Pourahmadi, M. Fotuhi-Firuzabad, and P. Dehghanian, "Application of game theory in reliability centered maintenance of electric power systems," *IEEE Transactions on Industry Applications*, vol. 53, no. 2, pp. 936–946, 2017.

[45] ——, "Identification of critical generating units for maintenance: A game theory approach," *IET Generation, Transmission & Distribution*, vol. 10, no. 12, pp. 2942–2952, 2016.

[46] H. Sabouhi, A. Abbaspour, M. Fotuhi-Firuzabad, and P. Dehghanian, "Identifying critical components of combined cycle power plants for implementation of reliability centered maintenance," *IEEE CSEE Journal of Power and Energy Systems*, vol. 2, no. 2, pp. 87–97, 2016.

[47] ——, "Reliability modeling and availability analysis of combined cycle power plants," *International Journal of Electrical Power and Energy Systems*, vol. 79, pp. 108–119, 2016.

[48] R. Ghorani, M. Fotuhi-Firuzabad, P. Dehghanian, and W. Li, "Identifying critical component for reliability centered maintenance management of deregulated power systems," *IET Generation, Transmission, and Distribution*, vol. 9, no. 9, pp. 828–837, 2015.

[49] P. Dehghanian, M. Fotuhi-Firuzabad, F. Aminifar, and R. Billinton, "A comprehensive scheme for reliability centered maintenance implementation in power distribution systems- part II: Numerical analysis," *IEEE Transactions on Power Delivery*, vol. 28, no. 2, pp. 771–778, 2013.

[50] H. Tarzamni, E. Babaei, F. Panahandeh Esmaeelnia, P. Dehghanian, S. Tohidi, and M. B. Bannae Sharifian, "Analysis and reliability evaluation of a high step-up soft switching push-pull dc-dc converter," *IEEE Transactions on Reliability*, pp. 1–11, 2020.

[51] H. Tarzamni, F. Panahandeh Esmaeelnia, M. Fotuhi-Firuzabad, F. Tahami, S. Tohidi, and P. Dehghanian, "Comprehensive analytics for reliability evaluation of conventional isolated multi-switch pwm

dc-dc converters," *IEEE Transactions on Power Electronics*, vol. 35, no. 5, pp. 5254–5266, May 2020.

[52] D. Henry and J. E. Ramirez-Marquez, "Generic metrics and quantitative approaches for system resilience as a function of time," *Reliability Engineering & System Safety*, vol. 99, pp. 114–122, 2012.

[53] B. T. Shinde, "Real-time stability surveillance in power systems: A deep learning approach," [Online] Available at: https://cpb-us-e1.wpmucdn.com/blogs.gwu.edu/dist/f/1618/files/2020/01/Bhavesh-Shinde_MSc-Thesis.pdf, Accessed:2020.

[54] P. Dehghanian and M. Kezunovic, "Impact assessment of power system topology control on system reliability," in *IEEE Conference on Intelligent Systems Applications to Power Systems (ISAP)*, 2015, pp. 1–6.

[55] ——, "Probabilistic impact of transmission line switching on power system operating states," in *IEEE Power and Energy Society (PES) Transmission and Distribution (T&D) Conference and Exposition*, 2016, pp. 1–6.

[56] P. Dehghanian, B. Zhang, T. Dokic, and M. Kezunovic, "Predictive risk analytics for weather-resilient operation of electric power systems," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 1, pp. 3–15, 2019.

[57] P. Dehghanian, S. Aslan, and P. Dehghanian, "Maintaining electric system safety through an enhanced network resilience," *IEEE Transactions on Industry Applications*, vol. 54, no. 5, pp. 4927–4937, 2018.

[58] B. Zhang, P. Dehghanian, and M. Kezunovic, "Optimal allocation of PV generation and battery storage for enhanced resilience," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 535–545, 2017.

[59] M. Nazemi, M. Moeini-Aghtaie, M. Fotuhi-Firuzabad, and P. Dehghanian, "Energy storage planning for enhanced resilience of power distribution networks against earthquakes," *IEEE Transactions on Sustainable Energy*, vol. 11, no. 2, pp. 795–806, April 2020.

[60] P. Dehghanian, S. Aslan, and P. Dehghanian, "Quantifying power system resiliency improvement using network reconfiguration," in *IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1–5.

[61] S. Wang, P. Dehghanian, L. Li, and B. Wang, "A machine learning approach to detection of geomagnetically-induced currents in power grids," in *IEEE Industry Applications Society (IAS) Annual Meeting*, 2019, pp. 1–7.

[62] Z. Yang, P. Dehghanian, and M. Nazemi, "Enhancing seismic resilience of electric power distribution systems with mobile power sources," in *IEEE Industry Applications Society (IAS) Annual Meeting*, 2019, pp. 1–7.

[63] J. Su, P. Dehghanian, M. Nazemi, and B. Wang, "Distributed wind power resources for enhanced power grid resilience," in *The 51st North American Power Symposium (NAPS)*, 2019, pp. 1–6.

[64] D. Wang, Y. Li, P. Dehghanian, and S. Wang, "Power grid resilience to electromagnetic (EMP) disturbances: A literature review," in *The 51st North American Power Symposium (NAPS)*, 2019, pp. 1–6.

[65] M. Babakmehr, F. Harirchi, M. Nazir, S. Wang, P. Dehghanian, and J. Enslin, "Sparse representation-based classification of geomagnetically induced currents," in *Clemson University Power Systems Conference*, 2020, pp. 1–6.

[66] Z. Yang, M. Nazemi, P. Dehghanian, and M. Barati, "Toward resilient solar-integrated distribution grids: Harnessing the mobility of power sources," in *IEEE Power and Energy Society (PES) Transmission and Distribution (T&D) Conference and Exposition*, 2020, pp. 1–5.

[67] B. Shinde, S. Wang, P. Dehghanian, and M. Babakmehr, "Real-time detection of critical generators in power systems: A deep learning HCP approach," in *The 4th IEEE Texas Power and Energy Conference (TPEC)*, 2020, pp. 1–6.

[68] M. Nazemi and P. Dehghanian, "Seismic-resilient bulk power grids: Hazard characterization, modeling, and mitigation," *IEEE Transactions on Engineering Management*, vol. 67, no. 3, pp. 614–630, Aug. 2020.

[69] S. Wang, P. Dehghanian, L. Li, and B. Wang, "A machine learning approach to detection of geomagnetically-induced currents in power grids," *IEEE Transactions on Industry Applications*, vol. 56, no. 2, pp. 1098–1106, March/April 2020.

[70] Z. Yang, P. Dehghanian, and M. Nazemi, "Seismic-resilient electric power distribution systems: Harnessing the mobility of power sources," *IEEE Transactions on Industry Applications*, vol. 56, no. 3, pp. 2304–2313, May/June 2020.

[71] Nazemi, Mostafa and Dehghanian, Payman and Alhazmi, Mohannad, and Wang, Fei, "Multivariate uncertainty characterization for resilience planning in electric power systems," in *IEEE/IAS 56th Industrial and Commercial Power Systems (I&CPS) Technical Conference*, 2020, pp. 1–7.

[72] P. Jamborsalamati, M. Hossain, S. Taghizadeh, A. Sadu, G. Konstantinou, M. Manbachi, and P. Dehghanian, "Enhancing power grid resilience through an IEC61850-based ev-assisted load restoration," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1799–1810, March 2020.

[73] S. Wang, P. Dehghanian, M. Alhazmi, and M. Nazemi, "Advanced control solutions for enhanced resilience of modern power-electronic-interfaced distribution systems," *Journal of Modern Power Systems and Clean Energy*, vol. 7, no. 4, pp. 716–730, July 2019.

[74] S. Wang, P. Dehghanian, M. Alhazmi, J. Su, and B. Shinde, "Resilience-assured protective control of DC/AC inverters under unbalanced and fault scenarios," in *The 10th IEEE Power and Energy Society (PES) Conference on Innovative Smart Grid Technologies-North America (ISGT-NA)*, 2019, pp. 1–5.

[75] M. Babakmehr, F. Harirchi, P. Dehghanian, and J. Enslin, "Artificial intelligence-based cyber-physical event classification for islanding detection in power inverters," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, pp. 1–11, 2020.

[76] P. Dehghanian, "Power system topology control for enhanced resilience of smart electricity grids," Ph.D. dissertation, Texas A&M University, 2017.

[77] D. Deka, S. N. Backhaus, M. Chertkov, A. Lokhov, S. Misra, M. D. Vuffray, and K. Dvijotham, "Machine learning for the grid," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2016.

[78] P. Dehghanian and M. Kezunovic, "Probabilistic decision making for the bulk power system optimal topology control," *IEEE Transactions on Smart Grid*, vol. 7, no. 4, pp. 2071–2081, 2016.

[79] P. Dehghanian, Y. Wang, G. Gurrala, E. Moreno-Centeno, and P. Kezunovic, "Flexible implementation of power system corrective topology control," *Electric Power System Research*, vol. 128, pp. 79–89, 2015.

[80] M. Alhazmi, P. Dehghanian, S. Wang, and B. Shinde, "Power grid optimal topology control considering correlations of system uncertainties," in *IEEE/IAS 55th Industrial and Commercial Power Systems (I&CPS) Technical Conference*, 2019, pp. 1–7.

[81] M. Kezunovic, T. Popovic, G. Gurrala, P. Dehghanian, A. Esmaeilian, and M. Tasdighi, "Reliable implementation of robust adaptive topology control," in *The 47th Hawaii International Conference on System Science (HICSS)*, 2014, pp. 1–10.

[82] M. Alhazmi, P. Dehghanian, S. Wang, and B. Shinde, "Power grid optimal topology control considering correlations of system uncertainties," *IEEE Transactions on Industry Applications*, vol. 55, no. 6, pp. 5594–5604, November/December 2019.

[83] M. Nazemi, P. Dehghanian, and M. Lejeune, "A mixed-integer distributionally robust chance-constrained model for optimal topology control in power grids with uncertain renewables," in *13th IEEE Power and Energy Society (PES) PowerTech Conference*, 2019, pp. 1–6.

[84] M. A. Saffari, M. S. Misaghian, M. Kia, A. Heidari, D. Zhang, P. Dehghanian, and J. Aghaei, "Stochastic robust optimization for smart grid considering various arbitrage opportunities," *Electric Power Systems Research*, vol. 174, pp. 1–14, September 2019.

[85] M. Moeini-Aghtaie, A. Abbaspour, M. Fotuhi-Firuzabad, and P. Dehghanian, "Optimized probabilistic phev demand management in the context of energy hubs," *IEEE Transactions on Power Delivery*, vol. 30, no. 2, pp. 996–1006, 2015.

[86] ——, "Phev's centralized/decentralized charging control mechanisms: Requirements and impacts," in *The 45th North American Power Symposium (NAPS)*, 2013, pp. 1–6.

[87] M. S. Misaghian, M. Saffari, M. Kia, A. Heidari, P. Dehghanian, and B. Wang, "Electric vehicles contributions to voltage improvement and loss reduction in microgrids," in *North American Power Symposium (NAPS)*, 2018, pp. 1–6.

[88] B. Wang, P. Dehghanian, S. Wang, and M. Mitolo, "Electrical safety considerations in large electric vehicle charging stations," *IEEE Transactions on Industry Applications*, vol. 55, no. 6, pp. 6603–6612, November/December 2019.

[89] B. Wang, P. Dehghanian, and D. Zhao, "Chance-constrained energy management system for power grids with high proliferation of renewables and electric vehicles," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2324–2336, May 2020.

[90] B. Wang, D. Zhao, P. Dehghanian, Y. Tian, and T. Hong, "Aggregated electric vehicle load modeling in large-scale electric power systems," *IEEE Transactions on Industry Applications*, pp. 1–14, 2020.

[91] B. Wang, P. Dehghanian, D. Hu, and S. Wang, "Adaptive operation strategies for electric vehicle charging stations," in *IEEE Industry Applications Society (IAS) Annual Meeting*, 2019, pp. 1–7.

[92] B. Wang, J. A. Camacho, G. M. Pulliam, A. H. Etemadi, and P. Dehghanian, "New reward and penalty scheme for electric distribution utilities employing load-based reliability indices," *IET Generation, Transmission & Distribution*, vol. 12, no. 15, pp. 3647–3654, 2018.

[93] L. Hao, B. Tianshu, X. Quan, C. Shijie, S. Bonian, and X. Ancheng, "Scheme and prospect of high-precision synchrophasor measurement technology for distribution network," *Automation of Electric Power Systems*, [Online] Available at: http://doi.org/10.7500/AEPS20190814002, Accessed:2020.

[94] "openPDC," Phasor Data Concentrator, [Online] Available at: https://www.openpdc.codeplex.com.

[95] North American Synchrophasor Initiative (NASPI), [Online] Available at: https://www.naspi.org.

[96] S. V. Wiel, R. Bent, E. Casleton, and E. Lawrence, "Identification of topology changes in power grids using phasor measurements," *Applied Stochastic Models in Business and Industry*, vol. 30, no. 6, pp. 740–752, 2014.

[97] "Phasor measurement unit," Wikipedia, [Online] Available at: https://en.wikipedia.org/wiki/Phasor_measurement_unit, Accessed:2020.

[98] A. Silverstein and J. Follum, "High-resolution, time-synchronized grid monitoring devices," North American Synchrophasor Initiative (NASPI), [Online] Available at: https://www.naspi.org/sites/default/files/reference_documents/pnnl_29770_naspi_hires_synch_grid_devices_20200320.pdf.

[99] H. Kirkham, "Pure and applied metrology," *IEEE Instrumentation & Measurement Magazine*, vol. 19, no. 6, pp. 19–24, 2016.

[100] KEMA Inc., "Substation communications: Enabler of automation / technologies," UTC — United Telecom Council, pp. 3–40, 2006.

[101] M. Copeland, "Difference between artificial intelligence, machine learning, and deep learning," SGInnovate, [Online] Available at: https://www.sginnovate.com/blog/difference-between-artificial-intelligence-machine-learning-and-deep-learning, Accessed:2017.

[102] "Cyber security with artificial intelligence in 10 question," NormShield, [Online] Available at: https://www.normshield.com/cyber-security-with-artificial-intelligence-in-10-question, Accessed:2020.

