Anomaly Detection in Smart Distribution Grids with Deep Neural Network

by

Ming Zhou

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

With the rapid development of smart grids, the detection of anomalies is essential to improve the quality and security protection of the grid. The identification of anomalies not only saves valuable time but also reduces maintenance costs. Due to the increasing deployment of distributed energy resources, traditional methods of protecting the grid that rely on simple linear models and manual inspections are no longer sufficient. Meanwhile, the massive amount of data generated by smart meters and phasor measurement units provide opportunities to better monitor and control power grids in real-time. Due to this advantage of data availability, various machine learning and deep learning methods have been proposed and are currently demonstrating successful results in anomaly detection in power systems.

While previously proposed artificial intelligence techniques can successfully detect anomalies, most of them tend to require large amounts of simulated data of all different types of anomalies for training their framework. However, anomalous data may be rare in power distribution systems. In addition, their static training model makes them vulnerable to new data from different distributions entering the system. To address these drawbacks, we propose data-driven frameworks based on deep learning network models to directly detect anomalies in power distribution systems. Anomalies are generally defined as observations that deviate from standard, normal or expected values. Specifically, this work is divided into two phases. In the first phase, we consider anomalies as events caused by changes in the distribution system load, such as customer disconnection from the grid. A long short-term memory network is proposed to predict the next time step of the voltage magnitude of all

buses in the distribution system. A threshold function based on Euclidean distance is then used to detect voltage anomalies by utilizing only normal data. The results corresponding to this proposed framework have been successfully tested using a real distribution network.

In the second phase, we aim to classify faults and locate faulted lines in partially observable distribution systems using convolutional neural networks. To improve the robustness of the classification and localization performance, we extract feature vectors with measurements in the observable buses as inputs to the proposed classifier. In addition, we incorporate an online continuous learning algorithm to accommodate variations in the level of integration of distributed energy resources and changes in the load of the distribution system over time. Unlike previous data-driven approaches, the proposed method also deals with imbalanced learning tasks, as fault data are often rare. The performance of the method has been tested and validated by simulating ten faults on a real distribution feeder model.

Preface

Some of the work in this thesis is a result of a publication, described in article M. Zhou, and P. Musilek, "Real-Time Anomaly Detection in Distribution Grids Using Long Short Term Memory Network," published in *2021 IEEE Electrical Power and Energy Conference (EPEC)*. Chapter 3 of this thesis is adopted from the publication. Section 2.4.1, anomaly detection state-of-the-art, Section 2.2.2, distribution test system, and Section 2.6.4, recurrent neural network and long short-term memory network, in Chapter 2 is also adopted from the publication. Chapter 4 of this thesis has been submitted to *Applied Soft Computing*, as M. Zhou, and P. Musilek, "Real-time Fault Classification and Localization in Partially Observable Distribution Systems using Convolutional Neural Networks," and the article is currently under review. The rest of this thesis is the sole original work of the author.

Acknowledgements

The author expresses her appreciation and gratitude to Dr. Petr Musilek for his supervision of this work. His advice and assistance in the preparation of this thesis are thankfully acknowledged. The author also wishes to thank Electrical and Computer Engineering Department and the Faculty of Engineering at the University of Alberta.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Electricity is essential in our daily life. Due to the high cost of obtaining energy and the limited energy resources, efficient and operational use of energy is an important aspect of the economy and society in every country. The rapid development of smart grids and increased penetration of distributed energy resources (DERs) has become a key solution for more efficient delivery of electricity and brings many significant benefits, such as system scalability and reduced demand peaks [1]. It also provides a platform for integrating customer-owned generation, including renewable energy systems [1].

In addition, the increasing number of advanced metering infrastructures in the smart grids leads to massive amounts of available data. This, in turn, provides opportunities for better real-time monitoring and control of electric utilities. However, the increased complexity of the grid has also increased the fundamental challenges for system operation. As a result of these changes in the distribution grid, traditional approaches to power system protection, based on simple linear models and manual checks, have limited performance when facing the new complexity and huge amounts of data. Failure to detect anomalies can result in significant financial losses and may even lead to power system collapse [2]. Therefore, it is important and urgent to develop effective methods that can address these challenges and detect anomalies as

quickly as possible. Subsequently, additional actions can be taken at the discretion of the distribution system operator to mitigate the anomaly or to analyze it further.

Machine learning (ML) techniques have undergone tremendous advances, recently achieving near-human performance on a variety of applications, such as email filtering, speech recognition, and computer vision. ML algorithms have also been widely used to effectively detect anomalies in various domains in power systems, such as energy consumption, cyber-attacks, and power grid line outages.

Anomalies are usually defined as observations that deviate from the standard, normal or expected values. The anomaly detection problem of interest in this work is based on analyzing the measured values of smart meters and sub-station data collected in the distribution network. Specifically, we aim to investigate two types of anomalies. 1. Events caused by load changes in the distribution system, such as customer disconnections from the grid, unexpected voltage drops, or voltage rises caused by DERs. 2. Electrical faults caused by human error, equipment failure, or weather conditions.

While previously proposed ML techniques can successfully classify or locate anomalies, most techniques tend to require a large amount of simulated data on all different types of anomalies. The anomaly data, however, is usually rare. In addition, the static training model of many ML techniques makes them irresponsive to new data from different distributions. Considering the latest trends and challenges in anomaly detection for power distribution systems, we aim to explore and develop advanced ML techniques to overcome these issues.

1.2 Thesis Objectives

The goal of this work is to explore the application of existing ML and deep learning techniques to detect or localize anomalies and subsequently find ways of adapting these techniques to benefit anomaly detection in radially distributed systems. Several deep learning methods originating from natural language processing and image

recognition will be explored.

There are two main objectives in this work:

1. The first objective is to apply unsupervised ML techniques that use only normal data to detect events caused by load changes in the distribution system. Three types of events are considered in this study: customer disconnection from the grid, unexpected voltage drops, and voltage rises caused by DER.

2. The second objective is to develop a CNN-based framework for fault classification and location identification in partially observable distribution systems. Four types of fault are considered in this study: line-to-line fault (LLF), line-to-ground fault (LGF), double line-to-ground fault (LLGF), and three-line-to-ground fault (LLLGF).

1.3 Thesis Outline

This thesis is structured as follows. Chapter 2 begins with an overview of a typical power distribution system and the anomalies of the distribution system relevant to this study. A brief literature review on anomaly detection in power systems is presented next. Then, a brief literature review of anomaly detection in power system is given. It is followed by a brief introduction to deep learning and its fundamental techniques. Then, a closer look at the machine learning mechanisms in the field of power system is presented. The problem of real-time anomaly detection in distribution grids using long short term memory network is presented in Chapter 3. Chapter 4 describes the real-time fault classification and location in partially observable distribution systems using convolutional neural networks. Conclusions and possible directions for future work are presented in Chapter 5.

Chapter 2

Background

2.1 Introduction

As described in Chapter 1, the goal of this project is to develop accurate and effective deep learning techniques for detecting or locating anomalies in radial distribution systems. This chapter describes a typical distribution system and reviews the commonly encountered anomalies on it, which are the anomalies of interest in this work. An overview of recently developed anomaly detection techniques for power systems is given. This chapter also provides a brief introduction to deep learning and its applications in power systems.

2.2 A Typical Distribution System

Distribution networks of an electric power system connects bulk sources of energy to customers' facilities. Depending on the type of construction, distribution systems can be divided into overhead and underground systems, with underground systems typically being more expensive and time-consuming to maintain. Distribution lines typically operate in a radial pattern. Loads are typically tapped along the line and can be single-phase and/or multi-phase taps. Line construction is usually non-homogeneous, as distribution lines extend as the load grows. If a fault occurs at any point in the radial system, the supply system beyond the point of failure is isolated and the supply to the customer is interrupted. It is estimated that 80% of interruptions are caused

by faults in the distribution system [3].

Figure 2.2 shows a typical radial distribution system with sub-transmission circuits, substations, primary feeders, transformers, secondary circuits, and service to customers' facilities. Sub-transmission lines carry energy from bulk energy sources to distribution substations. Distribution substations typically consist of supply lines, power transformers, buses, switch gear, capacitors, circuit breakers, isolators and reclosers. Transformers reduce the voltage of the supply line to the voltage of the local distribution level.

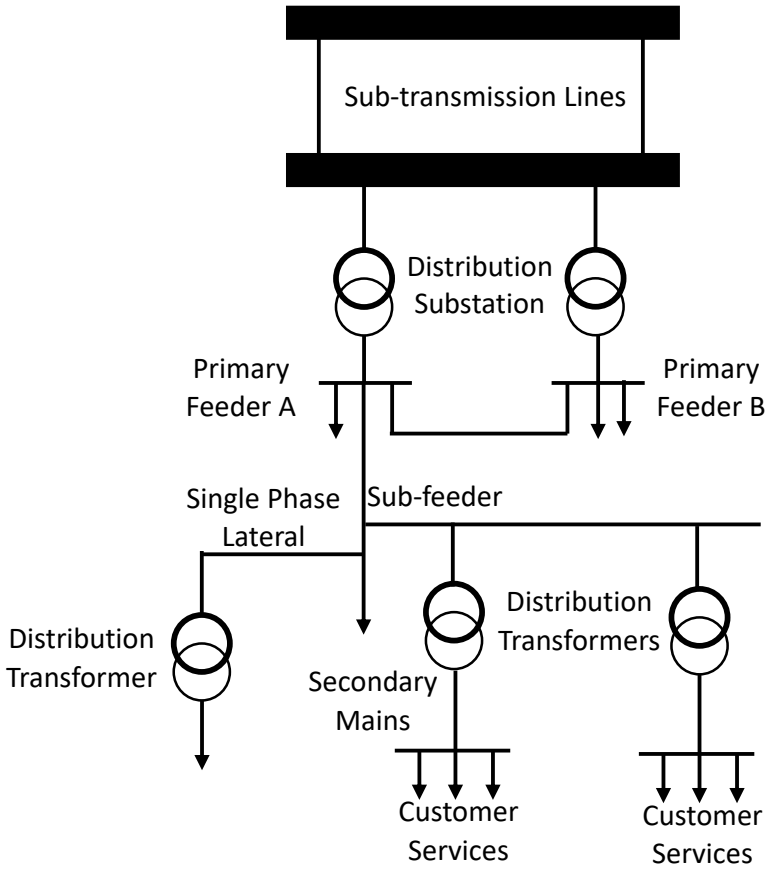


Figure 2.1: A typical radial distribution system.

Three-phase primary distribution feeders distribute energy to the load centers. The feeders carry high currents to the feeding points, and the size of the feeders is determined by the current-carrying capacity. From these feeding points, the circuit branches into three-phase feeders and single-phase laterals, consisting of overhead

lines and/or underground cables. Distribution transformers are installed to reduce the feeder voltage to service level. The secondary circuit distributes energy from the primary distribution feeder to the customer through the service drop.

2.2.1 Distribution System Protection

Devices commonly used to detect and isolate abnormal or faulty circuits in a power distribution system are phase to phase-overcurrent relays and grounding overcurrent relays [3]. Relays identify abnormal circuits in a system by detecting abnormal currents, abnormal voltages, abnormal frequencies, or a combination of them. Relays are installed to continuously monitor the distribution system and to send a signal to trip a circuit breaker or recloser in the event of an abnormal condition.

2.2.2 Distribution Test System

The test system used in this work is a real distribution network located in the Midwestern United States. This fully observable network, where all customers have smart meters installed, is operated by a municipal utility company. The test system consists of 240 primary network nodes, 3 feeder lines, and 23 miles of primary feeder conductors involving more than 1,120 customers [4]. Electrical components include overhead and underground cables with various phase configurations, load tap-change transformers, various secondary distribution transformers, shunt capacitor banks and circuit breakers [4]. Data include all customer smart meter measurements for one year (January to December 2017), system component parameters, and detailed network topology. To simulate this system, we used a commonly used open source solver, the Open Distribution System Simulator (OpenDSS).

2.3 Anomalies in Distribution Systems

Anomalies, or outliers, are data points that deviate from the pattern or distribution of the majority of the data [5]. In fact, there are different types of anomalous behaviors

in the power distribution grid. These anomalous behaviors can be caused by abnormal consumption patterns of users, faulty grid infrastructure, outages, energy fraud, or external cyberattacks [6]. Specifically, in this work, we are interested in detecting two types of anomalies: events caused by load changes in the distribution system and electrical faults.

2.3.1 Events Caused by Changes of Load

In this analysis, there are three types of events. Persistent outages are defined as the loss of service for longer than the normal re-closing interval, or the disconnection of a customer from the grid. According to statistics provided by the U.S. Energy Information Administration, the average loss of power per customer in 2016 was about 4 hours [7]. Power outages can be caused by accidents in the distribution system, animal disturbances and weather events. In addition to the safety aspects of outages, power outages mean lost time, lost customers and lost productivity.

Voltage drop is another major issue when it comes to quality maintenance of the power system. According to the *European standard EN50160*, a voltage drop is a sudden reduction in the effective voltage value to between 90% and 1% of the stipulated nominal value, followed by an "immediate" restoration of that voltage. The duration of the voltage drop is between half a cycle (10ms with 50Hz grids) and one minute [8]. One known cause of voltage drops is start-up or inrush current from capacitors, motors and other equipment. That is, startup of large loads can cause voltage drops. Voltage drops can lead to high costs and continuous processes that are particularly affected by this.

The event of voltage rise caused by DERs have also been studied. Overvoltage problems typically occur when DERs generate more power than the load on the feeder. The situation does get worse when DERs generate their peak power during periods of low demand [9]. This problem can affect the power supply, shorten the life of equipment through overheating, increase power consumption, and even cause

transformer failures.

2.3.2 Distribution System Faults

Distribution systems are usually subject to shunt faults. Shunt faults are classified as symmetrical faults and asymmetrical faults. Symmetrical faults are severe faults that occur infrequently, while asymmetrical faults are common but less severe [10]. Asymmetrical faults are classified as LGF, LLGF, and LLF. Three-phase faults LLLF and LLGF are two categories of symmetrical faults. In this work, we focus on four of these faults: LGF, LLGF, LLF and LLLGF, as shown in the following.

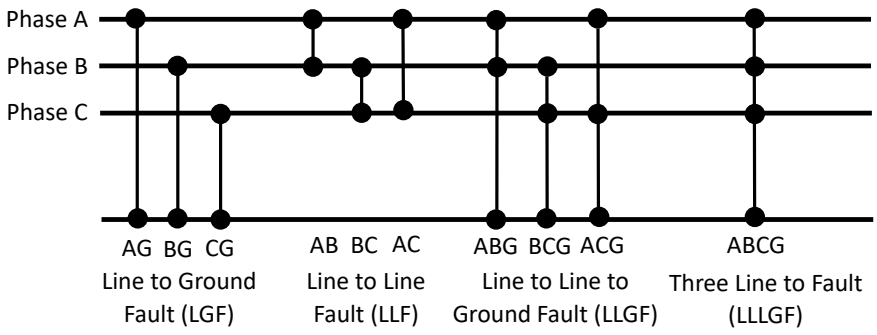


Figure 2.2: Fault types in power system [11].

LGF, also known as short circuit fault, occurs when one of the three-phase conductors of the distribution system reaches the ground due to wind, falling wood, animal contact or line drop [12]. LGF occurs at the rate of 70%, which makes it the most common fault in the distribution system network [13]. LLGF is a severe occurrence that causes severe asymmetry that can become LLGF if not resolved within a specific period of time. LLGF occurs when high winds or falling wood cause one phase to touch another phase, where 10% of faults in the distribution system are due to LLGF [13]. LLGF can occur on overhead or underground electrical lines due to high winds or a short circuit in both conductors. Approximately 15% of defects in the distribution network are due to line-to-line defects [13].

Symmetrical defects, also known as LLLGF, occur when there is an equipment

failure, when conductors reach different phases, or when towers connecting the remaining phases fall to the ground. LLLGF is the least common fault, accounting for only 5 percent of line faults, however, it would be the most detrimental [13].

