NEURAL NETWORK FAULT RECOGNITION IN POWER SYSTEMS WITH HIGH PENETRATION OF INVERTER-BASED RESOURCES

A Thesis Submitted to the

College of Graduate and Postdoctoral Studies

In Partial Fulfillment of the Requirements

For the Degree of Master of Science

In the Department of Electrical and Computer Engineering

University of Saskatchewan

Saskatoon

by

Connor A. Tant

B. Eng., P. Eng.

Permission to Use

Acknowledgments

I would like to express my heartfelt gratitude to Prof. Rama Gokaraju for his unwavering support and guidance throughout my M.Sc. program. His patience and mentorship have been instrumental in shaping my knowledge and expertise in the field of power system protection, EMT modelling, renewable energy, and various other topics. The education I have received under his tutelage has been invaluable, and I am deeply thankful for the knowledge and skills I have gained.

I am also deeply grateful to my friends and family for their constant support and motivation, especially my fiance Kaylee, whose encouragement and belief in me have been a driving force behind my achievements.

I extend my thanks to the entire team and leadership at Team Power Solutions for their continuous support. Their understanding and flexibility in granting me leave when needed have allowed me to pursue my academic goals while contributing to the advancement of the electrical power and affiliate industries. The contemporary approach to electrical engineering and innovation demonstrated by the leadership and my colleagues has been a great source of inspiration in my educational journey.

Furthermore, I would like to acknowledge and thank all the members of the Power Systems Simulation Laboratory research group for creating a welcoming and supportive environment during my time at the University of Saskatchewan. Their camaraderie and encouragement have made my academic experience truly enjoyable.

Abstract

The growing demand for renewable energy resources (RER) has led to increased integration of inverter-based resources (IBRs), into existing power distribution and transmission networks. However, RER locations are often not ideally suited for direct integration, necessitating a restructuring of the grid from a traditional radial network to a more complex mesh network topology. This transition presents challenges in terms of protection and coordination, as IBRs exhibit atypical responses to power system anomalies compared to conventional synchronous generation.

To address these challenges and support existing power system protection infrastructure, this work explores the incorporation of machine learning algorithms. Specifically, an optimized convolutional neural network (CNN) is developed for real-time application in power system protection schemes. The focus is on prioritizing key performance metrics such as recall, specificity, speed, and the reduction of computational resources required for effective protection.

The machine learning model is trained to differentiate between healthy system dynamics and hazardous conditions, such as faults, in the presence of IBRs. By analyzing data retrieved from an IEEE 34-bus 24kV distribution network, the model's application is demonstrated and its performance is evaluated.

A photovoltaic (PV) source was incorporated into the IEEE 34-bus distribution feeder model at the end of the feeder. By adding a PV source at the end of the feeder, IBR characteristics, such as its response to system anomalies can be monitored through the model.

Once the modified IEEE 34-bus distribution feeder model with the PV source was set up, various system anomalies were simulated to create a diverse dataset for training the machine learning (ML) model. These anomalies included; load rejection - a sudden and complete removal of load from the distribution network, simulating a scenario where a significant portion of the load disconnects from the grid, load addition - a sudden and significant increase in load demand, representing a scenario where new loads are connected to the grid, islanding - a scenario where the distribution feeder becomes electrically isolated from the main grid, with the PV source acting as a microgrid and supplying power to the local loads, and various types of faults, such as short-circuits or ground faults, occurring at different locations along the distribution line.

To create a diverse dataset, model parameters were varied through 50 different iterations of each simulated anomaly scenario. These parameters included the PV system's capacity, the location of the anomaly on the feeder, the severity and duration of the anomaly, and other relevant grid parameters.

For each iteration and anomaly scenario, the responses of the system were recorded, including voltage levels, current flows, and other relevant synchorphasors at the PV source's point of common coupling (PCC). These responses formed the dataset for training the ML model.

The accumulated dataset was then used to train the various ML models, including the optimized convolutional neural network (CNN), to identify patterns and hidden characteristics in the data corresponding to different system anomalies. The training process involved feeding the model with input data from the various iterations and scenarios, along with corresponding labels indicating the type of anomaly present.

By exposing the ML model to diverse scenarios and varying parameters, the model learns to generalize its understanding of system dynamics and accurately distinguish between healthy system states and hazardous conditions. The models in this work were specifically trained to recognize the various fault characteristics on the system. The trained model's ability to process time-series data and recognize anomalies from the accumulated dataset enhances power system protection infrastructure's capability to respond rapidly and accurately to various grid disturbances, ensuring the reliable and stable operation of the distribution network, especially in the presence of PV and other IBRs.

The results show that the optimized CNN outperforms traditional machine learning models used in time-series data analysis. The model's speed and reliability make it an effective tool for identifying hidden characteristics in power system data without the need for extensive manual analysis or rigid programming of existing protection relays. This capability is particularly valuable as power grids integrate a higher penetration of IBRs, where traditional protection infrastructure may not fully account for their unique responses.

The successful integration of the optimized CNN into power system protection infrastructure enhances the grid's ability to detect and respond to anomalies, such as faults, in a more efficient and accurate manner. By leveraging machine learning techniques, power system operators can better adapt to the challenges posed by the increasing presence of IBRs and ensure the continued stability and reliability of the distribution network.

Table of Contents

List of Tables

List of Figures

List of Acronyms & Abbreviations

ACSR - Aluminum Conductor Steel Reinforced

ADC - Analog-to-Digital Converter

ANN - Artificial Neural Network

BPNN - Back-Propagation Neural Network

BPTT - Back-Propagation Through Time

CB - Circuit Breaker

CNN - Convolutional Neural Network

EMT - Electromagnetic Transient

FFT - Fast Fourier Transform

FN - False Negative

FP - False Positive

GLVQ - Generalized Learning Vector Quantization

GFOV - Ground Fault Overvoltage

GRU - Gated Recurrent Unit

IBR - Inverter-Based Resource

IPP - Independent Power Producer

LCD - Line Current Differential

LSTM - Long Short-Term Memory

LVQ - Learning Vector Quantization

MPPT - Maximum Power Point Tracking

ML	- Machine Learning
OC	- Overcurrent
OV	- Overvoltage
PCC	- Point of Common Coupling
PMU	- Phasor Measurement Unit
PSCAD	- Power Systems Computer-Aided Design
PU	- Per Unit
PV	- Photovoltaic
RER	- Renewable Energy Resource
ROCOF	- Rate of Change of Frequency
RMS	- Root-Mean-Square
RNN	- Recurrent Neural Network
SVM	- Support Vector Machine
SVC	- Support Vector Classification
TN	- True Negative
TOV	- Transient Overvoltage
TP	- True Positive
VSC	- Voltage Source Converter

Chapter 1

Introduction

1.1 Background

The deployment of distributed energy resources (DERs) is increasing. These sources are integrated into the distribution grid throughout the system. Introducing DERs into a system quickly converts the traditional radial network to a mesh network topology. Power distribution groups must significantly restructure protection schemes. Due to the physics and intermittency of renewable sources, these DERs often rely on power inverters to tie into the distribution grid, adding an increased concern to system protection and reliability.

Traditional generation had been mainly composed of rotating machines called synchronous generators. Sources reliant on inverters are often referred to as inverter-based resources (IBRs). The physics behind synchronous generators and IBRs are fundamentally different. This is apparent in the short-circuit characteristics of the inverters and their response to faults. Synchronous generator fault characteristics are defined by their high-magnitude current, dictated by the source parameters and the impedance of the short-circuit path to ground. In contrast, IBRs typically produce low-magnitude short-circuit currents, limited by the power electronics of the inverters as a means of self-preservation. These characteristics are defined by the inverter manufacturer and largely undisclosed - leading to inconsistencies in the response between the various IBRs integrated into a distribution grid.

Traditional distribution feeders are set up as radial feeders with a single source of fault current. Because of this, overcurrent (OC) protection was predominantly used for fault detection and mitigation. OC protection devices will monitor feeder current and operate a fault-clearing device such as a circuit breaker (CB) or recloser once the current exceeds a predefined value. This operation will be triggered after a predefined period of time in order to coordinate with other protection devices on the system in an attempt to protect system infrastructure, the public and the environment while mitigating the number of affected loads. When distributed generation is connected through a distribution system, another source of fault current is introduced. In the case of a ground fault, a second source will create a parallel path to ground, partially offsetting the original source fault current contribution - directly affecting the response of the original OC protection equipment [1].

As power systems begin to integrate high levels of IBRs, traditional infrastructure can be unreliable during anomalies in the network. The difference in power delivery characteristics between synchronous generators and IBRs as well as the inconsistencies between IBRs themselves not only make traditional protection equipment ineffective but require new types of protection methods that can effectively accommodate the inconsistencies between all types of distributed RERs.

The objective of the research work reported in this thesis is to deal with the modified power system topology and the dynamic characteristics of IBRs during transient conditions through the development and application of a deep learning algorithm optimized for mitigating the effects of harmful system anomalies. As standard industry design and implementation of IBRs is developed, machine learning models can be applied to monitor electrical distribution system health and react as needed during grid abnormalities such as faults.

Machine learning models have been developed to analyze time-series data. These models can be trained to continuously monitor system data and recognize the difference between typical system characteristics and anomalies such as the occurrence of a fault on the power system [2]. Implementing this type of algorithm into existing protection schemes can increase the reliability and effectiveness of most overcurrent and overvoltage protection. In this work, a convolutional neural network model is developed to reduce the speed and computational resources required to recognize harmful system abnormalities and effectively incorporate a deep learning algorithm into power system protection schemes.

1.2 Inverter-Based Resource Transient Characteristics & Protection

1.2.1 DER Overcurrent Characteristics

Traditional distribution feeders are set up as radial feeders with a single source of fault current. Because of this, overcurrent (OC) protection was predominantly used for fault detection and mitigation. OC protection devices will monitor feeder current and operate a fault-clearing device such as a circuit breaker (CB) or recloser once the current exceeds a predefined value. This operation will be triggered after a predefined period of time in order to coordinate with other protection devices on the system in an attempt to protect system infrastructure, the public and the environment while mitigating the number of affected loads. When distributed generation is connected through a distribution system, another source of fault current is introduced. In the case of a ground fault, a second source will create a parallel path to ground, partially offsetting the original source fault current contribution - directly affecting the response of the original OC protection equipment.

Synchronous Generators

Synchronous generators' equivalent Thevenin circuit can be effectively represented and modelled as a voltage source in series with a relatively low impedance in comparison to the impedance of the distribution network. During a fault event, the fault current level and characteristics are defined by the generator capacity, the generator impedance and the impedance of the distribution network between the generator and the fault location. When a fault occurs in the distribution network, the synchronous generator's low internal impedance produces a significant fault current.

Protection relays take advantage of the characteristics of synchronous generator sources during overcurrent events to effectively protect the distribution network. Overcurrent protection relays monitor the current flowing through the protected circuit and are set to operate when the current exceeds a predefined threshold. Due to the severity of the OC events produced by the synchronous generators, the operation threshold can in most cases be clearly defined in order to differentiate between fault OC events and other system anomalies.

The low internal impedance of synchronous generators ensures that they contribute a substantial amount of fault current during fault events. This characteristic is advantageous for overcurrent protection, as the relays can clearly differentiate between fault conditions and other system anomalies, they can quickly detect and initiate the isolation process, minimizing potential damage to equipment and ensuring the safety and stability of the distribution network. These characteristics also ease the coordination of the various protection relays in the distribution network, ensuring that the nearest relay to the fault location operates first while allowing downstream relays enough time to clear the fault without causing unnecessary tripping. This coordinated protection strategy helps maintain the integrity of the distribution system while effectively handling overcurrent events from synchronous generator sources.

Inverter-Based Resources

IBRs have considerably different fault characteristics than conventional synchronous generators. Reports on IBRs characterize their fault current response as a low-magnitude, positive-sequence contribution and as negligible sources of negative- and zero-sequence current regardless of the fault type. The short-circuit currents are typically limited to 1.2 per unit (pu) by the inverter controller in order to protect the electronics [3]. However, the initial transient current can be as high as 2.5 pu for 1/4 to 2 cycles before it is reduced by the controller to the limited output within the first 6 cycles [4].

Unlike conventional synchronous generators, these sources are not represented as a voltage source. A photovoltaic (PV) inverter, will regulate its DC voltage to achieve the maximum power at the specific irradiance level on the panels. Through power and current control loops, the inverter will regulate the output current in order to maintain the maximum power generation [5]. In other words, as the interconnection voltage fluctuates, the inverter will vary the output current to the maximum power output - therefore operating as a voltage-controlled current source. This specific mode of control is referred to as maximum power point tracking (MPPT).

PV inverters can typically be grouped into two operating modes - DC voltage control and maximum power point tracking (MPPT) control. In DC voltage control mode, the inverter's booster converter maintains the DC voltage so that the voltage source converter (VSC) can control the active power on the DC link. This mode of operation is utilized when the reactive power output of the inverter is a priority. In MPPT control mode, the maximum real power yield is taken from the PV array. Since small independent power producer (IPP) commercial compensation is dependent on the amount of real power injected into the utility, IPPs will typically operate their inverters in MPPT control mode to optimize the real power output. Operating at near unity power factor by limiting the reactive power output, can produce drastically different characteristics when compared to synchronous generators or inverters operating in DC voltage control mode [1]. Furthermore, the various inverter operating modes can have a significant effect on the expected fault current produced by an IBR.

IEEE 1547 introduced ride-through requirements for IBRs [6]. These requirements force IBRs to inject reactive power into the system for voltage support. This standardization provides helpful metrics for protection engineers carrying out system protection studies and creating protection schemes. During fault conditions, the reduced system voltage will cause the inverter controls to increase the current regulator target output in order to maintain constant power. The transient response of IBR in system fault scenarios is driven by the inverter control schemes. The first consideration these control schemes prioritize is to manage and maintain current magnitudes within the thermal withstand capability of the internal power electronics. As the power electronic devices used in inverters are sensitive to overcurrent beyond their rated values, these devices can fail if exposed to overcurrent levels several times their rated values for less than a cycle. Modern inverters will therefore recognize the transient conditions and limit the current to a sustainable output.

Reference [7] discusses the various levels of system protection that are impacted by introducing distributed IBRs. This list includes line distance protection, memory-polarized zero sequence directional protection, negative sequence-based directional ground fault protection, negative sequence overcurrent elements, pilot protection, line current differential (LCD), rate of change of frequency (ROCOF) and power swing protection.

Although PV inverters will typically act as a current source, in severe transient overvoltage conditions, the inverter controller can become saturated [5]. The output of the inverter is then determined by the DC voltage, causing the inverter to momentarily act as a voltage source, adding further complexity to the distribution system protection scheme.

1.2.2 Transient Overvoltage Concerns

The increase in DER introduces a growing concern for TOV and a need for restructuring TOV mitigation practices. Distribution surge arresters are applied to mitigate lighting strikes overvoltages. Arresters are therefore sized to have a minimum rating above the ground fault overvoltage (GFOV) magnitude [8]. Other voltage causes in distribution networks include ferroresonance, load rejection, loss of ground and transformer inrush.

When a significant load is isolated from the system and there is no substantial load to sink the current, load rejection overvoltage (LROV) conditions may occur until the abnormality is recognized and the inverters are shut down by protective devices. LROV is a balanced positive sequence overvoltage that is independent of the system grounding.

Transformer inrush can also lead to overvoltages. These inrushes can often saturate the power transformer's iron core, leading to resonance with the distribution system's capacitance.

GFOV is an unbalanced overvoltage dependent on the system grounding and how much the system neutral shifts during a fault. During ground fault conditions, the voltage on the unfaulted phases increases while the faulted phase voltage moves to the potential at the system neutral.

In DER networks, transmission source transformers acting as ground sources will be isolated from the system during ground faults on the sub-transmission system. When distribution source transformers that typically act as a source to the LV side backfeed into the transmission system, they will act as an ungrounded source during GF conditions, resulting in loss of ground overvoltage conditions.

Although the mitigation of overvoltages is often dependent on physical system characteristics, differentiating between the various causes of overvoltage is a useful metric to react accordingly while meeting operating standards defined by IEEE. One such mitigation strategy is the connection of zero-sequence ground sources. However, without implementing these strategies in a calculated approach, this leads to system concerns such as ground fault relaying desensitization. Similar concerns are inherent with techniques utilized to mitigate the various problems resulting from the integration of DERs.

1.2.3 Industry Requirements

As the implementation of DERs in distribution networks increases, more specifically IBRs, standards such as IEEE 1547 have been developed to standardize the response of IBRs developed by independent manufacturers. These standards establish minimum requirements for the system protection and control schemes and the responsibility of both the utility and the IPP. The IEEE Standard for Interconnection and Interoperability of Distributed Energy Resouces with Associated Electric Power Systems Interfaces, reference [6], is one such standard that defines the technical requirements of IBRs. This standard defines the reactive power injection requirements, IBR response to abnormal conditions and power quality requirements required to maintain distribution network reliability.

To contribute to the stability of the interconnected system, IEEE 1547-18 defines the required characteristics and capabilities for the various applications of DER.

Normal Performance Categories:

Category A - The minimum performance capabilities needed for Area EPS voltage regulation and are reasonably attainable by all state-of-the-art DER technologies. This level of performance is deemed adequate for applications where the DER penetration in the distribution system is lower, and where the DER power output is not subject to frequent large variations.

Category B - Covers all requirements within Category A and specifies supplemental capabilities needed to adequately integrate the DER in local Area EPS where the DER penetration is higher or where the DER power output is subject to frequent large variations.

Abnormal Performance Categories:

Category I - The minimal bulk power system (BPS) reliability needs and is reasonably attainable by all DER technologies that are in common usage today.

Category II - Performance covers all BPS reliability needs and coordinates with the existing BPS reliability standard, NERC PRC-024-2 [B26], developed to avoid adverse tripping of bulk system generators during system disturbances. Additional voltage ride-through capability is specified for DERs, beyond mandatory voltage ride-through defined by NERC PRC-024-2 [B26], to account for the potential for fault-induced delayed voltage recovery on the distribution system, due to distribution load characteristics.

Category III - Provides the highest disturbance ride-through capabilities, intended to address integration issues such as power quality and system overloads caused by DER tripping in local Area EPS with very high DER penetration levels. This category also provides increased bulk power system security by further reducing the potential loss of DER during bulk system events. These requirements are based on the California Rule 21 [B4] Smart Inverter requirements.

As implied by these performance categories, as the penetration of DERs increases, the system requirements for voltage regulation and power quality during normal operation increase and the burden abnormal conditions have on the resources is also exacerbated.

1.3 Research Objectives

The increasing deployment of DERs is resulting in a transformation of traditional radial network topologies into mesh network configurations. These DERs, often IBRs relying on power inverters to connect to the distribution grid, pose new challenges to system protection and reliability due to their unique physics and intermittency. IBRs exhibit low-magnitude short-circuit currents, unlike synchronous generators, making traditional overcurrent protection methods less effective. Therefore, the need for new protection methods, especially utilizing machine learning algorithms, becomes essential to address system anomalies and ensure grid stability.

Deep learning models, particularly CNNs, have recently been deployed in research and have proven effective in analyzing time-series data; recognizing anomalies in various domains. By harnessing the intrinsic capabilities of these ML structures, these models can continuously monitor electrical distribution system health, including fault occurrences, and react promptly to grid abnormalities. Implementing such algorithms into existing protection schemes has the potential to improve the reliability and effectiveness of overcurrent and overvoltage protection, critical for accommodating the inconsistencies among distributed IBRs and other types of DERs.

The development of a convolutional neural network model specifically tailored for power system protection will also contribute to reducing the computational resources required for anomaly recognition, making it more feasible to incorporate deep learning algorithms into real-world power distribution systems.

In summary, this research aims to contribute to the protection of diverse power systems in the context of increasing DER integration by developing and applying an optimized deep learning algorithm, particularly a CNN model, capable of efficiently mitigating the effects of harmful system anomalies. The particular research objectives of this thesis are as follows:

1. Review existing Convolutional Neural Network (CNN) structures and their suitability for time-series data diagnostics.

2. Develop an optimized CNN model structure tailored for effective time-series data diagnostics in the context of power system anomalies, considering the modified power system topology and dynamic characteristics of IBRs during transient conditions.

3. Evaluate the effectiveness of optimized CNN model structures compared to common machine learning models for enhancing power system protection schemes, with a focus on fault recognition.

1.4 Related Work

Over the years, there has been a growing interest in applying machine learning techniques to analyze electrical signal time series data sets. The earliest works explored the use of model architectures such as Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), and Learning Vector Quantization (LVQ) for this purpose. However, models often struggled with hidden feature extraction, necessitating expert/manual feature extraction techniques like Fast Fourier Transform (FFT) to assist the models.

Several studies demonstrated the potential of ANN, SVM, and LVQ models for fault diagnosis and protection in power systems. ANN-based schemes were proposed for fault location in transmission lines [9], and intelligent backup decisions in HVDC systems [10]. SVM combined with wavelet analysis was applied to diagnose faults in three-phase PWM inverters [11]. LVQ structures have shown advantages over SVM models in similar applications [12], references [13] and [14] demonstrate these benefits for use in fault diagnosis in three-phase induction motors and power transformers.

In works utilizing RNN model architectures such as LSTM and GRU, researchers have demonstrated their suitability for time series data analysis. RNNs have the ability to capture temporal dependencies making them well-suited for tasks such as fault detection, load forecasting, and system modelling. By integrating RNNs into online prediction models, they have demonstrated the ability to provide insights into system stability during both steady-state and transient conditions.