[103] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. D. Jesús, *Neural network design (2nd Edition)*. Martin Hagan, 2014.

[104] M. Valueva, N. Nagornov, P. Lyakhov, G. Valuev, and N. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation*, 2020.

[105] W. Zhang *et al.*, "Shift-invariant pattern recognition neural network and its optical architecture," in *Proceedings of annual conference of the Japan Society of Applied Physics*, 1988.

[106] W. Zhang, K. Itoh, J. Tanida, and Y. Ichioka, "Parallel distributed processing model with local space-invariant interconnections and its optical architecture," *Applied optics*, vol. 29, no. 32, pp. 4790–4797, 1990.

[107] A. Rao, "Convolutional neural network tutorial (cnn) – developing an image classifier in python using tensorflow," edureka.co, [Online] Available at: https://www.edureka.co/blog/convolutional-neural-network, Accessed:2020.

[108] "Convolutional neural network," Wikipedia, [Online] Available at: https://en.wikipedia.org/wiki/Convolutional_neural_network, Accessed:2020.

[109] C. P. Steinmetz, "Complex quantities and their use in electrical engineering," in *Proceedings of the International Electrical Congress, Chicago*. Illinois 1893 conference of the AIEE: American Institute of Electrical Engineers, 1893, p. 33–74.

[110] A. G. Phadke, "Synchronized phasor measurements-a historical overview," in *IEEE/PES Transmission and Distribution Conference and Exhibition*, vol. 1, 2002, pp. 476–479.

[111] W. Kersting, "A method to teach the design and operation of a distribution system," *IEEE Transactions on Power apparatus and Systems*, no. 7, pp. 1945–1952, 1984.

[112] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[113] L. Li, M. Doroslovački, and M. H. Loew, "Discriminant analysis deep neural networks," in *2019 53rd annual conference on information sciences and systems (CISS)*. IEEE, 2019, pp. 1–6.

[114] ——, "Loss functions forcing cluster separations for multi-class classification using deep neural networks," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2019, pp. 2106–2110.

[115] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[116] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1798–1828, 2013.

[117] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing*. Springer, 1990, pp. 227–236.

[118] F. F. Wu and A. J. Monticelli, "Critical review of external network modelling for online security analysis," *International Journal of Electrical Power and Energy Systems*, vol. 5, pp. 222–235, 1983.

[119] R. Lugtu, D. Hackett, K. Liu, and D. Might, "Power system state estimation: Detection of topological errors," *IEEE Transactions on Power Apparatus and Systems*, no. 6, pp. 2406–2412, 1980.

[120] M. R. Dorostkar-Ghamsari, M. Fotuhi-Firuzabad, M. Lehtonen, and A. Safdarian, "Value of distribution network reconfiguration in presence of renewable energy resources," *IEEE Transactions on Power Systems*, vol. 31, no. 3, pp. 1879–1888, 2015.

[121] S. Bolognani, N. Bof, D. Michelotti, R. Muraro, and L. Schenato, "Identification of power distribution network topology via voltage correlation analysis," in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 1659–1664.

[122] S. Xu, R. C. De Lamare, and H. V. Poor, "Dynamic topology adaptation for distributed estimation in smart grids," in *2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2013, pp. 420–423.

[123] J. Huang, V. Gupta, and Y.-F. Huang, "Electric grid state estimators for distribution systems with microgrids," in *2012 46th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2012, pp. 1–6.

[124] L. F. Alvarado, "Determination of external system topology errors," *Power Engineering Review, IEEE*, pp. 34–35, 1981.

[125] C. N. Lu, K. C. Liu, and S. Vemuri, "An external network modeling approach for online security analysis," *IEEE Transactions on Power Systems*, vol. 5, no. 2, pp. 565–573, 1990.

[126] K. L. Lo, L. J. Peng, J. F. Macqueen, A. O. Ekwue, and D. T. Y. Cheng, "An extended ward equivalent approach for power system security assessment," *Electric Power Systems Research*, vol. 42, pp. 181–188, 1997.

[127] H. Singh and F. L. Alvarado, "Network topology determination using least absolute value state estimation," *IEEE Transactions on Power Systems*, vol. 10, no. 3, pp. 1159–1165, 1995.

[128] P. Dimo, "Nodal analysis of power systems," 1975.

[129] G. Irisarri, A. M. Sasson, and J. F. Dopazo, "Real-time external system equivalent for on-line contingency analysis," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-98, no. 6, pp. 2153–2171, 1979.

[130] A. S. Debs, "Estimation of steady-state power system model parameters," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-93, no. 5, pp. 1260–1268, 1974.

[131] K. M. Rogers, R. D. Spadoni, and T. J. Overbye, "Identification of power system topology from synchrophasor data," in *2011 IEEE/PES Power Systems Conference and Exposition*. IEEE, 2011, pp. 1–8.

[132] C. M. Davis, J. E. Tate, and T. J. Overbye, "Wide area phasor data visualization," in *2007 39th North American Power Symposium*. IEEE, 2007, pp. 246–252.

[133] K. R. Gabriel, "The biplot graphical display of matrices with application to principal component analysis," *Biometrika*, vol. 58, no. 3, p. 453–467, 1971.

[134] E. Lawrence, S. V. Wiel, and R. Bent, "Model bank state estimation for power grids using importance sampling," *Technometrics*, vol. 55, no. 4, pp. 426–435, 2013.

[135] G. Cavraro and R. Arghandeh, "Power distribution network topology detection with time-series signature verification method," *IEEE Transactions on Power Systems*, vol. 33, no. 4, pp. 3500–3509, 2017.

[136] G. Cavraro, R. Arghandeh, K. Poolla, and A. Von Meier, "Data-driven approach for distribution network topology detection," in *2015 IEEE power & energy society general meeting*. IEEE, 2015, pp. 1–5.

[137] D. Deka, S. Talukdar, M. Chertkov, and M. Salapaka, "Graphical models in loopy distribution grids: Topology estimation, change detection and limitation," *arXiv preprint arXiv:1905.06550*, 2019.

[138] ——, "Graphical models in meshed distribution grids: Topology estimation, change detection & limitations," *IEEE Transactions on Smart Grid*, 2020.

[139] D. Deka, M. Chertkov, and S. Backhaus, "Topology estimation using graphical models in multi-phase power distribution grids," *IEEE Transactions on Power Systems*, 2019.

[140] P. K. Ghosh and A. Tahabilder, "Optimal pmu placement for complete system observability and fault observability using graph theory," in *2017 International Electrical Engineering Congress (iEECON)*. IEEE, 2017, pp. 1–4.

[141] Y. Weng, Y. Liao, and R. Rajagopal, "Distributed energy resources topology identification via graphical modeling," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 2682–2694, 2016.

[142] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[143] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[144] "Matlab," Wikipedia, [Online] Available at: https://en.wikipedia.org/wiki/MATLAB, Accessed:2020.

[145] "Simulink," Wikipedia, [Online] Available at: https://en.wikipedia.org/wiki/Simulink, Accessed:2020.

[146] C. D. Bodemann and F. De Rose, "The successful development process with matlab simulink in the framework of esa's atv project," in *55th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law*, pp. U–3.

[147] J. Reedy and S. Lunzman, "Model based design accelerates the development of mechanical locomotive controls," SAE Technical Paper, Tech. Rep., 2010.

[148] A. Santacruz, "Why it is important to work with a balanced classification dataset," amsantac.co, [Online] Available at: http://amsantac.co/blog/en/2016/09/20/balanced-image-classification-r.html, Accessed:2016.

[149] "powergui," MathWorks, [Online] Available at: https://www.mathworks.com/help/physmod/sps/powersys/ref/powergui.html, Accessed:2006.

[150] "Pycharm," Wikipedia, [Online] Available at: https://en.wikipedia.org/wiki/PyCharm, Accessed:2020.

[151] "Cuda|nvidia developer," NVIDIA, [Online] Available at: https://developer.nvidia.com/about-cuda, Accessed:2020.

[152] "Cuda," Wikipedia, [Online] Available at: https://en.wikipedia.org/wiki/CUDA, Accessed:2020.

[153] "Convolutional neural nework model - deep learning and neural networks with python and pytorch p.6," PythonProgramming.net, [Online] Available at: https://pythonprogramming.net/convnet-model-deep-learning-neural-network-pytorch, Accessed:2019.

[154] "Run parallel simulations," MathWorks, [Online] Available at: https://www.mathworks.com/help/simulink/ug/running-parallel-simulations.html, Accessed:2020.

[155] J. Lai, X. Lu, F. Wang, P. Dehghanian, and R. Tang, "Broadcast gossip algorithms for distributed peer-to-peer control in AC microgrids," *IEEE Transactions on Industry Applications*, vol. 55, no. 3, pp. 2241–2251, May/June 2019.

[156] F. Wang, B. Xiang, K. Li, J. Lai, and P. Dehghanian, "Day-ahead forecast of aggregated loads for smart households under incentive-based demand response programs," in *IEEE Industry Applications Society (IAS) Annual Meeting*, 2019, pp. 1–10.

[157] S. Dehghan-Dehnavi, M. Fotuhi-Firuzabad, M. Moeini-Aghtaie, P. Dehghanian, and F. Wang, "Estimating participation abilities of industrial customers in demand response programs: A two-level decision-making tree analysis," in *IEEE/IAS 56th Industrial and Commercial Power Systems (I&CPS) Technical Conference*, 2020, pp. 1–7.

[158] F. Pourahmadi, H. Heidarabadi, S. H. Hosseini, and P. Dehghanian, "Dynamic uncertainty set characterization for bulk power grid flexibility assessment," *IEEE Systems Journal*, vol. 14, no. 1, pp. 718–728, March 2020.

[159] R. Azizpanah-Abarghooee, P. Dehghanian, and V. Terzija, "A practical multi-area bi-objective environmental economic dispatch equipped with a hybrid gradient search method and improved jaya algorithm," *IET Generation, Transmission & Distribution*, vol. 10, no. 14, pp. 3580–3596, 2016.

[160] M. Khoshjahan, P. Dehghanian, M. Moeini-Aghtaie, and M. Fotuhi-Firuzabad, "Harnessing ramp capability of spinning reserve services

for enhanced power system flexibility," *IEEE Transactions on Industry Applications*, vol. 55, no. 6, pp. 7103–7112, November/December 2019.

[161] M. Zareian Jahromi, M. Tajdinian, J. Zhao, P. Dehghanian, M. Allahbakhshi, and A. Seifi, "An enhanced sensitivity-based decentralized framework for real-time transient stability assessment in bulk power grids with renewable energy resources," *IET Generation, Transmission, and Distribution Systems*, vol. 14, no. 4, pp. 665–674, February 2020.

[162] M. Tajdinian, M. Allahbakhshi, A. R. Bagheri, H. Samet, P. Dehghanian, and O. P. Malik, "An enhanced sub-cycle statistical algorithm for inrush and fault currents classification in differential protection schemes," *International Journal of Electrical Power and Energy Systems*, vol. 119, pp. 1–17, July 2020.

[163] Z. Li, K. Li, F. Wang, Z. Mi, and P. Dehghanian, "An auto-encoder neural network approach to monthly electricity consumption forecasting using hourly data," in *IEEE/IAS 56th Industrial and Commercial Power Systems (I&CPS) Technical Conference*, 2020, pp. 1–7.

[164] M. Lejeune and P. Dehghanian, "Optimal power flow models with probabilistic guarantees: A boolean approach," *IEEE Transactions on Power Systems*, pp. 1–4, July 2020.

[165] B. Zhang, P. Dehghanian, and M. Kezunovic, "Simulation of weather impacts on the wholesale electricity market," in *10th International Conference on Deregulated Electricity Market Issues in South Eastern Europe (DEMSEE)*, 2015, pp. 1–6.

[166] Z. Li, Z. Xuan, K. Li, F. Wang, Z. Mi, P. Dehghanian, W. Li, and M. Fotuhi-Firuzabad, "Monthly electricity consumption forecasting based on two-stage forecasting step reduction strategy and auto-encoder neural network," *IEEE Transactions on Industry Applications*, pp. 1–11, 2020.

[167] M. Khoshjahan, M. Moeini-Aghtaie, M. Fotuhi-Firuzabad, P. Dehghanian, and H. Mazaheri, "Advanced bidding strategy for participation of energy storage systems in joint energy and flexible ramping product market," *IET Generation, Transmission, and Distribution*, pp. 1–11, 2020.

[168] T. Dokic, P. Dehghanian, P.-C. Chen, M. Kezunovic, Z. Medina-Cetina, J. Stojanovic, and Z. Obradovic, "Risk assessment of a transmission line insulation breakdown due to lightning and severe weather," in *The 49th Hawaii International Conference on System Science (HICSS)*, 2016, pp. 1–8.

[169] B. Zhang, P. Dehghanian, and M. Kezunovic, "Spatial-temporal solar power forecast through gaussian conditional random fields," in *IEEE Power and Energy Society (PES) General Meeting*, 2016, pp. 1–5.

[170] F. Pourahmadi and P. Dehghanian, "A game-theoretic loss allocation approach in power distribution systems with high penetration of distributed generations," *Mathematics*, vol. 6, no. 9, pp. 1–14, 2018.

[171] F. Pourahmadi, S. H. Hosseini, P. Dehghanian, E. Shittu, and M. Fotuhi-Firuzabad, "Uncertainty cost of stochastic producers: Metrics and impacts on power grid operational flexibility," *IEEE Transactions on Engineering Management*, pp. 1–12, 2020.

[172] F. Wang, B. Xiang, K. Li, X. Ge, H. Lu, J. Lai, and P. Dehghanian, "Smart households' aggregated capacity forecasting for load aggregators under incentive-based demand response programs," *IEEE Transactions on Industry Applications*, vol. 56, no. 2, pp. 1086–1097, March/April 2020.

[173] M. Zareian Jahromi, M. Tajdinian, J. Zhao, P. Dehghanian, M. Allahbakhshi, and A. Seifi, "An enhanced sensitivity-based decentralized framework for real-time transient stability assessment in bulk power grids with renewable energy resources," *IET Generation, Transmission, and Distribution Systems*, vol. 14, no. 4, pp. 665–674, February 2020.