2.4 Anomaly Detection in Power Systems

Due to the rapid development of smart grids, anomaly detection is needed to improve the quality and safety protection of the grid. Anomaly detection has been widely studied and applied in many different areas of the power system. In this section, the recent advances in anomaly detection in power systems are first outlined in Section 2.4.1. Then, Section 2.4.2 presents related work for specific types of anomalies, *i.e.*, fault detection in power systems.

2.4.1 Anomaly Detection State-of-the-Art

The existing anomaly detection methods can be categorized into two groups: machine learning-based and statistics-based. Machine learning methods have been widely used to efficiently detect anomalies in various areas, such as energy consumption, cyber attacks, and grid line outages. In [14], a micro-moments–based deep neural network model was proposed to detect and classify energy consumption anomalies. In [15] and [16], an LSTM neural network was employed to find a change in the behaviour of a system and detect anomalies in energy consumption. The proposed model shows improved accuracy in electricity theft identification. In [17], a neuro-cognitive inspired architecture based on Hierarchical Temporal Memory was proposed for real-time anomaly detection in smart grid using micro-phasor measurement unit (PMU) data. In [18], a supervised AI-based load estimator was developed to detect false data injection attacks by comparing the values of the estimated loads and their actual measurements. In [19], a data-driven framework combining a breadth-first search-based mechanism, a generative adversarial network, and a zone coordination process was proposed to detect and localize outage events in partially observable distribu-

tion systems by capturing the changes in smart meters' data distribution. In [20], a residual-based anomaly detection framework utilizing batch principal component analysis was developed to detect missing and bad data anomalies in primary distribution voltage magnitude measurements. In [21], various representative anomaly detection algorithms such as multivariate Gaussian distribution and one-class SVM are proposed to determine implausible inspection reports and assist in re-inspecting.

Statistics-based models have also been proposed to detect abnormal events. These models focus on extracting statistical features of the normal time-series data and utilize statistical techniques for measuring the difference between the observed patterns and the learned normal patterns. In [22], an alternating current power flow model and statistical change detection scheme were developed to detect dynamic outages using voltage phase angle data collected from PMUs. The proposed model can also capture system dynamics since it retains the time-variant and nonlinear nature of the power system. In [23], a random matrix theory and mean spectral radius based architecture was proposed to conduct anomaly detection in smart grid and to locate the abnormal sources. In [24], a distributed, multi-agent maximum likelihood (ML) approach was employed to detect anomalies in smart grid with reduced computational complexity. The proposed method also intended to preserve data privacy among different players in the network. In [25], the authors propose an ensemble Kalman filter based anomaly detector using a relaxation-based solution to detect cyber-attacks.

2.4.2 Fault Detection in Power Systems

In recent years, several studies have explored techniques for locating faults in power systems. These methods can be divided into two categories: conventional and artificial intelligence algorithms.

Conventional methods include traveling wave-based and impedance-based methods. Both methods are beneficial for transmission systems. But when considering the increasing complexity of the distribution system's topology, these methods are

not very effective. Traveling wave-based techniques are based on the principle of reflection and transmission of traveling waves between the line terminal and the fault location. The advantage of the method is that it is independent of system configuration and load variance. However, it can be costly to implement, as it requires high-speed data acquisition devices and sensors to capture the transient waveform for fault location [26–28]. Impedance-based algorithms are popular due to their simplicity and cost-effectiveness in comparison with the traveling wave-based techniques. Impedance-based algorithms use measured voltage and current to compute the bus impedance. From the calculated impedance, the fault distance to the measured point can be determined [29, 30]. These algorithms depend on high values of the fault current at the main substation to calculate the fault resistance, which can lead to multiple estimations of the fault location [31]. Additionally, these methods are vulnerable to random noise and changes in system parameters.

Due to the complexity of distribution grids and the availability of massive amounts of data, AI-based algorithms have attracted increasing attention. Since AI methods can automatically extract features and learn from historical information, they are considered a promising tool for application in power systems [32, 33]. There have been numerous studies exploring AI models for fault classification and location identification in power distribution systems. They include the decision tree (DT) [34, 35], random forest (RF) [36], k-nearest neighbor (KNN) [37], support vector machine (SVM) [38–40], artificial neural network (ANN) [41], convolutional neural network (CNN) [42–44], and many others. In [35], a combination of empirical mode decomposition, and DT algorithm was proposed to perform fault detection, classification and localization in the presence of solar photovoltaic (PV) distributed generation (DG). An SVM-based fault detection, which can simultaneously detect islanding and grid faults, was presented in [38]. Lin *et al.* [45] used the support vector data description method for fault detection and classification with only normal data for its training process. In [43], the authors proposed a faulted line localization method based on

a CNN classifier using bus voltages. It is capable of locating the faulted line with high probability under low observability of the buses. In [44], a fault type classification and a possible fire localization algorithm based on independent CNN classifiers were presented with a single observability characteristic. While the previously proposed techniques can successfully classify or locate faults, most of them require large amounts of simulation data on all different types of faults. The fault data, however, can be rare in the distribution system. In addition, their static training mode makes them vulnerable to new data with different distributions that enter the system.

2.5 Deep Learning

This section serves as an introduction to deep learning. A brief introduction to deep neural networks is given, and then the training process of neural networks is described. In addition, an overview of neural network regularization is presented.

2.5.1 Neural Network

A deep feedforward network or multilayer perceptron (MLP) is one of the basic artificial neural networks. As a classifier or approximator, MLP has been shown to be an effective alternative to other traditional statistical techniques [46]. Unlike traditional statistical methods, MLP does not make prior assumptions about the input distribution. It can model highly nonlinear functions and can be trained to generalize accurately when new or unseen data is encountered. These features of MLPs allow it to develop more complex models efficiently.

An MLP consists of interconnected neurons or nodes, as shown in Figure 2.3. An MLP is a model that represents a nonlinear mapping between an input vector x and an output vector y. The nodes or perceptrons are connected by weights w_n and the output signal is a function of the sum of node inputs modified by a simple nonlinear transfer or activation function. The MLP receives inputs at the input layer, passes through some hidden layers, and finally reaches the output layer. The number of

layers describes the depth of the network. Each hidden layer contains cells, which define the width of the network. A small network can contain one layer, while a large deep network may contain hundreds of layers or more.

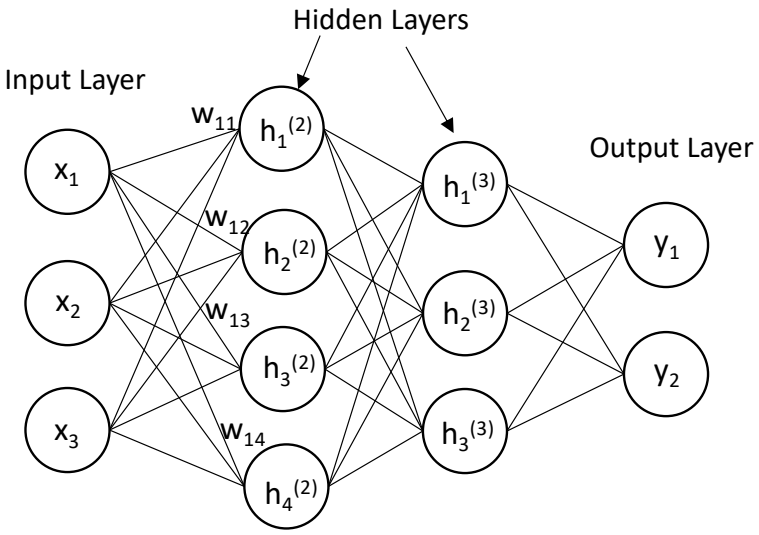


Figure 2.3: A mutilayer perception (MLP) contains two hidden layers.

For a single hidden layer MLP, the output y_k can be determined using the Equation (2.1) and Equation (2.2), where x_i is the input, w_{ij} and w_{jk} are the weights, and b_i and b_j are the biases or offsets. h_j is the intermediate state of the hidden layer. $f(*)$ is the activation function, which is the key to determine the nonlinear decision boundary. It can be set as a hyperbolic tangent or a sigmoid function. For MLPs involving more than one hidden layer, the expressions can be extended depending on the architecture.

$$h_j = f(\sum_i w_{ij} * x_i + b_i) \tag{2.1}$$

$$y_k = f(\sum_j w_{jk} * h_j + b_j) \tag{2.2}$$

2.5.2 Neural Network Training

The goal of a neural network is to approximate some function f^* that best generalizes the given data set. Given the following mapping relation, $y = f(x; \theta)$, where x is the

input and y is the output, a learning algorithm is applied to optimize the parameters of the neural network, θ, to approximate a desired function during the training process. By choosing an appropriate set of connection weights and activation functions, it has been shown that the neural network can approximate any smooth, measurable function between the input and output vectors [46].

To illustrate the training process in detail, let us consider a fully-connected multilayer (MLP). MLPs are considered supervised learning networks, and training such networks requires a set of labeled data. The data set is reasonably divided into training data and test data. During the training process, the MLP is repeatedly presented with training data. Supervised loss is used to define the difference between the predicted output and the target value [47]. The weights are updated until the output matches the target value, minimizing the supervised loss. Finally, the trained MLP will be examined using test data. A well-trained network will have zero generalization error and perform accurately on new previously unseen data from the same generation process. Having a low generalization error indicates that the network has low bias and low variability. Bias is a measure of the error between the predicted output and the target value; the model has a high bias that does not fit the data distribution. Variance is a measure of the variability of the model's prediction results when presented with dataset with different distributions. High variance indicates that the model is overfitted to one dataset and may not generalize well to other datasets. To achieve generalization, different optimization and normalization techniques have been designed [47].

Parameter optimization is usually performed by gradient algorithms through backpropagation. Gradient descent is an algorithm that is used to search for the local minima of the objective function during the optimization process. Based on this technique, the network parameters are updated by small increments in the negative direction of the gradient. The gradient is calculated by the backpropagation algorithm using the chain rule of calculus. The gradient update is performed during training in

the following manner [47]:

$$\widehat{\mathbf{g}} \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(\mathbf{x}^{(i)};\theta), \mathbf{y}^{(i)}) \tag{2.3}$$

$$\theta \leftarrow \theta - \epsilon\widehat{\mathbf{g}} \tag{2.4}$$

where θ represents the model parameters, $\mathbf{x}$ and $\mathbf{y}$ are the input and target output vectors of the model, respectively. $L(*)$ is the loss function that the algorithm aims to minimize. m is the size of the training set, or in the case of stochastic gradient descent, the size of the minimum batch. ϵ is the learning rate.

Unlike batch gradient descent, which processes the entire training set simultaneously in a large batch, stochastic gradient descent uses only a minibatch of the training set at a time. Small batches can offer a regularizing effect due to the noise they add to the learning process, which can be compensated for as long as the learning rate is small enough [47]. It is important that these mini-batches are randomly selected from the dataset to avoid bias towards a particular group. To reduce training error, it is usually desirable to have multiple epochs, *i.e.*, multiple passes of the training set [47]. The momentum approach was introduced to accelerate learning. It accumulates an exponentially decaying moving average of past gradients and accelerates toward its direction [47]. The update rule becomes:

$$\mathbf{v} \leftarrow \alpha\mathbf{v} - \epsilon\nabla_\theta(\frac{1}{m}\sum_i^m L(f(\mathbf{x}^{(i)};\theta), \mathbf{y}^{(i)})) \tag{2.5}$$

$$\theta \leftarrow \theta + \mathbf{v} \tag{2.6}$$

The velocity v accumulates the gradient element $\nabla_\theta(*)$, which represents the direction and speed of the gradient moving through the parameter space. The α is a hyperparameter in the range of 0 to 1 that is used to determine the exponential decay rate of the previous gradient contribution. The larger the α relative to ϵ, the greater the

effect of previous gradients on the current direction [47]. In practice, the learning rate ϵ is set to gradually decrease over time to speed up this process. The learning rate setting can greatly affect the performance of the model because if a large learning rate is set, the model will never converge. Algorithms such as AdaGrad, RMSProp, and Adam were proposed to accommodate the learning rate during training [47].

Training a deep neural network can be challenging. Since the input distribution of the layers in the network may change when the weights are updated after each minibatch, the learning algorithm may spend a lot of time optimizing the parameters. Batch normalization is an adaptive reparametrization method that aims to improve the speed of network convergence. It standardizes the input layers for each minibatch, and it can be applied to any input or hidden layer of the network [47].

2.5.3 Neural Network Regularization

Many regularization strategies are used in machine learning to achieve small test errors and thus perform well on new inputs. In doing so, some regularization strategies add extracted terms to the objective function. Some add additional constraints and penalties to the parameter values [47]. In this section, we review several regularization strategies that have been implemented in deep natural network models.

One regularization technique is to penalize parameter norms, such as weight decay or L^2 norms. By adding a parameter norm penalty, $\Omega(\theta) = \frac{1}{2} \parallel w \parallel_2^2$, it leads to an increase in the variance of the input features and a decrease in the weights of the features that have a weak relationship with the output target. An alternative approach is to use L^1 regularization by adding $\Omega(\theta) = \parallel w \parallel_1$ to the objective function. By doing this, the weights of features that have a weak relationship with the objective become zero, which leads to a more sparse solution. Both the L^2 norm and the L^1 norm can be used to avoid overfitting, while the L^1 norm is usually used for feature selection in sparse feature spaces [47].

Dropout is a regularization algorithm that mimics the behaviour of bagging for

ensembles of very many large neural networks. Specifically, it randomly selects the units in the hidden layer and applies a mask, thus creating a sub-network. In the case of bagging, these models are trained independently. Dropout differs from bagging in that its models share parameters, and each model inherits a different subset of parameters from the parent neural network [47]. With dropout, randomly selected neurons are ignored in the training so that the network is less sensitive to the specific weights of the neurons. The number of selected neurons is controlled by a parameter in dropout.

Early stopping is another popular regularization technique because of its effectiveness and simplicity. When training large models given sufficient representational capacity, as the training error steadily decreases, the validation error begins to rise, which can lead to overfitting of the model. By stopping early, the training process stops when the model's performance on the validation dataset starts to degrade. And returns the parameter settings for the stopping point where the validation error is minimized. However, this technique can be expansive, as it may require additional computational resources to achieve optimal training time. In addition, this technique requires maintaining a copy of the optimal parameters, which requires additional memory [47].

2.6 Machine Learning in Power Systems

Machine learning techniques are used in many different areas of power systems due to their complexity and various uncertainties that are difficult to solve with traditional techniques. In general, such techniques require information provided by sensors or devices installed along the line, such as feeder measurements. This information can be analyzed by machine learning techniques to detect anomalies. This section briefly describes the various machine learning techniques applied to power systems for detecting, classifying or locating anomalies.

2.6.1 Support Vector Machine

Support vector machine (SVM) is a supervised learning method that is widely used in classification and regression problems [48]. The number of support vectors is determined by the SVM algorithm given a specific set of inputs, while in neural networks the number of hidden layers is determined by trial and error. SVM is effective in high-dimensional spaces and it does not require any training effort like neural networks to achieve good performance. However, if the number of features is much larger than the number of input samples, it is crucial to avoid overfitting when choosing kernel functions and regularization terms.

The concept of SVM classification is shown in Figure 2.4. The goal is to train an SVM model that classifies the input points between two classes (class 1 and class 0). The black circles represent class 1, while the empty circles represent class 0. The input features are the voltage magnitude, V and phase angle, ϕ, from the measurement

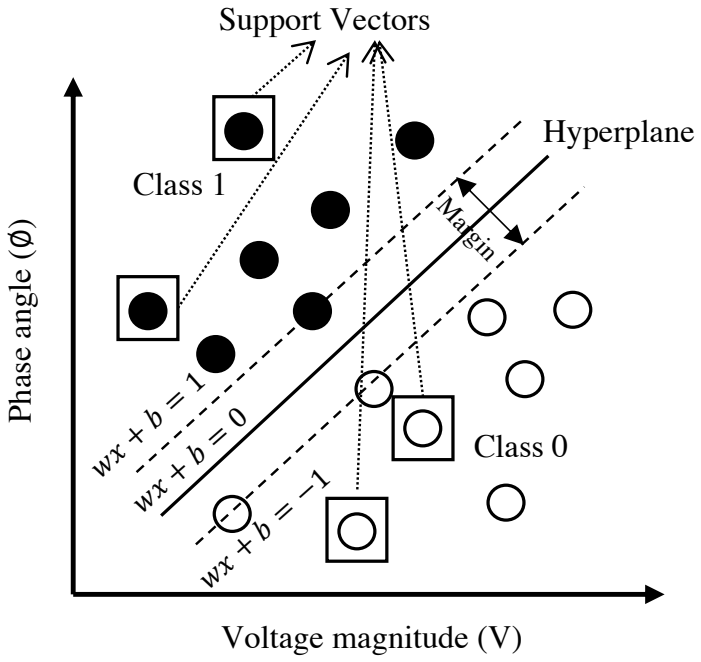


Figure 2.4: Linear classification using SVM.

node, and the target output is the type of anomaly. The support vector is an element of the training data, which is used to determine the margin for identifying the optimal hyperplane separating the two classes. The optimal hyperplane is defined as the linear decision function with the maximum margin between the vectors of the two classes. The margin is defined as the sum of the minimum distances between the training dataset and the separating hyperplane [48]. If the pattern is nonlinear, the SVM uses a kernel function to transform the original data into a high-dimensional feature space.