One of the earliest works utilizing an LSTM, reference [15], addressed the challenge of learning to store information over extended time intervals using recurrent backpropagation. As research in RNNs progressed, researchers explored more sophisticated units with gating mechanisms. Reference [16] compared LSTM and the recently proposed Gated Recurrent Unit (GRU) on music and speech signal modelling tasks. Other research, such as reference [17] explored the application of recurrent neural networks to model reduction with memory effect in power system equations. The proposed approach, inspired by the Mori-Zwanzig theory, demonstrated good performance in short-term prediction and long-term statistical properties. Both LSTM and GRU units outperformed traditional models utilizing tanh activation functions, showcasing the benefits of gating mechanisms in capturing temporal dependencies. In other research, reference [18] introduced the RNN Encoder-Decoder model, consisting of two RNNs for sequence encoding and decoding. The proposed model improved the performance of statistical machine translation systems by learning meaningful representations of linguistic phrases.

In recent years, further improvements have been made to RNN models for specific power system analysis tasks. Reference [19] introduced novel Deep Recurrent Neural

Network (DRNN) models employing LSTM for fault detection, classification, and location prediction in large-scale power systems. The use of sequential deep learning allowed the models to make reliable decisions based on PMU data. Other research, reference [20], proposed an LSTM-based framework for short-term load forecasting for individual residential energy consumers, demonstrating LSTMs' ability to outperform other algorithms, highlighting its potential for future grid planning and operation.

Although RNNs structure provides inherent benefits for analyzing time series data, the utilization of Convolutional Neural Networks (CNNs) has also gained significant attention in power system diagnostics [21]. The traditional use of CNN models deploys deep structures that allow the recognition of minute features for classifying inputs. However, these deep models are not suitable for applications requiring fast inference times due to the computational burden of the model. With that said, shallow models often struggle with hidden feature extraction. Therefore, researchers have incorporated the implementation of expert/manual feature extraction techniques like Fast Fourier Transform (FFT) to assist these models and effectively reduce their computational burden. Recent works have demonstrated the effectiveness of such CNNs in fault diagnosis and classification utilized for system-level [22] or component-specific application [2] [23] when utilizing application-specific optimizations.

With the appropriate data processing, CNNs have the ability to efficiently process multivariate time series data making them well-suited for power system applications where numerous signals need simultaneous analysis. For example, reference [24] utilizes a stacked auto-encoder to extract features during the data processing stage. Similarly, reference [25] explores domain adaptation techniques using deep models for fault diagnosis, focusing on adapting CNN-based fault diagnosis models to new and diverse datasets, improving the generalizability of the models. Furthermore, advancements in CNN model architectures, such as utilizing wide-area kernels [23] and product quantization [2], have contributed to improved accuracy and reduced model loss, ensuring reliable fault recognition and diagnosis. These methods increase the prediction accuracy and reduce the high computational burden of training that is characteristic of CNN networks.

The literature demonstrates a steady progression in the application of machine learning techniques for time series data diagnostics in power systems. From the early works utilizing shallow models to the recent focus on RNNs and the significant benefits of CNN models, these works highlight the potential of deep learning in enhancing power system reliability and fault detection capabilities. Although the use of deep learning models, particularly CNNs, has shown promising results, certain challenges and concerns remain. Real-time application of these models in power systems requires careful consideration due to the high computational burden involved, which may hinder immediate responses to critical faults.

1.5 Thesis Organization

The thesis is organized into six distinct sections, each addressing a specific aspect of the research on the application of machine learning models for power system protection on distribution networks with high penetrations of IBRs.

Chapter 1 provides an overview of the research, focusing on the impact of high penetrations of IBRs on electrical distribution networks. It explores the fundamental differences between traditional synchronous generators and IBRs concerning the control and transient characteristics of each resource. The introduction sets the context for the research, highlighting the challenges faced when integrating IBRs and the need for novel protection methods supplemented by machine learning algorithms.

Chapter 2 is a comprehensive review of various machine learning model structures. The discussion covers Support Vector Machines (SVM), Learning Vector Quantization (LVQ), and Recurrent Neural Networks (RNNs). However, the primary focus is on Convolutional Neural Network (CNN) model structures. The chapter delves into the various layers and parameters utilized in CNNs, providing a basis for selecting the most suitable model for time-series data diagnostics in the subsequent chapters.

Chapter 3 details the electromagnetic transient modelling and analysis process used in the research. It describes the utilization of the IEEE-34 bus distribution network as a test system, providing its structural and parameter definitions. Additionally, the chapter breaks down the IBR/solar plant model used in the simulations and explains the configuration and iterations carried out using the PSCAD simulation tool. The results obtained from the simulations are also presented and analyzed.

Chapter 4 is dedicated to the development of the CNN model for time-series data feature extraction. It outlines the specific requirements for the CNN model, including the input data structure and preprocessing steps. The chapter delves into the CNN model's architecture, detailing the various layers and hyperparameters used for optimization to ensure its suitability for effectively diagnosing power system anomalies.

Chapter 5 focuses on testing the machine learning models, particularly the CNN model, and presenting the results. It discusses the different test structures used to evaluate the model's performance. The chapter explores the optimization procedures involving hyperparameter tuning and layer modifications to enhance the model's effectiveness. The model's performance is thoroughly reviewed and compared to other machine learning model structures, such as SVM and RNNs.

In the final chapter, Chapter 6, the thesis draws conclusions based on the research findings and the performance of the CNN model for time-series data diagnostics. It highlights the contributions of the research in addressing the challenges posed by high-penetrations of IBRs in electrical distribution networks and the potential benefits of integrating deep learning algorithms into power system protection. The chapter concludes with suggestions for future work, identifying areas for further investigation and potential improvements to the developed model and methodologies.

Chapter 2

Machine Learning Models

Although IBR transient characteristics are atypical when compared to conventional synchronous generation, their respective response to various system anomalies is consistent and can be recognized with tuned protection equipment. The tuning required may contradict typical protection scheme parameters, leading to miscoordination and nuisance tripping. Not all methods can be consistently relied on to adequately protect heavily integrated IBR systems due to newly introduced overvoltage concerns and inconsistent fault current characteristics.

Due to the rapid development of machine learning research, the concepts are widely used in power systems. These data-driven prediction methods treat the evaluation of transient power system data as a pattern classification problem. The health of the system can be generalized as a binary classification - zero (0) meaning healthy system data and one (1) meaning undesirable system status. The structure of pattern classification problems includes a data pre-processing layer, feature selection layer, offline training and online rule extraction [26].

The main advantage of machine-learning methods is that these models do not require a predefined threshold and set points to adequately identify system anomalies. Artificial neural networks (ANN), support vector machine (SVM), learning vector quantization (LVQ), k-nearest neighbour (KNN) and recurrent neural networks (RNN) are several popular data-driven methods that have been utilized in power system applications. These models have shallow architectures that are traditionally limited to one hidden layer.

2.1 Learning Vector Quantization (LVQ)

Learning Vector Quantization (LVQ), is an ANN structure and a supervised vector quantization technique used to classify input vectors [12]. LVQs are a competitive learning model that is well-suited for discrete classifications. These models approximate the class distributions of input vectors by utilizing trained prototypes. These prototypes are trained through unsupervised vector quantization that produces a set of representative vectors used to classify input data.

Basic LVQs assign one prototype for each number of classes in a problem. During training, the Euclidean distance between data points and each of the initialized prototypes is calculated to identify the prototype with the smallest distance. If the closest prototype belongs to the same class as the input data point, the prototype is adjusted, moving it closer to the data point. If the closest prototype belongs to a different class than the data point, the prototype is moved away from the input data point.

The architecture of a basic LVQ network is shown in the below figure. LVQ structures consist of an input layer, a Kohonen layer and an output layer [13]. In the input layer, neurons are connected to each neuron on the subsequent Kohonen layer through a weighted filter. The output layer neurons are then connected to a specific group of Kohonen layer neurons and used to classify the input vector.

The weights of the filters between the Kohonen layer and the output neurons are either zero (0) or one (1) - representing the Kohonen layer neurons that affect the specific output layer neurons. As described in references [27] and [14], a basic LVQ network can be represented by the below equations.

Continuous input vectors:

$$X = (x_1, x_2, \cdots, x_M) \tag{2.1}$$

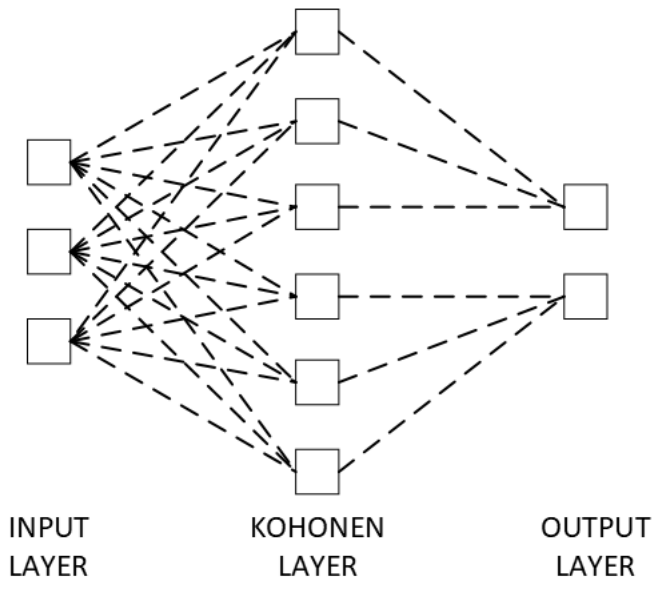


Figure 2.1: LVQ Basic Architecture

Connection weights vectors between the input layer and the Kohonen layer:

$$W^1 = (w_1^1, w_2^1, \cdots, w_p^1) w_i^1 = (w_{i1}^1, w_{i2}^1, \cdots, w_{iM}^1) \tag{2.2}$$

where $i = 1, 2, \cdots, p$.

Connection weights vectors between the Kohonen layer and the output layer:

$$W^2 = (w_1^2, w_2^2, \cdots, w_p^2) w_i^2 = (w_{i1}^2, w_{i2}^2, \cdots, w_{iM}^2) \tag{2.3}$$

where $k = 1, 2, \cdots, N$, and

$$W_{kr}^2 = \begin{cases} 1 & r \in k \\ 0 & r \notin k \end{cases} \tag{2.4}$$

Suppose the training mode results as follows:

$$\{x_1, t_1\}, \{x_2, t_2\}, \cdots, \{x_Q, t_Q\} \tag{2.5}$$

where $t_j (j = 1, 2, \cdots, Q)$

The hidden layer is calculated as follows:

$$V = W^1 X \tag{2.6}$$

Then the output vector is:

$$T = W^2 V \tag{2.7}$$

To train the network, W^1 can be adjusted as follows:

For every input vector, the network will give a classification result. If the result of the classification is correct, the connection weights values can be corrected by 2.8.

$$w_i^1(t+1) = w_i^1(t) + \alpha(t)(x(t) - w_i^1(t)) \tag{2.8}$$

If the result of the classification is false, the connection weights values can be corrected by 2.9.

$$w_i^1(t+1) = w_i^1(t) - \alpha(t)(x(t) - w_i^1(t)) \tag{2.9}$$

Where $x(t)$ is the input, α is the scalar gain, $\alpha \in (0, 1)$, which is decreasing in time and $w_i^1(t)$ is the connection weights value of the wth neuron at t time.

The advantage of LVQ is that it creates prototypes that are easy to interpret and that they can be applied to multi-class classification in a natural way [28].

2.2 Support Vector Machine (SVM)

Similar to LVQ models, SVMs utilize prototypes denoted as support vectors. This also utilizes supervised vector quantization. However, unlike LVQ, the support vectors of SVMs define a boundary between the input vectors of the various classes. Rather than attempt to define a prototype vector as close as possible to the input vectors of each class like in LVQ, SVMs attempt to define vectors with as large of a margin between the separate classes as possible

A support vector machine (SVM), also referred to as support vector classification (SVC), is a non-probabilistic binary classifier. SVMs produce a representation of the input vectors as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. The goal of the SVM classifier is to maximize the gap between groups of outputs. New input vectors are then mapped into that same space and predicted to belong to a category based on the region where they fall. Therefore, these algorithms are recognized as a pattern recognition method.

As SVMs were originally designed for binary classification [29], these algorithms have naturally been used in power system fault detection [11]. As mentioned, the objective of an SVM is to establish a separating boundary around support vectors used to identify the binary classification of the input data. These boundaries are separated from the respective input vector classes, by a maximum distance that is calculated using training data. The maximum separation is defined by solving the following equations, as defined by reference [11]. The Lagrange multipliers $\{\alpha_i\}_{i=1}^{N}$ are determined, by solving 2.10 with the defined constraints.

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{2.10}$$

where $W(\alpha)$ is subject to:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{2.11}$$

$$0 \leq \alpha_i \leq C, i = (1, 2, \cdots, N) \tag{2.12}$$

where α_i is a Lagrande multiplier, (x_i, y_i) is a training data set in which x_i is the input vectors and y_i is the output classification $\epsilon\{-1, 1\}$ and C is a constant to balance algorithm performance and the generalization.

$$K(x, x_i) = [(x \bullet x_i) + 1]^d \tag{2.13}$$

where $K(x, x_i)$ is a polynomial kernel function that performs an inner product of the input vector and support vector that performs the non-linear mapping into the feature space and d defines the order of the polynomial that controls the complexity of the classification boundary [30].

and a separating function can be defined as:

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) - b\right) \tag{2.14}$$

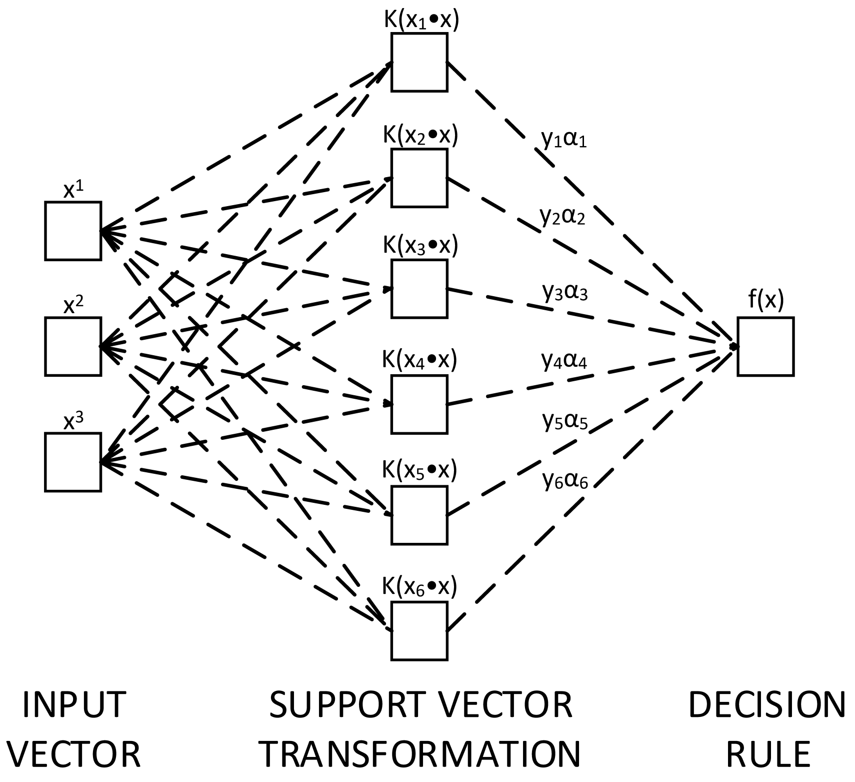


Figure 2.2: SVM Basic Architecture

where N is the number of support vectors x_i, x is the input vector and b is a bias term.

By solving the decision function $f(x)$, the SVM predicts the class label of the input vector x based on the output sign. If the value of $f(x)$ is positive, the predicted class label is true, and if the value of $f(x)$ is negative, the predicted class label is false.

2.3 Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is a neural network with recurrent connections that effectively handles time series data. These networks utilize recurrent connections to link values in a neural network to themselves. These links create a dependency on the respective values from the previous time step, creating a memory of the prior input, and influencing the output from the current input. RNNs also share weight parameters between the individual model layers before they are adjusted in the backpropagation stage.

Backpropagation is an algorithm utilized by RNNs and other neural networks. This stage updates the randomly initialized parameters/weights based on the output error through a technique called gradient descent [31].

A one-layer RNN with an input time series dataset, x_t, can be represented using equations 2.15 and 2.16, where h_t are the hidden layer values and y_t are the output values at time step t [17]. Note, multi-layer RNNs are created by feeding the output of y_t into another RNN.

$$h_t = \alpha(Wh_{t-1} + Ux_t) \tag{2.15}$$

$$y_t = \alpha(Vh_t) \tag{2.16}$$

Where U and W are the weight metrices, V is the projection matrix, and α are the respective non-linear activation functions.

RNNs utilize the same group parameters for all time steps. This allows RNNs to learn information in a long time series without introducing a significant number of parameters.

Training RNNs is similar to training regular feed-forward neural networks such as CNNs. However, RNNs consider the gradient through the time direction. For example, the gradient of y_t with respect to h_s, where $s < t$. These networks are trained utilizing the back-propagation through time (BPTT) method. In this back-propagation method, as the model is trained by calculating the errors from the output layer to the input layer, the errors are summed at each time step. This varies from traditional back-propagation, where the errors are not summed, as parameters are not traditionally shared across the model layers.

These networks are susceptible to concerns with exploding gradients and vanishing gradients. Gradients are defined as the slope of the loss function along the error curve produced when training the model. Vanishing gradient occurs when the gradient is too small and continues to decrease through training, causing the weight parameters to near zero, leading them to have an insignificant effect on the layer output [20]. Exploding gradient refers to the opposite effect, where weight parameters exponentially increase.

2.3.1 Long Short-Term Memory (LSTM) Network

To solve the gradient problems, Long Short-Term Memory (LSTM) RNNs were developed [15]. LSTM networks group nodes into a cell containing four gates that produce two outputs - a hidden vector, h, and a memory vector, m. The four gates include an input gate, g^u, output gate, g^o, forget gate, g^f and a cell state gate, g^c. Each gate is defined by equations 2.17 through 2.20 [19] [32].

$$g^u = \sigma(W^u h_{t-1} + U^u x_t) \tag{2.17}$$

$$g^o = \sigma(W^o h_{t-1} + U^o x_t) \tag{2.18}$$

$$g^f = \phi(W^f h_{t-1} + U^f x_t) \tag{2.19}$$

$$g^c = \sigma(W^c h_{t-1} + U^c x_t) \tag{2.20}$$

Where W and U are the recurrent weight matrices and the projection matrices, ϕ represents the tanh activation function and σ represents the sigmoid activation function.

The resultant output vectors can be described as:

$$m_t = g^f \odot m_{t-1} + g^u \odot g^c \tag{2.21}$$

$$h_t = \phi(g^o \odot m_t) \tag{2.22}$$

Where $\odot$ is the element-wise multiplication of the respective matrices.

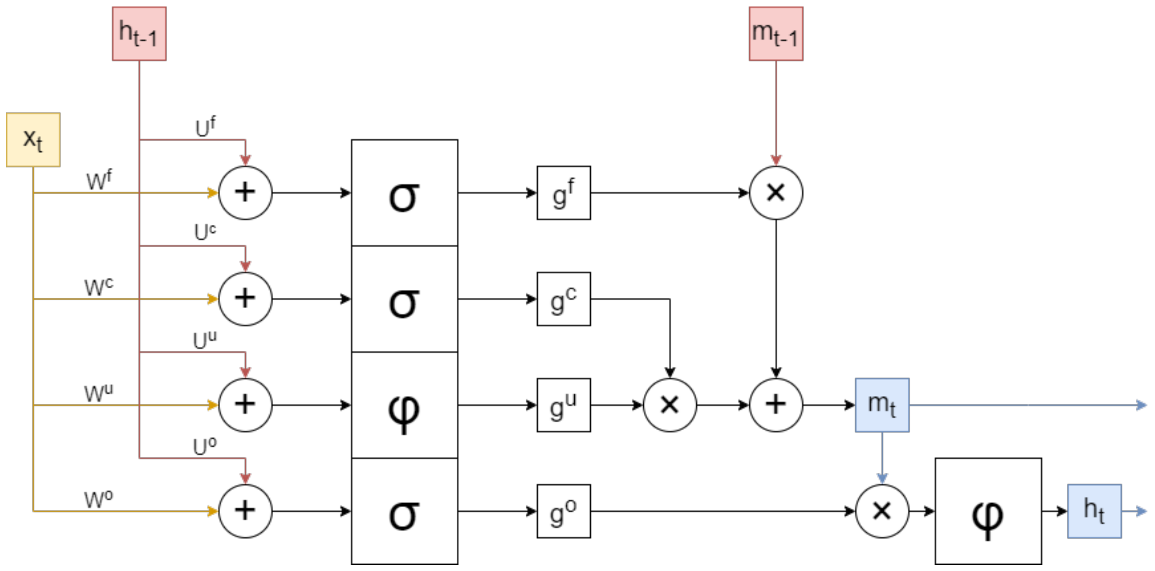


Figure 2.3: LSTM Block Structure

2.3.2 Gated Recurrent Unit (GRU) Network

Similar to the LSTM model, a Gated Recurrent Unit (GRU) model was later proposed by Cho [18] to solve the vanishing gradient problem, utilizing a reduced gating structure and a single output vector, h_t. The GRU cell is structured with two gates - an update gate, z_t, and a reset gate, r_t defined as equations 2.23 and 2.24. The resultant output vector, h_t, is then defined as equation 2.25 [32]. Although the GRU network utilizes a simplified LSTM structure, certain applications show improved results when utilizing a GRU model [16].