[174] M. Moeini-Aghtaie, P. Dehghanian, M. Fotuhi-Firuzabad, and A. Abbaspour, "Multi-agent genetic algorithm: An online probabilistic view on economic dispatch of energy hubs constrained by wind availability," *IEEE Transactions on Sustainable Energy*, vol. 5, no. 2, pp. 699–708, 2014.

[175] P. Dehghanian, S. H. Hosseini, M. Moeini-Aghtaie, and S. Arabali, "Optimal siting of dg units in power systems from a probabilistic multi-objective optimization perspective," *International Journal of Electrical Power and Energy Systems*, vol. 51, pp. 14–26, 2013.

[176] M. Alhazmi, P. Dehghanian, M. Nazemi, and M. Mitolo, "Optimal integration of the interconnected water andelectricity networks," in *2020 IEEE Industry Applications Society Annual Meeting*. IEEE, 2020, pp. 1–7.

Appendix A: IEEE 34 Node Test Feeder

A.1 Introduction

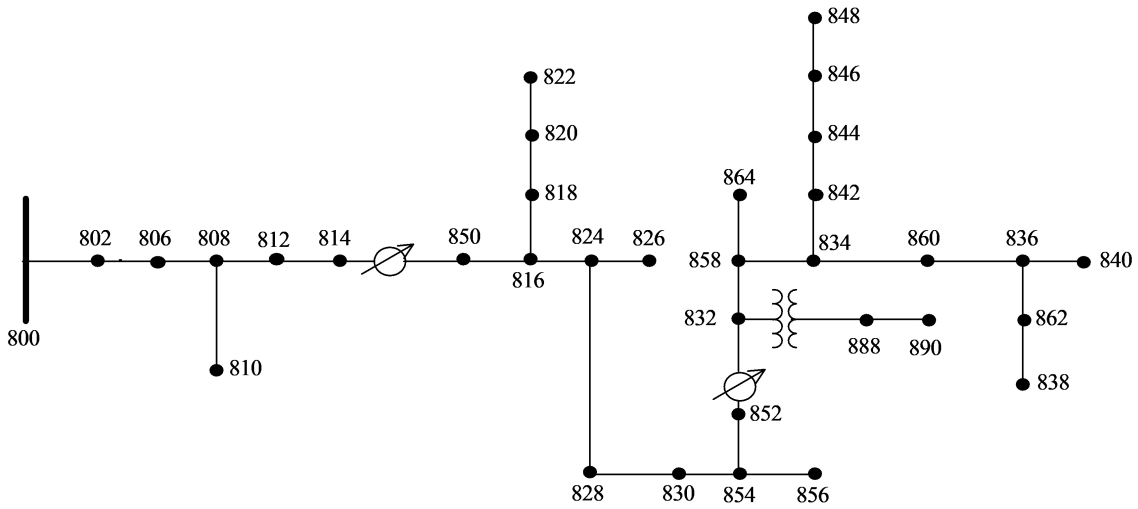


Figure A.1: IEEE 34-Node Test Feeder [6]

This feeder is an actual feeder located in Arizona. The feeder's nominal voltage is 24.9 kV. It 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

Because of the length of the feeder and the unbalanced loading, the system may at times have a convergence problem.

A.2 System Data

Here are the data forms originated from the IEEE PES AMPS DSAS test feeder working group [6].

Table A.1: Overhead Line Configurations [6]

Config.	**Phasing**	**Phase ACSR**[1]	**Neutral ACSR**	**Spacing ID**
300	BACN	1/0	1/0	500
301	BACN	#2 6/1	#2 6/1	500
302	AN	#4 6/1	#4 6/1	510
303	BN	#4 6/1	#4 6/1	510
304	BN	#2 6/1	#2 6/1	510

[1] ACSR: Aluminum conductor steel reinforced.

Table A.2: Line Segment Data [6]

Node A	**Node B**	**Length** (ft.)	**Config.**
800	802	2580	300
802	806	1730	300
806	808	32230	300
808	810	5804	303
808	812	37500	300
812	814	29730	300
814	850	10	301
816	818	1710	302
816	824	10210	301
818	820	48150	302
820	822	13740	302
824	826	3030	303
824	828	840	301
828	830	20440	301
830	854	520	301
832	858	4900	301
832	888	0	XFM-1
834	860	2020	301
834	842	280	301
836	840	860	301
836	862	280	301
842	844	1350	301
844	846	3640	301
846	848	530	301
850	816	130	301
852	832	10	301
854	856	23330	303
854	852	36830	301
858	864	1620	302
858	834	5830	301
860	836	2680	301
862	838	4860	304
888	890	10560	300

Table A.3: Transformer Data [6]

	kVA	kV - high	kV - low	**R** - %	**X** - %
Substation	2500	69 - D	24.9 - Gr.W	1	8
XFM-1	500	24.9 - Gr.W	24.9 - Gr.W	1.9	4.08

Table A.4: Spot Loads [6]

Node	**Load Model**	**Ph-1** kW	**Ph-1** kVAr	**Ph-2** kW	**Ph-2** kVAr	**Ph-3** kW	**Ph-4** kVAr
860	Y-PQ	20	16	20	16	20	16
802	Y-I	9	7	9	7	9	7
806	Y-Z	135	105	135	105	135	105
808	D-PQ	20	16	20	16	20	16
808	D-I	150	75	150	75	150	75
812	D-Z	10	5	10	5	25	10
Total		344	224	344	224	359	229

Table A.5: Distributed Loads [6]

Node A	**Node B**	**Load Model**	**Ph-1** kW	**Ph-1** kVAr	**Ph-2** kW	**Ph-2** kVAr	**Ph-3** kW	**Ph-3** kVAr
802	806	Y-PQ	0	0	30	15	25	14
808	810	Y-I	0	0	16	8	0	0
818	820	Y-Z	34	17	0	0	0	0
820	822	Y-PQ	135	70	0	0	0	0
816	824	D-I	0	0	5	2	0	0
824	826	Y-I	0	0	40	20	0	0
824	828	Y-PQ	0	0	0	0	4	2
828	830	Y-PQ	7	3	0	0	0	0
854	856	Y-PQ	0	0	4	2	0	0
832	858	D-Z	7	3	2	1	6	3
858	864	Y-PQ	2	1	0	0	0	0
858	834	D-PQ	4	2	15	8	13	7
834	860	D-Z	16	8	20	10	110	55
860	836	D-PQ	30	15	10	6	42	22
836	840	D-I	18	9	22	11	0	0
862	838	Y-PQ	0	0	28	14	0	0
842	844	Y-PQ	9	5	0	0	0	0
844	846	Y-PQ	0	0	25	12	20	11
846	848	Y-PQ	0	0	23	11	0	0
Total			262	133	240	120	220	114

Table A.6: Shunt Capacitors [6]

Node	**Ph-A** kVAr	**Ph-B** kVAr	**Ph-C** kVAr
844	100	100	100
848	150	150	150
Total	250	250	250

Table A.7: Regulator Data [6]

Regulator ID	1			2		
Line Segment	814-850			852-832		
Location	814			852		
Phases	A-B-C			A-B-C		
Connection	3-Ph, LG			3-Ph, LG		
Monitoring Phase	A-B-C			A-B-C		
Bandwidth	2.0 volts			2.0 volts		
PT Ratio	120			120		
Primary CT Rating	100			100		
Compensator Settings	**Ph-A**	**Ph-B**	**Ph-C**	**Ph-A**	**Ph-B**	**Ph-C**
R-Setting	2.7	2.7	2.7	2.5	2.5	2.5
X-Setting	1.6	1.6	1.6	1.6	1.6	1.6
Vlotage Level	122	122	122	124	124	124

A.3 Impedances

```
Configuration 300:

————————— Z & B Matrices  Before Changes —————————

          Z (R +jX) in ohms per mile
 1.3368   1.3343    0.2101   0.5779    0.2130   0.5015
                    1.3238   1.3569    0.2066   0.4591
                                       1.3294   1.3471
         B in micro Siemens per mile
           5.3350    −1.5313    −0.9943
                      5.0979    −0.6212
                                 4.8880
Configuration 301:

          Z (R +jX) in ohms per mile
 1.9300   1.4115    0.2327   0.6442    0.2359   0.5691
                    1.9157   1.4281    0.2288   0.5238
                                       1.9219   1.4209
         B in micro Siemens per mile
           5.1207    −1.4364    −0.9402
                      4.9055    −0.5951
                                 4.7154

Configuration 302:

          Z (R +jX) in ohms per mile
 2.7995   1.4855    0.0000   0.0000    0.0000   0.0000
                    0.0000   0.0000    0.0000   0.0000
                                       0.0000   0.0000
         B in micro Siemens per mile
           4.2251     0.0000     0.0000
                      0.0000     0.0000
                                 0.0000

Configuration 303:

          Z (R +jX) in ohms per mile
 0.0000   0.0000    0.0000   0.0000    0.0000   0.0000
```

```
                    2.7995   1.4855    0.0000   0.0000
                                       0.0000   0.0000
          B in micro Siemens per mile
            0.0000      0.0000      0.0000
                        4.2251      0.0000
                                    0.0000

Configuration 304:

          Z (R +jX) in ohms per mile
0.0000   0.0000    0.0000   0.0000    0.0000   0.0000
                   1.9217   1.4212    0.0000   0.0000
                                      0.0000   0.0000
          B in micro Siemens per mile
            0.0000      0.0000      0.0000
                        4.3637      0.0000
                                    0.0000
```

A.4 Power Flow Results

A.4.1 Radial Flow Summary

```
– R A D I A L  F L O W  S U M M A R Y – DATE:  6–24–2004 AT 16:34:11 HOURS ——
SUBSTATION:  IEEE 34;   FEEDER:  IEEE 34
_____________________________________________________________________________

SYSTEM          PHASE              PHASE              PHASE              TOTAL
INPUT ————————(A)————————|————————(B)————————|————————(C)————————|————————————————
kW    :         759.136    |         666.663    |         617.072    |        2042.872
kVAr  :         171.727    |          90.137    |          28.394    |         290.258
kVA   :         778.318    |         672.729    |         617.725    |        2063.389
PF    :           .9754    |           .9910    |           .9989    |          .9901

LOAD   ——(A–N)————(A–B)—|——(B–N)————(B–C)—|——(C–N)————(C–A)—|———WYE———DELTA—
kW    :   359.9      246.4|    339.3      243.3|    221.8      359.0|    921.0      848.8
 TOT  :         606.322    |         582.662    |         580.840    |       1769.824
                           |                    |                    |
kVAr  :   230.9      128.7|    216.9      128.7|    161.8      184.6|    609.6      441.9
 TOT  :         359.531    |         345.609    |         346.407    |       1051.547
                           |                    |                    |
```

```
kVA   :   427.6     278.0|    402.7     275.3|    274.6     403.7|   1104.5     957.0
 TOT  :        704.903    |         677.452    |         676.293    |        2058.647
                          |                    |                    |
PF    :   .8417     .8864|    .8425     .8840|    .8078     .8894|    .8339     .8870
 TOT  :         .8601     |          .8601     |          .8589     |          .8597

LOSSES ------(A)--------|--------(B)--------|--------(C)--------|--------------------
kW    :        114.836    |          80.389    |          77.824    |         273.049
kVAr  :         14.200    |          10.989    |           9.810    |          34.999
kVA   :        115.711    |          81.137    |          78.440    |         275.283

CAPAC --(A-N)-----(A-B)--|--(B-N)-----(B-C)--|--(C-N)-----(C-A)--|---WYE------DELTA--
R-kVA:   250.0         .0|    250.0         .0|    250.0         .0|    750.0         .0
 TOT  :        250.000    |         250.000    |         250.000    |         750.000
                          |                    |                    |
A-kVA:   265.7         .0|    264.8         .0|    265.9         .0|    796.3         .0
 TOT  :        265.658    |         264.760    |         265.869    |         796.287
```

A.4.2 Voltage Profile

```
--- V O L T A G E   P R O F I L E --- DATE:  6-24-2004 AT 16:34:18 HOURS ---
SUBSTATION:  IEEE 34;   FEEDER:  IEEE 34
_____________________________________________________________________________
NODE  |   MAG       ANGLE  |    MAG       ANGLE  |    MAG       ANGLE |mi.to SR
_____________________________________________________________________________
______|_______ A-N ______  |_______ B-N _______  |_______ C-N _______ |
800   |  1.0500 at    .00   |  1.0500 at -120.00   |  1.0500 at  120.00 |    .000
802   |  1.0475 at   -.05   |  1.0484 at -120.07   |  1.0484 at  119.95 |    .489
806   |  1.0457 at   -.08   |  1.0474 at -120.11   |  1.0474 at  119.92 |    .816
808   |  1.0136 at   -.75   |  1.0296 at -120.95   |  1.0289 at  119.30 |   6.920
810   |                     |  1.0294 at -120.95   |                    |   8.020
812   |   .9763 at  -1.57   |  1.0100 at -121.92   |  1.0069 at  118.59 |  14.023
814   |   .9467 at  -2.26   |   .9945 at -122.70   |   .9893 at  118.01 |  19.653
RG10  |  1.0177 at  -2.26   |  1.0255 at -122.70   |  1.0203 at  118.01 |  19.654
850   |  1.0176 at  -2.26   |  1.0255 at -122.70   |  1.0203 at  118.01 |  19.655
816   |  1.0172 at  -2.26   |  1.0253 at -122.71   |  1.0200 at  118.01 |  19.714
818   |  1.0163 at  -2.27   |                      |                    |  20.038
820   |   .9926 at  -2.32   |                      |                    |  29.157
822   |   .9895 at  -2.33   |                      |                    |  31.760
```