2.6.2 K-nearest Neighbor

K-nearest neighbors (KNN) is another popular and easy-to-implement non-parametric supervised learning algorithm [49]. KNN assumes that similar instances exist in close proximity. To classify an unknown sample or instance represented by a feature vector in the feature space, the KNN classifier computes the distance between that point and the points in the training dataset. The Euclidean distance is usually used as a distance metric. Then, the points are sorted according to the ascending order from smallest to largest distance. The top k entries are selected and the points are assigned to one of these k nearest neighbor classes, where k is a parameter set before training the network.

The concept of KNN classification is illustrated in Figure 2.5. The black boxes represent class 1 and the empty circles represent class 2. Given a point shown in the orange star, if k is set to 1, the unknown point belongs to class 1; if k is set to 5, the point belongs to class 0, since the empty circle class is the majority class of the five nearest points. Since KNN does not require offline training, its main computation is to "search" online for the k nearest neighbors of a given test point and determine the class of that point.

To calculate the distance between points A and B in the feature space, various distance functions are used in the literature, among which Euclidean distance is the most widely used one. Let $A = (x_1, x_2, ..., x_n)$, $B = (y_1, y_2, ..., y_n)$, where n is the

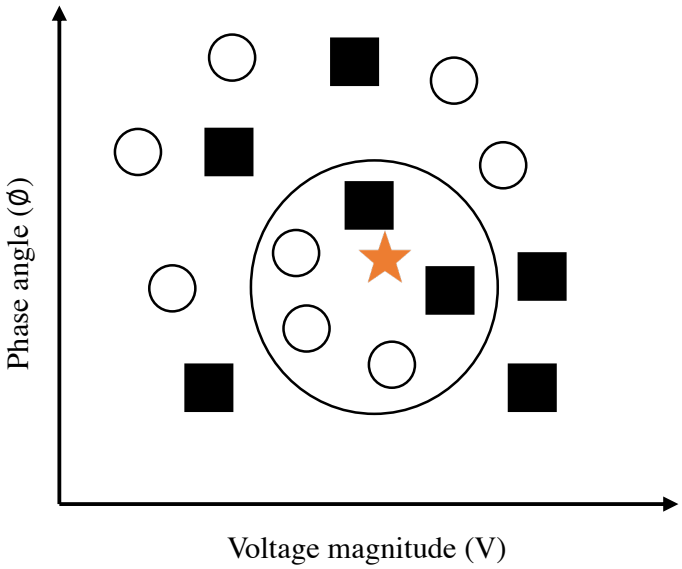


Figure 2.5: K-nearest neighbor classification.

dimensionality of the feature space. The normalized Euclidean metric is generally used, as follows:

$$dist(A, B) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}} \tag{2.7}$$

2.6.3 Convolutional Neural Network

Deep learning has gained great popularity, with many achievements described in the literature [50, 51]. As a type of deep network, CNNs have shown excellent performance on many machine learning problems. Examples include image recognition, financial time series, medical image analysis, and natural language processing [52–54]. CNN can automatically extract local features and recognize complex patterns in input data. It is robust to input distortions or offsets due to three important concepts that differ from traditional feedforward neural networks: local receptive fields, weight sharing, and pooling [55]. As a supervised learning technique, given a set of labeled voltage or current measurements, the goal of a CNN is to learn patterns and identify anomalous types of newly observed data.

To design the structure of CNN for application in power system, researcher usually follows the common practice of adopting a model that has already shown competitive

performance in other fields. For example, the AlexNet model [56]: the use of one-dimensional CNNs with hyper-parameters designed to fit the input. There are five main layers in this type of CNN: input layer, convolutional layer(s), pooling layer(s), fully connected layer(s), and output layer, as shown in Figure 2.7. These layers can be described as follows:

(i) The *input layer* accepts multi-dimensional raw data for processing in the network. It is usually specified by its width, height, and several channels. When the input data are images, the number of channels is often set to three to account for the colour channels (red, green, and blue). In this study, the input layer has $1 \times N$ neurons, where N denotes a variable number of features.

(ii) The primary purpose of *convolutional layers* is to extract features from the input data. Convolution preserves the spatial relationship between data by learning features using a small part of the input data. Essentially a filter or kernel is used during convolution, and the feature map is formed by sliding the filter over the entire input and computing the dot product. The size of feature map is controlled by three parameters: depth, stride, and padding. Depth corresponds to the number of filters used in the convolutional layers. Stride denotes the number of steps by which the window moves after each operation. A larger stride produces smaller feature maps. Sometimes, it is convenient to pad the input matrix with zeros around the border to obtain the feature map of the same size as the input matrix. The convolutional layer is followed by the non-linear rectified linear unit (ReLU) activation function which discards the negative values of feature maps without changing the size.

(iii) The *pooling layer* is used to reduce the dimensionality of the feature map while retaining information from the input feature map. It reduces the computational cost by reducing the number of parameters and prevents overfitting, therefore increasing the overall performance and accuracy of the network. In this study, the max-pooling is applied, where the largest elements are taken from the rectified feature map within a specific window.

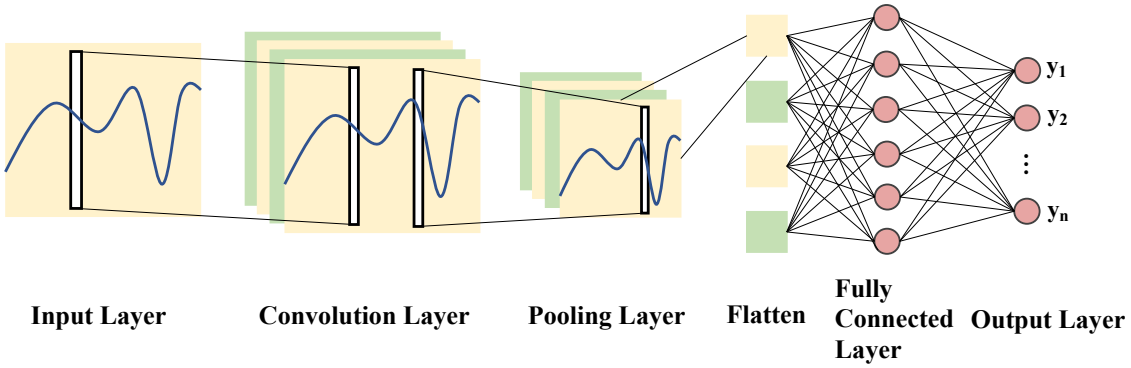


Figure 2.6: Representation of the one-dimensional convolutional neural network (CNN) architecture with data input connected to the convolution, pooling, flatten, fully connected and the output layer.

(iv) After several convolution and pooling operations, the original data is represented by a series of feature maps. The feature maps are then flattened into a one-dimensional vector which can be fed into the fully connected layer network. The *fully connected layer* contains numerous neurons that are connected to all nodes in the preceding layers.

(v) The *output layer* has n neurons, corresponding to n classes of the input data. It is fully connected to the feature layer. Depending on the type of output, the output layer uses a different type of activation function.

2.6.4 Recurrent Neural Network and Long short-Term Memory Network

The main drawback of traditional neural networks is that they do not remember information throughout time. As a result, they do not perform well enough when modeling data sequences (*i.e.*, time series). For example, if we want to classify what kind of events occur at each point in a movie. It is not clear how traditional neural networks use their inference about previous events in the movie to inform later events. Recurrent neural networks solve this problem. Recurrent neural networks, also known as RNNs, are a class of neural networks that allow previous outputs to be used as inputs while having hidden states [57]. Their typical characteristics are as follows.

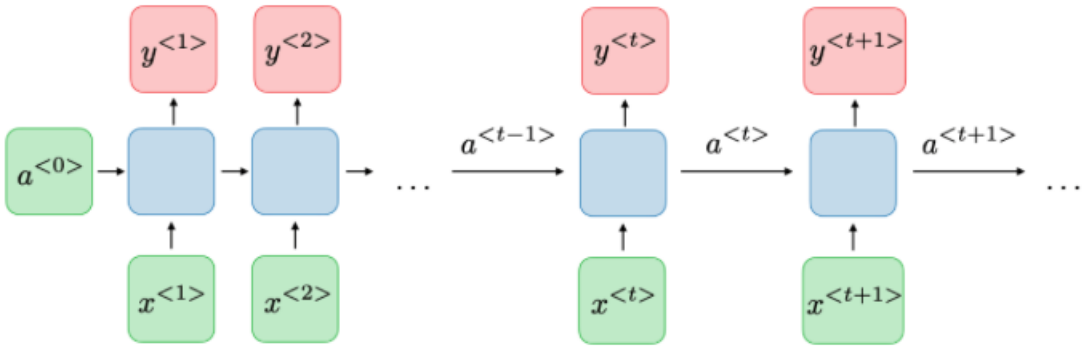


Figure 2.7: Representation of the recurrent neural network (RNN).

For each timestep t, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows [57]:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{2.8}$$

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \tag{2.9}$$

where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 are activation functions.

With the above architecture, it is possible for RNNs to handle inputs of any length, and the size of the model does not increase with the size of the input, and most importantly, the weights are shared across time. However, there are some drawbacks, including too slow computation and difficulty in accessing information from long ago [57]. Vanishing and exploding gradient phenomena are frequently encountered in the context of RNNs. They occur because it is difficult to capture long-term dependencies, as the multiplicative gradient may decrease or increase exponentially with the number of layers [57]. To address this gradient disappearance problem, long short-term memory networks (LSTM) have been proposed.

The LSTMs [58] are a special kind of recurrent neural network (RNN). Due to their effectiveness in handling long-term dependencies, LSTM networks are especially

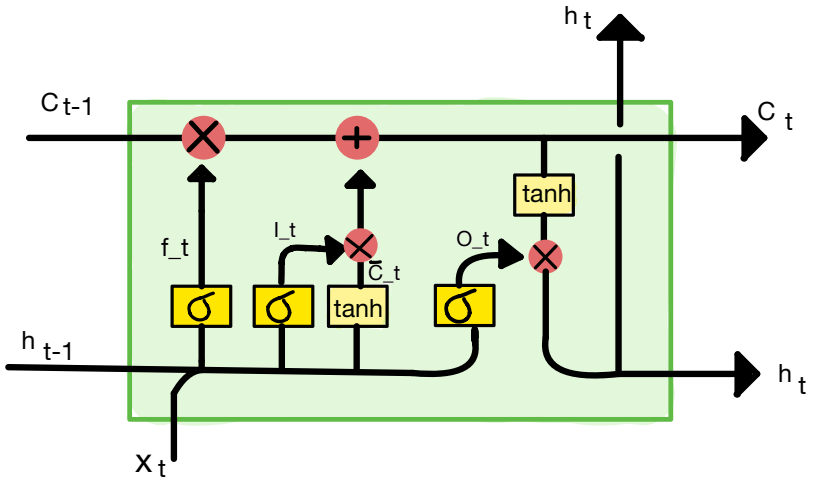


Figure 2.8: Structure of an LSTM memory block [59], where x_t, c_t, h_t denotes the input, cell state, and hidden state of the cell at time t, respectively.

suitable for processing sequential data such as sounds, written natural language, or time series data collected from sensors. The purpose of a LSTM network is to identify the correlation between the input sequence $x = (x_1, x_2, ...x_t)$ and the output sequence $y = (y_1, y_2, ...y_t)$.

The structure of LSTM network is similar to the standard RNN. It consists of one input layer, one hidden layer, and one output layer, where the hidden layer contains one or multiple memory blocks. An example of a single memory block is shown in Figure 2.8.

The main difference between LSTM and RNN is the cell state, which is the top horizontal path from C_{t-1} to C_t. The cell state is a sequence chain that connects all LSTM cells. Ideally, it can carry information throughout a serial learning process. Unlike hidden states which care mostly about a few previous states, cell states can store information from the earliest time stamps throughout the processing of the sequences. The LSTM network has the capability of adding or removing information to the cell state using different gates in the cell. Gates can optionally let information through the cell state using a sigmoid neural network layer and multiplication operations. The sigmoid layer outputs a number between 0–1, which determines how

much of the information should be let through the gate. For example, a value of 0 indicates nothing will get through the gate, while a value close to 1 let the information through the gate. An LSTM network has three of these gates that control the amount of information passing through the cell state: input gate, output gate, and forget gate [60]. The input gate determines what new information is stored in the cell state. The forget gate learns to filter out information from the input, x_t and last hidden state, h_{t-1}. Finally, the output gate decides how much of the updated cell state should be given as the output. The overall workflow of the LSTM cell is shown in Algorithm 1, where the equations are processed iteratively [59].

Algorithm 1:

Input: last hidden state, h_{t-1}; input, x_t; last cell state, c_{t-1}

Parameters: Weight matrices, W_f, W_i, W_C, W_o; bias vectors, b_f, b_i, b_C, b_o

Functions: logistic sigmoid function, $\sigma()$; tangent activation functions, tanh()

Output: hidden state, h_t; cell state, c_t

1: $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

2: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$

3: $\overline{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$

4: $C_t = f_t \cdot C_{t-1} + i_t \cdot \overline{C}_t$

5: $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

6: $h_t = o_t \cdot \tanh(C_t)$

Chapter 3

Real-Time Anomaly Detection in Distribution Grids Using Long Short Term Memory Network

3.1 Introduction

The massive amount of data generated by smart meters provides opportunities to better monitor and control power utilities in real-time. Meanwhile, the rapid deployment of distributed energy resources (DERs) and electric vehicles (EVs) rises fundamental challenges for system operation. For example, the reverse power flow (caused, *e.g.*, by solar panels, EVs, and energy storage systems) makes voltage regulation using conventional tools difficult [61]. In addition, frequent plug-and-charge EV charging, sudden changes of load, or the occurrence of line faults can cause momentary drops of system voltage [62].

In the distribution grid, traditional safeguards monitor and control voltage and frequency levels through internal relays that control the tap changer operation. However, the performance of relay actions is restricted, because they only try to keep the frequency and voltage within acceptable limits. Other anomalies, such as unbalanced three-phase voltage, system swings, and line outages are not detectable by the traditional method. Failure to detect these anomalies can result in significant financial losses and may even lead to power system collapse. Therefore, it is of critical

importance to detect anomalies as soon as possible.

To overcome these issues, we develop a real-time method for anomaly detection in distribution systems. The proposed method uses long-short-term memory (LSTM) model to analyze the voltage magnitude measurements in the system. LSTM network is a special type of recurrent neural network (RNN) [58]. LSTM networks are suitable for complex tasks including speech recognition and time-series forecasting. Due to its effectiveness in learning long-term dependencies, we use LSTM model to predict the next time step of voltage magnitude for all buses in a distribution system. Voltage anomalies are then detected using a threshold function based on the Euclidean distance. The performance of the data-driven anomaly detection algorithm is verified using a real distribution grid located in the Midwestern United States. The test system is described in Section 2.2.2.

The anomaly detection problem of interest in this work is based on analysing nodal voltage data collected in a power distribution grid. Anomalies are generally defined as observations that deviate from standard, normal, or expected values. Specifically, we consider anomalies as events caused by changes in the distribution system load. For example, a customer disconnected from the grid, unexpected voltage drop, or voltage rise caused by DERs. The sole input to the anomaly detection system is the stream of voltage magnitudes at each bus. The proposed framework aims to detect any anomalies in real-time. Subsequently, additional actions can be taken at the discretion of the distribution system operator (DSO) to mitigate the anomalies or to analyse them further.