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{2.23}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{2.24}$$

Where x_t is the input vector at time step t, W are the trainable weight matrices for the update and reset gates, and U are the respective trainable recurrent weight matrices.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \phi(W_h x_t + U_h(r_t \odot h_{t-1})) \tag{2.25}$$

Where $\odot$ is the element-wise multiplication of the respective matrices.

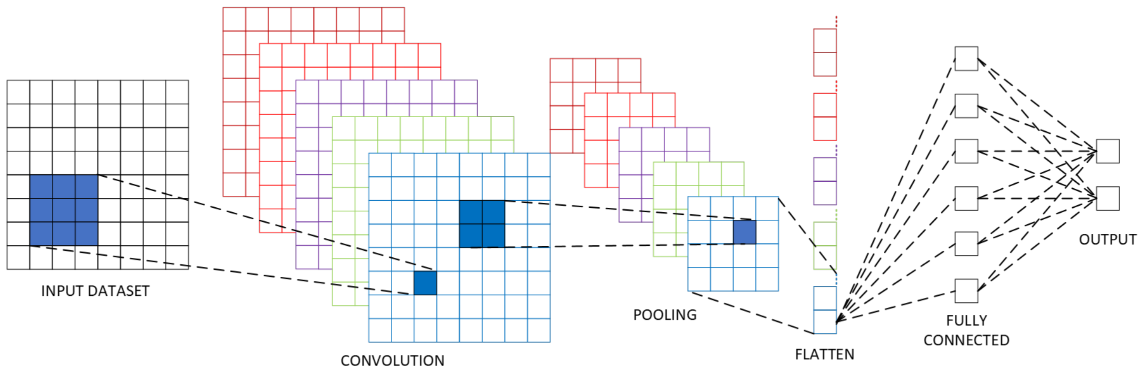


Figure 2.4: CNN Basic Architecture

2.4 Convolutional Neural Network (CNN)

Deep networks such as convolutional neural network (CNN) models typically have several hidden layers of various types. Deep networks can approximate the class of compositional functions with the same accuracy as shallow networks but with an exponentially lower number of training parameters as well as VC-dimension (the number of training examples that are needed in order to train, compute or approximate a specific target function) [33].

A CNN is a multi-stage neural network which consists of filter stages and one classification stage [34]. The filter stages are used to extract features from the inputs, which typically contain four types of layers - the convolution layer, batch normalization layer, activation layer and the pooling layer. The classification stage is a multi-layer prediction composed of one or more fully-connected layers.

The CNN training process is composed of two processes - forward propagation and backward propagation. In forward propagation, the model receives the input data, processes the information and generates an output. In backward propagation, the model calculates the error from the previous forward propagation and updates the parameters of the network.

One advantage of convolutional networks is the ease with which they can be implemented in hardware and require significantly less data to train an accurate and reliable model. However, these networks can be prone to several problems that will reduce efficacy if they are not structured appropriately.

Vanishing and Exploding Gradient

Vanishing and exploding gradient problems are issues that can arise during the training of neural networks such as CNNs. Both the vanishing and exploding gradient hinder the training of CNNs by impeding the convergence of the network's weights.

Vanishing gradient occurs when the gradients calculated during the backpropagation stage diminish rapidly as they propagate from the output layers to the input layers. This significantly reduces the error correction in the weights of the earlier layers, impeding their ability to effectively contribute to the model output. This reduces the network's ability to learn complex patterns as the lower layers are unable to extract meaningful features for the input data. This problem is prevalent in activation functions that have near-zero derivatives such as sigmoid and hyperbolic tangent functions.

Exploding gradient is the opposite of the vanishing gradient problem. It occurs when the gradients calculated during backpropagation become extremely large as they propagate through the network. This leads to large weight updates and instability during training, making it difficult for the network to converge to an optimal solution. Models are more susceptible to exploding gradient when activation functions with steep derivatives, such as ReLU, are utilized.

These concerns are mitigated through the application of activation functions with suitable derivatives and through normalization techniques that assist in stabilizing training while expediting model convergence [35].

Internal Covariate Shift

Internal covariate shift refers to the changing distribution of each respective layer's inputs during the learning process due to the varying outputs of the previous layer.

As the parameters of the preceding layers change through backpropagation there is a shift in the weights for inputs to subsequent layers. Model's susceptible to internal covariate shifts require reduced gradients during training, reducing the learning speed of the model [36].

As the distribution of weights and resulting inputs to each layer change, the gradients calculated based on the current distribution may not be suitable for updating the parameters in an optimal manner. This can result in slower convergence, vanishing gradients, or difficulty in training deep networks.

For these models, careful parameter initialization is required as it can also lead to the saturation of activation functions [37]. When the input distribution to a layer becomes heavily weighted, sigmoid and hyperbolic tangent activation functions can saturate. This saturation causes the activation functions to push the respective outputs toward the extremes of the function's range, where the gradient becomes close to zero, also known as the diminishing gradient phenomenon.

Internal Covariate Shift highlights the need for normalization techniques in CNNs. Normalization will stabilize the distribution of inputs to each layer by normalizing them across multiple sets of data inputs.

Overfitting

Overfitting refers to the phenomenon where the network effectively memorizes the data utilized for training, including the noise or irrelevant patterns in the dataset, rather than capturing the underlying general patterns. This is caused by factors such as insufficient data, excessive model complexity and ineffective regularization techniques intended to prevent the network from overcompensating for the training data set [38].

When the available training data is limited, the network may not encounter enough diverse examples to generalize the layer weights to adequately identify the underlying patterns. This concern is exacerbated when the respective model has a large number of layers or neurons. Complex models with an increased number of parameters enable them to closely fit the training data.

Techniques utilized for mitigating overfitting issues include expanding the size of the training data set with diverse, representative data. This can be accommodated through data augmentation techniques used to artificially increase the size and diversity of the training data set by applying random transformations to the existing data. Another common mitigating measure is the implementation of an early-stopping algorithm. Training a CNN involves iterating over multiple epochs. Early stopping is a technique where the training process is halted before convergence, based on a predefined metric.

2.4.1 Convolution Layer

The convolution layer convolves the input vectors with filter kernals before going through an activation function to generate the output features. Each filter uses the same kernel to extract the local feature of the input vector. This is referred to as weight sharing. One filter corresponds to one frame in the next layer and the number of frames corresponds to the layer depth. The convolution process can be described as shown below.

$$y^{l(i,j)} = W_i^l \bullet X^{l(r^j)} = \sum_{j'=0}^{w} W_i^l(j')X^{l(j+j')} \tag{2.26}$$

where W_i^l is used to denote the weights of the i-th filter kernel in layer l, $X^{l(r^j)}$ denotes the j-th local region in the convolution layer l, w is the width of the kernel, $W_i^l(j')$ denotes the j-th weight of the kernel and the notation $\bullet$ denotes the dot product between the kernel and the local regions.

2.4.2 Batch Normalization

The batch normalization layer is a technique introduced in CNN models to reduce the shift of internal covariance, accelerate the training process and mitigate the susceptibility to vanishing and exploding gradient problems [39] [40]. In some applications, the introduction of batch normalization can make the use of dropout layers unnecessary [36].

Batch normalization will typically follow a convolution layer or fully-connected layer and ahead of the activation layer. For a q-dimension input to the l-th batch normalization layer, $y^l = (y^{l(1)}, \cdots, y^{l(q)})$. The output of the batch normalization layer can be described as shown below.

$$\hat{y}^{l(i,j)} = \frac{y^{l(i,j)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.27}$$

where $\mu_B = E[y^{l(i,j)}]$, $\sigma_B^2 = Var[y^{l(i,j)}]$ and ϵ is a numerical stability constant.

$$z^{l(i,j)} = W^{l(i)}\hat{y}^{l(i,j)} + \beta^{l(i)} \tag{2.28}$$

where $W^{l(i)}$ are the respective model weights and $\beta^{l(i)}$ are the learned biases of the respective neurons.

2.4.3 Activation Function

The activation function is placed at the end of the convolution and fully connected layers. The most common activation functions can be grouped as either a ridge, radial or fold function. Specific activation functions are selected depending on the desired output and model requirements. The three activation functions utilized in this work are the sigmoid, rectified linear unit and hyperbolic tangent functions as depicted in Figure 2.5.

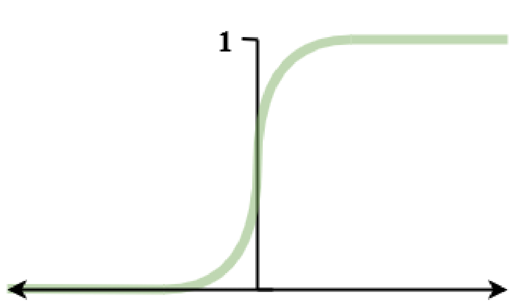
Sigmoid

The sigmoid activation function is typically used at the end of the model when the desired result is a binary output [41]. This function is described below by 2.29.

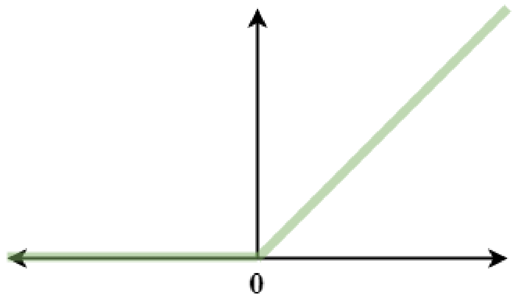
$$\alpha(z^{l(i,j)}) = Sigmoid(z^{l(i,j)}) = \frac{1}{1 + e^{-z^{l(i,j)}}} \tag{2.29}$$

Rectified Linear Unit (ReLU)

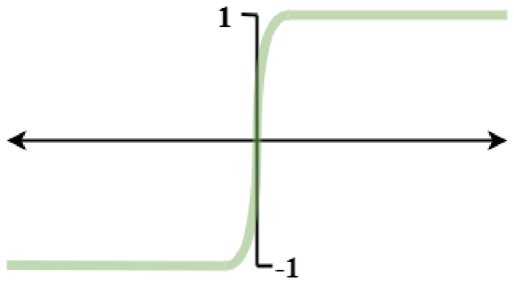
The ReLu activation function, described by 2.30, typically follows the convolution layer. This is a non-saturating function that does not suffer from the vanishing



(a) Sigmoid



(b) ReLU



(c) TanH

Figure 2.5: Activation Functions

gradient problem [42]. This function accelerates the convergence of neural networks during the back-propagation stage of model training.

$$\alpha(z^{l(i,j)}) = ReLu(z^{l(i,j)}) = max\{0, z^{l(i,j)}\} = \begin{cases} 0, z < 0 \\ z, z \geq 0 \end{cases} \tag{2.30}$$

Hyperbolic Tangent (TanH)

The hyperbolic tangent (tanh) function is defined as 2.31. This is a zero-centred function, symmetric around the origin, with outputs ranging from -1 to 1. Tanh is a preferred function when negative input values play a signification role in dataset classification [43].

$$\alpha(z^{l(i,j)}) = tanh(z^{l(i,j)}) = 2Sigmoid(2z^{l(i,j)}) - 1 = 1 - \frac{2}{e^{2z^{l(i,j)}} + 1} \tag{2.31}$$

2.4.4 Pooling Layer

Pooling layers are designed to reduce the dimensionality of the features, speed up the training and improve the robustness of the extracted features. This operation is a static attribute in the neural network, containing no parameters [23]. The operation will not change the depth of the input data. However, the dimension of each output is reduced. Pooling layers are typically added after a convolution layer in CNN architectures. Pooling is utilized as a down-sampling operation to reduce the spatial size of the features and parameters of the network. The most common pooling layer is the max-pooling layer. This layer performs the local max operation over the input features to reduce the parameters and obtain location-invariant features [34]. This transformation function can be described by 2.32.

$$p^{l(i,j)} = max_{(j-1)w+1 \geq t \geq jw}\{\alpha^{l(i,j)}\} \tag{2.32}$$

where $\alpha^{l(i,j)}$ denotes the value of the t-th neuron in the i-th frame of layer l, w is the width of the pooling region and $t \in [(j-1)w+1, jw]$.

2.4.5 Fully Connected Layer

In a fully connected layer, each neuron or node is connected to every neuron in the preceding layer, forming a dense set of connections. This layer is a traditional neural network that serves as a bridge between the preceding layers and the output layer, generating an output based on the features previously extracted from the preceding network layers.

Fully connected layers can only receive 1-D vectors. Therefore, prior to the fully connected layer, a flattening operation is typically required to convert the feature maps of the previous layers into an appropriate array.

This layer performs two operations on the incoming data - a linear transformation and a non-linear transformation (activation function). The linear transformation of the input data is carried out by computing a weighted sum of the inputs. The weighted sum is calculated by multiplying each input value with its corresponding weight and summing up the results. The linear transformation is then passed through an activation function, such as the sigmoid, hyperbolic tangent (tanh), or rectified linear unit (ReLU). This introduces non-linear elements, enabling the network to capture non-linear relationships between the inputs and the outputs.

A simple representation of these layers can be described by 2.33, utilizing the sigmoid activation function for demonstration purposes.

$$Output(X) = Sigmoid(W^T \bullet X + B) \tag{2.33}$$

As suggested, fully connected layers are introduced in the later stages of neural networks, where they can capture high-level features and learn complex representations. For this reason, these layers are also utilized as the output layer in neural network models for final derivations.

2.4.6 Flatten Layer

The flatten layer is used to transform multi-dimensional input data into a one-dimensional array. The flatten layer is typically applied after convolutional or recurrent layers, which produce output tensors with multiple dimensions.

The flatten layer does not introduce any trainable parameters. Rather, the flattening process preserves the batch dimension while merging all the other dimensions into a one-dimensional vector. Each element of the output vector corresponds to a specific feature or attribute of the input data. This allows the subsequent fully connected layers to process the data and learn complex relationships. The flatten layer is an important step in preparing the input for further processing and making predictions in the network.

2.4.7 Dropout Layer

Dropout is a technique utilized to improve model generalization by preventing overfitting, increase regularization by preventing the network from relying heavily on specific neurons and inject uncertainty into the input data to improve the model's ability to handle noisy inputs [34]. This is accomplished by randomly deactivating neurons and their connections during a model's training stage.

This technique can be added to many layers of a model, including the convolution layer. As convolution layers have significantly fewer parameters than a fully connected layer, it may seem unnecessary. However, dropout applied to a higher layer will have an effect on all subsequent layers [44].

Care must be taken when utilizing dropout and selecting the dropout rate as improper application can lead to suboptimal results. For example, applying dropout to recurrent connections in RNN networks can disrupt the temporal dependencies and reduce the model's ability to capture sequential information [45].

If $z^{(l)}$ denotes the vector of inputs into layer l, the feed-forward operation of a standard CNN can be described by 2.34. When a dropout layer has been introduced, the typical feed-forward operation becomes 2.35 [44].

$$z_i^{(l+1)} = W_i^{(l+1)} X^l + b_i^{(l+1)} \tag{2.34}$$

$$z_i^{(l+1)} = W_i^{(l+1)} \tilde{X}^l + b_i^{(l+1)} \tag{2.35}$$

$$\tilde{X}^l = r^{(l)} \odot X^l \tag{2.36}$$

$$r_j^{(l)} \sim Bernoulli(p) \tag{2.37}$$

Where W^l and b^l are the hidden layer weights and biases at layer l, X^l is the layer input vector, $\odot$ denotes an element-wise multiplication and $r^{(l)}$ is a vector of Bernoulli random binary variables with p designating the probability of each variable being 1.

By carrying out an element-wise multiplication of the Bernoulli vector r, as described in 2.36, a percentage of the input vector variables are set to 0, effectively removing that amount of random inputs from the model layer.

2.5 Summary

This chapter introduced various machine learning models that have been previously implemented for time-series data classification applications, including Learning

Vector Quantization (LVQ), Support Vector Machines (SVM), Recurrent Neural Networks (RNN) with a focus on Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), and Convolutional Neural Networks (CNN). This chapter lays the foundation for the subsequent experimentation and optimization of CNN models in the context of time-series data classification for power system protection and control.

The chapter began with an overview of LVQ, which is a supervised learning algorithm traditionally used for classification tasks. The general structure of LVQ involves training prototypes for each class and adjusting them to minimize the classification error. LVQ models are particularly suitable for tasks where interpretability and transparency are essential.

Next, SVM, a supervised learning algorithm for classification and regression tasks, was introduced. SVM aims to find the optimal hyperplane that best separates classes in a high-dimensional feature space. The general structure of SVM involves mapping the data into higher dimensions using a kernel function and maximizing the distance between the various data input types to achieve effective classification.

RNN models are also discussed, with a focus on LSTM and GRU, both of which are specialized RNN architectures designed to overcome the vanishing gradient problem in traditional RNNs. LSTM and GRU models introduce gating mechanisms to retain previous time-step information and utilizing it for future input characterization.

The central focus of the chapter was on CNNs, traditionally deep learning architectures widely used for image recognition. The purpose and utilization of each of the essential layers in CNNs were discussed, providing a preface for how CNNs ability to learn hierarchical representations have the potential to be well-suited for time-series data analysis and classification tasks.

Chapter 3

Electromagnetic Transient Modeling and Analysis

3.1 Electromagnetic Transient Simulations

Electromagnetic Transient (EMT) modelling is a crucial tool for studying the transient behaviour of electrical power systems, including the integration of renewable energy resources like photovoltaics [46]. EMT studies provide valuable insights into system characteristics and performance, allowing the optimization of control and protection strategies that is not possible with fundamental frequency positive sequence (RMS) models.

EMT models utilize computational techniques in the time-domain to analyze the dynamic behaviour of electrical power systems and devices. These simulations model the transient response of the system to sudden changes or disturbances, considering the time-varying behaviour of voltages, currents, and electromagnetic fields.

Renewable energy resources, such as PV, can significantly impact power system dynamics due to their intermittent nature and variability in inverter control philosophies. EMT simulations play a crucial role in understanding and mitigating the impact of PV and other IBR DG integration on the grid. For example, EMT simulations are used to investigate the behaviour of PV systems during faults, such as short circuits or other disturbances in the power system; they consider the response of supporting infrastructure on this distribution grid such as voltage regulators; and they are able to effectively model real-world fluctuations in power quality parameters due to harmonics.

Therefore, these simulations are employed in this research through Power Systems Computer Aided Design (PSCAD) software, to study the transient phenomena that occur during various events, such as load rejection, switching operations and faults while integrating photovoltaic resources. These simulations provide valuable insights into the magnitudes and phase angles of the system's electrical phasor quantities, allowing for the extraction of the data required for the assessment of power system characteristics by ML models in real time.

3.2 IEEE 34-Bus Feeder Model

An IEEE 34-bus test feeder model initially developed by PSCAD, was utilized for this research, as depicted in Figure 3.1. This model represents an actual radial distribution system in Arizona operating at a nominal voltage of 24.94 kV, which is a common configuration for medium-voltage distribution networks in North America. It includes various types of components such as step-down transformers, voltage regulators, shunt capacitors and diverse load characteristics from residential, commercial, and industrial consumers. This diversity allows for a realistic representation of the power consumption patterns observed in real distribution systems and the analysis of unbalanced conditions. The model consists of multiple feeders with different lengths, which introduces variations in the impedance and voltage drop along the distribution lines. This reflects the inherent characteristics of distribution systems, where line losses and voltage drops can vary due to varying line lengths.

The IEEE 34 Bus model is particularly suitable for modelling the integration of DERs, such as PV due to the feeder lengths and diversity in load and feeder characteristics. Its size makes it computationally efficient while still providing sufficient complexity to capture the dynamics of a distribution system [47]. This scalability makes it suitable for studying the impact of DER integration without excessive com-

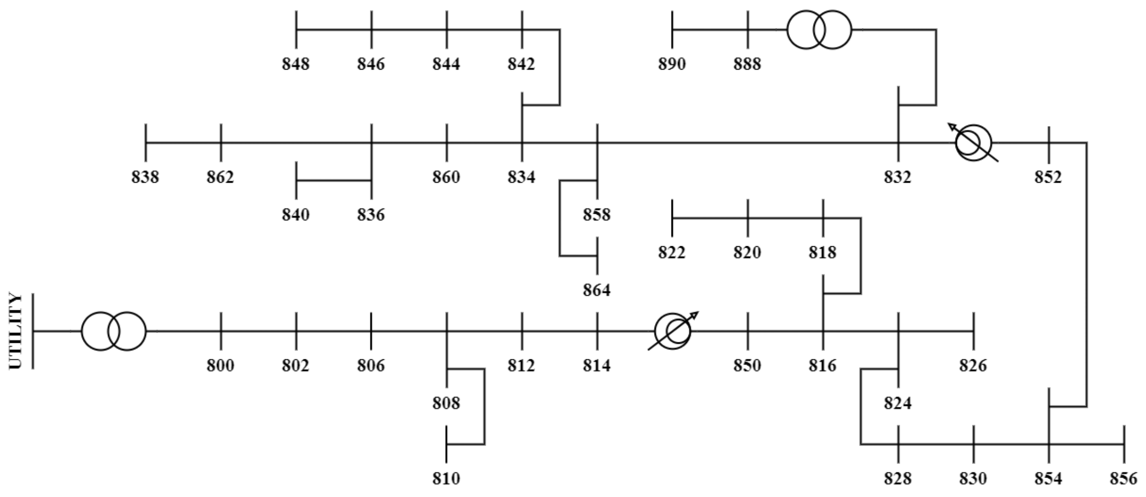


Figure 3.1: IEEE 34 Bus Feeder

putational burden. The incorporation of residential, commercial, and industrial loads reflects the diversity of electricity consumption in real systems. This provides insights into the interaction between PV resources and various load types.