824	1.0082 at −2.37	1.0158 at −122.94	1.0116 at 117.76	21.648
826		1.0156 at −122.94		22.222
828	1.0074 at −2.38	1.0151 at −122.95	1.0109 at 117.75	21.807
830	.9894 at −2.63	.9982 at −123.39	.9938 at 117.25	25.678
854	.9890 at −2.64	.9978 at −123.40	.9934 at 117.24	25.777
852	.9581 at −3.11	.9680 at −124.18	.9637 at 116.33	32.752
RG11	1.0359 at −3.11	1.0345 at −124.18	1.0360 at 116.33	32.752
832	1.0359 at −3.11	1.0345 at −124.18	1.0360 at 116.33	32.754
858	1.0336 at −3.17	1.0322 at −124.28	1.0338 at 116.22	33.682
834	1.0309 at −3.24	1.0295 at −124.39	1.0313 at 116.09	34.786
842	1.0309 at −3.25	1.0294 at −124.39	1.0313 at 116.09	34.839
844	1.0307 at −3.27	1.0291 at −124.42	1.0311 at 116.06	35.095
846	1.0309 at −3.32	1.0291 at −124.46	1.0313 at 116.01	35.784
848	1.0310 at −3.32	1.0291 at −124.47	1.0314 at 116.00	35.885
860	1.0305 at −3.24	1.0291 at −124.39	1.0310 at 116.09	35.169
836	1.0303 at −3.23	1.0287 at −124.39	1.0308 at 116.09	35.677
840	1.0303 at −3.23	1.0287 at −124.39	1.0308 at 116.09	35.839
862	1.0303 at −3.23	1.0287 at −124.39	1.0308 at 116.09	35.730
838		1.0285 at −124.39		36.650
864	1.0336 at −3.17			33.989
XF10	.9997 at −4.63	.9983 at −125.73	1.0000 at 114.82	32.754
888	.9996 at −4.64	.9983 at −125.73	1.0000 at 114.82	32.754
890	.9167 at −5.19	.9235 at −126.78	.9177 at 113.98	34.754
856		.9977 at −123.41		30.195

A.4.3 Voltage Regulator Data

```
————————— VOLTAGE REGULATOR DATA ——— DATE:  6–24–2004 AT 16:34:22 HOURS —
SUBSTATION:  IEEE 34;   FEEDER:  IEEE 34

--------------------------------------------------------------------------------
[NODE]--[VREG]-----[SEG]------[NODE]          MODEL                    OPT    BNDW
814     RG10       850         850     Phase A & B & C, Wye            RX    2.00
        ..............................................................................
        PHASE  LDCTR   VOLT HOLD  R–VOLT   X–VOLT  PT RATIO  CT RATE     TAP
          1              122.000    2.700    1.600   120.00    100.00      12
          2              122.000    2.700    1.600   120.00    100.00       5
          3              122.000    2.700    1.600   120.00    100.00       5

--------------------------------------------------------------------------------
[NODE]--[VREG]-----[SEG]------[NODE]          MODEL                    OPT    BNDW
```

```
852      RG11        832          832      Phase A & B & C, Wye              RX     2.00
        ..................................................................
        PHASE  LDCTR   VOLT HOLD  R-VOLT   X-VOLT  PT RATIO  CT RATE     TAP
          1            124.000    2.500    1.500    120.00    100.00     13
          2            124.000    2.500    1.500    120.00    100.00     11
          3            124.000    2.500    1.500    120.00    100.00     12
```

A.4.4 Radial Power Flow

```
- R A D I A L  P O W E R  F L O W ---  DATE:  6-24-2004 AT 16:34:32 HOURS ---
SUBSTATION:  IEEE 34;   FEEDER:  IEEE 34
_____________________________________________________________________________

   NODE       VALUE          PHASE A          PHASE B          PHASE C      UNT O/L<
                             (LINE A)         (LINE B)         (LINE C)        60.%
-------------------------*---------A---------*---------B---------*---------C---------*----------
NODE: 800        VOLTS:   1.050     .00   1.050 -120.00   1.050   120.00 MAG/ANG
 kVll  24.900             NO LOAD OR CAPACITOR REPRESENTED AT SOURCE NODE

TO NODE 802    .......:   51.56  -12.74   44.57 -127.70   40.92   117.37 AMP/DG
<802    > LOSS=  3.472:    (  1.637)        (   .978)        (   .858)     kW
-------------------------*---------A---------*---------B---------*---------C---------*----------
NODE: 802        VOLTS:   1.047    -.05   1.048 -120.07   1.048   119.95 MAG/ANG
                  -LD:     .00      .00     .00      .00     .00      .00 kW/kVR
 kVll  24.900     CAP:              .00              .00             .00 kVR

FROM NODE 800  .....:   51.58  -12.80   44.57 -127.76   40.93   117.31 AMP/DG
<802    > LOSS=  3.472:    (  1.637)        (   .978)        (   .858)     kW
TO NODE 806    .......:   51.58  -12.80   44.57 -127.76   40.93   117.31 AMP/DG
<806    > LOSS=  2.272:    (  1.102)        (   .618)        (   .552)     kW
-------------------------*---------A---------*---------B---------*---------C---------*----------
NODE: 806        VOLTS:   1.046    -.08   1.047 -120.11   1.047   119.92 MAG/ANG
                  -LD:     .00      .00     .00      .00     .00      .00 kW/kVR
 kVll  24.900     CAP:              .00              .00             .00 kVR

FROM NODE 802  .....:   51.59  -12.83   42.47 -126.83   39.24   118.52 AMP/DG
<806    > LOSS=  2.272:    (  1.102)        (   .618)        (   .552)     kW
TO NODE 808    .......:   51.59  -12.83   42.47 -126.83   39.24   118.52 AMP/DG
<808    > LOSS= 41.339:    ( 20.677)        ( 10.780)        (  9.882)     kW
-------------------------*---------A---------*---------B---------*---------C---------*----------
```

```
NODE: 808          VOLTS:    1.014    -.75    1.030 -120.95    1.029   119.30 MAG/ANG
                     -LD:      .00     .00      .00     .00      .00      .00 kW/kVR
 kVll  24.900        CAP:                .00              .00              .00 kVR

FROM NODE 806     .....:    51.76  -13.47   42.46 -127.59   39.28   117.76 AMP/DG
<808    > LOSS= 41.339:     ( 20.677)        ( 10.780)        (  9.882)     kW
TO NODE 810       .....:                     1.22 -144.62                    AMP/DG
<810    > LOSS=   .002:                     (   .002)                       kW
TO NODE 812       .....:    51.76  -13.47   41.30 -127.10   39.28   117.76 AMP/DG
<812    > LOSS= 47.531:     ( 24.126)        ( 11.644)        ( 11.761)     kW
-------------------*---------A---------*---------B---------*---------C---------*---------
NODE: 810          VOLTS:                   1.029 -120.95                    MAG/ANG
                     -LD:                     .00     .00                     kW/kVR
 kVll  24.900        CAP:                             .00                     kVR

FROM NODE 808     .....:                     .00     .00                     AMP/DG
<810    > LOSS=   .002:                     (   .002)                       kW
-------------------*---------A---------*---------B---------*---------C---------*---------
NODE: 812          VOLTS:    .976   -1.57   1.010 -121.92   1.007   118.59 MAG/ANG
                     -LD:      .00     .00      .00     .00      .00      .00 kW/kVR
 kVll  24.900        CAP:                .00              .00              .00 kVR

FROM NODE 808     .....:    51.95  -14.18   41.29 -127.99   39.33   116.90 AMP/DG
<812    > LOSS= 47.531:     ( 24.126)        ( 11.644)        ( 11.761)     kW
TO NODE 814       .....:    51.95  -14.18   41.29 -127.99   39.33   116.90 AMP/DG
<814    > LOSS= 37.790:     ( 19.245)        (  9.140)        (  9.404)     kW
-------------------*---------A---------*---------B---------*---------C---------*---------
NODE: 814          VOLTS:    .947   -2.26    .994 -122.70    .989   118.01 MAG/ANG
                     -LD:      .00     .00      .00     .00      .00      .00 kW/kVR
 kVll  24.900        CAP:                .00              .00              .00 kVR

FROM NODE 812     .....:    52.10  -14.73   41.29 -128.69   39.37   116.23 AMP/DG
<814    > LOSS= 37.790:     ( 19.245)        (  9.140)        (  9.404)     kW
TO NODE RG10  .<VRG>.:    52.10  -14.73   41.29 -128.69   39.37   116.23 AMP/DG
<RG10   > LOSS=   .000:     (   .000)        (   .000)        (   .000)     kW
-------------------*---------A---------*---------B---------*---------C---------*---------
NODE: RG10         VOLTS:   1.018   -2.26   1.026 -122.70   1.020   118.01 MAG/ANG
                     -LD:      .00     .00      .00     .00      .00      .00 kW/kVR
 kVll  24.900        CAP:                .00              .00              .00 kVR

FROM NODE 814   <VRG>:    48.47  -14.73   40.04 -128.69   38.17   116.23 AMP/DG
```

```
<RG10  > LOSS=    .000:     (    .000)        (    .000)        (    .000)     kW
TO NODE 850    .......:    48.47  -14.73    40.04 -128.69    38.17  116.23 AMP/DG
<850   > LOSS=    .017:     (    .008)        (    .005)        (    .005)     kW
--------------------*---------A---------*---------B---------*---------C---------*----------
NODE: 850         VOLTS:    1.018   -2.26    1.026 -122.70    1.020  118.01 MAG/ANG
                    -LD:      .00     .00      .00     .00      .00     .00 kW/kVR
 kVll  24.900       CAP:              .00               .00              .00 kVR

FROM NODE RG10  .....:    48.47  -14.73    40.04 -128.69    38.17  116.23 AMP/DG
<850   > LOSS=    .017:     (    .008)        (    .005)        (    .005)     kW
TO NODE 816    .......:    48.47  -14.73    40.04 -128.69    38.17  116.23 AMP/DG
<816   > LOSS=    .538:     (    .254)        (    .145)        (    .139)     kW
--------------------*---------A---------*---------B---------*---------C---------*----------
NODE: 816         VOLTS:    1.017   -2.26    1.025 -122.71    1.020  118.01 MAG/ANG
                    -LD:      .00     .00      .00     .00      .00     .00 kW/kVR
 kVll  24.900       CAP:              .00               .00              .00 kVR

FROM NODE 850   .....:    48.47  -14.74    40.04 -128.70    38.17  116.23 AMP/DG
<816   > LOSS=    .538:     (    .254)        (    .145)        (    .139)     kW
TO NODE 818    .......:    13.02  -26.69                                      AMP/DG
<818   > LOSS=    .154:     (    .154)                                       kW
TO NODE 824    .......:    35.83  -10.42    40.04 -128.70    38.17  116.23 AMP/DG
<824   > LOSS=  14.181:     (   4.312)        (   5.444)        (   4.425)     kW
--------------------*---------A---------*---------B---------*---------C---------*----------
NODE: 818         VOLTS:    1.016   -2.27                                      MAG/ANG
                    -LD:      .00     .00                                      kW/kVR
 kVll  24.900       CAP:              .00                                      kVR

FROM NODE 816   .....:    13.03  -26.77                                      AMP/DG
<818   > LOSS=    .154:     (    .154)                                       kW
TO NODE 820    .......:    13.03  -26.77                                      AMP/DG
<820   > LOSS=   3.614:     (   3.614)                                       kW
--------------------*---------A---------*---------B---------*---------C---------*----------
NODE: 820         VOLTS:     .993   -2.32                                      MAG/ANG
                    -LD:      .00     .00                                      kW/kVR
 kVll  24.900       CAP:              .00                                      kVR

FROM NODE 818   .....:    10.62  -28.98                                      AMP/DG
<820   > LOSS=   3.614:     (   3.614)                                       kW
TO NODE 822    .......:    10.62  -28.98                                      AMP/DG
<822   > LOSS=    .413:     (    .413)                                       kW
```

```
————————————*————A————*————B————*————C————*————
NODE: 822          VOLTS:     .990   -2.33                              MAG/ANG
                    -LD:     .00      .00                               kW/kVR
 kVll  24.900       CAP:              .00                               kVR

FROM NODE 820     .....:     .00      .00                               AMP/DG
<822    > LOSS=    .413:     (    .413)                                 kW
————————————*————A————*————B————*————C————*————
NODE: 824          VOLTS:   1.008   -2.37   1.016 -122.94   1.012  117.76 MAG/ANG
                    -LD:     .00      .00     .00      .00     .00      .00 kW/kVR
 kVll  24.900       CAP:              .00              .00              .00 kVR

FROM NODE 816     .....:   35.87   -10.70   39.82 -129.02   38.05  116.25 AMP/DG
<824    > LOSS= 14.181:     (   4.312)       (   5.444)       (   4.425)   kW
TO NODE 826       .....:                     3.10 -148.92                  AMP/DG
<826    > LOSS=    .008:                    (    .008)                    kW
TO NODE 828       .....:   35.87   -10.70   36.93 -127.39   38.05  116.25 AMP/DG
<828    > LOSS=   1.108:     (    .361)       (    .393)       (    .354)   kW
————————————*————A————*————B————*————C————*————
NODE: 826          VOLTS:                   1.016 -122.94                  MAG/ANG
                    -LD:                      .00      .00                  kW/kVR
 kVll  24.900       CAP:                               .00                  kVR

FROM NODE 824     .....:                      .00      .00                  AMP/DG
<826    > LOSS=    .008:                    (    .008)                    kW
————————————*————A————*————B————*————C————*————
NODE: 828          VOLTS:   1.007   -2.38   1.015 -122.95   1.011  117.75 MAG/ANG
                    -LD:     .00      .00     .00      .00     .00      .00 kW/kVR
 kVll  24.900       CAP:              .00              .00              .00 kVR

FROM NODE 824     .....:   35.87   -10.72   36.93 -127.41   37.77  116.42 AMP/DG
<828    > LOSS=   1.108:     (    .361)       (    .393)       (    .354)   kW
TO NODE 830       .....:   35.87   -10.72   36.93 -127.41   37.77  116.42 AMP/DG
<830    > LOSS= 26.587:     (   8.443)       (   9.214)       (   8.930)   kW
————————————*————A————*————B————*————C————*————
NODE: 830          VOLTS:    .989   -2.63    .998 -123.39    .994  117.25 MAG/ANG
                   D-LD:    9.95     4.98    9.86     4.93   24.55     9.82 kW/kVR
 kVll  24.900      Y CAP:              .00              .00              .00 kVR

FROM NODE 828     .....:   35.43   -11.06   36.91 -127.92   37.79  115.96 AMP/DG
<830    > LOSS= 26.587:     (   8.443)       (   9.214)       (   8.930)   kW
```

```
TO NODE 854    . . . . . :    34.22   -9.97    36.19 -127.47    36.49   116.26 AMP/DG
<854    > LOSS=    .635:     (    .197)         (    .227)         (    .211)     kW
-----------------------*---------A---------*---------B---------*---------C---------*---------
NODE: 854         VOLTS:     .989   -2.64     .998 -123.40     .993   117.24 MAG/ANG
                    -LD:      .00     .00      .00     .00      .00      .00 kW/kVR
 kVll   24.900      CAP:                .00               .00               .00 kVR

FROM NODE 830    . . . . . :    34.23   -9.99    36.19 -127.48    36.49   116.25 AMP/DG
<854    > LOSS=    .635:     (    .197)         (    .227)         (    .211)     kW
TO NODE 852    . . . . . :    34.23   -9.99    35.93 -127.72    36.49   116.25 AMP/DG
<852    > LOSS=  44.798:     ( 13.996)         ( 15.778)         ( 15.023)     kW
TO NODE 856    . . . . . :                      .31  -98.70                     AMP/DG
<856    > LOSS=    .001:                     (    .001)                        kW
-----------------------*---------A---------*---------B---------*---------C---------*---------
NODE: 852         VOLTS:     .958   -3.11     .968 -124.18     .964   116.33 MAG/ANG
                    -LD:      .00     .00      .00     .00      .00      .00 kW/kVR
 kVll   24.900      CAP:                .00               .00               .00 kVR

FROM NODE 854    . . . . . :    34.35   -11.00   35.90 -128.66    36.52   115.41 AMP/DG
<852    > LOSS=  44.798:     ( 13.996)         ( 15.778)         ( 15.023)     kW
TO NODE RG11  .<VRG>.:    34.35   -11.00   35.90 -128.66    36.52   115.41 AMP/DG
<RG11   > LOSS=    .000:     (    .000)         (    .000)         (    .000)     kW
-----------------------*---------A---------*---------B---------*---------C---------*---------
NODE: RG11        VOLTS:    1.036   -3.11    1.035 -124.18    1.036   116.33 MAG/ANG
                    -LD:      .00     .00      .00     .00      .00      .00 kW/kVR
 kVll   24.900      CAP:                .00               .00               .00 kVR

FROM NODE 852   <VRG>:    31.77   -11.00   33.59 -128.66    33.98   115.41 AMP/DG
<RG11   > LOSS=    .000:     (    .000)         (    .000)         (    .000)     kW
TO NODE 832    . . . . . :    31.77   -11.00   33.59 -128.66    33.98   115.41 AMP/DG
<832    > LOSS=    .011:     (    .003)         (    .004)         (    .004)     kW
-----------------------*---------A---------*---------B---------*---------C---------*---------
NODE: 832         VOLTS:    1.036   -3.11    1.035 -124.18    1.036   116.33 MAG/ANG
                    -LD:      .00     .00      .00     .00      .00      .00 kW/kVR
 kVll   24.900      CAP:                .00               .00               .00 kVR

FROM NODE RG11   . . . . . :    31.77   -11.00   33.59 -128.66    33.98   115.41 AMP/DG
<832    > LOSS=    .011:     (    .003)         (    .004)         (    .004)     kW
TO NODE 858    . . . . . :    21.31     .47    23.40 -116.89    24.34   128.36 AMP/DG
<858    > LOSS=   2.467:     (    .643)         (    .997)         (    .827)     kW
TO NODE XF10   . . . . . :    11.68   -32.29   11.70 -152.73    11.61    87.39 AMP/DG <
```

```
<XF10   > LOSS=  9.625:     (  3.196)        (  3.241)        (  3.187)     kW
————————————————*————————A————————*————————B————————*————————C————————*————————
NODE: 858        VOLTS:   1.034   −3.17   1.032 −124.28   1.034   116.22 MAG/ANG
                   –LD:     .00     .00     .00     .00     .00      .00 kW/kVR
 kVll  24.900      CAP:                .00                .00              .00 kVR

FROM NODE 832    .....:   20.86     .86   23.13 −116.39   24.02   128.48 AMP/DG
<858    > LOSS=  2.467:     (  .643)        (  .997)        (  .827)     kW
TO NODE 834      .......: 20.73    1.01   23.13 −116.39   24.02   128.48 AMP/DG
<834    > LOSS=  2.798:     (  .717)        (  1.145)       (  .936)     kW
TO NODE 864      .......:   .14  −22.82                                  AMP/DG
<864    > LOSS=   .000:     (  .000)                                     kW
————————————————*————————A————————*————————B————————*————————C————————*————————
NODE: 834        VOLTS:   1.031   −3.24   1.029 −124.39   1.031   116.09 MAG/ANG
                   –LD:     .00     .00     .00     .00     .00      .00 kW/kVR
 kVll  24.900      CAP:                .00                .00              .00 kVR

FROM NODE 858    .....:   20.29    2.18   22.37 −116.07   23.23   130.06 AMP/DG
<834    > LOSS=  2.798:     (  .717)        (  1.145)       (  .936)     kW
TO NODE 842      .......: 14.75   34.68   16.30  −95.63   15.12   151.05 AMP/DG
<842    > LOSS=   .064:     (  .015)        (  .032)        (  .017)     kW
TO NODE 860      .......: 11.16  −43.05    9.09 −154.82   10.60    99.34 AMP/DG
<860    > LOSS=   .141:     (  .021)        (  .104)        (  .017)     kW
————————————————*————————A————————*————————B————————*————————C————————*————————
NODE: 842        VOLTS:   1.031   −3.25   1.029 −124.39   1.031   116.09 MAG/ANG
                   –LD:     .00     .00     .00     .00     .00      .00 kW/kVR
 kVll  24.900      CAP:                .00                .00              .00 kVR

FROM NODE 834    .....:   14.74   34.67   16.30  −95.64   15.12   151.03 AMP/DG
<842    > LOSS=   .064:     (  .015)        (  .032)        (  .017)     kW
TO NODE 844      .......: 14.74   34.67   16.30  −95.64   15.12   151.03 AMP/DG
<844    > LOSS=   .306:     (  .068)        (  .156)        (  .083)     kW
————————————————*————————A————————*————————B————————*————————C————————*————————
NODE: 844        VOLTS:   1.031   −3.27   1.029 −124.42   1.031   116.06 MAG/ANG
                 Y–LD:  143.41   111.54  142.97   111.20  143.51   111.62 kW/kVR
 kVll  24.900  Y CAP:             106.23             105.90           106.31 kVR

FROM NODE 842    .....:   14.47   37.12   16.29  −95.71   15.11   150.97 AMP/DG
<844    > LOSS=   .306:     (  .068)        (  .156)        (  .083)     kW
TO NODE 846      .......:  9.83   78.88    9.40  −63.87    9.40  −170.67 AMP/DG
<846    > LOSS=   .323:     (  .043)        (  .212)        (  .068)     kW
```