3.2 Methods and Procedure

The proposed solution consists of a forecasting technique exploiting the voltage magnitude measurements extracted from the test system. It aims to identify the distribution system anomalies through monitoring the difference between the predicted and real voltage data. As shown in Figure 3.1, the two main operational phases of this

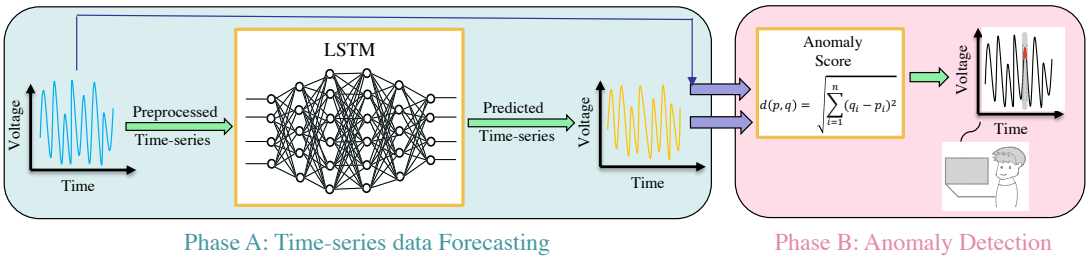


Figure 3.1: Systematic diagram of the proposed anomaly detection framework. The proposed method is divided into two phases: phase A focuses on time-series forecasting using LSTM and phase B incorporates the real and predicted time-series into an anomaly score function. This function detects anomalies that are further examined by the grid operator or other expert.

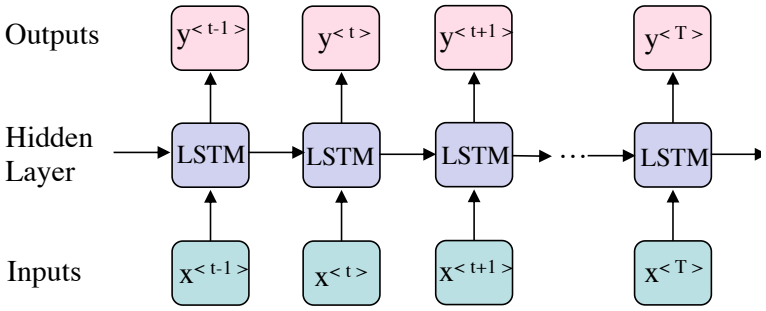


Figure 3.2: LSTM architecture. The hidden layer involves multiple memory blocks.

framework are (A) time-series forecasting, and (B) anomaly detection. More details of this workflow are presented next.

3.2.1 Data Preprocessing

The input data consists of 1-year period voltage magnitude measurements expressed per unit. The voltage dataset includes all three phase magnitudes for a total of 240 nodes or buses which are recorded hourly. To preprocess the data, we began by scaling all values to the $[0, 1]$ interval. This normalization step changes the range and scale of the data, thereby improving its uniformity. A normalized dataset also speeds up the learning. In particular, we apply the min-max scaler, where the original value is subtracted by the minimum and then divided by range (max-min).

To obtain the training $(X_{\text{train}}, Y_{\text{train}})$ and testing $(X_{\text{test}}, Y_{\text{test}})$ samples, a sliding window with of size t is introduced to divide the original time series into N sub-

sequences of input/output pairs. The inputs are $X = (x_n^{i,\dots i+t})_{i=1}^N$, and the outputs are $Y = (y_n^{i+t})_{i=1}^N$. Moreover, we split the dataset so that 2/3 are used for training, and 1/3 is reserved for testing. Note that all splits are chronologically ordered, so that there is no overlap between the datasets.

3.2.2 Time-series Forecasting

The time-series forecasting has been implemented using the LSTM network due to its capability of learning long-term dependencies. The input data are the hourly voltage time series measured at 240 nodes and various transformers (overall, values from 909 sensors were recorded). The obtained preprocessed data are fed into the LSTM network for predicting the next time-step value. Hence, the output data of the network is the predicted time series. The architecture of the LSTM network is shown in Figure 3.2, where the hidden layer consists of LSTM memory blocks illustrated in Figure 2.8.

The structure of the implemented LSTM network consists of an input layer of size 909, a hidden layer with 23 LSTM neurons, and an output layer that makes the next time step value prediction. The default sigmoid activation function is used for the LSTM memory block. The LSTM model only depends on historical data for providing the next time-step data. Moreover, the model automatically learns how much of the previous information contributes to future predictions. Thus, it can be used for real-time forecasting.

3.2.3 Unsupervised Anomaly Detection

The core operational phase of the proposed framework is anomaly detection. In this work, we are interested in detecting anomalies at any time instance by examining the difference between the real and predicted time series. Conventional approaches often suffer from a lack of data that represent anomalous behaviour. As anomalous behaviour is rare and detrimental to the system, it is difficult to gather enough

anomalous data. The technique proposed in this contribution addresses this data imbalance issue. In particular, it implements an unsupervised technique that does not require labeled data.

Given the real and predicted time series, their Euclidean distance can be calculated at each instant. Then a threshold T_h, based on the Euclidean distance function, is introduced. Since the data is acquired from multiple sensors, the differences are first calculated for each sensor and then accumulated for all sensors. In this work, the threshold is set to $T_h = \mu + 5\sigma$, where μ is the mean and σ is the standard deviation of the differences. If the distance calculated at a specific instant is greater than the threshold, the corresponding instant is considered a potential abnormal point. Since the uncertainty or fault occurring in the distribution system can last for a while, we aim to detect it once it happens, in real-time. The anomaly detector takes three types of input, the real data, $x_i = (x_1, x_2, ..., x_t)$, predicted data, $y_i = (y_1, y_2, ..., y_t)$, and a threshold, T_h. The distance between the real data and the predicted data $d(x, y)$ can be calculated as $d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$. Then, the calculated distance is compared to the preset threshold, T_h, if $d(x, y)$ is larger than T_h, then the particular time instance t is stored as an anomaly.

3.3 Experimental Results

The proposed framework is built using the open source deep learning tool Pytorch in Google Colab and tested on a real distribution feeder with corresponding 1-year voltage magnitude measurements. The distribution feeder consists of 240 nodes which involve more than 1120 customers [4]. To simulate potential disturbances in the distribution system, three abnormal cases have been considered: 1) multiple customers disconnected at various nodes, 2) unexpected voltage drop due to overload, and 3) voltage rise due to excess solar power. All three abnormal cases are randomly assigned to various nodes at different points in time, as detailed in Table 3.1. Corresponding abnormal cases are incorporated into the simulation, and the obtained results are

integrated into the data set.

The anomaly detection framework is trained and verified in two stages. In the first stage, three training datasets are prepared, each involving one type of abnormal cases where each dataset are trained and tested separately. In the second stage, the performance of the system is tested using a data set including all three types of abnormal cases.

Table 3.1: Designed anomalies with corespondent time.

Abnormal cases	**Time (M-D-Y)**
Case #1.1	3-06-17 5:00PM-8:00PM
Case #1.2	4-14-17 8:00AM-11:00AM
Case #2.1	4-28-17 8:00AM-11:00AM
Case #2.2	7-25-17 8:00AM-12:00PM
Case #3.1	2-04-17 1:00PM-4:00PM
Case #3.2	11-19-17 11:00PM-2:00AM

3.3.1 Performance Measures

For anomaly detection with high confidence, it is of critical importance that the LSTM network performs accurate forecasting. To evaluate the implemented LSTM model, three metrics have been used: root mean squared error (RMSE), mean squared error (MSE), and mean absolute error (MAE). Each metric has its own advantages and disadvantages.

MAE is a relatively simple and straightforward metric for evaluating the correctness of the predicted value, y_i *w.r.t.* true value, x_i. As shown in Eqn. 3.1, where n is the sample size, it calculates errors on a similar scale, which implies that it treats large and small errors equally.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|(x_i - y_i)| \tag{3.1}$$

Although it is widely used, it is not sufficient to appropriately evaluate forecast

accuracy. A large error is much less desirable than two or more smaller ones whose sum is about the same as the large error. For this reason, it is preferred to use a metric that is better at detecting larger errors. Therefore, compared to MAE, MSE is a more suitable metric. It is simply an average of the squared differences between the target value and the value predicted by the model.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{3.2}$$

As it squares the differences, it penalizes even small errors which may lead to an overestimation of how bad the model is. On the other hand, it is preferred over other metrics because it is differentiable and hence it can be optimized in a more straightforward way.

RMSE, a widely used statistical metric, is a quadratic scoring rule that also measures the average magnitude of the error. It is the square root of the average of the squared differences between predictions and the actual observations. RMSE penalizes larger errors as all errors are squared. This gives comparatively higher weight to larger values. Moreover, RMSE is an absolute error measure that squares the deviations to keep the positive and negative deviations from canceling one another out.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.3}$$

The two main advantages of MSE or RMSE are that they provide a quadratic loss function and that they are also measures of forecasting uncertainty.

3.3.2 Performance Evaluation

The proposed LSTM model was trained using the preprocessed voltage time series data which contained 2/3 of the overall data set. In the training of the LSTM model, ADAM was used as the optimizer, with a learning rate of 0.01, batch size 256, and 500 epochs. The MSE was selected as the loss function. Once the training process was completed, the model was tested using the remaining 1/3 of the data. The prediction

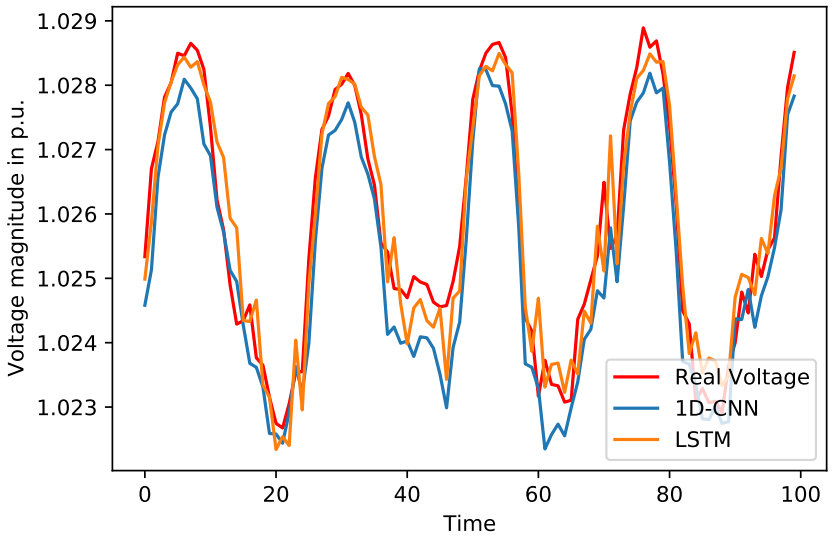


Figure 3.3: The prediction results for a particular node's voltage data using 1-D CNN and LSTM.

results of the proposed LSTM are compared with another state-of-the-art method, 1-D CNN [63]. The prediction performance is evaluated using MAE, MSE, and RMSE. Smaller values of the error metrics indicate higher prediction accuracy.

Table 3.2: Prediction results comparison between 1D-CNN and LSTM, using MAE, MSE, and RMSE.

Model	MAE	MSE	RMSE
LSTM	**0.0010341**	**2.25965×10^{-6}**	**0.0015032**
1-D CNN	0.0012261	2.78637×10^{-6}	0.0016692

All forecasting error metrics for the nodal voltage data are listed in Table 3.2. Compared with 1-D CNN, the proposed LSTM has lower MAE, MSE, and RMSE values. The detailed prediction results for a specific bus are shown in Figure 3.3. Compared with the 1-D CNN, the voltage curves predicted by the proposed LSTM are visually closer to the actual voltage curves. This further confirms that the proposed LSTM outperforms the alternative model.

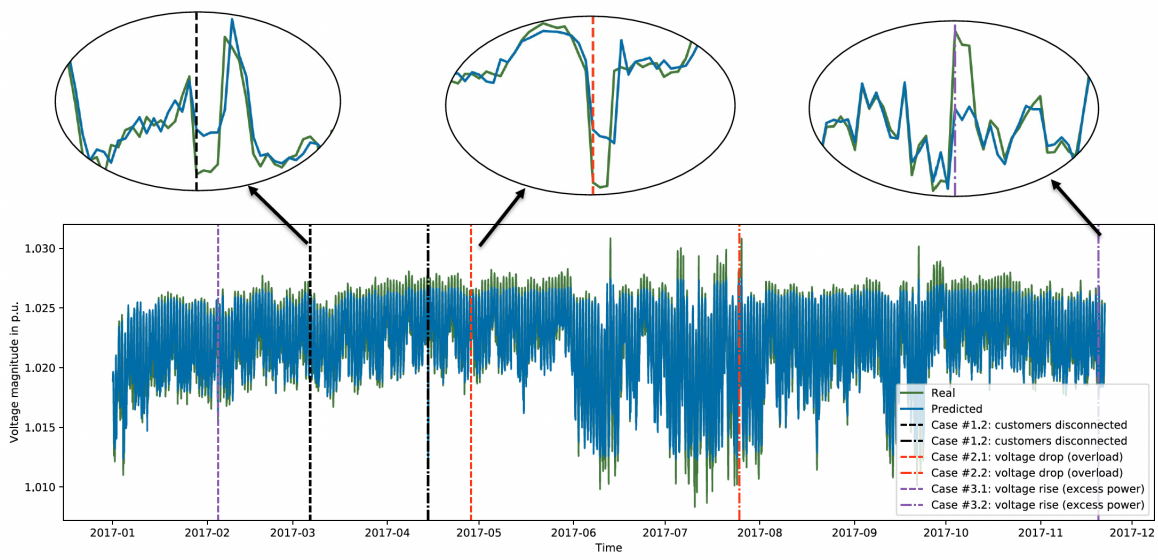


Figure 3.4: Performance of anomaly detection.

Finally, the proposed anomaly detection framework was assessed by inspecting if it can detect all considered anomalies. The performance of the anomaly detection for one of the voltage streams is shown in Figure 3.4. It can be observed that the proposed framework can detect all 6 anomalies at the instants when they occur, in real-time. More importantly, the framework did not detect any false positive samples, which means that it provides high confidence in the detected anomalies. Also, it is worth noting that the framework is designed to analyze all voltage streams at each time point to avoid detecting false positives. This is important, because other environmental or human factors, such as noises, may occur at a particular sensor, resulting in a large discrepancy between the true and predicted data for that particular sensor. In this study, we concentrated on three example anomaly types. However, it is expected that the developed framework will be able to detect multiple types of anomalous data.

3.4 Conclusions

In this chapter, we present a new data-driven framework to detect anomalies in distribution systems using voltage magnitudes measurement. The proposed LSTM-based approach is able to predict highly accurate next time step data with MAE of 0.0010341

and RMSE of 0.0015032. The developed anomaly detector successfully detected all pre-designed anomalies with a high degree of confidence. The proposed anomaly detector operates in an unsupervised manner, which addresses the data imbalance problem caused by the lack of anomalous data. The results corresponding to the proposed framework have been successfully tested using a real distribution grid.

Chapter 4

Real-time Fault Classification and Localization in Partially Observable Distribution Systems using Convolutional Neural Networks

4.1 Introduction

Fault classification and location identification are critical tasks in smart distribution networks. Identification of faults can not only save valuable time, but also reduce maintenance costs. With the rapid deployment of DERs, traditional fault detection methods that rely on simple linear models and human inspection are no longer sufficient. This is because they are not capable of handling the increased complexity and inverse power flows in distribution systems. While the previously proposed ML techniques can successfully classify or locate faults, most of them often require large amounts of simulation data on all different types of faults. The fault data, however, can be rare in the distribution system. In addition, their static training mode makes them vulnerable to new data with different distribution that enters the system. By taking into account the recent trends and challenges in distribution system fault classification and location identification, this section proposes a new real-time, data-driven framework for fault classification and localization in partially observable

distribution systems, based on CNN models. In particular, this article concentrates on the first stage of the fault localization problem, *i.e.*, locating the faulted lines or faulted area. Considering that most data are obtained under normal operating conditions of the distribution system and the fault data are rare, we propose a data augmentation technique to create synthetic fault data during the data preprocessing phase. Furthermore, instead of only using voltage or current measurements taken from the distribution system, we add special features that fundamentally contribute to the fault to improve model performance. Finally, we incorporate transfer learning and calibration modules to dynamically train the network to enable online continual learning.