3.2.1 Distribution Feeder Characteristics

The IEEE 34-bus distribution system PSCAD model includes five aluminum conductor steel reinforced (ACSR) conductor overhead line configurations, as described below.

Table 3.1: IEEE 34-Bus System Modelled Overhead Line Configurations

Identifier	**Phasing**	**Phase Conductor**	**Neutral Conductor**
300	BACN	1/0	1/0
301	BACN	#2 6/1	#2 6/1
302	AN	#4 6/1	#4 6/1
303	BN	#4 6/1	#4 6/1
304	BN	#2 6/1	#2 6/1

3.2.2 System Loads

The base IEEE 34-bus feeder model utilizes two categories of system loads described as spot loads and distributed loads. Spot loads are individual loads connected at a specific location in the distribution system. These represent localized points of

Table 3.2: IEEE 34-Bus System Modelled Line Segments

Node A	Node B	Length [ft]	Config. Type	Node A	Node B	Length [ft]	Config. Type
800	**802**	2580	300	**834**	**842**	280	301
802	**806**	1730	300	**836**	**840**	860	301
806	**808**	32230	300	**836**	**862**	280	301
808	**810**	5804	303	**842**	**844**	1350	301
808	**812**	37500	300	**844**	**846**	3640	301
812	**814**	29730	300	**846**	**848**	530	301
814	**850**	10	301	**850**	**816**	310	301
816	**818**	1710	302	**852**	**832**	10	301
816	**824**	10210	301	**854**	**856**	23330	303
818	**820**	48150	302	**854**	**852**	36830	301
820	**822**	13740	302	**858**	**864**	1620	302
824	**826**	3030	303	**858**	**834**	5830	301
824	**828**	840	301	**860**	**836**	2680	301
828	**830**	20440	301	**862**	**838**	4860	304
830	**854**	520	301	**888**	**890**	10560	300
832	**858**	4900	301	**860**	**836**	2680	301
832	**888**	0	XFM-1	**862**	**838**	4860	304
834	**860**	2020	301	**888**	**890**	10560	300

demand, typically associated with commercial and industrial facilities. Distributed loads refer to loads that are distributed across a segment of a distribution feeder. These loads are comprised of multiple smaller loads that do not have a significant individual effect on the system but contribute to the overall system demand.

Each load is further defined by its load type. Although these specifications are dependent on the available information of the actual system, these details help to define the load's specific characteristics, increasing the accuracy of the model's transient characteristics. Wye (Y) connected loads connect each phase to a common neutral and are often grounded at the neutral. In a delta (D) connection, each phase is connected in a closed loop without a neutral connection.

3.2.3 Shunt Capacitors

Shunt capacitors provide reactive power compensation and voltage regulation for distribution feeders, contributing to improved power factor and voltage stabil-

Table 3.3: IEEE 34-Bus System Modelled Spot Loads

Node	Load Type	Phase-A [kW]	Phase-A [kVAR]	Phase-B [kW]	Phase-B [kVAR]	Phase-C [kW]	Phase-C [kVAR]
860	Y-PQ	20	16	20	16	20	16
840	Y-I	9	7	9	7	9	7
844	Y-Z	135	105	135	105	135	105
848	D-PQ	20	16	20	16	20	16
890	D-I	150	75	150	75	150	75
830	D-Z	10	5	10	5	25	10

Table 3.4: IEEE 34-Bus System Modelled Distributed Loads

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

ity. When significant system demand is comprised of inductive loads, connecting capacitors in parallel generates reactive power, improving the system's power factor. This compensation reduces the burden on the power supply by increasing efficiency and allowing for better utilization of the power supply's capacity. Shunt capacitors assist in voltage regulation by injecting reactive power into the system as the load on the feeder varies. By compensating for reactive power losses and voltage drops, capacitors help stabilize and regulate the voltage profile of the system.

As reactive loads, shunt capacitors can have a significant impact on the system's transient characteristics and response to various power system events. For instance, during load rejection events, when a significant load is suddenly disconnected from the system, shunt capacitors can help mitigate voltage sags and stabilize the voltage profile. As the load is removed, the capacitors continue to inject reactive power, compensating for the sudden drop in demand and helping to maintain voltage stability. During a fault, the capacitors discharge their stored potential energy, contributing to a higher fault current.

Table 3.5: IEEE 34-Bus System Modelled Shunt Capacitors

Node	Phase-A [kVAR]	Phase-B [kVAR]	Phase-C [kVAR]
844	100	100	100
848	150	150	150

3.2.4 Regulator Data

The IEEE 34-bus feeder model includes two voltage regulators, as defined below. Like shunt capacitors, voltage regulators are installed at specific points in the distribution feeder to maintain system voltages within a defined range as significant loads and long distribution lines cause system voltages to drop.

Table 3.6: IEEE 34-Bus System Modelled Regulators

Regulator	Node	Connection	Bandwidth [V]	PT Ratio	Phase R Setting	Phase X Setting	Setpoint [V]
1	814	3PH LG	2	120	2.7	1.6	120
2	852	3PH LG	2	120	2.5	1.5	124

3.3 Distributed Inverter-Based Resource

A solar farm model developed by PSCAD is introduced into the IEEE 34-bus system at node 848. Node 848 was selected as the distribution grid point of integration as it would be a strong integration point candidate in the actual system to assist with voltage fluctuations and power quality concerns at the end of the distribution line. This voltage support improves the voltage profile, enhances voltage regulation, and minimizes voltage deviations experienced by connected loads toward the end of the feeder. Although not considered in this research, placing the DER at node 848 could also increase system resilience to unplanned system events by providing localized generation capacity, reducing reliance on the primary generation sources. By having a distributed energy resource local to the end-of-line loads, the system could potentially ride-through disruptions or outages in the transmission network and continue to supply power to nearby critical loads.

Integration at this location will also best demonstrate the transient characteristics of DER. Modelling the DER at node 848 best represents the transient characteristics of the DER. As the grid is weakest at this point in the feeder, the transient behaviour and response of the DER during unplanned events will have a greater effect on the surrounding feeder. Therefore, modelling the DER at node 848 allows for an accentuated analysis of the DER's dynamic performance, response to disturbances and interaction with the grid.

3.3.1 PV Model Definition

The PV model developed by PSCAD is comprised of 5 major components including the PV array, the power plant controller, the boost converter and the DC-AC inverter. A scaling component is also introduced to augment the number of parallel DC-AC inverters.

The PV array is the solar farm generation source, consisting of multiple photovoltaic modules or solar panels. These modules absorb sunlight (irradiation) and convert it into direct current (DC) electricity through the photovoltaic effect. The power output of the PV array is a function of two parameters, the irradiation and the ambient temperature. Although not considered in this research, these parameters may be adjusted to vary the model output and characteristics for specific irradiation and temperature conditions. In this thesis, the PV array irradiation and temperature parameters are set to 1000 W/m^2 and 28°C respectively to maximize the power output of the PV array.

The power plant controller is implemented in a basic form to monitor the output of the solar farm and the distribution network conditions at the point of common coupling (PCC). Based on the measured voltage, active power and reactive power output and the mode of operation it adjusts the active and reactive power references for the inverters in the solar farm. The power plant controller model can also detect low and high-voltage ride-through condition setpoints, simulating the power stability requirements of IEEE Std 1547-2018.

The power plant controller model's three modes of operation include POC voltage mode, fixed reactive power and power factor control. In POC voltage mode, the PV inverter controller adjusts the active power output of the PV system based on a predefined DC voltage limit or curve. The voltage reference curve or maximum power point tracking (MPPT) curve defines the desired relationship between the grid voltage and the active power output. The controller continuously monitors the grid voltage and compares it to the voltage MPPT curve. Based on this comparison, the MPPT algorithm varies the DC input voltage of the inverter to find the optimal point on the current-voltage (I-V) curve of the PV array. This ensures that the PV system operates at its maximum power efficiency. In fixed reactive power mode, the PV inverter controller adjusts the reactive power output of the PV system to maintain a predetermined level of reactive power injection. The controller continuously monitors the grid's reactive power demand and adjusts the inverter's output to supply the fixed level of reactive power required. In power factor control mode, also referred to as volt/var control, the PV inverter controller adjusts the power factor of the PV system's output to achieve a desired value. This control strategy regulates the feeder voltage level and power factor by adjusting the reactive power output of the inverter.

The boost converter, also known as the DC-DC converter, modulates the PV controller output while operating in both MPPT and DC voltage control modes. This element converts the DC power generated by the PV array to a higher voltage DC level defined by the controller, enabling the optimization of the power transmission from the PV array to the DC-AC inverter.

The DC-AC Inverter is the main power electronic component, responsible for converting the DC power generated by the PV array into the AC voltage suitable for grid connection and distribution. The inverter performs the conversion through power electronic devices, such as insulated gate bipolar transistors (IGBTs) or silicon carbide (SiC) devices.

3.3.2 Step-Up Distribution Transformer

Systems integrating DERs require effective grounding to avoid severe overvoltage during ground fault conditions. Since inverter neutrals are typically designed for sensing purposes and not rated for carrying ground fault currents, utilities will require a separate zero-sequence source for PV systems to assist with ground fault mitigation required by IEEE Std 1547-2018 [8]. This zero-sequence source is typically introduced through a zig-zag grounding transformer or through a specific interconnection transformer grounding configuration such as wye grounded-delta. These transformers typically require a level of derating, introduced from normal distribution network imbalances. System zero-sequence return current on the feeder due to imbalances will result in a zero-sequence voltage imposed on the wye side of the transformer. On the delta side, this zero-sequence voltage will result in a circulating current through the delta loop that is a function of the transformer zero sequence impedance.

As the focus of this thesis is on the IBR transient characteristics and the ML models' ability to recognize these conditions, a YGyg interconnection transformer was utilized at the distribution network PCC to not restrict these attributes. In practice, a YGd grounding configuration would likely be introduced as a zero-sequence current source to limit overvoltage conditions. This transformer would also incorporate an impedance on the transformer neutral to balance the amount of zero-sequence current divided between the transformers and the overvoltage severity during ground faults, preventing ground fault relaying desensitization [5].

3.4 PSCAD Simulations

A minimum number of samples are required to prevent overtraining of the deep learning model. Overtraining occurs when the model becomes too familiarized with the training data and becomes specifically fitted for the training data set. As the model becomes more closely fitted for the training data, its performance with other data sets will begin to deteriorate. Overtraining can be expected when the training set size is smaller than ten times the number of connections of the network [48].

To retrieve large, diverse and accurate data sets to effectively train the ML model and prevent overfitting, simulations were performed under a full range of network parameters and operating conditions. Variables included PV DER capacity, the inverter control methodology and the distribution feeder system demand. Utilizing an iterative process carried out by a Python script, combinations of these variables are simulated and data is retrieved for 48 unique runtime events. These events, detailed in Table 3.10, demonstrate various system anomalies resulting in transient fluctuations in power system attributes.

Two specific PV inverter control scenarios, as seen in Table 3.7, were simulated to capture the variances in differing DERs controller responses to unplanned events. These specific control types were utilized due to their robust capabilities with the diverse combinations of the modelled system attributes and simulated conditions.

To simulate the effect the DER/load ratio has on the system's transient characteristics, the demand of spot load 844 is also varied between iterations. The load capacity is shown in Table 3.9. As the load size is increased, transients such as voltage

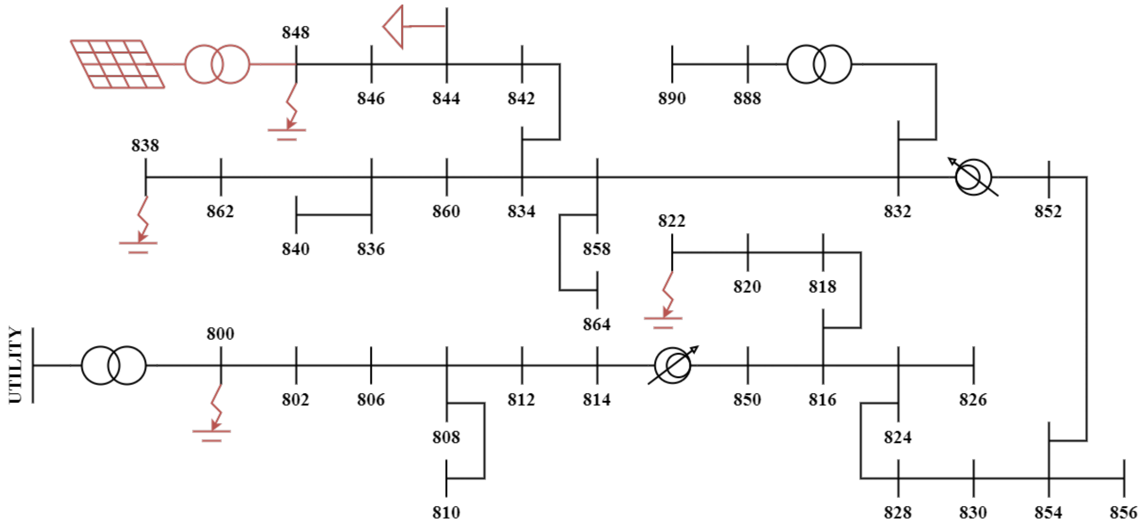


Figure 3.2: Modified IEEE 34 Bus Feeder

fluctuations to load rejection events are exacerbated, while OV transient conditions are mitigated during system islanding and ground fault events.

Lastly, the PV farm capacities listed in Table 3.8 are also cycled. The sizes of the solar farm are varied to simulate the variance of transient conditions between a strong and weak grid system [49].

Table 3.7: PSCAD Model Iterations - PV System Control Scenarios

PV Control Type	**Reactive Power Control**	**Active Power Control**
1	POC Voltage	DC Voltage
2	POC Voltage	MPPT

Table 3.8: PSCAD Model Iterations - PV Size Scenarios

PV Size	**Inverter Quantity**	**Inverter Capacity [kW]**	**Total Capacity [MW]**
1	2	250	0.5
2	4	250	1.0
3	8	250	2.0
4	12	250	3.0
5	20	250	5.0

Table 3.9: PSCAD Model Iterations - Load Demand Scenarios

Load 844 Demand	**Active Power [MW/Phase]**	**Reactive Power [MVAR/Phase]**	**Total Demand [MVA]**
1	0.135	0.105	0.513
2	0.270	0.210	1.026
3	0.540	0.420	2.052
4	0.810	0.630	3.078
5	1.350	1.050	5.131

Table 3.10: PSCAD Runtime Events

Event	**Time [s]**	**Type**	**Node**	**Action**	**Event**	**Time [s]**	**Type**	**Node**	**Action**
1	2.500	Load Energization	844	Close Breaker	**25**	15.583	AG Fault	848	Remove Fault
2	3.500	Utility Islanding	800	Open Breaker	**26**	16.583	BG Fault	848	Apply Fault
3	3.667	Utility Synchronization	800	Close Breaker	**27**	16.666	BG Fault	848	Remove Fault
4	4.667	AG Fault	800	Apply Fault	**28**	17.666	CB Fault	848	Apply Fault
5	4.750	AG Fault	800	Remove Fault	**29**	17.750	CB Fault	848	Remove Fault
6	5.750	BG Fault	800	Apply Fault	**30**	18.750	ABG Fault	848	Apply Fault
7	5.833	BG Fault	800	Remove Fault	**31**	18.833	ABG Fault	848	Remove Fault
8	6.833	CB Fault	800	Apply Fault	**32**	19.833	ACG Fault	848	Apply Fault
9	6.917	CB Fault	800	Remove Fault	**33**	19.916	ACG Fault	848	Remove Fault
10	7.917	ABG Fault	800	Apply Fault	**34**	20.916	BCG Fault	848	Apply Fault
11	8.000	ABG Fault	800	Remove Fault	**35**	21.000	BCG Fault	848	Remove Fault
12	9.000	ACG Fault	800	Apply Fault	**36**	22.000	ABCG Fault	848	Apply Fault
13	9.083	ACG Fault	800	Remove Fault	**37**	22.083	ABCG Fault	848	Remove Fault
14	10.083	BCG Fault	800	Apply Fault	**38**	23.083	AB Fault	848	Apply Fault
15	10.167	BCG Fault	800	Remove Fault	**39**	23.166	AB Fault	848	Remove Fault
16	11.167	ABCG Fault	800	Apply Fault	**40**	24.166	AC Fault	848	Apply Fault
17	11.250	ABCG Fault	800	Remove Fault	**41**	24.249	AC Fault	848	Remove Fault
18	12.250	AB Fault	800	Apply Fault	**42**	25.249	BC Fault	848	Apply Fault
19	12.333	AB Fault	800	Remove Fault	**43**	25.333	BC Fault	848	Remove Fault
20	13.333	ACG Fault	800	Apply Fault	**44**	26.333	AG Fault	838	Apply Fault
21	13.416	ACG Fault	800	Remove Fault	**45**	26.416	AG Fault	838	Remove Fault
22	14.416	BCG Fault	800	Apply Fault	**46**	27.416	BG Fault	822	Apply Fault
23	14.500	BCG Fault	800	Remove Fault	**47**	27.499	BG Fault	822	Remove Fault
24	15.500	AG Fault	848	Apply Fault	**48**	28.499	Load Rejection	844	Open Breaker

3.4.1 Simulation Results

The events simulated in Table 3.10 are intended to provide a diverse collection of transient conditions for the ML model to consider. This includes events such as significant load connections and load rejections that may lead to severe fluctuations in the distribution system synchrophasors. It is essential for the ML model to differentiate between system operating anomalies such as these and more severe transients from unplanned events such as faults. Three such anomalies are demonstrated in Figure 3.3; including a load connection, load rejection and an utility islanding event.

During a load connection event, as described in Figure 3.3(a), the phase voltage magnitude at the IBR may experience a significant sag. Conversely, during this period there is an influx of current flowing from the inverter-based resource to the load.

During a load rejection event, as described in Figure 3.3(b), the phase voltage magnitude of the IBR may exhibit a spike that exceeds acceptable distribution system levels. The sudden disconnection of a large load can cause a temporary voltage rise due to the inductive nature of the grid. The phase current magnitude during a load rejection event depends on the behaviour of the inverter. In some cases, the inverter may limit the output current. However, if the inverter is not designed with specific current-limiting features, the phase current may rise temporarily before stabilizing. It is important to note that the behaviour of an inverter-based resource can vary depending on the specific control strategies and protection schemes implemented in the system. The characteristics described provide a general overview of the phase voltage and current behaviour during these events however these characteristics are significantly influenced by the specific design and configuration of the IBR.

A utility islanding scenario is also simulated in the PSCAD modelling. Figure 3.3(c) demonstrates severe transient conditions during this event. In this scenario, the loss of the utility initiates significant instability in the distribution system and the system is unable to re-stabilize until the utility is resynchronized. In practice, these conditions would lead to the de-energization of the entire distribution feeder. However, in this work, the ML model is trained to differentiate between these islanding anomalies and a distribution system fault and not classify these as a triggering event.

Samples of various fault transient conditions are shown in Figure 3.4. As in Figure 3.3, these characteristics have been retrieved from the IBR POC to demonstrate how the IBR in particular responds to the various event types. During a short-circuit event, the fault current contribution from inverter-based resources is typically lower compared to traditional sources. [50] demonstrates that fault current increases are often greater in the synchronous machine compared to IBRs of similar capacity. IBRs typically incorporate additional current-limiting features than those modelled in this work that prevent excessive current flow to protect their internal components. These additional current limiting features will often result in significantly lower fault currents than those shown through these simulations.

The results demonstrated in Figure 3.4, with the exception of the line-to-ground fault in Figure 3.4(b), show IBR fault current contributions higher than those previously discussed. However, these fault current magnitudes are significantly less than the magnitudes produced by synchronous generators of comparable capacity.

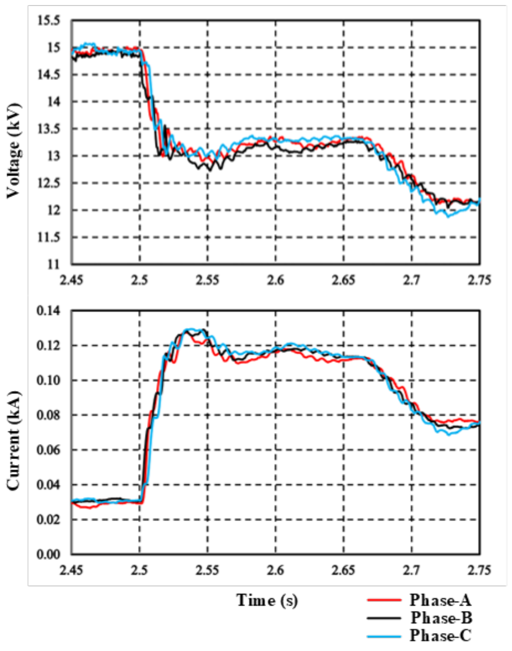
3.5 Summary

This chapter provides an in-depth exploration of the Electromagnetic Transient (EMT) modeling utilized to produce the application specific dataset for training and testing the machine learning models. The chapter begins by introducing the concept of EMT modeling and its ability to capture the behavior and transient characteristics of inverter-based DERs.