		A		B		C		
NODE: 846	VOLTS:	1.031	−3.32	1.029	−124.46	1.031	116.01	MAG/ANG
	−LD:	.00	.00	.00	.00	.00	.00	kW/kVR
kVll 24.900	CAP:		.00		.00		.00	kVR
FROM NODE 844	.....:	9.76	78.80	9.40	−52.54	9.78	−161.93	AMP/DG
<846 > LOSS=	.323:	(	.043)	(	.212)	(	.068)	kW
TO NODE 848	.....:	9.76	78.80	9.40	−52.54	9.78	−161.93	AMP/DG
<848 > LOSS=	.048:	(	.007)	(	.031)	(	.010)	kW
NODE: 848	VOLTS:	1.031	−3.32	1.029	−124.47	1.031	116.00	MAG/ANG
	D−LD:	20.00	16.00	20.00	16.00	20.00	16.00	kW/kVR
kVll 24.900	Y CAP:		159.43		158.86		159.56	kVR
FROM NODE 846	.....:	9.76	78.79	9.77	−42.47	9.78	−161.94	AMP/DG
<848 > LOSS=	.048:	(	.007)	(	.031)	(	.010)	kW
NODE: 860	VOLTS:	1.030	−3.24	1.029	−124.39	1.031	116.09	MAG/ANG
	Y−LD:	20.00	16.00	20.00	16.00	20.00	16.00	kW/kVR
kVll 24.900	Y CAP:		.00		.00		.00	kVR
FROM NODE 834	.....:	5.87	−33.62	7.68	−156.52	5.29	86.10	AMP/DG
<860 > LOSS=	.141:	(	.021)	(	.104)	(	.017)	kW
TO NODE 836	.....:	4.16	−30.19	5.96	−154.63	3.60	90.25	AMP/DG
<836 > LOSS=	.039:	(	−.035)	(	.103)	(	−.028)	kW
NODE: 836	VOLTS:	1.030	−3.23	1.029	−124.39	1.031	116.09	MAG/ANG
	−LD:	.00	.00	.00	.00	.00	.00	kW/kVR
kVll 24.900	CAP:		.00		.00		.00	kVR
FROM NODE 860	.....:	1.49	−19.83	4.42	−150.74	1.74	68.08	AMP/DG
<836 > LOSS=	.039:	(	−.035)	(	.103)	(	−.028)	kW
TO NODE 840	.....:	1.50	−20.01	2.33	−151.97	1.75	68.00	AMP/DG
<840 > LOSS=	.002:	(	−.014)	(	.026)	(	−.010)	kW
TO NODE 862	.....:	.00	.00	2.09	−149.38	.00	.00	AMP/DG
<862 > LOSS=	.000:	(	−.005)	(	.009)	(	−.004)	kW
NODE: 840	VOLTS:	1.030	−3.23	1.029	−124.39	1.031	116.09	MAG/ANG
	Y−LD:	9.27	7.21	9.26	7.20	9.28	7.22	kW/kVR
kVll 24.900	Y CAP:		.00		.00		.00	kVR

```
FROM NODE 836     .....:       .79   -41.11       .79  -162.26       .79    78.21 AMP/DG
<840    > LOSS=    .002:     (   -.014)         (    .026)         (   -.010)     kW
-----------------------*----------A----------*----------B----------*----------C----------*----------
NODE: 862         VOLTS:    1.030   -3.23    1.029  -124.39    1.031   116.09 MAG/ANG
                    -LD:      .00     .00      .00     .00      .00      .00 kW/kVR
 kVll  24.900       CAP:              .00               .00               .00 kVR

FROM NODE 836     .....:       .00     .00     2.09  -149.50       .00     .00 AMP/DG
<862    > LOSS=    .000:     (   -.005)         (    .009)         (   -.004)     kW
TO NODE 838      .......:                       2.09  -149.50                     AMP/DG
<838    > LOSS=    .004:                       (    .004)                     kW
-----------------------*----------A----------*----------B----------*----------C----------*----------
NODE: 838         VOLTS:                     1.029  -124.39                     MAG/ANG
                    -LD:                       .00     .00                     kW/kVR
 kVll  24.900       CAP:                               .00                     kVR

FROM NODE 862     .....:                       .00     .00                     AMP/DG
<838    > LOSS=    .004:                       (    .004)                     kW
-----------------------*----------A----------*----------B----------*----------C----------*----------
NODE: 864         VOLTS:    1.034   -3.17                                     MAG/ANG
                    -LD:      .00     .00                                     kW/kVR
 kVll  24.900       CAP:              .00                                     kVR

FROM NODE 858     .....:       .00     .00                                     AMP/DG
<864    > LOSS=    .000:     (    .000)                                     kW
-----------------------*----------A----------*----------B----------*----------C----------*----------
NODE: XF10        VOLTS:    1.000   -4.63     .998  -125.73    1.000   114.82 MAG/ANG
                    -LD:      .00     .00      .00     .00      .00      .00 kW/kVR
 kVll   4.160       CAP:              .00               .00               .00 kVR

FROM NODE 832     .....:     69.90   -32.29    70.04  -152.73    69.50    87.39 AMP/DG <
<XF10   > LOSS=   9.625:     (  3.196)         (  3.241)         (  3.187)     kW
TO NODE 888      .......:    69.90   -32.29    70.04  -152.73    69.50    87.39 AMP/DG
<888    > LOSS=    .000:     (    .000)         (    .000)         (    .000)     kW
-----------------------*----------A----------*----------B----------*----------C----------*----------
NODE: 888         VOLTS:    1.000   -4.64     .998  -125.73    1.000   114.82 MAG/ANG
                    -LD:      .00     .00      .00     .00      .00      .00 kW/kVR
 kVll   4.160       CAP:              .00               .00               .00 kVR

FROM NODE XF10    .....:     69.90   -32.29    70.04  -152.73    69.50    87.39 AMP/DG
<888    > LOSS=    .000:     (    .000)         (    .000)         (    .000)     kW
```

```
TO NODE 890     . . . . . :    69.90  −32.29    70.04 −152.73    69.50    87.39 AMP/DG
<890    > LOSS= 32.760:      ( 11.638)        (  9.950)        ( 11.173)     kW
————————————————————*————————A————————*————————B————————*————————C————————*————————
NODE: 890          VOLTS:      .917   −5.19     .924 −126.78     .918   113.98 MAG/ANG
                     D–LD:   139.11    69.55  137.56    68.78  137.01    68.50 kW/kVR
 kVll   4.160       Y CAP:                .00                .00                .00 kVR

FROM NODE 888    . . . . . :    69.91  −32.31    70.05 −152.75    69.51    87.37 AMP/DG
<890    > LOSS= 32.760:      ( 11.638)        (  9.950)        ( 11.173)     kW
————————————————————*————————A————————*————————B————————*————————C————————*————————
NODE: 856          VOLTS:                        .998 −123.41                    MAG/ANG
                     –LD:                        .00       .00                   kW/kVR
 kVll  24.900        CAP:                                  .00                   kVR

FROM NODE 854    . . . . . :                     .00       .00                   AMP/DG
<856    > LOSS=    .001:                      (    .001)                        kW
```