The main contributions of this article can be summarized as follows:

1. Development of a novel CNN-based framework for fault classification and location identification in partially observable distribution systems.

2. Delta information capturing pre-fault and fault state changes is added to the feature set to classify and localize faults.

3. The positive-, negative-, and zero-sequence components that contribute to power systems analysis under faulted or unbalanced conditions are extracted to classify faults.

4. Synthetic minority over-sampling technique (SMOTE) is utilized to augment the limited amount of simulated fault data.

5. An online continual learning algorithm is proposed based on transfer learning and calibration modules to accommodate variations in the integration level of DERs and loading in distribution systems over time.

4.2 Data Acquisition and Feature Extraction

In this section, we briefly describe the test system used in this study and the data acquisition process, including the number of samples generated for each fault scenario. The feature vectors are also introduced for fault classification and faulted line (FL) localization.

4.2.1 Data Acquisition

The simulation system we used is described in Section 2.2.2. To ensure that the photovoltaic (PV) system was modeled as realistically as possible, we obtained hourly irradiance and temperature data for Chicago in 2017 from the NASA Prediction of Worldwide Energy Resources (POWER) project [64]. Each PV source was installed at the location of the existing load and scaled according to the load size and the desired PV penetration level. As for the location, the PV sources were randomly distributed on each feeder. For simplicity, the PV sources were assumed to provide only active power and penetration of 10% of these sources was considered as the baseline.

The simulation steps were performed as follows. First, load information was assigned to each bus on the network. Next, the quasi-static time series power flow over 1 year was solved via the Matlab-OpenDSS interface using the yearly mode in OpenDSS. Finally, the voltage magnitudes and phase angles of the selected buses were extracted for further analysis. According to the study presented in [65], smart meters connected at the end of the line/branches of a radial distribution network give the most relevant information for fault identification applications. Therefore, in this study, buses located at the end of each line or branch of the test system (23% of the total buses) were selected to place the phasor measurement units (PMUs), thus minimizing communication requirements. To evaluate the fault classification and FL identification framework, various fault scenarios were simulated and generated. In each simulation, different types of fault were randomly assigned at different distances

between buses. It is assumed that only one fault occurred at a time. The fault resistance was set to change in the range of 0.01 to 0.05 per unit (p.u.). Table 4.1 shows the number of samples generated for the normal operating state and each type of fault.

Table 4.1: Number of samples generated for normal operation state and for each fault type (g represents gound).

Fault Type	**Phase Involved**	**Number of Samples**
Line to ground (LG)	Phase a - g	200
	Phase b - g	200
	Phase c - g	200
Line to line (LL)	Phase a - Phase b	180
	Phase b - Phase c	180
	Phase a - Phase c	180
Double line to ground (LLG)	Phase a - Phase b - g	180
	Phase b - Phase c - g	180
	Phase a - Phase c - g	180
Three phase to ground (LLLG)	Phase a - Phase b - Phase c - g	100
Normal	-	8760

4.2.2 Feature Extraction

A power distribution network consisting of 240 buses with 55 observable nodes is considered in this study. A single fault may be one of the following four types: LG, LL, LLG, and LLLG. We are interested in real-time fault classification and FL localization using PMU measurements collected before and during the fault occurrence. Specifically, voltage magnitudes and phase angles will be collected to extract feature vectors to be fed into the proposed framework. Suppose that a fault occurs on the line

between an upstream bus p and a downstream bus s. Prior to the fault occurrence, its admittance model can be described as

$$\begin{bmatrix} I_{ps}^{pre-F} \\ I_{sp}^{pre-F} \end{bmatrix} = \begin{bmatrix} Y_{ps}^{pp} & Y_{ps}^{ps} \\ Y_{ps}^{sp} & Y_{ps}^{ss} \end{bmatrix} \begin{bmatrix} V_{p}^{pre-F} \\ V_{s}^{pre-F} \end{bmatrix} \tag{4.1}$$

where, I_{ps}^{pre-F} and I_{sp}^{pre-F} are the vectors of phase currents flowing from bus p to bus s, and bus s to bus p on the line, V_{p}^{pre-F} and V_{s}^{pre-F} are the vectors of phase voltages at bus p and bus s respectively. Y_{ps}^{pp} and Y_{ps}^{ss} are the self-admittance at bus p and bus s, and Y_{ps}^{ps} and Y_{ps}^{sp} are the mutual admittance matrices between bus p and bus s, and bus s and bus p respectively.

When a fault occurs on the line segment between bus p and bus s, the during-fault admittance model can be represented as follows

$$\begin{bmatrix} I_{ps}^{F} \\ I_{sp}^{F} \end{bmatrix} = \begin{bmatrix} Y_{ps}^{'pp} & Y_{ps}^{'ps} \\ Y_{ps}^{'sp} & Y_{ps}^{'ss} \end{bmatrix} \begin{bmatrix} V_{p}^{F} \\ V_{s}^{F} \end{bmatrix} \tag{4.2}$$

The during-fault current can then be derived as

$$\begin{aligned} I_{ps}^{F} &= Y_{ps}^{'pp} V_{p}^{F} + Y_{ps}^{'ps} V_{s}^{F} \\ I_{pp}^{F} &= Y_{ps}^{'sp} V_{p}^{F} + Y_{ps}^{'ss} V_{s}^{F} \end{aligned} \tag{4.3}$$

Therefore, the difference between the pre-fault and during-fault voltage, $\Delta V = V_{p/s}^{pre-F} - V_{p/s}^{F}$, can be constructed as

$$\begin{aligned} \Delta V_p &= V_{p}^{pre-F} - \frac{I_{ps}^{'} - Y_{ps}^{'ps} V_{s}^{'}}{Y_{ps}^{'pp}} \\ \Delta V_s &= V_{s}^{pre-F} \frac{I_{sp}^{'} - Y_{ps}^{'ss} V_{s}^{'}}{Y_{ps}^{'sp}} \end{aligned} \tag{4.4}$$

From (4.4), it is clear that the difference between the pre-fault and during-fault voltages contributes to the fault type classification and FL localization based on the admittance differences in each phase. Therefore, both its real and imaginary parts, $|\Delta V|$ and $\Delta\theta_v$, are added to the feature vectors to classify and locate faults.

In addition, symmetrical components are also incorporated during the feature vector extraction. Symmetrical component method has been widely used in fault analysis

by converting a three-phase unbalanced system into two sets of balanced phasors and a set of single-phase phasors, or symmetrical components [66]. These sets of positive, negative, and zero-sequence components may contain valuable information about the fault. Therefore, these three sequence components are added to the feature vectors, with the objective of improving the accuracy of fault type classification. To convert a set of phase quantities into symmetrical components, the following calculation can be performed

$$\begin{bmatrix} V_0 \\ V_1 \\ V_2 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 \\ 1 & \alpha^2 & \alpha \end{bmatrix} \begin{bmatrix} V_A \\ V_B \\ V_C \end{bmatrix} \tag{4.5}$$

where α is defined as $1\angle 120$, V_0, V_1, and V_2 are the zero, positive, and negative sequence components, respectively, and V_A, V_B, and V_C are the voltages for phase A, B, and C, respectively.

This results in the following equations

$$\begin{aligned} V_0 &= \frac{1}{3}(V_A + V_B + V_C) \\ V_1 &= \frac{1}{3}(V_A + V_B + V_C) \\ V_2 &= \frac{1}{3}(V_A + V_B + V_C) \end{aligned} \tag{4.6}$$

Consequently, the feature vectors for fault type classification (C) and faulted line localization (L) are defined as:

$$\phi_C = \{|V^{ABC}|, \theta_v^{ABC}, |\Delta V^{ABC}|, \Delta\theta_v^{ABC}, V_{0,1,2}\} \tag{4.7}$$

$$\phi_L = \{|V^{ABC}|, \theta_v^{ABC}, |\Delta V^{ABC}|, \Delta\theta_v^{ABC}\} \tag{4.8}$$

4.3 Method and Procedure

To design the CNN structure for application in fault classification and location identification, we follow the common practice of adopting a model that has already shown

competitive performance in other fields. Specifically, the CNN is based on the AlexNet model [56], which uses one-dimensional CNNs with hyperparameters designed to fit the input. There are five main layers in this type of CNN: input layer, convolutional layer(s), pooling layer(s), fully connected layer(s), and output layer, as shown in Figure 2.7 in Section 2.6.3.

4.3.1 CNN Training Process

The CNN is trained in a supervised manner using a sequence of training examples $[(x_1, y_1), (x_2, y_2), ..., (x_K, y_K)]$, where $x_t \in \mathbf{R}^{1xN}$, $y_t \in \mathbf{R}^n$ for $1 \leq t \leq K$. Data x_t is given as input to the network, while vector y_t denotes the target output. Let Θ denote the set of CNN parameters. The CNN training process can be represented as an optimization problem with the objective of minimizing the expected loss on the training set [47]. The cross-entropy loss function is utilized for multiclass classification with a regularization term $\lambda \parallel \Theta \parallel_f^2$ to avoid overfitting, as shown below

$$L(f(x;\Theta), y) = \frac{1}{m} \sum_{i=1}^{m} y_i log(f(x_i;\Theta)) + \lambda \parallel \Theta \parallel_f^2 \tag{4.9}$$

where $f(x;\Theta)$ is the output probability of the CNN parameterized by Θ when the input is x, m is the number of training examples, and λ denotes the regularization coefficient.

To solve this optimization problem and find the optimal set of Θ that minimizes the above loss, the stochastic gradient descent algorithm and some of its variants (such as RMSProp [67] and Adam [68]) performed fairly robustly in various tasks. Adam is the most often used optimization algorithm, since it is fairly robust to the choice of hyperparameters. In addition, batch normalization and dropout layers are often introduced into the network to prevent overfitting. Batch normalization layer is used to prevent internal covariate shift by standardizing each element in the layer to zero mean and unit variance [69].

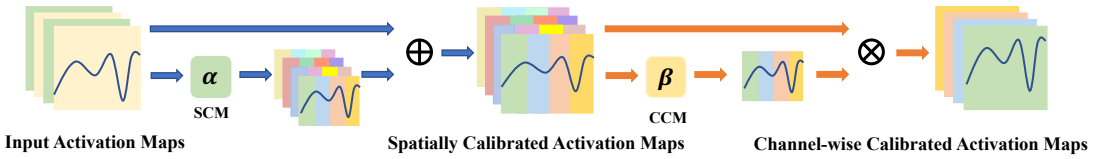


Figure 4.1: Calibration modules containing spatial and channel-wise calibration modules applied sequentially to the activation maps. Here $\oplus$ and $\otimes$ represent element-wise addition and channel-wise multiplication operation, respectively.

4.3.2 Incremental CNN with Calibration Modules

A static CNN model learned from past data may not fully capture the characteristics of future data. To solve this problem, incremental learning is proposed to update the CNN model and satisfy the need for online learning. Traditional incremental learning can be costly and time consuming, as it often requires retraining the model to adapt to changes in the system or data distribution over time. Moreover, it may not preserve previously acquired knowledge and can lead to catastrophic forgetting. An effective incrementally trained model must be able to learn from new data that arrive sequentially and still retain the knowledge gained from previous data set without retraining on all previously seen data. To address the above challenges, we adopted transfer learning and calibration modules, aimed at accommodating variations of the integration level of DERs and loading in distribution systems over time.

Transfer learning is a technique for transferring knowledge from one domain/data set to another using a specific weight adjustment strategy [70]. It selects partial knowledge gained from training the network on the source data set as supplements to the training set in the target domain by assigning appropriate weight values to these selected instances. Based on the above idea and inspired by [71], we use spatial and channel-wise calibration modules within the intermediate activation-maps of the CNN, as shown in Figure 4.1. Specifically, the spatial calibration modules (SCM) learn weights to calibrate each point in the feature map while the channel-wise calibration modules (CCM) learn weights to calibrate each channel in the feature map. The calibration module (CM) was added after each layer of the base CNN.

Suppose that the output activation map of the i^{th} CNN layer is M_i. Let α_i be the SCM operator added after the i^{th} layer of the base module. The SCM uses 1D-convolution with 3×3 kernel size and the output of α_i representing the spatial calibration weights. The calibration weights will be added element-wise to M_i to give the spatially calibrated activation maps, M_i^*, which are then fed as input to the channel-wise calibration module. Let β_i be the CCM operator added after the SCM operator for the i^{th} layer of the base module. The CCM operator first performs global average pooling (GAP) on M_i^*, and then applies a 1D convolution with kernel size 1×1. This is followed by a batch normalization operation that produces an output of β_i that represents the channel-wise calibrated activation maps. Each of the calibration weights is multiplied by the corresponding channel of M_i^* to produce the final calibrated activation maps M_i^{**} for the i^{th} layer. Algorithm 2 shows the workflow of CM_i, and the overall calibration process can be described as:

$$M_i^{**} = CM_i(M_i) = \beta_i(\alpha_i(M_i) \oplus M_i) \otimes \alpha_i(M_i) \oplus M_i \tag{4.10}$$

Algorithm 2:

Input: activation maps, M_i

Output: spatially calibrated activation maps, M_i^*; channel-wise calibrated activation maps, M_i^{**}

1: $\alpha_i(M_i) = \text{1D-CNN}(M_i)$

2: $M_i^* = \alpha_i(M_i) \oplus M_i$

3: $\beta(M_i^*) = BN(\text{1D-CNN}(GAP(M_i^*)))$

4: $M_i^{**} = \beta_i(M_i^*) \otimes M_i^*$

For the first task, a base CNN model with a classification layer and calibration module is trained using historical data, as shown in Figure 4.2. For the subsequent task, where part of the historical data and all new data are fed to the system, the parameters of the base CNN module Θ are kept frozen and only the data-adaptive

calibration module and the classification module are trained. This way, the model adapts features relevant to new data from the base model using calibration modules.

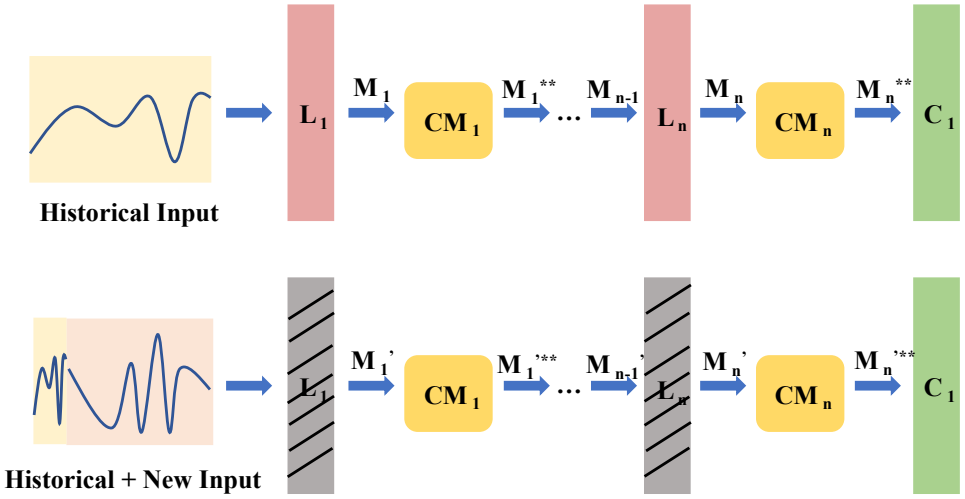


Figure 4.2: Proposed architecture for incremental learning. The top architecture is used for the first task, where the model is trained from historical data. The bottom architecture is for all subsequent tasks and the model is trained from part of historical data and all new data. $L_1 - L_n$ represent layers of the base CNN module. The calibration modules calibrate the output activation map M_i to produce M_i^{**} at layer i. C_1 is the classification module. To adapt new input data, the base CNN modules are frozen and not trainable. They are marked in grey color with the hatched pattern.