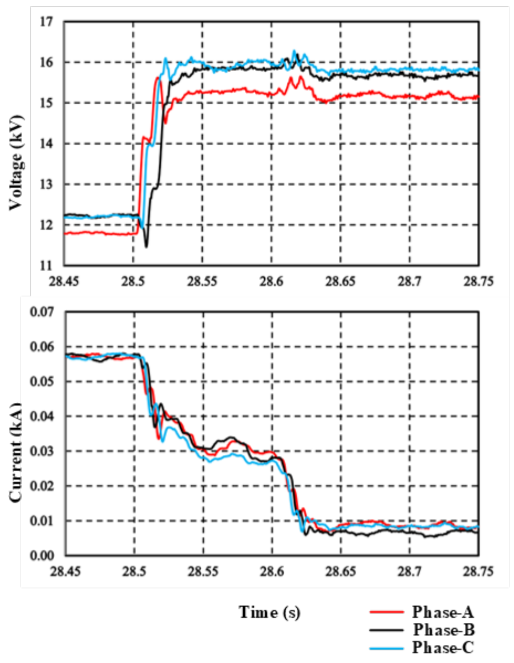
The IEEE 34-bus test feeder foundational model, based on the widely used benchmark model for DER integration studies, is then thoroughly described. The components of the test feeder, including transformers, distribution lines, and loads, are discussed prior to the introduction of the PV source and it's integration.

A detailed PV model, developed by PSCAD, is introduced. The integration of this PV model into the IEEE 34-bus test feeder allows for the investigation of its impact on the system under various operating conditions.

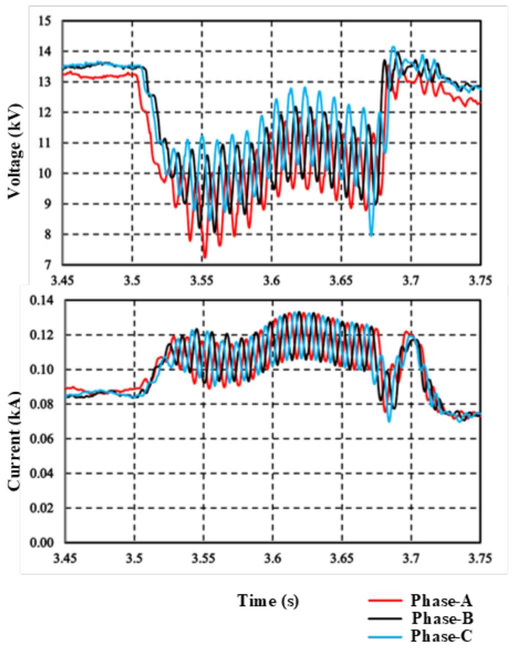
The chapter proceeds with the description of the simulated fault, islanding and



(a) Load Connection

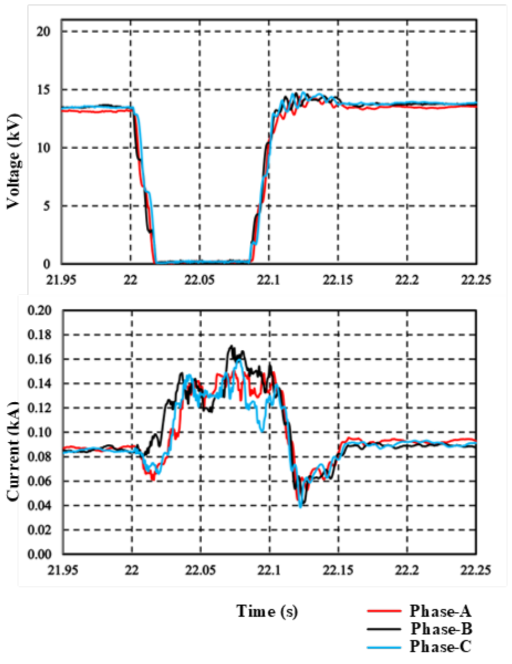


(b) Load Rejection

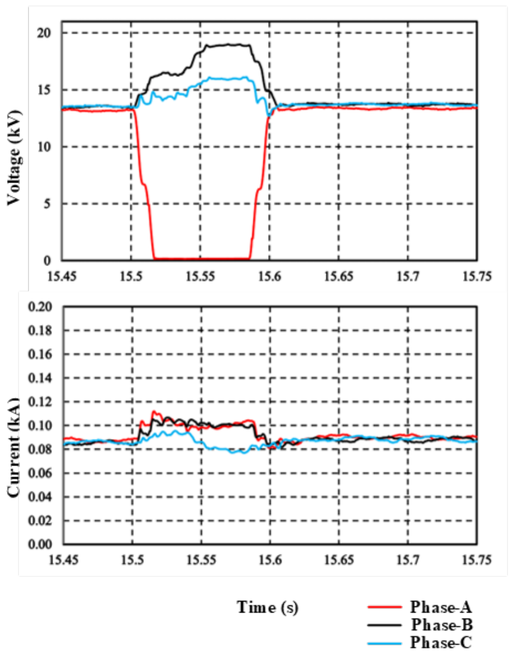


(c) Utility Islanding

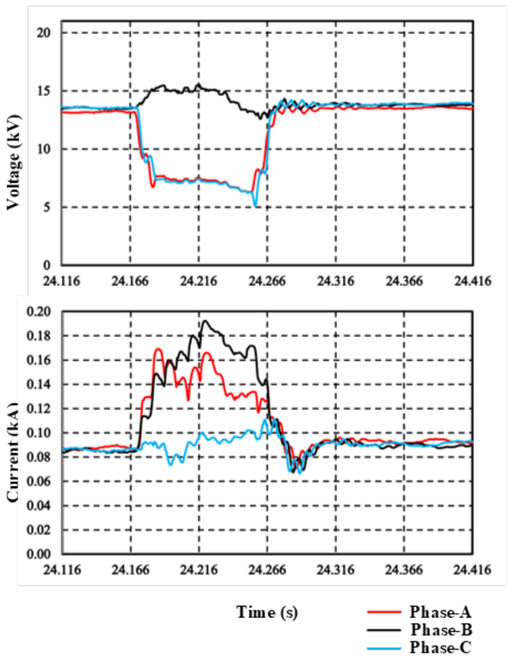
Figure 3.3: System Operating Transients at PCC



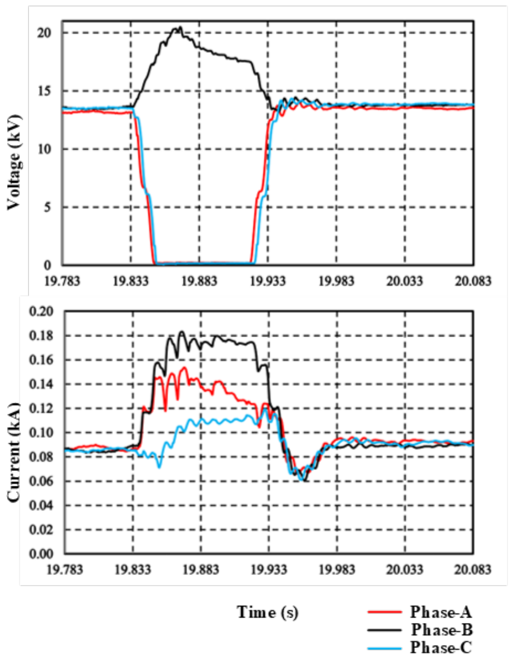
(a) ABC-G Fault



(b) A-G Fault



(c) AC Fault



(d) AC-G Fault

Figure 3.4: IBR Fault Characteristics

load connection/rejection events in the IEEE 34-bus test feeder with the integrated PV model. Throughout 50 model iterations, various system parameters are adjusted to observe their effects on system behavior and transient responses. These iterations allow for the extraction of a diverse set of transient events at the PV PCC.

Finally, an overview of the simulated results is presented, summarizing the observed transient voltage and current profiles during the simulated events. These results provide a visual representation of the input data that may be captured by online power system PMUs and utilized by machine learning algorithms.

Chapter 4

CNN Model Adaptation for Time-Series Data Feature Extraction

Convolutional Neural Networks (CNNs) have emerged as powerful models for various tasks requiring deep learning algorithms, including image classification, object detection, and natural language processing [21]. While CNNs were initially developed for analyzing grid-like data such as images, their application has extended to other domains, including time-series analysis. In this thesis, a CNN model has been optimized for handling normalized time-series data for real-time power system analysis.

CNN models have demonstrated remarkable performance when applied to normalized time-series datasets [2] [22] [23] [34]. Their ability to automatically learn hierarchical representations, capture local and global dependencies, and leverage various architectural extensions makes them well-suited for analyzing time-series data. By employing the appropriate normalization techniques to preprocess the data, CNN models can effectively extract meaningful patterns and relationships from time-series sequences.

Analyzing time-series data poses unique challenges due to its temporal nature, this typically requires capturing both local patterns and long-term dependencies within the sequence. Traditionally, recurrent neural networks (RNNs) and their variants, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), have been the primary models for time-series analysis, given their ability to retain the information of past data inputs. However, CNN models have emerged as a complementary or often superior alternative in certain scenarios. Their ability to incorporate various architectural extensions and techniques enhances their performance on time-series datasets. For instance, dilated convolutions, which incorporate exponentially increasing receptive fields, enable CNNs to capture patterns at multiple scales within the time series. Other techniques, such as the incorporation of residual connections, attention mechanisms, and skip connections in CNN architectures further enhances their ability to model past dependencies and attend to relevant temporal information.

4.1 Model Requirements

Reference [51] describes the general procedure for the real-time detection and classification of power quality disturbances. In their analysis, real-time detection and classification are divided into four major stages with a total of eight comprehensive steps. The four major stages identified include the input space, preprocessing, feature engineering and the decision space; which include the steps, input data preparation, data preprocessing, transformation, feature extraction, feature selection, detection, classification and characterization. This scheme is modified for this specific application as shown in Figure 4.1. Readers may refer to reference [51] for detailed information regarding each particular step not covered in this application.

The input stage in this thesis required the development of the EMT simulation to retrieve the relevant information required to train the ML models. However, in a typical application, the data source would be pre-existing PMUs, protection relays or the local inverter continuously monitoring the distribution system. These devices inherently carry out the feature engineering required for the application of ML models. The raw voltage and current inputs are converted to an analog signal through a sampling process, where the continuous input signals are measured at discrete time intervals. These samples are then converted from their analog form into digital values through an analog-to-digital converter (ADC). The devices are then able to transform the signals from the time domain to the frequency domain through Fourier transform

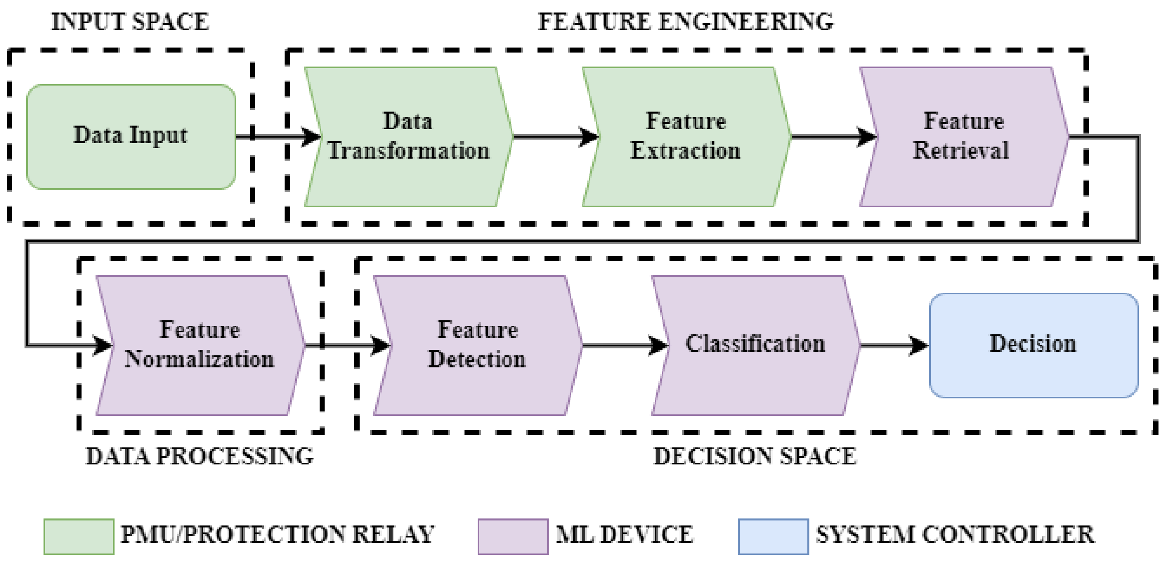


Figure 4.1: Real-Time Detection and Classification

techniques. From there, the devices then carry out further computation to extract features from data such as RMS values, sequential components, harmonic data and other such metrics. These features are then retrieved as required for the respective protection functions or for use by external devices.

The work in this thesis applies particularly to the steps pertaining to the decision space. These steps include the detection and classification of the extracted input features. Prior to the data entering the model, the respective features require an additional stage of preprocessing for the data normalization discussed in the subsequent sections. Once the data features have been normalized, the ML model will carry out inference, intrinsically detecting the state of the various features and classifying the time-series data window. Depending on the application, these steps can be carried out by a single device or by multiple, as described in Figure 4.1.

The research outlined in reference [23] discussed the importance of selecting the appropriate feature extraction to detect abnormality characteristics that are early indicators. It is crucial that the respective ML model can detect these early indicators in order to meet the clearing times outlined in IEEE 1547 [6]. For instance, Table 4.1, outlining the required DER response times for abnormal system voltages, requires the system DER to clear OVs exceeding 1.20 p.u. of the nominal voltage in 0.16 seconds (10 cycles).

Table 4.1: IEEE 1547 DER Required Response Times to Abnormal Voltages

Voltage [pu]	**Clearing Time [s]**
1.20	0.16
1.10-1.20	1.0-13.0
0.0-0.88	2-21.0
0.0-0.50	0.16-2.0

This clearing time is further restricted in IEEE 1547 when discussing the power quality introduced by the DER. This standard specifically states the DER shall not cause the instantaneous voltage on any portion of the distribution system to exceed the cumulative duration shown in the below figure - potentially requiring the DER to remove OV conditions in 1.6 milliseconds. In most applications, OV levels in this range cannot be mitigated through reactive processes such as protective relaying in conjunction with high-speed circuit breakers. In practice, it is more appropriate to mitigate these scenarios through long-term planning of system configurations and defined equipment parameters. For this reason, the handling of OVs requiring clearing within a time frame of 1.6 milliseconds is out of the scope of this work.

4.2 Input Feature Data

The extracted data features are retrieved at a sampling rate of 32 samples per cycle. This represents a typical sampling rate utilized by existing industry protection relays [52]. To mitigate model latency and effectively isolate the DER from the system in an acceptable period of time, half-cycle data sets are entered into the model. These data sets include 16 samples from the voltage and current phasor measurement units (PMUs), as described in 4.1 through 4.4 as well as the current sequence data, as described in 4.5. These features are extracted locally to the system's DER, at the

PCC.

$$I_{mag,ph} = \begin{bmatrix} I^{1}_{a,mag}, & I^{2}_{a,mag}, & \cdots, & I^{16}_{a,mag} \\ I^{1}_{b,mag}, & I^{2}_{b,mag}, & \cdots, & I^{16}_{b,mag} \\ I^{1}_{c,mag}, & I^{2}_{c,mag}, & \cdots, & I^{16}_{c,mag} \end{bmatrix} \tag{4.1}$$

$$I_{\theta,ph} = \begin{bmatrix} I^{1}_{a,\theta}, & I^{2}_{a,\theta}, & \cdots, & I^{16}_{a,\theta} \\ I^{1}_{b,\theta}, & I^{2}_{b,\theta}, & \cdots, & I^{16}_{b,\theta} \\ I^{1}_{c,\theta}, & I^{2}_{c,\theta}, & \cdots, & I^{16}_{c,\theta} \end{bmatrix} \tag{4.2}$$

$$V_{mag,ph} = \begin{bmatrix} V^{1}_{a,mag}, & V^{2}_{a,mag}, & \cdots, & V^{16}_{a,mag} \\ V^{1}_{b,mag}, & V^{2}_{b,mag}, & \cdots, & V^{16}_{b,mag} \\ V^{1}_{c,mag}, & V^{2}_{c,mag}, & \cdots, & V^{16}_{c,mag} \end{bmatrix} \tag{4.3}$$

$$V_{\theta,ph} = \begin{bmatrix} V^{1}_{a,\theta}, & V^{2}_{a,\theta}, & \cdots, & V^{16}_{a,\theta} \\ V^{1}_{b,\theta}, & V^{2}_{b,\theta}, & \cdots, & V^{16}_{b,\theta} \\ V^{1}_{c,\theta}, & V^{2}_{c,\theta}, & \cdots, & V^{16}_{c,\theta} \end{bmatrix} \tag{4.4}$$

$$I_{mag,seq} = \begin{bmatrix} I^{1}_{1,mag}, & I^{2}_{1,mag}, & \cdots, & I^{16}_{1,mag} \\ I^{1}_{2,mag}, & I^{2}_{2,mag}, & \cdots, & I^{16}_{2,mag} \\ I^{1}_{0,mag}, & I^{2}_{0,mag}, & \cdots, & I^{16}_{0,mag} \end{bmatrix} \tag{4.5}$$

The resulting dataset window, as seen by the respective ML models, is structured as described by 4.6.

$$X_i = \begin{bmatrix} V^1_{a,mag}, & V^2_{a,mag}, & \cdots, & V^{16}_{a,mag} \\ V^1_{a,\theta}, & V^2_{a,\theta}, & \cdots, & V^{16}_{a,\theta} \\ V^1_{b,mag}, & V^2_{b,mag}, & \cdots, & V^{16}_{b,mag} \\ V^1_{b,\theta}, & V^2_{b,\theta}, & \cdots, & V^{16}_{b,\theta} \\ V^1_{c,mag}, & V^2_{c,mag}, & \cdots, & V^{16}_{c,mag} \\ V^1_{c,\theta}, & V^2_{c,\theta}, & \cdots, & V^{16}_{c,\theta} \\ I^1_{a,mag}, & I^2_{a,mag}, & \cdots, & I^{16}_{a,mag} \\ I^1_{a,\theta}, & I^2_{a,\theta}, & \cdots, & I^{16}_{a,\theta} \\ I^1_{b,mag}, & I^2_{b,mag}, & \cdots, & I^{16}_{b,mag} \\ I^1_{b,\theta}, & I^2_{b,\theta}, & \cdots, & I^{16}_{b,\theta} \\ I^1_{c,mag}, & I^2_{c,mag}, & \cdots, & I^{16}_{c,mag} \\ I^1_{c,\theta}, & I^2_{c,\theta}, & \cdots, & I^{16}_{c,\theta} \\ I^1_{1,mag}, & I^2_{1,mag}, & \cdots, & I^{16}_{1,mag} \\ I^1_{2,mag}, & I^2_{2,mag}, & \cdots, & I^{16}_{2,mag} \\ I^1_{0,mag}, & I^2_{0,mag}, & \cdots, & I^{16}_{0,mag} \end{bmatrix} \tag{4.6}$$

PMU data analysis provides insight into the characteristics of the respective power system. In the case of fault analysis, PMUs provide fundamental information through synchrophasors, pertaining to the location of the disturbance and the classification of the fault [53]. Synchrophasors refer to the cosine wave magnitude and angle of system voltages and currents [54]. These waves can be represented as 4.7.

$$y(t) = Y_m \cos\left(w(t) + \theta\right) \tag{4.7}$$

By utilizing 4.7, the current and voltage vector magnitudes and angles are extracted. These elements provide electrical characteristics pertaining to an absolute value in time [55].

Symmetrical/sequence components are brought in to supplement the synchrophasor data for their ability to identify real-time system operational characteristics and fault classification [56].

4.2.1 Data Normalization

Before training the model, the input data set is normalized. Normalization refers to the process of transforming data to a common scale or range to facilitate meaningful comparisons and analyses. Normalizing the input data will reduce the model estimation errors and the calculation time needed in the training process [48]. If scales are dissimilar for the different features (i.e. voltage and current), the large-scaled feature data will have a higher contribution to the output error. This will result in the error reduction algorithm (back-propagation) focusing on the weights of higher values, neglecting the information from the small valued weights [48]. The type of input data normalization utilized and its effectiveness has a direct relationship with the input data characteristics and distribution [57]. The two most common types of normalization are the min-max method and z-score.

The min-max method will normalize data within the range of (0, 1) or (-1, 1) based on the maximum and minimum values of the training data subset. The model structure and respective characteristics should be considered in the determination of the normalized data range. For example, layer activation functions such as ReLU performance increases when utilizing a normalized dataset range of (0,1), while the converse is true for tanh activation functions [43]. This in turn affects the performance of the weight and bias initializer [58].

Min-max normalization is described in 4.8. This linear transformation equation maintains all the distance ratios of the original vectors after normalization.

$$\widehat{X}_{(i,j)} = \frac{X_{(i,j)} - X_{min}}{X_{max} - X_{min}}(u - l) + l \tag{4.8}$$

where $\widehat{X}_{(i,j)}$ is the normalized data, $X_{(i,j)}$ is the original data, X_{max} and X_{min} are the respective maximum and minimum values of X and u and l are the respective upper and lower values of the new normalized data, $\widehat{X}_{(i,j)} \in [l : u]$. The range of normalized values are typically $[u = 1, l = 0]$ or $[u = 1, l = -1]$.