Appendix B: PMU Data Generating Code

B.1 Generate scenarios

Because the model was built on MATLAB, the PMU data generating coding part was also wrote on MATLAB. The shown code is for 2019b version, and annotated some codes for other versions that have be to changed.

```
% 1) Load model==========================================
model = 'IEEE_34_node_2019b_scenarios';
load_system(model);

power_34NodeTestFeeder_loads_init
power_34NodeTestFeeder_init

% 2) Set up the sweep parameters=============================
S1 = [0,0,0,0,0];
S2 = [1,0,0,0,0];
S3 = [1,1,0,0,0];
S4 = [1,0,1,0,0];
S5 = [1,0,1,0,1];
S6 = [1,0,1,1,0];
S7 = [1,0,1,1,1];
S8 = [1,1,1,0,0];
S9 = [1,1,1,0,1];
S10 = [1,1,1,1,0];
S11 = [1,1,1,1,1];

Load1 = [Pa_824_828,Pp_824_828,Pn_824_828];
Load2 = [Pa_820_822,Pp_820_822,Pn_820_822];
Load3 = [Pa_858_834,Pp_858_834,Pn_858_834];
Load4 = [Pa_844,Pp_844,Pn_844];
Load5 = [Pa_840,Pp_840,Pn_840];

L1 = Load1;
L2 = Load2;
L3 = Load3;
L4 = Load4;
```

```
L5 = Load5;

% 3) Generate loads data for 9 scenarios====================
current_path=pwd; % read path
T=datestr(now,'mm-dd-yyyy-HH_MM');     % read current time

% For Scenario 1 (Topology 3)
SW=S3;
m=3;
n=1;
folder_name = ['Topology',num2str(m)];
Dirocry = [current_path,'\DataFolder\','DataGeneratedAt---',T,'\',folder_name];
mkdir(Dirocry);
for a=linspace(-0.05,0.05,40)
    for b=linspace(-0.05,0.05,40)
        Load = [L1 L2 L3 L4 L5];
        Load=[(1+a)*Load(1:9) (1+b)*Load(10:12) Load(13:end)];
        save_data(Load,SW,n,Dirocry);
        n=n+1;
    end
end
m=m+1;

% For Scenario 2 (Topology 4)
SW=S4;
n=1;
folder_name = ['Topology',num2str(m)];
Dirocry = [current_path,'\DataFolder\','DataGeneratedAt---',T,'\',folder_name];
mkdir(Dirocry);
for a=linspace(-0.05,0.05,13)
    for b=linspace(-0.05,0.05,13)
        for c=linspace(-0.05,0.05,13)
            Load = [L1 L2 L3 L4 L5];
            Load=[(1+a)*Load(1:9) Load(10:12) (1+b)*Load(13:21) Load(22:30) (1+c)*Load(31:
                end)];
            save_data(Load,SW,n,Dirocry);
            n=n+1;
        end
    end
end
m=m+1;
```

```
% For Scenario 3 (Topology 5)
SW=S5;
n=1;
folder_name = ['Topology',num2str(m)];
Dirocry = [current_path,'\DataFolder\','DataGeneratedAt---',T,'\',folder_name];
mkdir(Dirocry);
for a=linspace(-0.05,0.05,13)
    for b=linspace(-0.05,0.05,13)
        for c=linspace(-0.05,0.05,13)
            Load = [L1 L2 L3 L4 L5];
            Load=[(1+a)*Load(1:9) Load(10:12) (1+b)*Load(13:21) Load(22:30) (1+c)*Load(31:
                end)];
            save_data(Load,SW,n,Dirocry);
            n=n+1;
        end
    end
end
m=m+1;

% For Scenario 4 (Topology 6)
SW=S6;
n=1;
folder_name = ['Topology',num2str(m)];
Dirocry = [current_path,'\DataFolder\','DataGeneratedAt---',T,'\',folder_name];
mkdir(Dirocry);
for a=linspace(-0.05,0.05,7)
    for b=linspace(-0.05,0.05,7)
        for c=linspace(-0.05,0.05,7)
            for d=linspace(-0.05,0.05,7)
                Load = [L1 L2 L3 L4 L5];
                Load=[(1+a)*Load(1:9) Load(10:12) (1+b)*Load(13:21) (1+c)*Load(22:30) (1+d
                    )*Load(31:end)];
                save_data(Load,SW,n,Dirocry);
                n=n+1;
            end
        end
    end
end
m=m+1;
```

```
% For Scenario 5 (Topology 7)
SW=S7;
n=1;
folder_name = ['Topology',num2str(m)];
Dirocry = [current_path,'\DataFolder\','DataGeneratedAt---',T,'\',folder_name];
mkdir(Dirocry);
for a=linspace(-0.05,0.05,7)
    for b=linspace(-0.05,0.05,7)
        for c=linspace(-0.05,0.05,7)
            for d=linspace(-0.05,0.05,7)
                Load = [L1 L2 L3 L4 L5];
                Load=[(1+a)*Load(1:9) Load(10:12) (1+b)*Load(13:21) (1+c)*Load(22:30) (1+d
                    )*Load(31:end)];
                save_data(Load,SW,n,Dirocry);
                n=n+1;
            end
        end
    end
end
m=m+1;

% For Scenario 6 (Topology 8)
SW=S8;
n=1;
folder_name = ['Topology',num2str(m)];
Dirocry = [current_path,'\DataFolder\','DataGeneratedAt---',T,'\',folder_name];
mkdir(Dirocry);
for a=linspace(-0.05,0.05,7)
    for b=linspace(-0.05,0.05,7)
        for c=linspace(-0.05,0.05,7)
            for d=linspace(-0.05,0.05,7)
                Load = [L1 L2 L3 L4 L5];
                Load=[(1+a)*Load(1:9) (1+b)*Load(10:12) (1+c)*Load(13:21) Load(22:30) (1+d
                    )*Load(31:end)];
                save_data(Load,SW,n,Dirocry);
                n=n+1;
            end
        end
    end
end
m=m+1;
```

```
% For Scenario 7 (Topology 9)
SW=S9;
n=1;
folder_name = ['Topology',num2str(m)];
Dirocry = [current_path,'\DataFolder\','DataGeneratedAt---',T,'\',folder_name];
mkdir(Dirocry);
for a=linspace(-0.05,0.05,7)
    for b=linspace(-0.05,0.05,7)
        for c=linspace(-0.05,0.05,7)
            for d=linspace(-0.05,0.05,7)
                Load = [L1 L2 L3 L4 L5];
                Load=[(1+a)*Load(1:9) (1+b)*Load(10:12) (1+c)*Load(13:21) Load(22:30) (1+d
                    )*Load(31:end)];
                save_data(Load,SW,n,Dirocry);
                n=n+1;
            end
        end
    end
end
m=m+1;

% For Scenario 8 (Topology 10)
SW=S10;
n=1;
folder_name = ['Topology',num2str(m)];
Dirocry = [current_path,'\DataFolder\','DataGeneratedAt---',T,'\',folder_name];
mkdir(Dirocry);
for a=linspace(-0.05,0.05,5)
    for b=linspace(-0.05,0.05,5)
        for c=linspace(-0.05,0.05,5)
            for d=linspace(-0.05,0.05,5)
                for e=linspace(-0.05,0.05,5)
                    Load = [L1 L2 L3 L4 L5];
                    Load=[(1+a)*Load(1:9) (1+b)*Load(10:12) (1+c)*Load(13:21) (1+d)*Load
                        (22:30) (1+e)*Load(31:end)];
                    save_data(Load,SW,n,Dirocry);
                    n=n+1;
                end
            end
        end
```

```
    end
end
m=m+1;

% For Scenario 9 (Topology 11)
SW=S11;
n=1;
folder_name = ['Topology',num2str(m)];
Dirocry = [current_path,'\DataFolder\','DataGeneratedAt---',T,'\',folder_name];
mkdir(Dirocry);
for a=linspace(-0.05,0.05,5)
    for b=linspace(-0.05,0.05,5)
        for c=linspace(-0.05,0.05,5)
            for d=linspace(-0.05,0.05,5)
                for e=linspace(-0.05,0.05,5)
                    Load = [L1 L2 L3 L4 L5];
                    Load=[(1+a)*Load(1:9) (1+b)*Load(10:12) (1+c)*Load(13:21) (1+d)*Load
                        (22:30) (1+e)*Load(31:end)];
                    save_data(Load,SW,n,Dirocry);
                    n=n+1;
                end
            end
        end
    end
end
```

The "save_data" function is written as:

```
function save_data(Load,SW,n,Dirocry)
load1=Load(1,1:9);
load2=Load(1,10:12);
load3=Load(1,13:21);
load4=Load(1,22:30);
load5=Load(1,31:39);
result=table(SW,load1,load2,load3,load4,load5);
file_name = [Dirocry,'\',num2str(n),'.csv'];
writetable(result,file_name);
end
```

B.2 Generate PMU Data

Since the data corresponding to different percentages of loads have been generated under 9 topologies with different switch status scenarios, the PMU data could be generated with the MATLAB Simulink toolbox. Here, we used an approach which is a parallel simulation, that could save 80% time for running the entire process.