4.3.3 Fault Type and Fault Location Identification Process

The overall proposed framework for fault type and FL identification is presented in Figure 4.3. There are two major components in the framework, namely offline learning and online continual learning. For offline learning, a CNN-based model is proposed for identifying the 10 different types of faults that can occur in a distribution network. These faults are presented in Table 4.1. Identifying the type of fault that occurs in a distribution network is an essential operation for further determining the FL in the system. The historical voltage magnitudes and phase angles collected at the end of the line/branches were input into the system. Next, features ϕ_C shown in Section 4.2.2 were extracted and fed into the fault type identification model, CNN_C. The number

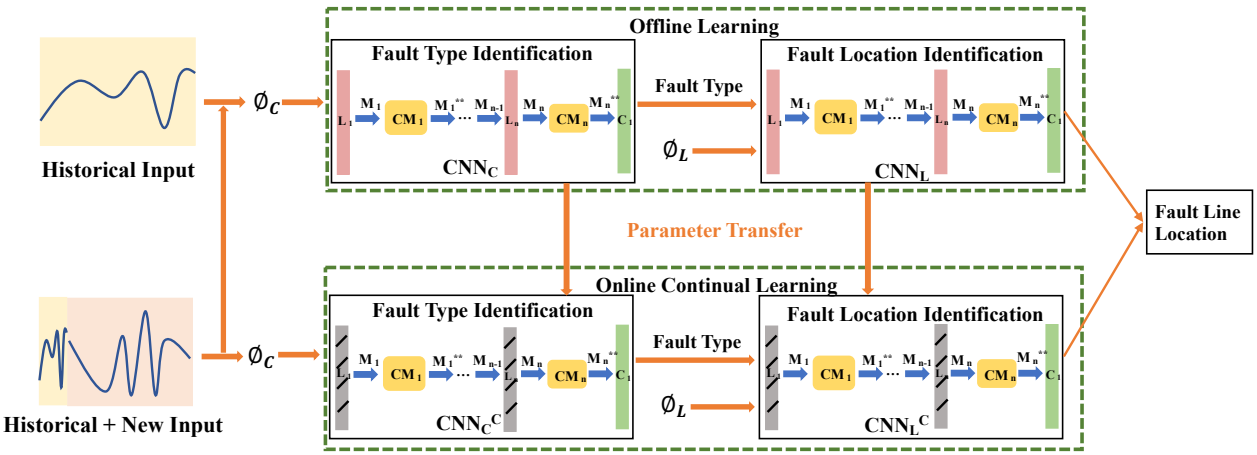


Figure 4.3: Overall fault type classification and fault location identification framework.

of features fed into the model was determined by the number of buses where the PMU were located. Using these features, the CNN_C was capable of determining the type of fault by recognizing the pattern in the bus voltage magnitude and phase angle values, the delta voltage information, and the calculated symmetrical components. The output of the CNN_C was a one-hot encoded vector of size 11, where each index denoted one of the 10 fault types and the normal type.

After the fault type was determined, the next step was to identify the FL. The fault type, along with the features ϕ_L, were fed into the fault location identification model, CNN_L. Note that if the fault type is normal, the framework will not proceed to the CNN_L. Since the features ϕ_L were all numerical values, the one-hot encoded fault type vector was converted to dummy encoding columns to ensure consistency within the input. In total, 11 additional columns were added and a value of 1 was placed in a specific column and 0 for all other columns, to indicate a particular fault type. CNN_L learned the correlations between the input features and the provided labels and was capable of identifying the FL. CNN_L was designed to produce a vector of size m, which represents the likelihood that the fault is located on a particular line in the set of all lines M in the distribution network. For example, using the test system described in Section 2.2.2, if a single line to ground fault occurs between node

2028 and node 2029, the output vector would indicate a value of 1 at the 2027^{th} index and a value of 0 for all other lines. Similarly, if a double line to ground fault occurs at node 2028, the output vector would indicate a value of 1 at the 2027^{th} index or the 2028^{th} index and a value of 0 for all other lines.

However, as mentioned in Section 4.3.2, a static CNN model may not fully capture the characteristics of the future data and is not suitable for online application. For example, consider historical data collected from a system with DER penetration levels of 0-20% and loading between 70-100%. When the system undergoes further increase in DER penetration levels or loading, the offline model cannot capture the pattern learned in the new data and its performance is likely to degrade. To address this problem, an incremental CNN model with transfer learning is used to provide online continual learning. Note that this model requires labeled data for training. Essentially, the model training process is similar to the offline learning model. The feature vectors ϕ_C and ϕ_L were extracted from part of the historical data that contained all different types of faults and all new data. The convolutional layers L_1 to L_n were frozen while the calibration module CM_i and the fully connected classifier C_1 were trained and fine-tuned using the new extracted feature vectors. The online model can adapt to changes when new data enter the system continuously within a short training time period. This is possible by using parameters transferred from the offline model. When needed, all subsequent new data and historical data can be further used together to train the offline model to enhance the model in recognizing patterns within a more complex data distribution.

4.4 Data Preprocessing and Model Setting

The test distribution system under normal condition and with four types of line fault of Section 1, were simulated and data were collected. As shown in Table 4.1, only a small amount of fault data were simulated compared to the normal data. This corresponds to the scarcity of fault data in real distribution systems. The preprocessing

of collected data started by converting the categorical labels to one-hot encoding to ensure the labels and the target outputs from the proposed model were in the same format. Next, to address the class-imbalance problem (normal vs. fault cases), we utilized the synthetic minority over-sampling technique (SMOTE) [72]. It over-sampled the fault classes so that the number of samples for each minority classes was the same as the majority class. Then, the over-sampled data was partitioned as follows: 70% of the data were used for training, 20% for validation, and the remaining 10% for model testing. Finally, a min-max scalar was applied and the data set was normalized to the interval [0, 1] to improve its uniformity and speed up learning.

4.4.1 Fault Type Classification Model

The fault type is classified using the CNN-based model. The structure of the CNN_C classifier is summarized in Table 4.2. There are a total of four major layers in this model: the input size of the model is $[-1, 1, 344]$, and there are two main convolutional layers (layer 1 and layer 4), one fully connected layer (layer 7), and one output layer. The last dimension of 344 corresponds to the total number of features. The "-1" in the output shapes are place holders for batch size. In each convolutional layer, the `convolution-1d` operation has a kernel size of 1 while stride and padding are set to 1. The `convolution-1d` operation is followed by a `maxpool-1d` operation that has a stride of 1. To prevent the model from overfitting, a 20% dropout is added after the `maxpool-1d` operation. Furthermore, after each convolutional layer, the spatial calibration module (layer 2 and layer 5) and the channel-wise calibration module (layer 3 and layer 6) are added to calibrate the output activation map from the previous convolutional layer. The output of this model is $[-1, 11]$, which represents the 11 types of faults.

The CNN_C model was trained using Adam optimizer with decay factor parameter $\alpha = 0.9$ and the learning rate or iteration step size was set to 0.001. The cross-entropy loss was selected as the loss function and the network was trained for 50 epochs. The

Table 4.2: Structure of the Fault Type Classification Model, CNN_C.

Layer No.	**Layer Type**	**Output**	**Param No.**
Layer 1	Convolution1d	[-1, 32, 344]	128
	MaxPool1d	[-1, 32, 114]	0
	Dropout	[-1, 32, 114]	0
Layer 2	Convolution1d	[-1, 32, 114]	3,104
	BatchNorm1d	[-1, 32, 114]	64
Layer 3	AdaptiveAvgPool1d	[-1, 32, 1]	0
	Convolution1d	[-1, 32, 1]	1,056
	BatchNorm1d	[-1, 32, 1]	64
Layer 4	Convolution1d	[-1, 32, 344]	128
	MaxPool1d	[-1, 32, 114]	0
	Dropout	[-1, 32, 114]	0
Layer 5	Convolution1d	[-1, 32, 114]	3,104
	BatchNorm1d	[-1, 32, 114]	64
Layer 6	AdaptiveAvgPool1d	[-1, 32, 1]	0
	Convolution1d	[-1, 32, 1]	1,056
	BatchNorm1d	[-1, 32, 1]	64
Layer 7	Flatten	[-1, 3648]	0
	Linear	[-1, 128]	467,072
	RELU	[-1, 128]	0
Layer 8	Linear (Output)	[-1, 11]	1,419

effectiveness of the CNN_C was evaluated by the fault type classification accuracy, η_C. The F1-score was also used to evaluate the performance of CNN; it is the harmonic mean of precision and recall which gives a better measure of the incorrectly classified cases than the accuracy metric. Both metrics are shown below

$$\eta_c = \frac{\text{the number of faults correctly categorized}}{\text{total number of faults}} \tag{4.11}$$

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{4.12}$$

where precision and recall can be calculated as:

$$\begin{aligned} \text{precision} &= \frac{\text{true positive (TP)}}{\text{true positive (TP)} + \text{false positive (FP)}} \\ \text{recall} &= \frac{\text{true positive (TP)}}{\text{true positive (TP)} + \text{false negative (FN)}} \end{aligned} \tag{4.13}$$

Table 4.3: Structure of the Fault Location Identification Model, CNN_L.

Layer No.	**Layer Type**	**Output**	**Param No.**
Layer 1	Convolution1d	[-1, 32, 260]	128
	MaxPool1d	[-1, 32, 86]	0
	Dropout	[-1, 32, 86]	0
Layer 2	Convolution1d	[-1, 32, 86]	3,104
	BatchNorm1d	[-1, 32, 86]	64
Layer 3	AdaptiveAvgPool1d	[-1, 32, 1]	0
	Convolution1d	[-1, 32, 1]	1,056
	BatchNorm1d	[-1, 32, 1]	64
Layer 4	Convolution1d	[-1, 32, 260]	128
	MaxPool1d	[-1, 32, 86]	0
	Dropout	[-1, 32, 86]	0
Layer 5	Convolution1d	[-1, 32, 86]	3,104
	BatchNorm1d	[-1, 32, 86]	64
Layer 6	AdaptiveAvgPool1d	[-1, 32, 1]	0
	Convolution1d	[-1, 32, 1]	1,056
	BatchNorm1d	[-1, 32, 1]	64
Layer 7	Flatten	[-1, 2752]	0
	Linear	[-1, 128]	352,384
	RELU	[-1, 128]	0
Layer 8	Linear (Output)	[-1, 239]	30,381

4.4.2 Fault Line Localization Model

Another CNN-based model, CNN_L, was developed to localize the faulted line for the ten fault types. The structure of the model is shown in Table 4.3. There are two major convolutional layers in which the `convolution-1d`, `maxpool-1d` and `dropout` has the same property as CNN_C. The spatial calibration module and channel-wise calibration module are also added after each convolutional layer. The input feature has a shape of [-1, 1, 260] while the output of the CNN_L model has a shape of [-1, 239] which indicates the 239 lines in the test system.

The Adam optimizer with the same properties as for CNN_C was used during the training process of CNN_L. The cross-entropy loss was selected as the loss function and the CNN_L was trained for 45 epochs. The effectiveness of CNN_L was evaluated using the faulted line localization accuracy, η_L, as well as the F1-score

$$\eta_l = \frac{\text{number of fault line correctly located}}{\text{total number of faults}} \tag{4.14}$$

4.5 Experimental Results

In this section, the performance of the proposed framework for fault type classification and FL identification in the distribution system is evaluated using the test system introduced in Section 2.2.2. We first compare the proposed framework with other frequently used models from the literature. This verifies the effectiveness and feasibility of the proposed model under a specific DER penetration level and loading condition, *i.e.*, during the offline learning mode. The robustness to noise and fault resistance is also analyzed. Finally, the online fault type and FL identification system are verified by the data collected under systems with different DER penetration levels and loading conditions.

4.5.1 Offline Performance

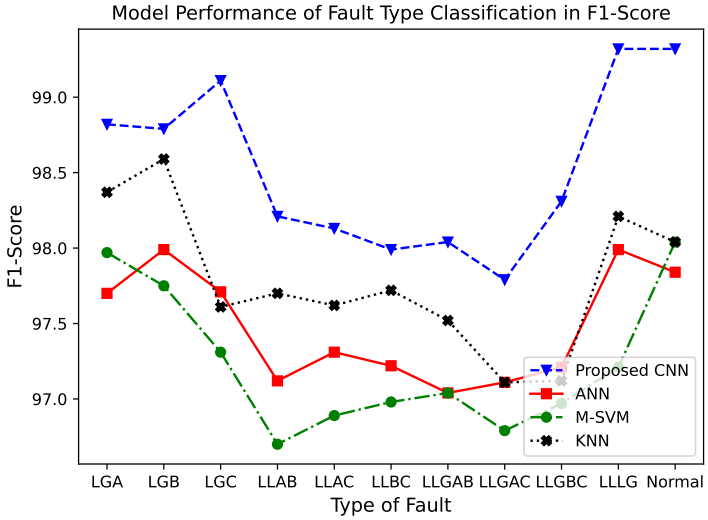
We first compare the proposed framework with the classifiers described in the previous literature to assess the performance of the proposed offline model. Next, we evaluate the effectiveness of the SMOTE approach. Finally, we investigate the robustness to noise. The offline model was trained and tested on the data set under 10% PV penetration level and 80% of the rated load condition. The fault resistance was set to change in the range of 0.01 to 0.05 p.u.

Model Comparison

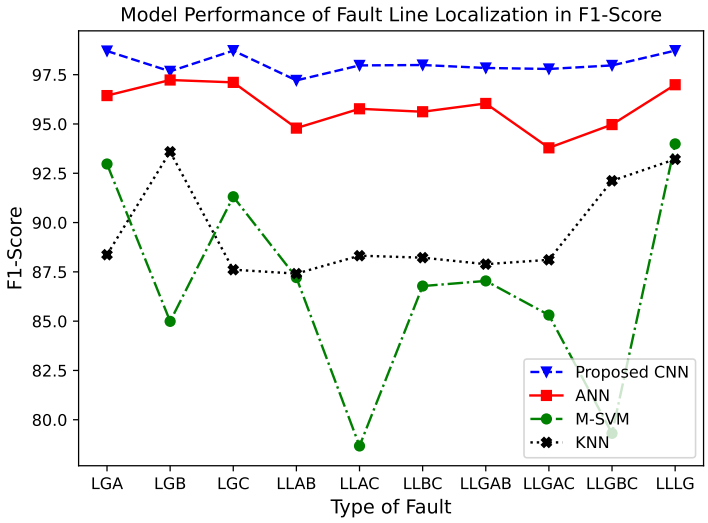
A total of 96,360 data points (70% for training, 20% for validation, and 10% for testing) including 86,000 SMOTE generated data points were used to train and test CNN_C. Meanwhile, a total of 2000 data points (70% for training, 20% for validation, and 10% for testing) including 580 points generated by SMOTE were employed to train and test the CNN_L. We compare the proposed CNN with that of three other machine learning classifiers, including multiclass support vector machine (MSVM) [73], fully connected artificial neural network (ANN), and k-nearest neighbour (KNN).

The MSVM used the coupled pairwise strategy and radial basis function kernel to find the globally optimal solution. ANN of two layers was implemented with 64 neurons in the first layer and 32 neurons in the second layer, 20% dropout, and a 1D-batch normalization layer were applied before the output layer. RELU was selected as the activation function, while the learning rate was set at 0.001. The k parameter in the KNN was set to 11 for the fault type classification case and 50 for the faulted line localization case. The simulation results of the different models and fault types are presented in Figure 4.4a and Figure 4.4b for the fault type classification and FL localization, respectively. The results demonstrate that the proposed CNN_C and CNN_L can detect, classify, and localize distribution system faults accurately and reliably. A high F1 score also indicates that there are fewer misclassified samples as there are fewer FN and FP. Moreover, both models perform much better for all

types of faults compared to the other three classifiers. In particular, the CNN_C model achieved an F1 score of more than 98.5% for the LG, LLLG and normal types, while its performance for the LLG, and LL fault types is only slightly worse. Table 4.4 shows the weighted averages of accuracy over all types of faults for different classifiers. The results further confirm the feasibility of the proposed method.



(a) Fault type classification



(b) Fault line localization

Figure 4.4: Performance comparison on the proposed CNN, ANN, M-SVM, and KNN for 10 and 11 type of faults, respectively, using F1-Score.