The z-score method is an ideal normalization method when the input data is normally distributed and when the data can be effectively characterized with parameters such as mean and standard deviation [57]. These attributes are well suited for the data collected through the PSCAD simulations, as shown in the distribution curve of the collected data below.

Z-score standardization is described in 4.9. Generally, standardization consists of subtracting a quantity related to a measure of localization (mean) and dividing by a measure of the scale (standard deviation). The z-score transforms the original data to obtain a new distribution with a mean ($\hat{\mu}$) of 0 and a standard deviation ($\hat{\sigma}$) of 1.

$$\widehat{X}_{(i,j)} = \frac{X_{(i,j)} - \mu}{\sigma} \tag{4.9}$$

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4.10}$$

$$\hat{\sigma} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})^2} \tag{4.11}$$

As the min-max normalization and z-score standardization have attributes that complement the raw input dataset and the traditionally preferred ReLU activation function, the two normalization methods have been tested and compared.

4.3 Optimized Convolutional Neural-Network

Traditional shallow machine learning algorithms such as SVM and back-propagation neural networks (BPNNs) have relatively poor complex feature extraction capabilities [59]. This makes it difficult to identify the features hidden in time-series data monitoring the health and characteristics of complex power systems. To solve this issue, a traditional CNN model has been restructured to effectively extract features from time-series data while placing an emphasis on speed and resource requirements.

Several variations of the traditional CNN model have been developed for use with time-series data classification. Model structures and parameter variations are tested to quantify the performance of the particular network variances. The three general CNN structures developed include a basic 2-dimensional (2D) CNN, a 1-dimensional (1D) CNN and a deep CNN model. The basic 2D CNN is utilized as the base case, intended to efficiently classify time-series data windows. The 1D CNN was developed to mitigate the effect of small 2D convolutional kernels on feature loss. Finally, a deep CNN model was developed to increase the model's ability to identify complex characteristics.

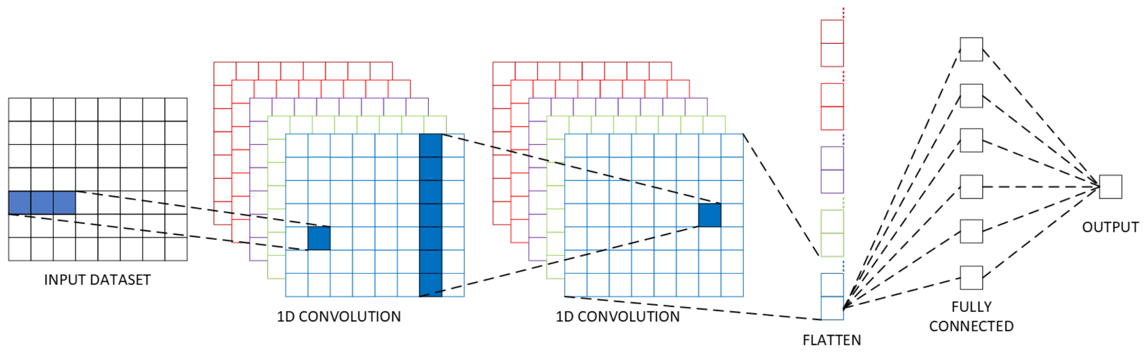


Figure 4.2: 1D Kernel CNN Architecture

4.3.1 Convolution Layers

One of the main strengths of CNNs lies in their ability to automatically learn hierarchical representations from the data. The convolutional layers in a CNN operate as local filters, capturing local patterns and features by convolving a set of learnable filters across the input data. This hierarchical feature extraction enables the model to learn representations at multiple levels of abstraction, which is crucial for understanding the complexity present in time-series data.

Model complexity and the effectiveness of the model are often trade-offs. For this reason, the optimal number of convolution layers is tested to determine the lowest number of convolution layers that can be utilized while still obtaining acceptable results. Models with one, two and three convolution layers are examined.

The i^th feature map at the l^th convolutional layer can be represented by 4.12.

$$z_i^l = \alpha(W_i^l \bullet X^{(l-1)} + b_i^l) \tag{4.12}$$

Where α denotes the activation function, W^l and b^l are the hidden layer weight and bias matrices at layer l, X^l is the layer input vector, $\bullet$ denotes a convolution operation.

Activation Function

A rectified linear unit (ReLU) activation function effectively reduces vanishing gradient and overfitting problems associated with CNN models [39]. This is notably apparent in deep neural networks, with a large number of convolutional layers [60].

Conversely, hyperbolic tangent (tanh) functions are typically applied to shallow models and provide an output range of $(-1, 1)$. As a result, the negative value of input data is mapped as negative [43], while ReLU functions drive all negative values to zero. Therefore, if the input data is not appropriately prepared when utilizing ReLU activation, negative values of input features that may play an intricate role in the dataset classification can be eliminated.

Initialization Schemes

In the Glorot initialization scheme, biases are initialized at 0 and the weights are set to a uniform distribution of values with a range proportional to the size of the number of nodes in the previous layer, as described by 4.13. As shown in [58], this scheme performs well with the use of sigmoid and tanh activation functions.

$$W_{(i,j)} \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right] \tag{4.13}$$

Where $U[-a, a]$ is the uniform distribution in the interval $(-a, a)$ and n is the number of nodes/columns of weights in the previous layer.

However, as suggested by [60], the Glorot activation scheme does not lend itself to its use with ReLU activation functions, often leading to exploding gradient concerns in deeper models. [60] proposed a modified Glorot function, suitable for use with ReLU activations. This initialization function is now referred to as He initialization and can be described by 4.14.

$$W_{(i,j)} \sim \sigma\left[-\sqrt{\frac{2}{n}}, \sqrt{\frac{2}{n}}\right] \tag{4.14}$$

Where σ is one standard deviation and n is the number of nodes/columns of weights in the previous layer.

Kernel Size

For traditional CNN networks utilizing image input arrays, different scales of filters extract different local features. Large-scale filters may find the symmetric property, while small-scale filters could find particular characteristics.

Traditional image identification CNN models utilize 3×3 convolutional kernels. 3×3 convolutional kernels reduce the number of model layer parameters when compared to larger kernel sizes such as 5×5. For example, a 3×3 kernel requires 9 parameters for each respective output element. Whereas, a 5×5 kernel requires 25 parameters. This significantly reduces the model training period and inference times. Reduced kernel sizes also improve the model's ability to detect patterns irrespective of their position in the input. As the kernel slides across the input, it convolves with different patches, allowing the model to learn spatially invariant features. This property is crucial for recognizing patterns in different parts of a time-series dataset.

When utilizing CNNs for time-series data, different scales of filters allow the extraction of features pertaining to the interaction between the various feature columns [22]. This is imperative for time-series data in this application, as the array location of the input feature columns with respect to one another is completely arbitrary. For these reasons, 3×3 kernels are utilized for all 2D convolutions in this application.

For the 1D CNN model, a modified wide-area kernel technique, as proposed in [23], is considered. In this approach, 1D kernels of various lengths are implemented. Initially, a 3×1 kernel is traced along the isolated feature columns. This kernel is intended to identify the low-level characteristics of each feature/parameter in the data window. Next, a wide 1×15 kernel is convolved across each set of features to identify system-level patterns. The use of these kernels is depicted in Figure 4.2

4.3.2 Batch Normalization

Previous works, [39] and [40], have shown batch normalization to improve training speed, prevent overfitting, reduce the effect of gradient explosion and gradient vanishing effects and increase the model's robustness to unseen data. The use of batch normalization in this application is investigated for its training speed and robustness improvement capabilities.

4.3.3 Dropout

Batch normalization is utilized in lieu of dropout as described in [36], as preliminary training demonstrated minimal susceptibility to overfitting. However, dropout should be considered in future works to provide increased resilience to imperfections/noise in the raw data and improve generalization.

4.3.4 Pooling Layer

CNN models can effectively capture both local and global dependencies within the time-series data. By utilizing pooling operations the CNN can downsample the learned representations while retaining the most salient features. This downsampling process allows the model to capture high-level patterns and long-term dependencies across the time series, effectively integrating information from multiple time steps. Thus, CNNs can learn to recognize important temporal patterns while being resilient to noise and irrelevant variations in the data.

Models developed in [23] utilize a global pooling technique to reduce the complexity introduced by traditional pooling operations and bypass the fully connected output layer. However, global max pooling has some limitations. It is shown in [61] that implementing global pooling techniques after batch normalization and ReLU activation layers will reduce the effectiveness of the global pooling classification.

$$GMP_c^l = max(X_c^{l-1}) \tag{4.15}$$

Where GMP_c^l is the c^{th} feature map of the l^{th} layer after the GMP operation, X_c^{l-1} is the overall feature map of the c^{th} channel, and $c = 1, 2, 3, ..., C$, where C is the number of channels.

Alternatively, a global pooling method considering the average value in a feature map was introduced in [62]. The advantage of global average pooling, especially when utilizing it in place of a fully connected layer, is its ability to categorize feature maps so the separate generalized features for the input data window can be compared in the output layer.

$$GAP_c^l = avg(X_c^{l-1}) \tag{4.16}$$

Where GAP_c^l is the c^{th} feature map of the l^{th} layer after the GAP operation, X_c^{l-1} is the overall feature map of the c^{th} channel, and $c = 1, 2, 3, ..., C$, where C is the number of channels.

Global pooling can also has a low resistance to noise. Since it only considers the maximum value within each channel, it can be sensitive to outliers. Global pooling also introduces a low utilization of information when dealing with large-scale data inputs. This concern is mitigated in this application due to the small input vectors.

A 1D pooling layer is also considered to accommodate the input data structure. As the characteristics of each feature/column of input data are often independent, introducing a max pooling layer and combining the respective feature data in an initial layer may introduce a loss of information [23].

4.3.5 Fully Connected Layer

The fully connected layer, also known as the dense layer, is a key component of traditional CNNs that follows the convolutional and pooling layers. While convolutional layers focus on extracting local features and preserving spatial relationships, fully connected layers enable global learning and decision-making.

When introduced, this layer captures high-level global information from the feature maps generated by the preceding convolutional layers. It performs a matrix multiplication between the input vector and a set of weights, followed by an element-wise activation function. This process allows the network to learn complex patterns and relationships across the entire time-series data window.

As done for the convolutional layer activation functions, the performance of both

ReLU and TanH activation functions in the fully connected layer are tested for their use in time-series data applications. These layers are represented by 4.17 and 4.18.

$$z_i^l = relu(W_i^l X^{(l-1)} + b_i^l) \tag{4.17}$$

$$z_i^l = tanh(W_i^l X^{(l-1)} + b_i^l) \tag{4.18}$$

Where W^l and b^l are the hidden layer weights and biases at layer l and X^l is the layer input vector.

The incorporation of a fully connected layer also requires the use of a flatten layer. The flatten layer transforms the multidimensional output of the convolutional layers into a one-dimensional vector that can be utilized by the fully connected layer. In this application, the convolutional layers will have a three-dimensional output, $y^{l(i,j)}$, this will be transformed into the resulting vector described by 4.19.

$$\hat{y}_{i*j*l\times 1} = y_{i\times j\times l} \tag{4.19}$$

Where $y^{l(i,j)}$ is used to denote the output of the previous convolutional layer, i designates the number of feature sets, j designated the number of features and l denotes the number of filters.

4.3.6 Output Layer

The output layer utilizes a sigmoid activation function for binary classification to identify if an abnormality has been detected, as described in 4.20. The sigmoid activation function will provide an output within the range of 0 and 1. This range represents the likelihood that the result is either true, a fault is detected, or false, the system is healthy. The CNN models trained in this research utilize a threshold of 50%. Therefore, if the model makes a fault-positive prediction greater than or equal to 50% likelihood, it will classify the input data array as a fault.

$$Output(X) = Sigmoid(W^T \bullet X + B) = \begin{cases} 0.5 \leq, \ a\ fault\ is\ detected \\ 0.5 >, \ no\ fault\ is\ detected \end{cases} \tag{4.20}$$

4.4 Summary

This chapter delves into the techniques and layers utilized to optimize CNN models for effective time-series data analysis and the modifications made to the traditional CNN architecture to tailor it to the unique characteristics of time-series data.

This chapter discusses the specific data retrieved from the EMT model simulations to be utilized for training and testing the adapted CNN and other ML model types. It outlines the crucial preprocessing steps applied to the power system EMT simulation data to optimize it's suitability for training the CNN model.

The adaptation of CNNs for time-series data involves redefining the input data structure to consider the temporal nature of the data and enable feature extraction across different time steps which is imperative for capturing temporal patterns present various IBR transient conditions.

Chapter 5

Testing & Results

5.1 Resource Definition & Preprocessing

Model training and testing were performed through Amazon Web Services (AWS) SageMaker application. This application hosts the computations on a remote server with the user-defined technical specifications. For this research, the model testing and training were carried out on a general-purpose 2-vCPU and 4-GiB instance with a TensorFlow 2.6, Python 3.8, CPU-optimized image, on a Python 3 kernel. The CNN models were developed utilizing Keras-defined layers. Although inference times would reduce through the use of a large GPU host, this was avoided as the application would have limited GPU computational capabilities available.

Adaptive moment estimation is utilized as an optimizer for all models with a learning rate of 0.0001. This optimization is a stochastic gradient descent method based on adaptive estimation of first-order and second-order moments. The loss function utilized is binary cross entropy.

During the hyperparameter and CNN model structure optimization testing, an early stopping algorithm is utilized to prevent overfitting of the model. The early-stopping algorithm calculates the sensitivity percentage of the validation dataset and stops the training iterations once the model validation sensitivity does not improve over 4 consecutive epochs. Once the early-stopping algorithm stops the training, the training iteration with the best sensitivity performance is selected to maximize model results. When training the CNN models, a batch size of 64 data windows is set to limit the number of batches that are trained without seeing instances of a fault.

Due to the inherent variability of trained models caused by the randomness of the weight initialization, for model performance validation tests, multiple tests are carried out for each model configuration.

Prior to training, the input data retrieved from the PSCAD simulations was pre-processed and stored for repeatability in training. The initial processing stage partially removed data from steady-state conditions through the PSCAD simulations. By doing so, the severity of the classification imbalances is reduced so the training data is comprised of 42,012 time steps. Where 38,344 time steps are considered healthy system representations and 3,668 time steps are during fault conditions from each of the 50 simulations. The original dataset is then duplicated and separately normalized through Z-score standardization and min-max normalization so each respective technique can be tested. As the magnitudes of each feature in the dataset varied, each feature was normalized separately so features with larger magnitudes did not have an unintended impact on influencing the ML model characterizations. The feature set summaries of each normalization technique are shown in Table 5.1 and Table 5.2.

Table 5.1: Z-Score Standardization Dataset Description

	Vam	**Vap**	**Vbm**	**Vbp**	**Vcm**	**Vcp**	**Iam**	**Iap**	**Ibm**	**Ibp**	**Icm**	**Icp**	**I1m**	**I2m**	**I0m**
count	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012
mean	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
std	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
min	-3.4901	-2.2579	-9.6509	-12.8086	-13.1514	-2.0194	-6.3242	-3.8423	-1.9458	-20.7022	-3.6516	-8.7331	-3.6270	-1.2060	-0.4221
25%	0.3033	-0.2731	-0.0003	0.0887	-0.0313	-0.2282	-0.0558	0.0088	-0.3037	0.0223	0.2242	-0.2995	0.2284	-0.5938	-0.2997
50%	0.3087	-0.2715	0.0112	0.0914	0.0034	-0.2131	0.1772	0.1029	-0.1912	0.0512	0.2998	-0.2310	0.2626	-0.2003	-0.2990
75%	0.3148	-0.2698	0.0399	0.0933	0.0234	-0.1813	0.4103	0.1736	-0.0787	0.0888	0.3602	-0.1166	0.3409	0.2783	-0.2963
max	0.8414	5.9884	2.3704	1.3163	2.9674	12.4136	7.7506	4.2740	5.2638	2.2938	2.6313	10.0917	2.3000	8.3554	4.8140

Once the data is normalized, it is broken into 16 time-step data windows. Each window will constitute an input to the ML model that will carry out one inference per data set window. Each of these windows will contain the full 15 feature set, for an array size of (16, 15). The final dataset for model training and testing contains 131,250 data windows.

Table 5.2: Min-Max Normalization Dataset Description

	Vam	**Vap**	**Vbm**	**Vbp**	**Vcm**	**Vcp**	**Iam**	**Iap**	**Ibm**	**Ibp**	**Icm**	**Icp**	**I1m**	**I2m**	**I0m**
count	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012	42012
mean	0.6304	0.2752	0.7177	0.9061	0.6424	0.5843	0.0383	0.3295	0.0348	0.9002	0.2695	0.4629	0.2686	0.0302	0.0533
std	0.1806	0.0877	0.0717	0.0707	0.0398	0.0120	0.0028	0.0315	0.0056	0.0432	0.0673	0.0527	0.0693	0.0248	0.1248
min	0.0002	0.0772	0.0259	0.0006	0.1189	0.5601	0.0206	0.2084	0.0239	0.0067	0.0237	0.0023	0.0173	0.0002	0.0006
25%	0.6852	0.2513	0.7177	0.9124	0.6412	0.5816	0.0382	0.3298	0.0331	0.9012	0.2846	0.4471	0.2844	0.0154	0.0159
50%	0.6861	0.2514	0.7185	0.9126	0.6426	0.5817	0.0388	0.3328	0.0337	0.9024	0.2897	0.4507	0.2868	0.0252	0.0160
75%	0.6872	0.2515	0.7206	0.9127	0.6434	0.5821	0.0395	0.3350	0.0344	0.9040	0.2938	0.4567	0.2922	0.0371	0.0163
max	0.7823	0.8004	0.8876	0.9992	0.7606	0.7333	0.0600	0.4642	0.0641	0.9992	0.4466	0.9951	0.4279	0.2375	0.6538

Finally, the structured, normalized dataset is divided into a training dataset and a testing dataset where 80% is allocated toward training and 20% is allocated toward testing.

5.2 Model Layer Testing & Confirmation

Reference [63] lists accuracy, availability and data timeliness as critical attributes when utilizing PMUs for protection and control applications. Data accuracy demands the synchrophasor measurements, such as phasor measurements, frequency and ROCOF estimates, and time synchronization, within acceptable errors. Data availability requires the measurement data to be complete, consistent, and without loss. Data timeliness refers to the measurement data delivered to their destinations within acceptable latencies. These same conclusions should be applied to the application of an ML algorithm utilizing PMU data for grid protection and control.

For this reason, the machine learning models are evaluated and compared using the metrics of accuracy, precision, sensitivity, availability and timeliness. In binary classification, an input dataset can be classified as one of two categories - the positive class and the negative class. As all ML models reviewed in this work utilize a sigmoid activation function in the output layer, they will produce a continuous output representing the estimated probability that a respective input will belong in the positive class. Typically, a model's classification threshold is set to 0.5. This threshold will cause any estimate with a probability above 50% to be classified as positive and any

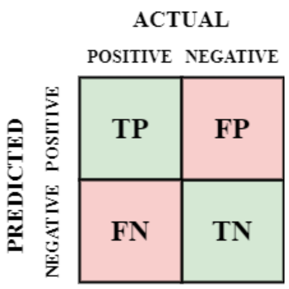


Figure 5.1: Confusion Matrix Structure

probability less than 50% to be classified as negative.

In binary classification, there are four possible categories for the estimated results. If an input dataset is positive and it is classified as positive, this result is listed as a true positive (TP). If an input dataset is negative and it is classified as positive, this result is listed as a false positive (FP). If an input dataset is negative and it is classified as positive, this result is listed as a false negative (FN). If an input dataset is negative and it is classified as negative, this result is listed as a true negative (TN). These results can be visualized in a matrix referred to as a confusion matrix or a contingency table, as depicted in Figure 5.1.

ML model performance assessment methods utilize these classification parameters (TP, TN, FP, FN). These parameters are used to calculate metrics such as accuracy, sensitivity, specificity and precision [64]. Accuracy, defined by 5.1, measures the overall correctness of a classification model. It calculates the portion of correctly classified instances out of the total number of input samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Sensitivity, defined by 5.2, also known as recall, measures the proportion of the correctly predicted positive instances out of the actual positive instances. This metric is useful when it is imperative that a model can effectively identify positive instances.

$$Sensitivity = \frac{TP}{TP + FN} \tag{5.2}$$

Conversely, specificity, defined by 5.3, measures the proportion of correctly predicted negative samples out of the actual number of negative samples.

$$Specificity = \frac{TN}{TN + FP} \tag{5.3}$$

Precision, defined by 5.4, measures the proportion of correctly predicted positive instances out of the total number of instances predicted as positive. Therefore, defining how reliable the model's positive predictions are.

$$Precision = \frac{TP}{TP + FP} \tag{5.4}$$

Depending on the dataset, these methods can fail to validate a model's performance when utilizing inappropriate parameters [65]. For example, if accuracy or precision is utilized to quantify the performance of a model, the metrics will change if the dataset samples become imbalanced or are more likely to be classified in one category. As the dataset produced for this work is inherently imbalanced, containing significantly more instances of healthy-system condition data windows, sensitivity and specificity are the preferred metrics when analyzing performance in this work.

A receiver operating characteristic (ROC) area curve is a graph that visualizes the trade-off between true positive rates and false positive rates and is often utilized for ML performance evaluation [66]. However, research has shown that precision-sensitivity curve (PRC) area graphs are more suitable for binary classifiers on unbalanced data [65]. Therefore, PRC curves have been utilized to visually compare the respective ML models in this work.