It should be noted that some functions would be affected by different versions of MATLAB, such as "readmatrix" and "writematrix" function could only be used in the newer versions of MATLAB 2019a; for previous versions, they should be replaced by "csvread" (or "dlmread") and "csvwrite" if the data need to be read from .csv file and save in .csv file. In addition, in MATLAB 2020a and Linux MATLAB, the "parsim" calculation should be defined detailed into each core with "parfevalOnAll" function.

```
% Run Parallel Simulations============================
simu_number=5;    % Set the number for one time parallel simulation
folder_name=[current_path,'\DataFolder\'];

for topology=3:11
    list=dir([folder_name,T,'\Topology',num2str(topology),'\','*.csv']);
    len=length(list);
    Dirocry = [folder_name,'PMUresults','\Topology',num2str(topology),'\'];
    mkdir(Dirocry);
    counter=1;
    num=1;

    for loop=1:(len/simu_number)    % 1st running
        clear simIn
        for idx = 1:simu_number
            filename=[folder_name,T,'\Topology',num2str(topology),'\',num2str(counter),'.
                csv'];
            SW=readmatrix(filename,'Range','A2:E2');      % After 2019a
            load=readmatrix(filename,'Range','F2:AR2');
            % SW=csvread(filename,1,0,['A2..E2']); % dlmread
```

```
% load=csvread(filename,1,0,['F2..AR2']);
% load=load(2,2:40);

% Switch
SW850_816 = SW(1,1);
SW818_820 = SW(1,2);
SW832_858 = SW(1,3);
SW834_842 = SW(1,4);
SW836_862 = SW(1,5);

% Load 1
Pa_824_828 = table2array(table([load(1,1) load(1,2) load(1,3)]));
Pp_824_828 = table2array(table([load(1,4) load(1,5) load(1,6)]));
Pn_824_828 = table2array(table([load(1,7) load(1,8) load(1,9)]));

% Load 2
Pa_820_822 = table2array(table([load(1,10)]));
Pp_820_822 = table2array(table([load(1,11)]));
Pn_820_822 = table2array(table([load(1,12)]));

% Load 3
Pa_858_834 = table2array(table([load(1,13) load(1,14) load(1,15)]));
Pp_858_834 = table2array(table([load(1,16) load(1,17) load(1,18)]));
Pn_858_834 = table2array(table([load(1,19) load(1,20) load(1,21)]));

% Load 4
Pa_844 = table2array(table([load(1,22) load(1,23) load(1,24)]));
Pp_844 = table2array(table([load(1,25) load(1,26) load(1,27)]));
Pn_844 = table2array(table([load(1,28) load(1,29) load(1,30)]));

% Load 5
Pa_840 = table2array(table([load(1,31) load(1,32) load(1,33)]));
Pp_840 = table2array(table([load(1,34) load(1,35) load(1,36)]));
Pn_840 = table2array(table([load(1,37) load(1,38) load(1,39)]));

% load(idx)
simIn(idx) = Simulink.SimulationInput(model);
simIn(idx) = simIn(idx).setVariable('SimulationMode','Accelerator');

simIn(idx) = simIn(idx).setVariable('SW850_816',SW850_816);
simIn(idx) = simIn(idx).setVariable('SW818_820',SW818_820);
```

```
        simIn(idx) = simIn(idx).setVariable('SW832_858',SW832_858);
        simIn(idx) = simIn(idx).setVariable('SW834_842',SW834_842);
        simIn(idx) = simIn(idx).setVariable('SW836_862',SW836_862);

        simIn(idx) = simIn(idx).setVariable('Pa_824_828',Pa_824_828);
        simIn(idx) = simIn(idx).setVariable('Pp_824_828',Pp_824_828);
        simIn(idx) = simIn(idx).setVariable('Pn_824_828',Pn_824_828);

        simIn(idx) = simIn(idx).setVariable('Pa_820_822',Pa_820_822);
        simIn(idx) = simIn(idx).setVariable('Pp_820_822',Pp_820_822);
        simIn(idx) = simIn(idx).setVariable('Pn_820_822',Pn_820_822);

        simIn(idx) = simIn(idx).setVariable('Pa_858_834',Pa_858_834);
        simIn(idx) = simIn(idx).setVariable('Pp_858_834',Pp_858_834);
        simIn(idx) = simIn(idx).setVariable('Pn_858_834',Pn_858_834);

        simIn(idx) = simIn(idx).setVariable('Pa_844',Pa_844);
        simIn(idx) = simIn(idx).setVariable('Pp_844',Pp_844);
        simIn(idx) = simIn(idx).setVariable('Pn_844',Pn_844);

        simIn(idx) = simIn(idx).setVariable('Pa_840',Pa_840);
        simIn(idx) = simIn(idx).setVariable('Pp_840',Pp_840);
        simIn(idx) = simIn(idx).setVariable('Pn_840',Pn_840);

        counter=counter+1;
    end

    simOut = parsim(simIn,'ShowSimulationManager','off');

    for round=1:simu_number
        PMU802=[simOut(round).logsout{34}.Values.Data(1,:) simOut(round).logsout{1}.
            Values.Data(1,:)];  %1
        PMU806=[simOut(round).logsout{35}.Values.Data(1,:) simOut(round).logsout{2}.
            Values.Data(1,:)];  %2
        PMU808=[simOut(round).logsout{36}.Values.Data(1,:) simOut(round).logsout{3}.
            Values.Data(1,:)];  %3
        PMU810=[simOut(round).logsout{37}.Values.Data(1,:) simOut(round).logsout{4}.
            Values.Data(1,:)];  %4
        PMU812=[simOut(round).logsout{38}.Values.Data(1,:) simOut(round).logsout{5}.
            Values.Data(1,:)];  %5
        PMU814=[simOut(round).logsout{39}.Values.Data(1,:) simOut(round).logsout{6}.
```

```
        Values.Data(1,:)];   %6
PMU850=[simOut(round).logsout{57}.Values.Data(1,:) simOut(round).logsout{24}.
        Values.Data(1,:)];   %7
PMU816=[simOut(round).logsout{40}.Values.Data(1,:) simOut(round).logsout{7}.
        Values.Data(1,:)];   %8
PMU818=[simOut(round).logsout{41}.Values.Data(1,:) simOut(round).logsout{8}.
        Values.Data(1,:)];   %9
PMU820=[simOut(round).logsout{42}.Values.Data(1,:) simOut(round).logsout{9}.
        Values.Data(1,:)];   %10
PMU822=[simOut(round).logsout{43}.Values.Data(1,:) simOut(round).logsout{10}.
        Values.Data(1,:)];   %11
PMU824=[simOut(round).logsout{44}.Values.Data(1,:) simOut(round).logsout{11}.
        Values.Data(1,:)];   %12
PMU826=[simOut(round).logsout{45}.Values.Data(1,:) simOut(round).logsout{12}.
        Values.Data(1,:)];   %13
PMU828=[simOut(round).logsout{46}.Values.Data(1,:) simOut(round).logsout{13}.
        Values.Data(1,:)];   %14
PMU830=[simOut(round).logsout{47}.Values.Data(1,:) simOut(round).logsout{14}.
        Values.Data(1,:)];   %15
PMU854=[simOut(round).logsout{59}.Values.Data(1,:) simOut(round).logsout{26}.
        Values.Data(1,:)];   %16
PMU856=[simOut(round).logsout{60}.Values.Data(1,:) simOut(round).logsout{27}.
        Values.Data(1,:)];   %17
PMU852=[simOut(round).logsout{58}.Values.Data(1,:) simOut(round).logsout{25}.
        Values.Data(1,:)];   %18
PMU832=[simOut(round).logsout{48}.Values.Data(1,:) simOut(round).logsout{15}.
        Values.Data(1,:)];   %19
PMU888=[simOut(round).logsout{65}.Values.Data(1,:) simOut(round).logsout{32}.
        Values.Data(1,:)];   %20
PMU890=[simOut(round).logsout{66}.Values.Data(1,:) simOut(round).logsout{33}.
        Values.Data(1,:)];   %21
PMU858=[simOut(round).logsout{61}.Values.Data(1,:) simOut(round).logsout{28}.
        Values.Data(1,:)];   %22
PMU864=[simOut(round).logsout{64}.Values.Data(1,:) simOut(round).logsout{31}.
        Values.Data(1,:)];   %23
PMU834=[simOut(round).logsout{49}.Values.Data(1,:) simOut(round).logsout{16}.
        Values.Data(1,:)];   %24
PMU842=[simOut(round).logsout{53}.Values.Data(1,:) simOut(round).logsout{20}.
        Values.Data(1,:)];   %25
PMU844=[simOut(round).logsout{54}.Values.Data(1,:) simOut(round).logsout{21}.
        Values.Data(1,:)];   %26
```

```
            PMU846=[simOut(round).logsout{55}.Values.Data(1,:) simOut(round).logsout{22}.
                Values.Data(1,:)];  %27
            PMU848=[simOut(round).logsout{56}.Values.Data(1,:) simOut(round).logsout{23}.
                Values.Data(1,:)];  %28
            PMU860=[simOut(round).logsout{62}.Values.Data(1,:) simOut(round).logsout{29}.
                Values.Data(1,:)];  %29
            PMU836=[simOut(round).logsout{50}.Values.Data(1,:) simOut(round).logsout{17}.
                Values.Data(1,:)];  %30
            PMU840=[simOut(round).logsout{52}.Values.Data(1,:) simOut(round).logsout{19}.
                Values.Data(1,:)];  %31
            PMU862=[simOut(round).logsout{63}.Values.Data(1,:) simOut(round).logsout{30}.
                Values.Data(1,:)];  %32
            PMU838=[simOut(round).logsout{51}.Values.Data(1,:) simOut(round).logsout{18}.
                Values.Data(1,:)];  %33
            PMU=[PMU802;PMU806;PMU808;PMU810;PMU812;PMU814;PMU850;PMU816;PMU818;PMU820;
                PMU822;PMU824;PMU826;PMU828;PMU830;PMU854;PMU856;PMU852;PMU832;PMU888;
                PMU890;PMU858;PMU864;PMU834;PMU842;PMU844;PMU846;PMU848;PMU860;PMU836;
                PMU840;PMU862;PMU838];
            file_name = [Dirocry,'\',num2str(num),'.csv'];
            writematrix(PMU,file_name);     % After 2019a
            %csvwrite(file_name,PMU);
            num=num+1;
        end
    end

    if rem(len,simu_number)~=0
        % if the remainder is not zero, run a 2nd time
        clear simIn
        for idx=1:(len-counter+1)       % 2nd running
            filename=[folder_name,T,'\Topology',num2str(topology),'\',num2str(counter),'.
                csv'];
            SW=readmatrix(filename,'Range','A2:E2');
            load=readmatrix(filename,'Range','F2:AR2');
            %SW=csvread(filename,1,0,['A2..E2']);    %dlmread
            %load=csvread(filename,1,0,['F2..AR2']);
            %load=load(2,2:40);

            % switch
            SW850_816 = SW(1,1);
            SW818_820 = SW(1,2);
            SW832_858 = SW(1,3);
```

```
SW834_842 = SW(1,4);
SW836_862 = SW(1,5);

% Load 1
Pa_824_828 = table2array(table([load(1,1) load(1,2) load(1,3)]));
Pp_824_828 = table2array(table([load(1,4) load(1,5) load(1,6)]));
Pn_824_828 = table2array(table([load(1,7) load(1,8) load(1,9)]));

% Load 2
Pa_820_822 = table2array(table([load(1,10)]));
Pp_820_822 = table2array(table([load(1,11)]));
Pn_820_822 = table2array(table([load(1,12)]));

% Load 3
Pa_858_834 = table2array(table([load(1,13) load(1,14) load(1,15)]));
Pp_858_834 = table2array(table([load(1,16) load(1,17) load(1,18)]));
Pn_858_834 = table2array(table([load(1,19) load(1,20) load(1,21)]));

% Load 4
Pa_844 = table2array(table([load(1,22) load(1,23) load(1,24)]));
Pp_844 = table2array(table([load(1,25) load(1,26) load(1,27)]));
Pn_844 = table2array(table([load(1,28) load(1,29) load(1,30)]));

% Load 5
Pa_840 = table2array(table([load(1,31) load(1,32) load(1,33)]));
Pp_840 = table2array(table([load(1,34) load(1,35) load(1,36)]));
Pn_840 = table2array(table([load(1,37) load(1,38) load(1,39)]));

simIn(idx) = Simulink.SimulationInput(model);
simIn(idx) = simIn(idx).setVariable('SimulationMode', 'Accelerator');

simIn(idx) = simIn(idx).setVariable('SW850_816', SW850_816);
simIn(idx) = simIn(idx).setVariable('SW818_820', SW818_820);
simIn(idx) = simIn(idx).setVariable('SW832_858', SW832_858);
simIn(idx) = simIn(idx).setVariable('SW834_842', SW834_842);
simIn(idx) = simIn(idx).setVariable('SW836_862', SW836_862);

simIn(idx) = simIn(idx).setVariable('Pa_824_828', Pa_824_828);
simIn(idx) = simIn(idx).setVariable('Pp_824_828', Pp_824_828);
simIn(idx) = simIn(idx).setVariable('Pn_824_828', Pn_824_828);
```

```
        simIn(idx) = simIn(idx).setVariable('Pa_820_822', Pa_820_822);
        simIn(idx) = simIn(idx).setVariable('Pp_820_822', Pp_820_822);
        simIn(idx) = simIn(idx).setVariable('Pn_820_822', Pn_820_822);

        simIn(idx) = simIn(idx).setVariable('Pa_858_834', Pa_858_834);
        simIn(idx) = simIn(idx).setVariable('Pp_858_834', Pp_858_834);
        simIn(idx) = simIn(idx).setVariable('Pn_858_834', Pn_858_834);

        simIn(idx) = simIn(idx).setVariable('Pa_844', Pa_844);
        simIn(idx) = simIn(idx).setVariable('Pp_844', Pp_844);
        simIn(idx) = simIn(idx).setVariable('Pn_844', Pn_844);

        simIn(idx) = simIn(idx).setVariable('Pa_840', Pa_840);
        simIn(idx) = simIn(idx).setVariable('Pp_840', Pp_840);
        simIn(idx) = simIn(idx).setVariable('Pn_840', Pn_840);

        counter=counter+1;
    end

    simOut = parsim(simIn, 'ShowSimulationManager', 'off');

    for round=1:(len-num+1)
        PMU802=[simOut(round).logsout{34}.Values.Data(1,:) simOut(round).logsout{1}.
            Values.Data(1,:)];  %1
        PMU806=[simOut(round).logsout{35}.Values.Data(1,:) simOut(round).logsout{2}.
            Values.Data(1,:)];  %2
        PMU808=[simOut(round).logsout{36}.Values.Data(1,:) simOut(round).logsout{3}.
            Values.Data(1,:)];  %3
        PMU810=[simOut(round).logsout{37}.Values.Data(1,:) simOut(round).logsout{4}.
            Values.Data(1,:)];  %4
        PMU812=[simOut(round).logsout{38}.Values.Data(1,:) simOut(round).logsout{5}.
            Values.Data(1,:)];  %5
        PMU814=[simOut(round).logsout{39}.Values.Data(1,:) simOut(round).logsout{6}.
            Values.Data(1,:)];  %6
        PMU850=[simOut(round).logsout{57}.Values.Data(1,:) simOut(round).logsout{24}.
            Values.Data(1,:)];  %7
        PMU816=[simOut(round).logsout{40}.Values.Data(1,:) simOut(round).logsout{7}.
            Values.Data(1,:)];  %8
        PMU818=[simOut(round).logsout{41}.Values.Data(1,:) simOut(round).logsout{8}.
            Values.Data(1,:)];  %9
        PMU820=[simOut(round).logsout{42}.Values.Data(1,:) simOut(round).logsout{9}.
```

```
        Values.Data(1,:)];   %10
PMU822=[simOut(round).logsout{43}.Values.Data(1,:) simOut(round).logsout{10}.
        Values.Data(1,:)];   %11
PMU824=[simOut(round).logsout{44}.Values.Data(1,:) simOut(round).logsout{11}.
        Values.Data(1,:)];   %12
PMU826=[simOut(round).logsout{45}.Values.Data(1,:) simOut(round).logsout{12}.
        Values.Data(1,:)];   %13
PMU828=[simOut(round).logsout{46}.Values.Data(1,:) simOut(round).logsout{13}.
        Values.Data(1,:)];   %14
PMU830=[simOut(round).logsout{47}.Values.Data(1,:) simOut(round).logsout{14}.
        Values.Data(1,:)];   %15
PMU854=[simOut(round).logsout{59}.Values.Data(1,:) simOut(round).logsout{26}.
        Values.Data(1,:)];   %16
PMU856=[simOut(round).logsout{60}.Values.Data(1,:) simOut(round).logsout{27}.
        Values.Data(1,:)];   %17
PMU852=[simOut(round).logsout{58}.Values.Data(1,:) simOut(round).logsout{25}.
        Values.Data(1,:)];   %18
PMU832=[simOut(round).logsout{48}.Values.Data(1,:) simOut(round).logsout{15}.
        Values.Data(1,:)];   %19
PMU888=[simOut(round).logsout{65}.Values.Data(1,:) simOut(round).logsout{32}.
        Values.Data(1,:)];   %20
PMU890=[simOut(round).logsout{66}.Values.Data(1,:) simOut(round).logsout{33}.
        Values.Data(1,:)];   %21
PMU858=[simOut(round).logsout{61}.Values.Data(1,:) simOut(round).logsout{28}.
        Values.Data(1,:)];   %22
PMU864=[simOut(round).logsout{64}.Values.Data(1,:) simOut(round).logsout{31}.
        Values.Data(1,:)];   %23
PMU834=[simOut(round).logsout{49}.Values.Data(1,:) simOut(round).logsout{16}.
        Values.Data(1,:)];   %24
PMU842=[simOut(round).logsout{53}.Values.Data(1,:) simOut(round).logsout{20}.
        Values.Data(1,:)];   %25
PMU844=[simOut(round).logsout{54}.Values.Data(1,:) simOut(round).logsout{21}.
        Values.Data(1,:)];   %26
PMU846=[simOut(round).logsout{55}.Values.Data(1,:) simOut(round).logsout{22}.
        Values.Data(1,:)];   %27
PMU848=[simOut(round).logsout{56}.Values.Data(1,:) simOut(round).logsout{23}.
        Values.Data(1,:)];   %28
PMU860=[simOut(round).logsout{62}.Values.Data(1,:) simOut(round).logsout{29}.
        Values.Data(1,:)];   %29
PMU836=[simOut(round).logsout{50}.Values.Data(1,:) simOut(round).logsout{17}.
        Values.Data(1,:)];   %30
```

```
            PMU840=[simOut(round).logsout{52}.Values.Data(1,:) simOut(round).logsout{19}.
                Values.Data(1,:)];   %31
            PMU862=[simOut(round).logsout{63}.Values.Data(1,:) simOut(round).logsout{30}.
                Values.Data(1,:)];   %32
            PMU838=[simOut(round).logsout{51}.Values.Data(1,:) simOut(round).logsout{18}.
                Values.Data(1,:)];   %33
            PMU=[PMU802;PMU806;PMU808;PMU810;PMU812;PMU814;PMU850;PMU816;PMU818;PMU820;
                PMU822;PMU824;PMU826;PMU828;PMU830;PMU854;PMU856;PMU852;PMU832;PMU888;
                PMU890;PMU858;PMU864;PMU834;PMU842;PMU844;PMU846;PMU848;PMU860;PMU836;
                PMU840;PMU862;PMU838];
            file_name = [Dirocry,'\',num2str(num),'.csv'];
            writematrix(PMU,file_name);
            %csvwrite(file_name,PMU);
            num=num+1;
        end
    end
end
```