Table 4.4: Model performance comparison using weighted average accuracy.

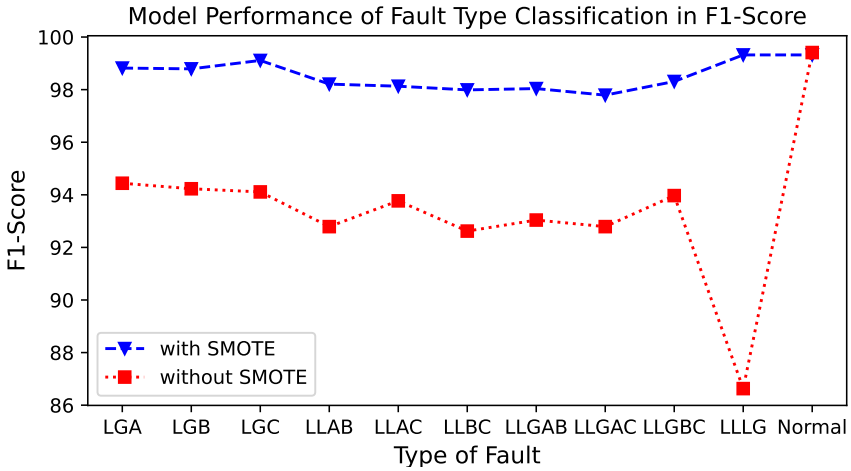
Model Type	**Weighted η_c**	**Weighted η_l**
Proposed CNN	**98.5%**	**97.9%**
ANN	96.3%	96.5%
M-SVM	96.7%	82.1%
KNN	97.1%	93.8%

Effectiveness of SMOTE

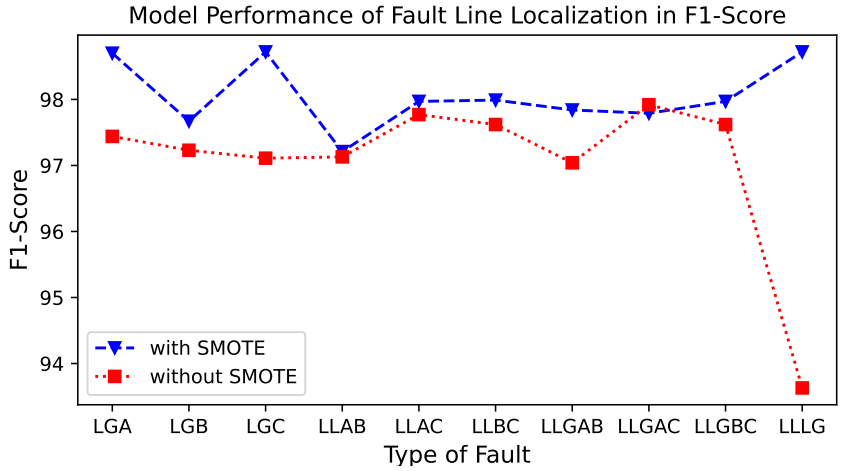
Two types of data set were used to evaluate the effectiveness of SMOTE for fault type classification: the unbalanced simulated data set described in Table 4.1 (10,360 samples in total), and its version augmented using SMOTE (96,360 samples in total). Similarly, the unbalanced data set in Table 4.1 except the normal case (1,420 samples in total) were employed with its augmented version using SMOTE (2,000 samples in total) to evaluate the effectiveness of SMOTE in fault line localization. The results shown in Figure 4.5a and Figure 4.5b demonstrate that the trained model tends to be biased towards the majority class. In particular, We can observe that there is a significant performance drop for the fault type LLLG compared to the other types of faults, since this class has the lowest amount of samples in the unbalanced data set. With SMOTE, the above problem is eliminated as we can observe that both CNN models achieved high F1 scores of about 97% for all types of faults. However, note that there is a trade-off between time and accuracy: as more data samples are generated, the model requires more time for training.

Robustness to Noise

The signal-noise-ratio (SNR) is a commonly used measure that compares the level of desired signal to the background noise, including in PMU data [74]. The experimental range of SNR from 40 dB to 100 dB was selected to test the robustness of the proposed model to noise. Gaussian noise of the same SNR was added to both the training and



(a) Fault type classification



(b) Fault line localization

Figure 4.5: Model performance comparison with and without SMOTE using F1-Score.

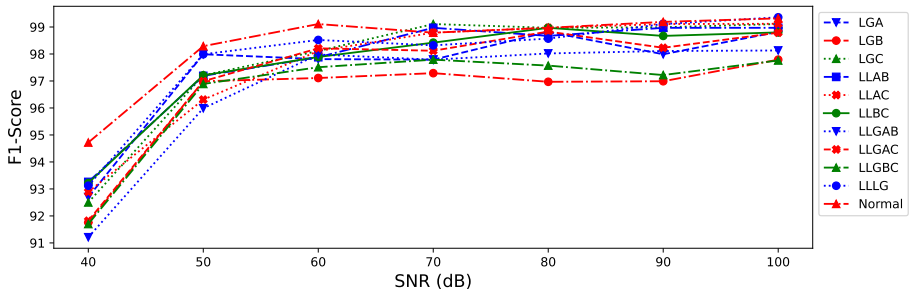
testing data set. Other preprocessing of the data set were the same as described in Section 4.5.1, including the use of SMOTE. The model setting also remained the same for both CNN_C and CNN_L.

Figure 4.6a shows the F1 score for fault type classification with different SNR levels, and Figure 4.6b shows the F1 score for FL localization. The results indicate that the sensitivities of CNN_C and CNN_L to different types of faults differ. For example, the normal class shows a decreasing trend when SNR is 70 and 80 dB, while the LLGAB fault indicates a continuously increasing trend. Moreover, the LGA and

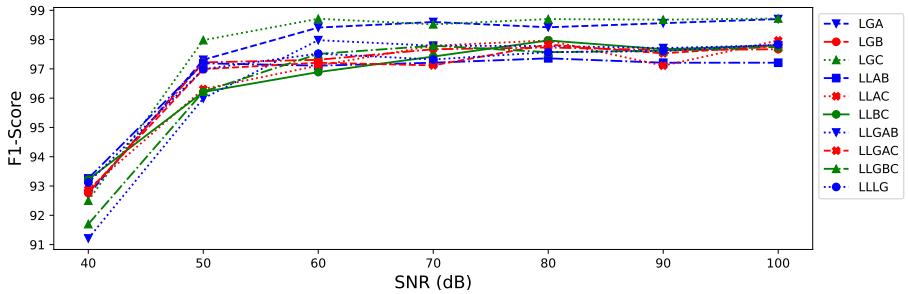
LGC faults are less robust to noise compared to other fault types, while the LLLG and normal case are more robust to noise. A relatively steady trends can be observed when the SNR is higher than 50 dB in both Figure 4.6a and Figure 4.6b where the F1 score can reach 95% or more. Table 4.5 shows the performance of CNN_C and CNN_L with different SNR using weighted average accuracy. When the SNR is 40 dB, a degradation of around 5% can be observed for both models. The influence of noise is contained when the SNR is greater than 60 dB, where the performance does not degrade noticeably.

Table 4.5: Model performance with different SNR (dB) using weighted average accuracy.

SNR (dB)	**40**	**50**	**60**	**70**	**80**	**90**	**100**
Weighted $\eta_c(\%)$	93.2	97.7	98.2	98.5	98.4	98.3	98.5
Weighted $\eta_l(\%)$	92.1	97.1	97.4	97.5	97.5	97.6	97.8



(a) Fault type classification



(b) Fault line localization

Figure 4.6: Model performance comparison with different SNR using F1-Score.

4.5.2 Online Performance

Section 4.5.1 confirmed the feasibility of the CNN-based framework under a distribution system with fixed PV penetration level and fixed rated load. However, in practice, the PV penetration levels and loading conditions may vary over time during the operation. Considering the situation where new data are continuously collected from systems with varying levels of PV penetration and loading, we conducted simulations to test the proposed online continual learning algorithm. As shown in Section 4.5.1, the data with 10% PV penetration level and 80% rated load condition were used to initialize the offline model.

PV Penetration Level Variation

With the recent rapid deployment of DERs, it is crucial that fault detection and localization systems can adapt to various PV penetration levels. To test the ability of the proposed online learning algorithm to handle data involving different levels of PV penetration, additional cases were simulated and data collected from the test distribution system. Specifically, 50 data samples of random fault type were simulated under PV penetration levels of 20%, 30%, and 40%. File data samples for each fault type in the original data along with the new collected data were fed as a data stream to update the trained offline model. Both CNN_C^C and CNN_L^C were trained for 15 epochs while other model settings were identical to the offline models CNN_C and CNN_L

The results for CNN_C^C and CNN_L^C are presented in Figure 4.7a and Figure 4.7b. Both figures show the performance comparison between data set with different PV penetration levels for all types of faults. Note that the line labelled PV-mix represents the data set containing 50 data samples of mixed PV penetration levels (20%, 30%, 40%). Overall, the F1-score for CNN_C^C and CNN_L^C is higher than 96%; it can be concluded that the proposed online learning model can quickly adapt to data with different distribution, in this case with different PV penetration levels. A similar

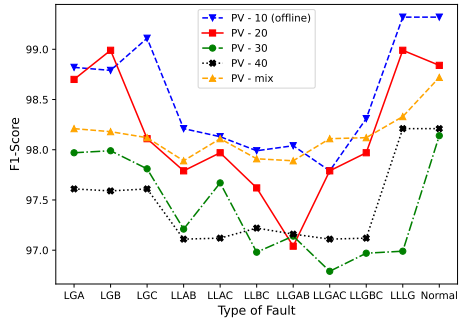
trend in performance over different types of faults can also be captured, e.g., the LG, LLLG and normal type performed relatively better than the LL and LLG fault type. Table 4.6 presents the weighted accuracy η_C and η_L for different levels of PV penetration. It could be seen that the results of the proposed online updating algorithm is quite promising. In the worst case, the accuracy of locating the faulted line decreased only by 1.2% (for PV penetration level of 40%). Furthermore, with the input of different new data, the model update times remain within 1.6 seconds. Therefore, the training process is suitable for practical implementation.

Table 4.6: Model performance with different PV penetration levels using weighted average accuracy.

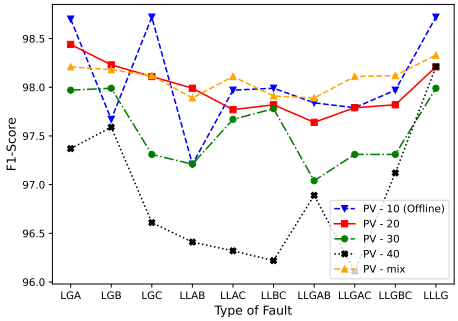
PV Penetration Level	**Weighted η_c**	**Weighted η_l**	**Time**
10% (offline)	98.5%	97.9%	252 s
20%	98.3%	97.5%	1.45 s
30%	97.7%	97.6%	1.56 s
40%	97.2%	96.7%	1.54 s
Mix	98.4%	97.7%	1.48 s

Load Variation

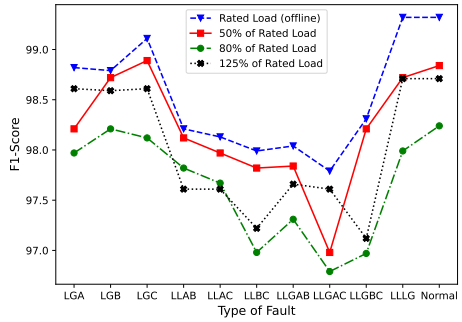
As the load in distribution systems can vary during normal operation over time, it is essential to evaluate the online accuracy of the model when the load is different than its rated value. In this scenario, the data set used for training the model involved 50% , 80%, and 125% of the rated load. In total, 50 data samples were simulated and collected for each case above. Note that other parameters, such as the level of PV penetration, remained the same as for the offline model. Both CNN_C^C and CNN_L^C were trained for 15 epochs while other model settings were identical to the offline models CNN_C and CNN_L. To demonstrate that the online model can effectively adapt to variation of loading, the F1-score was computed for different types of fault in Figure 4.7c and Figure 4.7d. The results demonstrate that the online model can



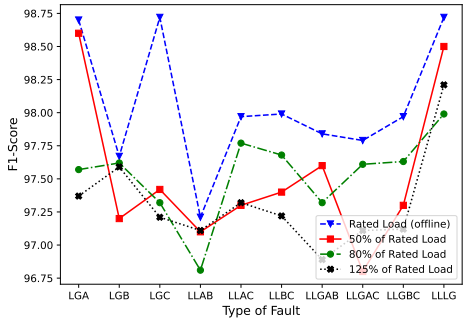
(a) Fault type classification - varying PV



(b) Fault line localization - varying PV



(c) Fault type classification - varying load



(d) Fault line localization - varying load

Figure 4.7: Performance of the proposed online learning model on fault type classification over varying PV penetration level and loading condition.

capture the patterns in data set with varying loading conditions within a short amount of time. Furthermore, the data set with different loading conditions were fed to the trained offline model and the weighted average accuracy were calculated. The results were compared with the online model as illustrated in Table 4.7. The performance of the offline model varies with differing loading conditions. In particular, when the dataset is at 50% of rated load, both offline model CNN_C and CNN_L resulted in a significant performance degradation. Due to the large difference of this new dataset compared to the original dataset which (100% of the rated load), the offline model was not capable to accurately classify and locate the faults. On the other hand, the online model achieved an average accuracy of above 97% in locating the fault lines. Moreover, the performance of online model does not degrade or vary significantly over

different loading conditions.

Table 4.7: Offline and online model performance comparison with different loading conditions using weighted average accuracy.

Metrics	**50% Load**	**80% Load**	**125% Load**
Weighted η_c (CNN_C^C)	98.5%	97.8%	97.5%
Weighted η_l (CNN_L^C)	97.3%	97.5%	97.3%
Weighted η_c (CNN_C)	84.2%	91.9%	92.4%
Weighted η_l (CNN_L)	81.7%	89.4%	91.1%

4.6 Conclusions

In this chapter, we describes a novel real-time data-driven framework for fault classification and location in partially observable distribution systems, based on CNN. The proposed framework uses special feature vectors extracted from the voltage magnitude and phase angles. The extracted feature vectors play an important role as they are highly correlated with the faults and fundamentally assist the CNN in identifying patterns within data. The fault classification and localization performance is further increased using SMOTE of the data prepossessing stage to augment the limited amount of simulated fault data. Moreover, an online continual learning algorithm based on transfer learning and calibration modules is proposed to accommodate variations of the DER integration level and loading in distribution systems over time. A real distribution grid located in the Midwestern United States was used to confirm the feasibility of the proposed method and to evaluate its properties. Testing results demonstrate that the proposed offline CNN-based models can accurately classify and locate faults with F1 scores greater than 97%. The simulation results also confirm the robustness of the proposed model to different levels of noise. In addition, the online model has shown its effectiveness in adapting to variation of PV penetration and loading condition within a short amount of time.

Chapter 5

Conclusions, Recommendations, & Future Work

5.1 Conclusions

This thesis explores the application of deep learning models to detect, classify and locate anomalies in smart distribution grids. To analyze the suitability of these techniques, a real distribution grid located in the Mid-western United States has been used. This fully observable test system includes one-year smart meter measurement of all customers, system component parameters, and detailed network topology. First, a long-short-term memory model with a threshold module was developed to detect events caused by changes in the distribution system load, such as customer disconnected from the grid and unexpected voltage drop. This model was compared to another neural network model-based approach. After identifying the anomalies, it is important to classify and locate them. Therefore, a convolutional neural network-based framework was proposed for fault classification and fault line location identification in partially observable distribution systems. To avoid expensive retraining process when processing data with distribution different from original training data, we incorporate transfer learning and calibration modules to dynamically train the network to enable online continual learning. For validation, the proposed model was compared to other classic machine learning models. Finally, experiments were conducted for both offline and online mode, and model robustness was also examined.