To construct the precision-recall curve, the trained model makes predictions on a separate set of data allocated for testing. For each input, the model produces a probability that the input belongs to the fault-positive class. Most binary classifiers, such as the models developed in this work, utilize a 50% threshold to classify the input. In this case, predictions with a probability greater than or equal to 0.5 are classified as fault-positive samples. When creating the PRC the threshold value is varied and the model's precision and sensitivity are calculated and plotted.

Model computational costs are expressed as the memory requirements and the inference time to execute a model on new data [67]. The number of parameters in a deep learning model can provide an estimate of its computational cost. Parameters represent the number of weights and biases of the model and directly impact the amount of computation required during training and inference. The higher the number of parameters, the more computations are typically needed [68].

As this work is more suitable for field implementation on resources with limited GPU performance, the model size (number of parameters) is used as the model resource and speed metric as opposed to FLOPs [69]. The parallel architecture of GPUs enables them to handle a larger number of FLOPs compared to CPUs. While a CPU typically has fewer cores, each core is usually more powerful and optimized for sequential processing. This reflects the results shown in this work, demonstrating limited inference time performance increases with deeper CNN models and fewer parameters.

5.2.1 LVQ Model

To test the performance of a basic LVQ model in this application, a generalized LVQ (GLVQ) model is implemented. This specific LVQ model structure overcomes the cost/loss function concerns with early LVQ models, discussed in reference [70]. This revised LVQ model optimizes the margin between the input vectors and respective prototypes, much like SVM models. However, in the case of GLVQs, the aim is to increase the difference between the distance from an input vector to the prototype of its own class and the distance from the prototype of the closest incorrect class. This difference is referred to as the hypothesis margin. This varies from the goal of SVMs, which attempt to maximize the separation margin between the input vectors and the support vector/decision boundary.

The LVQ model in this work assigns one prototype per class and a maximum number of training iterations of 100. LVQ models require a fixed-length feature vector representation for each data point. Therefore each 16x15 input data window (2D array) is arranged in a 1-dimensional array with a resultant vector length of 240

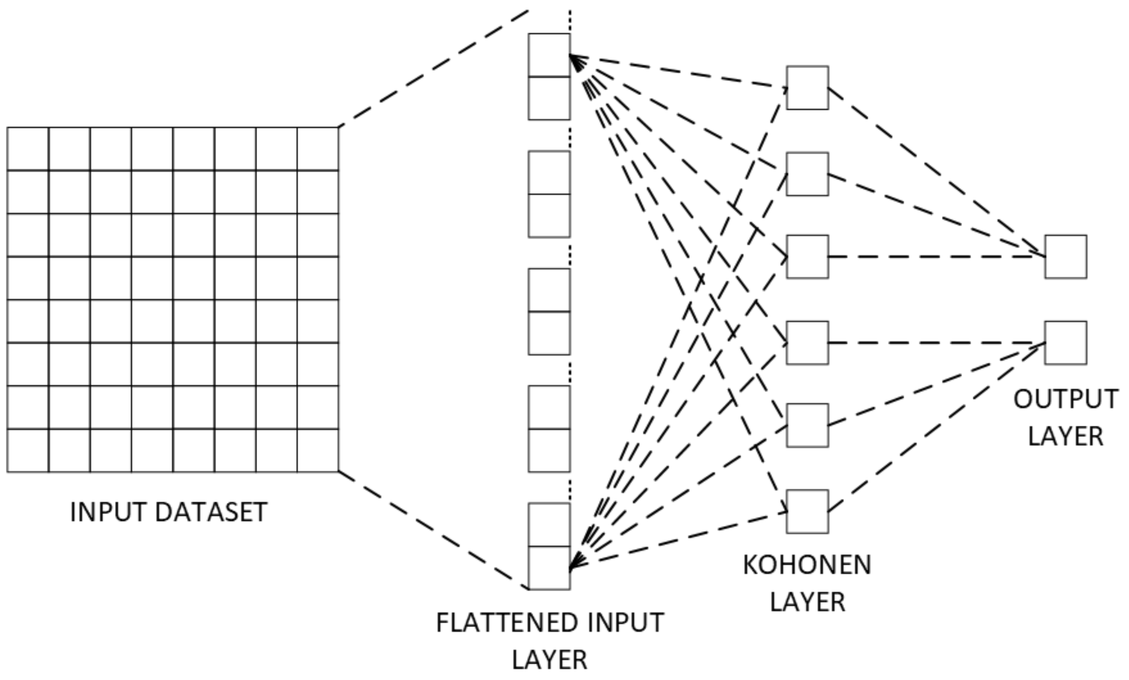


Figure 5.2: LVQ Model Architecture

floating point integers as shown in Figure 5.2.

5.2.2 SVM Model

Like LVQ models, SVM structures require a fixed-length feature vector representation for each data point. Therefore each 16x15 input data window (2D array) is arranged in a 1-dimensional array with a resultant vector length of 240 floating point integers.

SVM models were trained with linear, rbf and sigmoid kernel types. These kernels transform the input data window to a higher dimension, allowing the dataset to be classified into their respective groups.

Linear kernels carry out a standard dot product between the input vectors x and y. It represents a linear relationship between the data points in the original feature space.

$$K(x, y) = x^T \bullet y \tag{5.5}$$

The Radial Basis Function (RBF) kernel measures the similarity between data points based on their Euclidean distance in the higher-dimensional space. The parameter γ controls the width of the Gaussian distribution.

$$K(x, y) = exp(-\gamma * ||x - y||^2) \tag{5.6}$$

The sigmoid kernel maps data points to a higher-dimensional space using the hyperbolic tangent function.

$$K(x, y) = tanh(\alpha * x^T * y + c) \tag{5.7}$$

The results from training the SVM model with the three kernel types described in equations 5.5- 5.7 are shown in Table 5.3.

Table 5.3: SVM Performance for Varied Model Kernel Types

Kernel	**Sensitivity**	**Specificity**
RBF	0.9701	0.9993
Linear	0.9752	0.9987
Sigmoid	0.7416	0.9721

5.2.3 LSTM & GRU Model

The LSTM and GRU models are structured from single LSTM and GRU Keras layers. Each of these respective layers is fed into a fully connected layer with a sigmoid activation function to produce a probabilistic classification of the input data window.

The LSTM model's long short-term memory layer, defined by Keras, utilizes the activation functions defined in Chapter 2. This includes a hyperbolic tangent (tanh) activation function for the forget gate and hidden vector and a sigmoid activation function for the input, output and cell state gates.

Similarly, the GRU model's gated recurrent unit layer, utilized the activation functions as outlined in Chapter 2. A sigmoid function is utilized for the update and reset gate. While a hyperbolic tangent function is utilized for the output layer.

5.2.4 Basic CNN Optimization

Optimization was carried out for a 2D CNN model architecture's use in time-series dataset classifications. The optimization goals for all hyperparameters are sensitivity, specificity and parameters. This optimization process was divided into six separate stages.

The first stage compares the performance of the model while utilizing the data preconditioned through Min-Max Normalization and Z-Scare Standardization. As the layer activation functions and respective initialization methods determine how the input data will be handled, each normalization technique is tested with each combination of activation function and initialization method. Each of these CNN models utilizes one convolution layer with 256 filters and an output layer with 32 filters without the use of batch normalization and max pooling. The results shown in Table 5.4 demonstrate the best performance is achieved with the z-score standardization dataset.

Table 5.4: Data Normalization Technique Model Performance for Varied Activation Functions and Initialization Methods

Data Normalization	**Initialization**	**Activation Function**	**Sensitivity**	**Specificity**	**Epochs**	**Parameters**
Min-Max	He	tanh	0	1	4	1,490,976
Min-Max	He	relu	0.952	0.996	7	1,490,976
Min-Max	Glorot	tanh	0.9615	0.9986	16	1,490,976
Min-Max	Glorot	relu	0.9591	0.9984	15	1,490,976
Z-Score	He	tanh	0.9725	0.9981	13	1,490,976
Z-Score	He	relu	0.976	0.9988	20	1,490,976
Z-Score	Glorot	tanh	0.9761	0.9988	16	1,490,976
Z-Score	Glorot	relu	0.9721	0.9987	8	1,490,976

The next optimization stage focused on the CNN activation functions performance. The two activation functions analyzed are the relu and the tanh functions. Each activation function was tested with various numbers of convolution layers. The results of these tests are listed in Table 5.5. As outlined in , the best model performance came from the use of the relu activation function with He initialization for each number of model convolution layers.

Table 5.5: CNN 2D Convolution Model Varied Activation Function Performance

Conv Layers	Data Norm.	Initialization	Act. Function	Sensitivity	Specificity	Epochs	Parameters
1	Z-Score	Glorot	tanh	0.974	0.9993	21	186,753
1	Z-Score	He	ReLU	0.9748	0.9992	16	186,753
2	Z-Score	Glorot	tanh	0.9689	0.9988	8	144,801
2	Z-Score	He	ReLU	0.9764	0.998	19	144,801
3	Z-Score	Glorot	tanh	0.9752	0.998	23	111,041
3	Z-Score	He	ReLU	0.9709	0.9992	12	111,041

The third stage of testing was performed to determine the benefits of utilizing batch normalization and max pooling. Incorporating batch pooling demonstrated improved performance while utilized with all other variations in the network structure. Although max pooling significantly reduces the number of model parameters, it shows an adverse effect on the sensitivity performance.

Table 5.6: CNN 2D Convolution Model Varied Batch Normalization and Max Pooling Performance

Batch Norm.	Max Pool.	Sensitivity	Specificity	Epochs	Parameters
Yes	No	0.9768	0.9994	22	145,057
No	No	0.9764	0.998	19	144,801
Yes	Yes	0.9548	0.9986	28	13,985
No	Yes	0.9351	0.9957	10	13,729

The fourth stage was carried out to determine the optimal number of convolution layers. Utilizing the best-performing activation function and batch normalization the model was then tested to validate the effect the depth of the model has on model performance. This testing shows minor improvements in model sensitivity when the depth of the model increases.

The fifth stage of testing was performed to determine the optimal number of output layer filters. These results demonstrate the optimal number of output layer filters is 32.

The sixth stage of testing was performed to determine the performance of the

Table 5.7: CNN 2D Convolution Model Varied Convolution Layer Performance

Conv. Layers	Conv. Filters	Output Filters	Sensitivity	Specificity	Epochs	Parameters
1	16	32	0.9772	0.9995	19	93,473
1	32	32	0.9768	0.9992	17	186,881
1	64	32	0.9713	0.9985	8	373,697
1	128	32	0.9744	0.9985	8	747,073
1	256	32	0.9744	0.9987	9	1,494,593
2	16	32	0.9792	0.999	18	70,257
2	32	32	0.9776	0.9993	19	145,057
2	64	32	0.9776	0.9992	16	308,481
2	128	32	0.978	0.9996	14	690,625
3	16	32	0.9764	0.9992	25	51,137
3	32	32	0.9799	0.9993	21	111,425
3	64	32	0.9768	0.9993	14	259,649
3	128	32	0.9815	0.9995	28	666,689

Table 5.8: CNN 2D Convolution Model Varied Output Layer Filters

Conv. Filters	Output Filters	Sensitivity	Specificity	Epochs	Parameters
32	16	0.9764	0.9992	14	65,313
32	32	0.9799	0.9993	21	111,425
32	64	0.978	0.9992	23	203,649
32	128	0.9788	0.9991	16	388,097

model when replacing the fully connected output layer with a global pooling layer. Reference [23] demonstrates that the use of global pooling layers at the end of a CNN will greatly reduce model complexity and improve accuracy in some applications. In this stage, both global max pooling and global average pooling were tested.

Table 5.9: CNN 2D Convolution Model Varied Output Layer Type

Output Layer	Sensitivity	Specificity	Epochs	Parameters
Fully Connected	0.9799	0.9993	21	111,425
Global Max Pooling	0.9744	0.999	30	19,233
Global Avg Pooling	0.9654	0.9987	24	19,233

5.2.5 1D CNN Optimization

Utilizing the results from the 2D convolution CNN architecture optimization, the optimization process for a CNN network utilizing 1D convolution was repeated. Based on results published in other works, it is hypothesized that the use of a 1D kernel in the convolution layers will allow the model to prioritize the relationship between the combination of input features for each time step. This optimization utilizes five stages – activation function optimization, kernel padding performance confirmation, number of time-series convolution filter optimization, number of feature convolution filter optimization and number of output filter optimization.

The first stage repeats to activation function performance confirmation that was carried out for the 2D convolution CNN model. The performance of both the relu and tanh activation functions were tested for use with one, two and three convolution layers. Based on the results from the 2D convolution CNN model tests, the z-score standardized dataset was utilized, batch normalization was activated after each convolution layer and each convolution layer utilized 32 filters. The results depicted in Table 5.10 show better sensitivity performance utilizing the relu activation for two of the three CNN model depths. However, the highest sensitivity performance demonstrated by the tanh function had the lowest specificity performance.

Table 5.10: CNN 1D Convolution Model Varied Activation Function Performance

Conv. Layers	Data Normalization	Initialization	Activation Function	Sensitivity	Specificity	Epochs	Parameters
1	Z-Score	He	Relu	0.9811	0.9997	31	30,177
1	Z-Score	Glorot	tanh	0.978	0.9994	18	30,177
2	Z-Score	He	Relu	0.9827	0.9993	28	31,361
2	Z-Score	Glorot	tanh	0.9784	0.9994	13	31,361
3	Z-Score	He	Relu	0.9776	0.999	9	32,545
3	Z-Score	Glorot	tanh	0.9831	0.9984	26	32,545

The second stage of testing is intended to confirm the performance of the models using "true" and "same" padding. When carrying out the convolution, true padding convolves each parameter in the filter with an input parameter before sliding the kernel. This process is repeated until the last column of the filter matrix reaches the last column of the input array and the last convolution is carried out. When utilizing true padding, the resultant array is reduced. This method of convolution may lead to lost data, or data on the edges of the input array having less impact on the model characterization. In these cases, the same padding can be utilized so the resultant array is the same shape as the input array. The drawback to this method is an increased number of convolutions and model parameters.

To confirm the performance of the models when utilizing padding, four separate models were trained to confirm how kernel padding affects each convolution layer. The results showed that the addition of padding on any convolution layer did not improve the performance of the model in this application.

Table 5.11: CNN 1D Convolution Model Varied Kernel Padding Performance

Time-Series Padding	**Feature Padding**	**Sensitivity**	**Specificity**	**Epochs**	**Parameters**
Yes	Yes	0.9811	0.9973	15	264,833
Yes	No	0.9788	0.9994	16	35,457
No	Yes	0.9784	0.9995	12	203,393
No	No	0.9827	0.9993	28	31,361

The last three stages of tests are carried out to optimize the number of filters utilized for each convolution stage. A combination of time-series convolution filter quantities is tested for each number of convolution layers.

Using the results from the previous test, the optimal number of filters for each model depth is utilized to confirm the optimal number of feature convolution filters. This process is then repeated for a third time to confirm the optimal number of filters for the output layer.

Once the optimized hyperparameters for each CNN model depth were determined, each model was again tested utilizing the same input dataset and trained over the same number of epochs. The models were trained over 40 epochs. This number of iterations exceeded the number of epochs required to train the models in all other prior tests. Through training, the weight of each epoch was saved. After training was completed, the epoch with the highest sensitivity performance was selected as the optimized iteration. This ensures that although an excessive number of epochs were carried out if the model began overfitting the training data, the results would prioritize the earlier epochs where the test data showed the highest performance. This process was repeated three times and the average result was utilized for model

Table 5.12: CNN 1D Convolution Model Time-Series Convolution Varied Filter Quantity Performance

Conv. Layers	Time-Series Filters	Feature Filters	Output Filters	Sensitivity	Specificity	Epochs	Parameters
1	16	32	32	0.9764	0.9992	19	22,369
1	32	32	32	0.9776	0.9995	21	30,177
1	64	32	32	0.9792	0.9994	19	45,793
1	128	32	32	0.9831	0.9992	20	77,025
1	256	32	32	0.9803	0.9995	13	139,489
2	16	32	32	0.9803	0.9995	22	21,169
2	32	32	32	0.9851	0.9992	27	31,361
2	64	32	32	0.9835	0.9992	22	56,353
2	128	32	32	0.9835	0.9992	23	124,769
3	16	32	32	0.9788	0.9995	12	19,969
3	32	32	32	0.9866	0.9992	25	32,545
3	64	32	32	0.987	0.9994	26	66,913
3	128	32	32	0.9843	0.9996	14	172,513

Table 5.13: CNN 1D Convolution Model Varied Feature Convolution Filter Quantity Performance

Conv. Layers	Time-Series Filters	Feature Filters	Output Filters	Sensitivity	Specificity	Epochs	Parameters
1	128	16	32	0.9862	0.9996	26	39,057
1	128	32	32	0.9803	0.9993	11	77,025
1	128	64	32	0.9784	0.999	14	152,961
1	128	128	32	0.9827	0.9993	25	304,833
2	32	16	32	0.9744	0.9996	19	17,457
2	32	32	32	0.9862	0.9995	23	31,361
2	32	64	32	0.9772	0.9993	11	59,169
2	32	128	32	0.9807	0.9992	15	114,785
3	64	16	32	0.9855	0.999	27	46,353
3	64	32	32	0.9831	0.9995	22	66,913
3	64	64	32	0.9799	0.9994	12	108,033
3	64	128	32	0.9831	0.9992	14	190,273

performance comparisons.

5.3 Model Performance

The performance of the optimized CNN model architectures presented in this work is compared to common basic architectures, LVQ and SVM, and several simple RNN models typically utilized for time-series datasets, LSTM and GRU. The results

Table 5.14: CNN 1D Convolution Model Varied Output Convolution Filter Quantity Performance

Conv. Layers	Time-Series Filters	Feature Filters	Output Filters	Sensitivity	Specificity	Epochs	Parameters
1	128	16	16	0.9827	0.9994	18	35,441
1	128	16	32	0.9788	0.9991	18	39,057
1	128	16	64	0.9772	0.9993	10	46,289
1	128	16	128	0.9858	0.9994	25	60,753
2	32	32	16	0.9803	0.9993	33	25,185
2	32	32	32	0.9799	0.9996	21	31,361
2	32	32	64	0.9768	0.9994	16	43,713
2	32	32	128	0.9855	0.9995	26	68,417
3	64	32	16	0.9855	0.9994	19	61,761
3	64	32	32	0.978	0.9995	10	66,913
3	64	32	64	0.9819	0.9995	11	77,217
3	64	32	128	0.9831	0.999	16	97,825

Table 5.15: CNN 1D Convolution Model Optimized Performance

Conv. Layers	Time-Series Filters	Feature Filters	Output Filters	Sensitivity	Specificity	Epochs	Parameters
1	128	16	32	0.9835	0.9994	40	39,057
1	128	16	32	0.9851	0.9996	40	39,057
1	128	16	32	0.9843	0.9995	40	39,057
			Average	0.9843	0.9995	40	39,057
2	32	32	32	0.9843	0.9992	40	31,361
2	32	32	32	0.9847	0.9994	40	31,361
2	32	32	32	0.9835	0.9997	40	31,361
			Average	0.9842	0.9994	40	31,361
3	64	32	32	0.9851	0.9993	40	66,913
3	64	32	32	0.9858	0.9993	40	66,913
3	64	32	32	0.9878	0.9991	40	66,913
			Average	0.9862	0.9992	40	66,913

of these tests are listed in Table 5.16.

Table 5.16: Model Structure Comparison

Model	Sensitivity	Specificity	Input Data Windows	Total Time [s]	Inference Time [s]
LVQ	0.7102	0.9933	26,250	0.0277	1.0552×10^{-5}
SVM	0.9752	0.9987	26,250	3.4922	1.3304×10^{-4}
LSTM	0.9788	0.9996	26,250	4.9260	1.8766×10^{-4}
GRU	0.9784	0.9995	26,250	6.5790	2.5063×10^{-4}
2D CNN	0.9815	0.9995	26,250	19.6206	7.4745×10^{-4}
3-Layer 1D CNN	0.9878	0.9991	26,250	10.2900	3.9200×10^{-4}
1-Layer 1D CNN	0.9862	0.9996	26,250	5.1799	1.9733×10^{-4}

LVQs demonstrated the worst sensitivity performance compared to all other models. However, this architecture had a significantly reduced inference speed. Although LVQs are often considered more interpretable and require fewer parameters to tune than other architectures, these are not inherently optimized for time-series data. LVQs operate on fixed-length feature vectors and do not have the ability to capture complex spatial patterns within the data. These models, treat each data point independently without considering the temporal ordering or sequence. Therefore, explicit feature engineering would be required to increase the performance of LVQs and effectively extract relevant information from the data.

The attributes described for LVQs also apply to SVMs when reviewing their general performance with time-series data classification. Both model structures are also not structured to accommodate the high-dimensional data of the time-series data sets with multiple time steps and features. This data structure makes the training process for LVQ and SVM models more challenging.

As expected, the RNN models, LSTM and GRU, performed well with the application's dataset. As these models were specifically developed for use in time-series data classification, they show some of the best results for both sensitivity and specificity, with exceptional inference speeds due to the shallow architectures.

The optimized CNN architectures outperformed the other architectures in overall accuracy. As expected, the 3-layer 1D convolution CNN model demonstrated the highest average sensitivity. As CNNs are designed to learn hierarchical features through the convolutional layers, with the lower layers capturing simple patterns like edges and textures and the higher layers learning more abstract and complex features, the increased depth of the CNN increased the model's ability to consistently extract complex hidden features from the time-series data window.