Because the voltage and current have already been converted into per unit forms when building the Simulink model, the next step is to convert the angle values measured by PMUs into "radian/π". So, the values would be shrunk between -1 to 1, which be suited for neural convolution. The code to generate the PMU data into per unit values is shows below:

```
% Save data into one .mat file
for topology=3:11
    list=dir([folder_name,'\PMUresults','\Topology',num2str(topology),'\*.csv']);
    len=length(list);
    for counter=1:len
        filename=[folder_name,'PMUresults\Topology',num2str(topology),'\',num2str(counter)
            ,'.csv'];
        Data{topology-2,counter}=readmatrix(filename,'Range','A1:L33');
    end
end
save Data.mat

% Save in sorted folders
clear
```

```
load('Data.mat')
for topology=3:11
    Dirocry = [folder_name,'PMU_PUresults\',num2str(topology-2),'\'];
    mkdir(Dirocry);
    list=dir([folder_name,'\PMUresults','\Topology',num2str(topology),'\*.csv']);
    len=length(list);
    for counter=1:len
        file_name=[Dirocry,num2str(counter),'.csv'];
        rad=Data{topology-2,counter}(:,[4,5,6,10,11,12])*pi/180;
        % Convert to radian
        rad=rad/pi;    % let rad between -1 and 1
        voltage=Data{topology-2,counter}(:,1:3);
        current=Data{topology-2,counter}(:,7:9);
        save_data=[voltage rad(:,[1,2,3]) current rad(:,[4,5,6])];
        writematrix(save_data,file_name);
    end
end
```

Appendix C: Convolutional Neural Network Code

C.1 Sorting Data

For CNN part, the code bellow was ran in Pycharm which used Python language.

```
#Sorting the data into three folders
#Copy the data into a new folder first
import os, random, shutil
def moveFile_Train(fileDir):
        pathDir = os.listdir(fileDir)
            #Read the original path of the file
        filenumber=len(pathDir)
        rate=0.8    #Proportion of extracted files
        picknumber=int(filenumber*rate) #Extract files in proportion
        sample = random.sample(pathDir, picknumber)
            #Randomly select a proportional number of files
        print (sample)
        for name in sample:
                shutil.move(fileDir+'/'+name, tarDir+'/'+name)
                #shutil.copyfile(fileDir+name, tarDir+name)
        return


def moveFile_Test(fileDir):
    pathDir = os.listdir(fileDir)
    filenumber = len(pathDir)
    rate = 0.5
    picknumber = int(filenumber * rate)
    sample = random.sample(pathDir, picknumber)
    print(sample)
    for name in sample:
        shutil.move(fileDir+'/'+name, tarDir+'/'+name)
    return

def moveFile_Val(fileDir):
    pathDir = os.listdir(fileDir)
    filenumber = len(pathDir)
```

```
    rate = 1
    picknumber = int(filenumber * rate)
    sample = random.sample(pathDir, picknumber)
    print(sample)
    for name in sample:
        shutil.move(fileDir+'/'+name, tarDir+'/'+name)
    return


if __name__ == '__main__':
    for num in range(1,10):
        path_Test='G:/34NODES_RUNNING/DataFolder/CNN/Test/'+str(num)
        path_Train = 'G:/34NODES_RUNNING/DataFolder/CNN/Train/' + str(num)
        path_Val = 'G:/34NODES_RUNNING/DataFolder/CNN/Val/' + str(num)
        try:
            os.mkdir(path_Train)
            os.mkdir(path_Test)
            os.mkdir(path_Val)
        except OSError:
            print("Creation of the directory %s failed" % path_Train)
            print("Creation of the directory %s failed" % path_Test)
            print("Creation of the directory %s failed" % path_Val)
        else:
            print("Successfully created the directory %s" % path_Train)
            print("Successfully created the directory %s" % path_Test)
            print("Successfully created the directory %s" % path_Val)

    for num in range(1,10):
        fileDir = "G:/34NODES_RUNNING/DataFolder/CNN/PMU_PUresults/"+str(num)     #Source
            file folder path
        tarDir = 'G:/34NODES_RUNNING/DataFolder/CNN/Train/'+str(num)              #Move to
            new folder path
        moveFile_Train(fileDir)        #move to Train folder

    for num in range(1,10):
        fileDir = "G:/34NODES_RUNNING/DataFolder/CNN/PMU_PUresults/"+str(num)
        tarDir = 'G:/34NODES_RUNNING/DataFolder/CNN/Test/'+str(num)
        moveFile_Test(fileDir)        #move to Test folder

    for num in range(1,10):
        fileDir = "G:/34NODES_RUNNING/DataFolder/CNN/PMU_PUresults/"+str(num)
```

```
        tarDir = 'G:/34NODES_RUNNING/DataFolder/CNN/Val/'+str(num)
        moveFile_Val(fileDir)        #move to Val folder
```

C.2 Calculate the mean and standard deviation

```
import os
from glob import glob
import numpy as np
import pandas as pd

def mean_std(root):
    class_dir = glob(os.path.join(root, '*/'))
    for dir in class_dir:
        print(dir, os.path.isdir(dir))
    total_data = []

    for dir in class_dir:
            this_data = glob(os.path.join(dir, '*.csv'))
            total_data += this_data
    print(len(total_data))

    batch_size = 100
    num_batch = len(total_data) // batch_size

    batch_means = []
    batch_vars = []
    print('num_batch = ', num_batch)

    for i_batch in range(num_batch):
        print('processing batch ', i_batch)
        batch_data = None
        start = i_batch * batch_size
        end = (i_batch + 1) * batch_size
        for file in total_data[start:end]:
            #print('processing file ', file)
            df = pd.read_csv(file, header=None)
            im = df.to_numpy()        #Convert <pandas.core.frame.DataFrame> to <numpy.
                ndarray>
```

```
            if batch_data is None:
                batch_data = im
            else:
                batch_data = np.vstack((batch_data, im))
            #print(np.shape(batch_data))
        #print(np.shape(batch_data))

        batch_mean0 = np.mean(batch_data)
        batch_var0 = np.var(batch_data)

        batch_means.append(batch_mean0)
        batch_vars.append(batch_var0)

    batch_means = np.stack(batch_means, axis=0)
    batch_vars = np.stack(batch_vars, axis=0)

    print(batch_means.shape)
    print(batch_vars.shape)

    mean = np.mean(batch_means, axis=0)
    var = np.mean(batch_vars, axis=0) + np.var(batch_means, axis=0)

    print('mean = ', mean)
    print('var = ', var)
    print('std = ', np.sqrt(var))
    return mean, np.sqrt(var)
```

C.3 Two Layer Net

```
import torch.nn as nn
import torch.nn.functional as F
import numpy as np

# Weight initialization
def weight_init(m):
    if isinstance(m, nn.Linear):
        size = m.weight.size()
        fan_out = size[0]    # number of rows
        fan_in = size[1]     # number of columns
```

```
        variance = np.sqrt(2.0/(fan_in + fan_out))
        m.weight.data.normal_(0.0, variance)
    elif isinstance(m, nn.Conv2d):
        k1, k2 = m.kernel_size
        # in_c = m.in_channels
        out_c = m.out_channels
        n = k1 * k2 * out_c
        variance = np.sqrt(2.0 / n)
        m.weight.data.normal_(0.0, variance)


def num_flat_features(x):
    size = x.size()[1:]  # all dimensions except the batch dimension
    num_features = 1
    for s in size:
        num_features *= s
    return num_features

class TwoLayerConvNet(nn.Module):
    def __init__(self):
        super(TwoLayerConvNet, self).__init__()
        self.conv1 = nn.Conv2d(1, 15, kernel_size=(5, 3), stride=(1, 3))
        self.conv_bn1 = nn.BatchNorm2d(15)
        self.conv2 = nn.Conv2d(15, 20, kernel_size=3)
        self.conv_bn2 = nn.BatchNorm2d(20)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(20*27*2, 100)
        self.fc_bn1 = nn.BatchNorm1d(100)
        self.fc2 = nn.Linear(100, 9)
        weight_init(self.fc1)
        weight_init(self.fc2)



    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv_bn1(self.conv1(x)), kernel_size=(1, 1)))
        # print(x.shape)
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv_bn2(self.conv2(x))), kernel_size
            =(1,1)))
        # print(x.shape)
        x = x.view(-1, num_flat_features(x))
        x = F.relu(self.fc_bn1(self.fc1(x)))  # used to be F.tanh
        x = F.dropout(x, training=self.training)
```

```
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
```

C.4 Pandas Dataset Folder

Here a function is customized for the data input of the heatmap.

```
from torchvision.datasets import DatasetFolder
import torch
import numpy as np

class PandasDatasetFolder(DatasetFolder):
    def __getitem__(self, index):
        path, target = self.samples[index]
        sample = self.loader(path, header=None)
        # sample = np.as_matrix(sample)
        sample=sample.to_numpy()
        sample=sample[np.newaxis, ...]
        # print(np.shape(sample))
        # print('Sample Type: ', type(sample) )
        sample = torch.from_numpy(sample)
        # print(sample)
        if self.transform is not None:
            sample = self.transform(sample)
        if self.target_transform is not None:
            target = self.target_transform(target)

        return sample, target
```

C.5 Main Code

C.5.1 Cuda Parameters

```
no_cuda = False
is_cuda = not no_cuda and torch.cuda.is_available()
print("is_cuda = ", is_cuda)
```

```
kwargs = {'num_workers': 4, 'pin_memory': True} if is_cuda else {}
```

C.5.2 Load Data

```
data_path = 'G:\\XXXXXXXXXXXXXXXXXX'
data_file_names = ['Train', 'Val', 'Test']
data_sets = glob.glob(data_path)

for sub_data_path in data_sets:
    train_path = os.path.join(sub_data_path, 'Train')
    # ==== Get Mean STD ====
    print(train_path)
    mean, std = mean_std(train_path)    ## Commended before get MEAN std

currentloader = pd.read_csv

for data_file in data_file_names:
    data_file_path = os.path.join(data_path, data_file)
    if os.path.isdir(data_file_path) and data_file == 'Train':
        # train_dataset = DatasetFolder(root=data_file_path, loader=pd.read_csv,
            extensions='csv')
        train_dataset = PandasDatasetFolder(root=data_file_path, loader=currentloader,
            extensions='csv')
    elif os.path.isdir(data_file_path) and data_file == 'Val':
        # val_dataset = DatasetFolder(root=data_file_path, loader=pd.read_csv, extensions
            ='csv', transform=train_transform)
        val_dataset = PandasDatasetFolder(root=data_file_path, loader=currentloader,
            extensions='csv')
    elif os.path.isdir(data_file_path) and data_file == 'Test':
        # test_dataset = DatasetFolder(root=data_file_path, loader=pd.read_csv, extensions
            ='csv', transform=train_transform)
        test_dataset = PandasDatasetFolder(root=data_file_path, loader=currentloader,
            extensions='csv')
    else:
        raise RuntimeError('No data files found.')

train_loader = data_utils.DataLoader(dataset=train_dataset, batch_size=64, shuffle=True,
    **kwargs)
val_loader = data_utils.DataLoader(dataset=val_dataset, batch_size=32, shuffle=False, **
```

```
    kwargs)
test_loader = data_utils.DataLoader(dataset=test_dataset, batch_size=32, shuffle=False, **
    kwargs)

enumerate(train_loader)
```

C.5.3 Train Function

```
def train(epoch, print_period=100):
    model.train()
    # print(train_loader)
    for batch_idx, (data, target) in enumerate(train_loader):
        # convert data to tensor
        # data = torch.from_numpy(data.to_numpy())
        # normalize
        data -= mean
        data /= std
        # print(data.size())
        if is_cuda:
            data, target = data.cuda(), target.cuda()
            data = data.type('torch.cuda.FloatTensor')
            target = target.type('torch.cuda.LongTensor')
        else:
            data = data.type('torch.DoubleTensor')
            target = target.type('torch.LongTensor')

        # data = data[:, :2, :, :]
        target = target.squeeze()
        optimizer.zero_grad()
        output = model(data)
        with torch.enable_grad():
            loss = F.cross_entropy(output, target) #F.nll_loss(output, target)
            loss.backward()
        optimizer.step()

        if batch_idx % print_period == 0:
            print('Train Epoch: {} [{}/{}]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                loss.data))
```

```
    print('validation:')
    val_target_labels, val_pred_labels,val_acc = validate(model, val_loader, is_cuda=True)

    return val_target_labels, val_pred_labels, val_acc
```

C.5.4 Model and Parameter Setting

```
model = TwoLayerConvNet()
model = model.float()
if is_cuda:
    model.cuda()

optimizer = optim.Adam(model.parameters(), lr=1e-4,  weight_decay=0.00001)
scheduler = StepLR(optimizer, step_size=28, gamma=0.1)

num_epock = 1
print_period = 10

best_model_file_name = type(model).__name__ + '_best_checkpoint.pickle'
```

C.5.5 Training Loop

```
best_val_acc = 0
for epoch in range(num_epock):
    print('-------------------------------------------------------------')

    optimizer.zero_grad()
    val_target_labels, val_pred_labels, val_pred_acc = train(epoch, print_period=
        print_period)
    scheduler.step()
    if val_pred_acc > best_val_acc:
        best_val_acc = val_pred_acc

        torch.save({'state_dict':model.state_dict(),
                    'pred_labels':val_pred_labels,
                    'target_labels': val_target_labels,
                    'epoch': epoch,
                    'best_val_pred_acc': val_pred_acc,
```

```
                    'class2idx': train_dataset.class_to_idx,
                    'Mean_STD': np.array([mean, std]),
                    'Data_path': data_path}, best_model_file_name)

    print('Current best_val_acc: {:.4f}'.format(best_val_acc))
```

C.5.6 Training Loop

```
best_point = torch.load(best_model_file_name)
model.load_state_dict(best_point['state_dict'])

test_target_labels, test_pred_labels, test_acc = validate(model, test_loader, is_cuda=
    is_cuda)

print ('Best validation acc:\t', best_val_acc)
print ('Predicted test acc\t:', test_acc)

t = time.strftime("%Y%m%d_%H-%M-%S", time.localtime())
exist_check_point = os.path.isfile(best_model_file_name)
if exist_check_point:
    new_check_point_name = type(model).__name__ + 'best_chekpoint' + '{:.2f}'.format(
        best_val_acc) + '_' + t + '.pickle'
    os.rename(best_model_file_name, new_check_point_name)
```