The results confirmed that the proposed LSTM model with threshold modules are well suited for identifying anomalies when there are only data collected under normal conditions available and no anomaly data. The LSTM model's advantage in time series forecasting is due to its capability to learn long-term dependencies. This has lead to a 0.10% in MAE and 0.15% in RMSE score, setting a strong foundation for the threshold modules in identifying anomalies. This proposed anomaly detector operates in an unsupervised manner and solves the data imbalance problem caused by the lack of anomaly data. In addition, the results demonstrated the effectiveness of the proposed convolutional neural network-based framework for fault classification and location identification. The extracted special features play an important role because they are highly correlated with faults. The proposed CNN-based model achieved the best performance in the conducted experiments. Specifically, the proposed CNN-based offline model was able to classify and locate faults accurately with an F1 score higher than 97%. Furthermore, the online model shows its effectiveness in adapting to changes in PV penetration and load conditions in a short time.

In conclusion, this thesis proposed and validated deep learning approaches for detecting, classifying and locating anomalies in smart distribution networks. The contributions during the experiments validated the feasibility and effectiveness of using deep learning methods derived from natural language processing and image recognition for smart distribution network anomaly detection. The experimental results showed that the proposed system can be implemented on a real-time basis without the extensive work of retraining the entire model with newly collected data.

5.2 Contributions

In the course of the experiment, the following contributions were made to accomplish the two main objectives of this work.

1. The first objective is to apply unsupervised ML techniques that use only normal

data to detect events caused by load changes in the distribution system. Three types of events are considered in this study: customer disconnected from the grid, unexpected voltage drop, and voltage rises caused by DER.

- We first analyzed long-short-term memory (LSTM) model and their application in time-series forecasting.
- Next, a threshold function based on the Euclidean distance was introduced to detect anomalies by comparing the predicted and actual next time step voltage data.
- The performance of LSTM model was compared with an one-dimensional convolutional neural network (1D-CNN), and the proposed model is further validated to demonstrate the effectiveness of the LSTM.

2. The second objective is to develop a CNN-based framework for fault classification and location identification in partially observable distribution systems. Four fault types are considered in this study: line-to-line fault (LLF), line-to-ground fault (LGF), double line-to-ground fault (LLGF), three-line-to-ground fault (LLLGF).

   - First, additional features that contribute to fault type classification and fault line localization were explored and extracted.
   - Next, we investigated the statistical method, SMOTE, in augmenting the limited amount of simulated fault data.
   - Then, we explored and applied CNN models for fault classification and location identification. We also explored online continual learning algorithm to accommodate variations in power system over time.
   - Finally, further validation of proposed model was conducted and model robustness to noise was examined.

5.3 Future Work

Overall, this thesis demonstrated that deep learning techniques can be transferred to anomaly detection in smart distribution grids. In the future, this work will be extended to determine the exact location of faults along a line. Online models can also be optimized to achieve shorter model update times while maintaining model accuracy and adapting to different topologies of the distribution network. Finally, to potentially improve the performance of the model, more advanced enhancement techniques, such as generative adversarial networks, can be used to perform enhanced investigations of fault data.

Bibliography

[1] H. Jiang, K. Wang, Y. Wang, M. Gao, and Y. Zhang, "Energy big data: A survey," *IEEE Access*, vol. 4, pp. 3844–3861, 2016.

[2] K Moloi, M Ntombela, T. C. Mosetlhe, T. R. Ayodele, and A. A. Yusuff, "Feature extraction based technique for fault classification in power distribution system," in *2021 IEEE PES/IAS PowerAfrica*, IEEE, 2021, pp. 1–5.

[3] S. H. Horowitz and A. G. Phadke, *Power system relaying*. John Wiley & Sons, 2014.

[4] F. Bu, Y. Yuan, Z. Wang, K. Dehghanpour, and A. Kimber, "A time-series distribution test system based on real utility data," in *2019 North American Power Symposium (NAPS)*, IEEE, 2019, pp. 1–6.

[5] L. Ruff *et al.*, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021.

[6] J. E. Zhang, D. Wu, and B. Boulet, "Time series anomaly detection for smart grids: A survey," in *2021 IEEE Electrical Power and Energy Conference (EPEC)*, IEEE, 2021, pp. 125–130.

[7] *Average frequency and duration of electric distribution outages vary by states*. [Online]. Available: https://www.eia.gov/todayinenergy/detail.php?id=35652.

[8] C. Masetti, "Revision of european standard en 50160 on power quality: Reasons and solutions," in *Proceedings of 14th International Conference on Harmonics and Quality of Power - ICHQP 2010*, 2010, pp. 1–7.

[9] R. Tonkoski, D. Turcotte, and T. H. El-Fouly, "Impact of high pv penetration on voltage profiles in residential neighborhoods," *IEEE Transactions on Sustainable Energy*, vol. 3, no. 3, pp. 518–527, 2012.

[10] L. R. Almobasher and I. O. A. Habiballah, "Review of power system faults," *no. November*, 2020.

[11] M. Shafiullah and M. A. Abido, "A review on distribution grid fault location techniques," *Electric Power Components and Systems*, vol. 45, no. 8, pp. 807–824, 2017.

[12] A. K. Abbas, S. Hamad, and N. A. Hamad, "Single line to ground fault detection and location in medium voltage distribution system network based on neural network," *Indones. J. Electr. Eng. Comput. Sci*, vol. 23, pp. 621–632, 2021.

[13] P. K. Lim and D. S. Dorr, "Understanding and resolving voltage sag related problems for sensitive industrial customers," in *2000 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No. 00CH37077)*, IEEE, vol. 4, 2000, pp. 2886–2890.

[14] Y. Himeur, A. Alsalemi, F. Bensaali, and A. Amira, "A novel approach for detecting anomalous energy consumption based on micro-moments and deep neural networks," *Cognitive Computation*, vol. 12, no. 6, pp. 1381–1401, 2020.

[15] X. Wang, T. Zhao, H. Liu, and R. He, "Power consumption predicting and anomaly detection based on long short-term memory neural network," in *2019 IEEE 4th international conference on cloud computing and big data analysis (ICCCBDA)*, IEEE, 2019, pp. 487–491.

[16] S. Parsai and S. Mahajan, "Anomaly detection using long short-term memory," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, IEEE, 2020, pp. 333–337.

[17] A. Barua, D. Muthirayan, P. P. Khargonekar, and M. A. Al Faruque, "Hierarchical temporal memory based machine learning for real-time, unsupervised anomaly detection in smart grid: Wip abstract," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*, IEEE, 2020, pp. 188–189.

[18] K. Khanna, B. K. Panigrahi, and A. Joshi, "Ai-based approach to identify compromised meters in data integrity attacks on smart grid," *IET Generation, Transmission & Distribution*, vol. 12, no. 5, pp. 1052–1066, 2018.

[19] Y. Yuan, K. Dehghanpour, F. Bu, and Z. Wang, "Outage detection in partially observable distribution systems using smart meters and generative adversarial networks," *IEEE Transactions on Smart Grid*, vol. 11, no. 6, pp. 5418–5430, 2020.

[20] A. A. Imayakumar, A. Dubey, and A. Bose, "Anomaly detection for primary distribution system measurements using principal component analysis," in *2020 IEEE Texas Power and Energy Conference (TPEC)*, IEEE, 2020, pp. 1–6.

[21] B. Xiang, Z. Liu, and K. Zhang, "Flagging implausible inspection reports of distribution transformers via anomaly detection," *IEEE Access*, vol. 8, pp. 75 798–75 808, 2020.

[22] X. Yang, N. Chen, and C. Zhai, "A control chart approach to power system line outage detection under transient dynamics," *IEEE Transactions on Power Systems*, vol. 36, no. 1, pp. 127–135, 2020.

[23] X. He, Q. Ai, R. C. Qiu, W. Huang, L. Piao, and H. Liu, "A big data architecture design for smart grids based on random matrix theory," *IEEE transactions on smart Grid*, vol. 8, no. 2, pp. 674–686, 2015.

[24] R. Moslemi, M. Davoodi, and J. M. Velni, "A distributed approach for estimation of information matrix in smart grids and its application for anomaly detection," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, IEEE, 2020, pp. 1–7.

[25] H. Karimipour and H. Leung, "Relaxation-based anomaly detection in cyber-physical systems using ensemble kalman filter.," *IET Cyper-Phys. Syst.: Theory & Appl.*, vol. 5, no. 1, pp. 49–58, 2020.

[26] J. Ding, X. Wang, Y. Zheng, and L. Li, "Distributed traveling-wave-based fault-location algorithm embedded in multiterminal transmission lines," *IEEE Transactions on Power Delivery*, vol. 33, no. 6, pp. 3045–3054, 2018.

[27] F. V. Lopes, K. M. Silva, F. B. Costa, W. L. A. Neves, and D. Fernandes, "Real-time traveling-wave-based fault location using two-terminal unsynchronized data," *IEEE Transactions on Power Delivery*, vol. 30, no. 3, pp. 1067–1076, 2015.

[28] E. R. Sanseverino, V. L. Vigni, A. Di Stefano, and R. Candela, "A two-end traveling wave fault location system for mv cables," *IEEE Transactions on Industry Applications*, vol. 55, no. 2, pp. 1180–1188, 2018.

[29] I Dzafic and P Mohapatra, "Impedance based fault location for weakly meshed distribution networks," in *ISGT 2011*, IEEE, 2011, pp. 1–6.

[30] S.-J. Lee *et al.*, "An intelligent and efficient fault location and diagnosis scheme for radial distribution systems," *IEEE transactions on power delivery*, vol. 19, no. 2, pp. 524–532, 2004.

[31] F. Aboshady, D. Thomas, and M. Sumner, "A new single end wideband impedance based fault location scheme for distribution systems," *Electric Power Systems Research*, vol. 173, pp. 263–270, 2019.

[32] M. Esmalifalak, L. Liu, N. Nguyen, R. Zheng, and Z. Han, "Detecting stealthy false data injection using machine learning in smart grid," *IEEE Systems Journal*, vol. 11, no. 3, pp. 1644–1652, 2014.

[33] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 8, pp. 1773–1786, 2015.

[34] A. Malhotra, O. P. Mahela, and H. Doraya, "Detection and classification of power system faults using discrete wavelet transform and rule based decision tree," in *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, 2018, pp. 142–147.

[35] M. Shaik, S. K. Yadav, and A. G. Shaik, "An emd and decision tree-based protection algorithm for the solar pv integrated radial distribution system," *IEEE Transactions on Industry Applications*, vol. 57, no. 3, pp. 2168–2177, 2021.

[36] A. Zainab, S. S. Refaat, D. Syed, A. Ghrayeb, and H. Abu-Rub, "Faulted line identification and localization in power system using machine learning techniques," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 2975–2981. DOI: 10.1109/BigData47090.2019.9006377.

[37] Y. Liang, K.-J. Li, Z. Ma, and W.-J. Lee, "A multi-label classification model for type recognition of single-phase-to-ground fault based on knn-bayesian method," in *2020 IEEE/IAS Industrial and Commercial Power System Asia (I CPS Asia)*, 2020, pp. 929–936.

[38] H. R. Baghaee, D. Mlakić, S. Nikolovski, and T. Dragicević, "Support vector machine-based islanding and grid fault detection in active distribution networks," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 8, no. 3, pp. 2385–2403, 2019.

[39] R. Perez, C. Vásquez, and A. Viloria, "An intelligent strategy for faults location in distribution networks with distributed generation," *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 2, pp. 1627–1637, 2019.

[40] X. Zheng, X. Geng, L. Xie, D. Duan, L. Yang, and S. Cui, "A svm-based setting of protection relays in distribution systems," in *2018 IEEE Texas Power and Energy Conference (TPEC)*, IEEE, 2018, pp. 1–6.

[41] M. U. Usman, J. Ospina, and M. O. Faruque, "Fault classification and location identification in a smart dn using ann and ami with real-time data," *The Journal of Engineering*, vol. 2020, no. 1, pp. 19–28, 2020.

[42] M. Guo, X. Zeng, D. Chen, and N. Yang, "Deep-learning-based earth fault detection using continuous wavelet transform and convolutional neural network in resonant grounding distribution systems," *IEEE Sensors Journal*, vol. 18, no. 3, pp. 1291–1300, 2017.

[43] W. Li, D. Deka, M. Chertkov, and M. Wang, "Real-time faulted line localization and pmu placement in power systems through convolutional neural networks," in *2020 IEEE Power Energy Society General Meeting (PESGM)*, 2020, pp. 1–1.

[44] M. Zhao and M. Barati, "A real-time fault localization in power distribution grid for wildfire detection through deep convolutional neural networks," *IEEE Transactions on Industry Applications*, vol. 57, no. 4, pp. 4316–4326, 2021.

[45] Z. Lin *et al.*, "One-class classifier based fault detection in distribution systems with varying penetration levels of distributed energy resources," *IEEE Access*, vol. 8, pp. 130 023–130 035, 2020.

[46] S. Theodoridis and K. Koutroumbas, *Pattern recognition*. Elsevier, 2006.

[47] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, "Deep learning [http://www. deeplearningbook. org]," *MIT Press, Cambridge, MA*, 2016.

[48] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[49] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[50] S. Bhattacharya *et al.*, "Deep learning and medical image processing for coronavirus (covid-19) pandemic: A survey," *Sustainable cities and society*, vol. 65, p. 102 589, 2021.

[51] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and trends in signal processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[52] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification," *Neurocomputing*, vol. 321, pp. 321–331, 2018.

[53] S. Mehtab and J. Sen, "Analysis and forecasting of financial time series using cnn and lstm-based deep learning models," in *Advances in Distributed Computing and Machine Learning*, Springer, 2022, pp. 405–423.

[54] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-vgg16 cnn model for big data places image recognition," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2018, pp. 169–175.

[55] P. Swietojanski, A. Ghoshal, and S. Renals, "Convolutional neural networks for distant speech recognition," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1120–1124, 2014.

[56] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[57] L. Medsker and L. C. Jain, *Recurrent neural networks: design and applications*. CRC press, 1999.

[58] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," *Advances in neural information processing systems*, vol. 9, 1996.

[59] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[60] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[61] H. Hatta, M. Asari, and H. Kobayashi, "Study of energy management for decreasing reverse power flow from photovoltaic power systems," in *2009 IEEE PES/IAS Conference on Sustainable Alternative Energy (SAE)*, IEEE, 2009, pp. 1–5.

[62] K. Clement-Nyns, E. Haesen, and J. Driesen, "The impact of charging plug-in hybrid electric vehicles on a residential distribution grid," *IEEE Transactions on power systems*, vol. 25, no. 1, pp. 371–380, 2009.

[63] Z. Wang and T. Yan Weizhong dnd Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International joint conference on neural networks (IJCNN)*, IEEE, 2017, pp. 1578–1585.

[64] A. H. Sparks, "Nasapower: A nasa power global meteorology, surface solar energy and climatology data client for r," *The Journal of Open Source Software*, vol. 3, no. 30, p. 1035, 2018. DOI: 10.21105/joss.01035.

[65] F. C. Trindade and W. Freitas, "Low voltage zones to support fault location in distribution systems with smart meters," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2765–2774, 2016.

[66] M. Abdel-Akher and K. M. Nor, "Fault analysis of multiphase distribution systems using symmetrical components," *IEEE Transactions on Power Delivery*, vol. 25, no. 4, pp. 2931–2939, 2010.

[67] T. Tieleman, G. Hinton, *et al.*, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[68] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[69] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[70] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*, Springer, 2018, pp. 270–279.

[71] P. Singh, V. K. Verma, P. Mazumder, L. Carin, and P. Rai, "Calibrating cnns for lifelong learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 579–15 590, 2020.

[72] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[73] S Pöyhönen, A. Arkkio, P Jover, and H Hyötyniemi, "Coupling pairwise support vector machines for fault classification," *Control Engineering Practice*, vol. 13, no. 6, pp. 759–769, 2005.

[74] M. Brown, M. Biswal, S. Brahma, S. J. Ranade, and H. Cao, "Characterizing and quantifying noise in pmu data," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*, 2016, pp. 1–5. DOI: 10.1109/PESGM.2016.7741972.

[75] S. S. Gururajapathy, H. Mokhlis, and H. A. Illias, "Fault location and detection techniques in power distribution systems with distributed generation: A review," *Renewable and sustainable energy reviews*, vol. 74, pp. 949–958, 2017.