However, the added depth increased the computational burden on the CPU image used for testing these models, increasing the inference time of both the 2D CNN and 2-Layer 1D kernel CNN model. As noted, the resource utilized to carry out these tests has a high-performance CPU and is not a GPU as typically implemented with CNN models. GPUs excel at the fast matrix and vector multiplications required for CNN training, where they can speed up learning by a factor of 50 and more [71]. Therefore, if the GPU specifications of the computational resource were to be increased, the deeper CNN model inference time would increase. Further increasing the performance of the architecture in these applications.

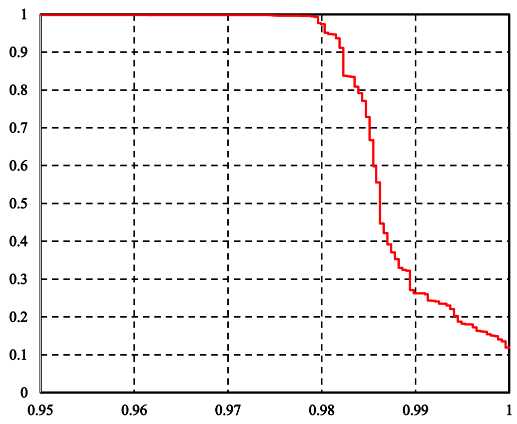
The 1-layer 1D kernel convolution CNN model demonstrated the highest overall performance. The exceptional sensitivity, specificity and inference times make it the best fit for this application.

The inconsistency between the multiple iterations of these CNNs demonstrates the significance of the initialization and training process of these models.

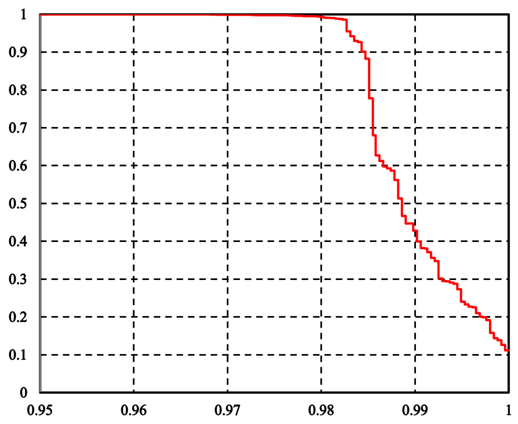
The PRCs of the top-performing models were plotted and are shown in Figure 5.3. The optimal model will produce a PRC with as much area under the curve as possible, indicating that the model is accurately predicting the fault-positive samples and minimizing the false fault-positive and false fault-negative predictions. As expected, based on prior results, the 3-layer 1D convolution CNN PRC demonstrates the model's accuracy in identifying faults while producing the least number of false classifications.

5.3.1 Resiliency

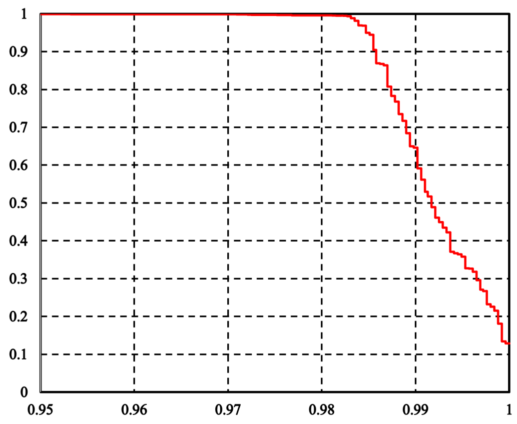
As these systems may be susceptible to noise, noise was injected to demonstrate the 1-layer 1D kernel convolution CNN model's resiliency and ability to generalize. The noise was injected into the dataset after the feature engineering stage, as the feature engineering stage could include filters to remove noise from the original dataset. More specifically, after the model was trained on the normalized training dataset, the normalized test dataset was altered by adding random values drawn from a normal distribution with mean zero and standard deviation of 1 to each data point in the test dataset. As shown in Table 5.17, the magnitude of the noise was altered to determine the resiliency of the model.



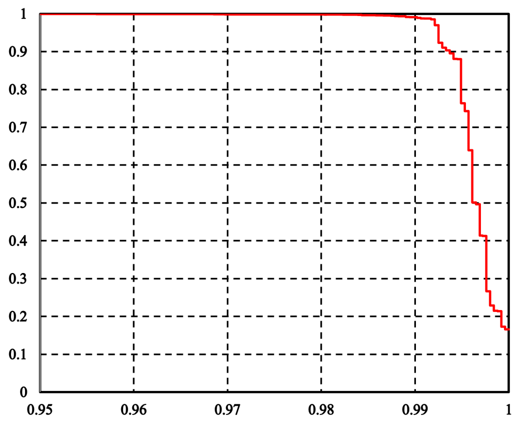
(a) Basic LSTM



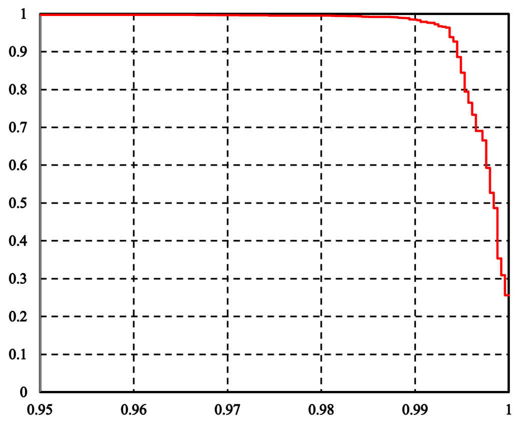
(b) Basic GRU



(c) 3-Layer 2D Convolution CNN



(d) 1-Layer 1D Convolution CNN



(e) 3-Layer 1D Convolution CNN

Figure 5.3: Model Precision-Recall Curves

The results in Table 5.17 demonstrate that the model's sensitivity is resilient to a significant amount of induced noise, indicating that the model maintains its ability to recognize fault conditions. However, the resilience of the precision is fairly poor as the number of false positive cases drastically increases with the introduction of noise.

Table 5.17: Model Resiliency

Noise Level [%]	**Accuracy**	**Sensitivity**	**Specificity**	**Precision**
0	0.9983	0.9862	0.9996	0.9964
1	0.9765	0.9843	0.9757	0.8127
2	0.941	0.9831	0.9364	0.6239
5	0.908	0.978	0.9005	0.5131
10	0.8817	0.9756	0.8716	0.449
15	0.8557	0.976	0.8428	0.3997
20	0.8413	0.9626	0.8283	0.3755

Although generalization techniques such as batch normalization have been introduced into the model architecture, additional steps can be taken to improve the precision of the model if noise is prevalent. The generalization and stability of the model can be improved through techniques including the introduction of augmented data with noise into the training dataset and regularization techniques such as dropout.

5.4 Summary

This chapter presents a comprehensive summary of the iterative optimization and testing process for various CNN model structures. Each CNN model structure is fine-tuned, taking into account different combinations of layers, hyperparameters and input data preprocessing techniques. This iterative process ensures the exploration of a wide range of configurations to identify the most effective model architecture for time-series data analysis in power systems.

The comparison of the optimized CNN structures with traditional LSTM, GRU, LVQ, and SVM models reveals the superiority transient classification performance of the CNN models. The CNN models consistently outperformed the traditional models in sensitivity, demonstrating their ability to effectively capture complex patterns and temporal dependencies in time-series data.

One of the key findings is the drastic reduction in inference period achieved by the optimized CNN model compared to the deeper CNNs. Despite this reduction in computational time, the optimized CNN model maintains or even improves the sensitivity and specificity performance compared to the deeper CNN structures.

The proper training techniques adopted for the CNN models enable them to reach and select optimal training epochs and converge to robust solutions. Overall, this chapter validates the effectiveness of the optimized CNN model structures for time-series transient data analysis. Superior sensitivity performance over traditional LSTM, GRU, LVQ, and SVM models, coupled with the reduced inference time for the shallow CNN model, establishes the potential of CNNs in supporting power system monitoring and protection applications with real-time inference capabilities.

Chapter 6

Conclusion & Future Works

6.1 Summary

This thesis has successfully demonstrated the efficacy of optimized machine learning models for real-time time-series data inferences on CPU-based resources. The experiments confirmed that large and expensive GPUs are not necessary for this application, as the models produced reliable and effective results on standard CPUs. The fastest inference time per input sample was $5.40 \times 10^{-5}s$, therefore these inferences can be made faster than the time it takes to retrieve the next data window, showing that they can be continually monitoring systems and acting in real-time. As this is well within transient and sub-transient time periods, this demonstrates the model's ability to be reliably implemented alongside power system protection and control hardware.

The development and optimization of a 1D convolutional kernel neural network (CNN) model showcased its superiority over traditional recurrent neural network (RNN) models for time-series data classification. By analyzing a time-series data window, the CNN model can utilize the architecture's inherent ability to recognize hidden features while considering the temporal transition of each input feature set through the data window. The optimized CNN's performance outperformed the LSTM and GRU model architectures in sensitivity, precision and inference time.

Potential future applications include built-in inverter protection. As the inverter characteristics play a key role in the system dynamics during anomalies, incorporating these models into the inverter themselves will allow a consistent data set to be developed for model training - mitigating the requirement for site-specific analysis.

6.2 Thesis Contributions

This thesis has demonstrated the effectiveness of optimized machine learning models for real-time time-series data analysis, showcasing their potential to improve power system protection and control infrastructure. The optimized 1D convolution CNN model's performance has surpassed that of traditional RNN models, making it a valuable addition to the field of power system protection. The potential future applications, such as built-in inverter protection and control, further expand the utility and practicality of these ML models in advancing power system stability and reliability. As the energy landscape continues to evolve with the integration of renewable resources, the contributions of this research hold immense promise for shaping a more robust and efficient power grid of the future.

6.3 Future Work

In light of the results obtained from this research, there are several avenues for future work that can further enhance the application and deployment of these models in power system protection and control. Key areas for future investigation include:

1. Further characterization of faults and anomalies in power distribution networks. By structuring the ML models to expand on the binary classification utilized in this work the model can provide further insights to the end user. This work should go as far as outlining the limitations of the model and its application in identifying conditions such as high-impedance faults and remote islanding.

2. The development of a real-time test bench that emulates existing devices and hardware used in power distribution systems. Ensuring the seamless interoperability of ML models with existing protection devices is essential for practical implemen-

tation. Constructing this system in a lab setting will enable the practical implementation and testing of machine learning models in real-world scenarios, providing validation of model performance under actual operating conditions and providing further insight into their integration into existing infrastructure. Research can focus on integration aspects such as communication protocols and interfaces to enable ML models to interact with protection hardware effectively.

3. The evaluation of ML models' performance on various power distribution network structures and characteristics with different grounding sources. This review is crucial for the broader applicability of the ML models. Investigating the effectiveness of supplementary layers, such as dropout, in handling network structural variations will help enhance model generalization and robustness.

4. The incorporation of real-world data sets from multiple power distribution networks with diverse operational conditions and renewable energy integration while further exploring data normalization and preprocessing techniques. This will aid in optimizing the utilization of raw data for the application. Investigating methods like batch normalization, feature scaling, and time-series normalization can enhance model training and improve the overall performance of ML models. Real-world data will further validate the models' effectiveness in practical applications. This will enable the identification of potential challenges and fine-tuning of ML models for real-world deployments.

In conclusion, work in this field holds significant potential to advance power system protection and control through the incorporation of machine learning models. By developing real-time test benches, further characterizing system transient conditions, evaluating models on diverse network structures, optimizing data preprocessing and considering deployment constraints the stability and reliability of power distribution systems with inverter-based DERs can be further enhanced.

Appendix A

PSCAD Network Parameters

Table A.1: Utility Three-Phase Source Parameters

Variable	**Value**	**Units**
Base MVA	12	[MVA]
Base Voltage (L-L RMS)	24.9	[kV]
Base Frequency	60	[Hz]
Voltage Input Time Constant	0.05	[sec]
Infinite Bus Ideal	No	-
Neg. Seq. Differs from Pos. Seq.	No	-
Pos. Seq. Impedance	1	[ohm]
Pos. Seq. Phase Angle	80	[deg]
Pos. Seq. Resistance	0.16038	[ohm]
Pos. Seq. Reactance	0.64151	[ohm]
Zero Seq. Impedance	1	[ohm]
Zero Seq. Phase Angle	80	[deg]
Zero Seq. Resistance	0.16977	[ohm]
Zero Seq. Reactance	0.50932	[ohm]

Table A.2: PV Power Plant Controller Parameters

Variable	**Value**	**Units**
System Frequency	60	[Hz]
Base Voltage (L-L RMS)	24.9	[kV]
Base Inverter Voltage (L-L RMS)	0.65	[kV]
Base DC Voltage	1	[kV]
Base Leakage Inductance of the Inverter	0.1	[pu]
Real Power Ramp Rate	1	[pu/s]
Reactive Power Ramp Rate	1	[pu/s]
Priority	P	-
Enable VRT Q Offset	Disable	-
VRT Deadband	0.2	[pu]
Slope of Iq Curve Outside of Deadband	2.5	[pu/pu]
Enable P(f) Mode	Disable	-
P(f) Deadband	0.4	[Hz]
Slope of P Curve Outside of Deadband	0.45	[pu/Hz]
P(f) Ramp Rate	0.5	[pu/s]

Table A.3: Solar Farm Parameters

Variable	**Value**	**Units**
Reactive Power Control Mode	POC Voltage	-
HVRT Detection Threshold	1.15	[pu]
LVRT Detection Threshold	0.85	[pu]
Pmax	1	[pu]
Pmin	0	[pu]
Qmax	0.6	[pu]
Qmin	-0.6	[pu]

Table A.4: PV Interconnection Transformer Parameters

Variable	**Value**	**Units**
Transformer Capacity	10	[MVA]
Base Operation Frequency	60	[Hz]
Winding #1 Type	WYE	-
Winding #2 Type	WYE	-
Positive Sequence Leakage Reactance	0.025	[pu]
Ideal Transformer Model	No	-
Eddy Current Losses	0	[pu]
Copper Losses	0.0001	[pu]
Tap Changer on Winding	None	-
Winding 1 Line to Line voltage (RMS)	24.9	[kV]
Winding 2 Line to Line voltage (RMS)	24.9	[kV]
Saturation Enabled	No	-
Place Saturation on Winding	Middle	-
Hysteresis	None	-
Inrush Decay Time Constant	0	[s]
Time to Release Flux Clipping	0	[s]
Air Core Reactance	0.2	[pu]
Magnetizing Current	2	[%]
Knee Voltage	1.17	[pu]

References

[1] G. Kou, L. Chen, P. Van Sant, and F. Velez-Cedeno, "Fault Characteristics of Distributed Solar Generation," *IEEE Transactions on Power Delivery*, vol. 35, pp. 1062–1064, Mar. 2019.

[2] S. Afrasiabi, M. Afrasiabi, B. Parang, and M. Mohammadi, "Integration of Accelerated Deep Neural Network Into Power Transformer Differential Protection," *IEEE Transactions on Industrial Informatics*, vol. 16, pp. 865–876, Feb. 2020.

[3] "IEEE Guide for Conducting Distribution Impact Studies for Distributed Resource Interconnection," *IEEE Std 1547.7-2013*, pp. 1–137, 2014.

[4] IEEE PES Industry Technical Support Task Force, "Impact of Inverter Based Generation on Bulk Power System Dynamics and Short-Circuit Performance," Technical Report PES-TR68, IEEE Power & Energy Society, Piscataway, NJ, USA, July 2018.

[5] X. Shi, T. Key, D. Van Zandt, and W. Wang, "Effective Grounding for Inverter-Connected DER," Final Report 3002020130, Electric Power Research Institute, 2021.

[6] "IEEE Standard for Interconnection and Interoperability of Distributed Energy Resources with Associated Electric Power Systems Interfaces," *IEEE Std 1547-2018 (Revision of IEEE Std 1547-2003)*, pp. 1–138, 2018.

[7] A. Haddadi, E. Farantatos, I. Kocar, and U. Karaagac, "Impact of Inverter Based Resources on System Protection," *Energies*, vol. 14(4), Feb. 2021.

[8] H. Zhang, "DER Effective Grounding Screening and Study Methodology, TOV and TRV Study Methodology," White Paper, PS Technologies Inc., Nov. 2020.

[9] J. Gracia, A. Mazon, and I. Zamora, "Best ANN Structures for Fault Location in Single-and Double-Circuit Transmission Lines," *IEEE Transactions on Power Delivery*, vol. 20, no. 4, pp. 2389–2395, 2005.

[10] R. C. Santos, S. Le Blond, D. V. Coury, and R. K. Aggarwal, "An Intelligent Backup Scheme for Current Source Converter High Voltage Direct Current Systems Based on Artificial Neural Networks," *Electric Power Components and Systems*, vol. 45, no. 17, pp. 1892–1904, 2017.

[11] D.-E. Kim and D.-C. Lee, "Fault Diagnosis of Three-Phase PWM Inverters using Wavelet and SVM," in *2008 IEEE International Symposium on Industrial Electronics*, pp. 329–334, IEEE, 2008.

[12] T. Villmann, A. Bohnsack, and M. Kaden, "Can Learning Vector Quantization be an Alternative to SVM and Deep Learning? Recent Trends and Advanced Variants of Learning Vector Quantization for Classification Learning," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 7, no. 1, pp. 65–81, 2017.

[13] G. Kumar, S. Sharma, and H. Malik, "Learning Vector Quantization Neural Network Based External Fault Diagnosis Model for Three Phase Induction Motor Using Current Signature Analysis," *Procedia Computer Science*, vol. 93, pp. 1010–1016, 2016. Proceedings of the 6th International Conference on Advances in Computing and Communications.

[14] J. Liu, Y. Liang, and X. Sun, "Application of Learning Vector Quantization Network in Fault Diagnosis of Power Transformer," in *2009 International Conference on Mechatronics and Automation*, pp. 4435–4439, 2009.

[15] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation*, vol. 9, pp. 1735–80, 12 1997.

[16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," 2014.

[17] C. Ma, J. Wang, *et al.*, "Model Reduction with Memory and the Machine Learning of Dynamical Systems," *Communications in Computational Physics*, vol. 25, no. 4, pp. 947–962, 2018.

[18] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," 2014.

[19] S. Belagoune, N. Bali, A. Bakdi, B. Baadji, and K. Atif, "Deep Learning through LSTM Classification and Regression for Transmission Line Fault Detection, Diagnosis and Location in Large-Scale Multi-Machine Power Systems," *Measurement*, vol. 177, p. 109330, 2021.

[20] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2019.

[21] I. Gogul and V. S. Kumar, "Flower Species Recognition System Using Convolution Neural Networks and Transfer Learning," in *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, pp. 1–6, 2017.

[22] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu, "Time Series Classification With Multivariate Convolutional Neural Network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4788–4797, 2019.

[23] S. Zhang, R. Wang, Y. Si, and L. Wang, "An Improved Convolutional Neural Network for Three-Phase Inverter Fault Diagnosis," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–15, 2022.

[24] S. K. Azman, Y. J. Isbeih, M. S. E. Moursi, and K. Elbassioni, "A Unified Online

Deep Learning Prediction Model for Small Signal and Transient Stability," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4585–4598, 2020.

[25] W. Lu, B. Liang, Y. Cheng, D. Meng, J. Yang, and T. Zhang, "Deep Model Based Domain Adaptation for Fault Diagnosis," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 3, pp. 2296–2305, 2017.

[26] S. Zhang, Z. Zhu, and Y. Li, "A Critical Review of Data-Driven Transient Stability Assessment of Power Systems: Principles, Prospects and Challenges," *Energies*, vol. 14, no. 21, 2021.

[27] T. Kohonen, "Improved Versions of Learning Vector Quantization," in *1990 International Joint Conference on Neural Networks (INCNN)*, pp. 545–550, IEEE, 1990.

[28] T. Kohonen and T. Kohonen, "Learning Vector Quantization," *Self-Organizing Maps*, pp. 175–189, 1995.

[29] V. Vapnik, "An Overview of Statistical Learning Theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.

[30] D. Sebald and J. Bucklew, "Support Vector Machine Techniques for Nonlinear Equalization," *IEEE Transactions on Signal Processing*, vol. 48, no. 11, pp. 3217–3226, 2000.

[31] L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada, *Artificial Intelligence and Soft Computing: 21st International Conference, ICAISC 2022, Zakopane, Poland, June 19–23, 2022, Proceedings, Part I*, vol. 13588. Springer Nature, 2023.

[32] F. Karim, S. Majumdar, and H. Darabi, "Insights Into LSTM Fully Convolutional Networks for Time Series Classification," *IEEE Access*, vol. 7, pp. 67718–67725, 2019.

[33] H. Mhaskar, Q. Liao, and T. Poggio, "Learning Functions: When Is Deep Better Than Shallow," *CBMM Memos*, 2016.

[34] W. Zhang, C. Li, G. Peng, Y. Chen, and Z. Zhang, "A Deep Convolutional Neural Network with New Training Methods for Bearing Fault Diagnosis Under Noisy Environment and Different Working Load," *Mechanical Systems and Signal Processing*, vol. 100, pp. 439–453, 2018.

[35] Z. Dai and R. Heckel, "Channel Normalization in Convolutional Neural Network Avoids Vanishing Gradients," 2019.

[36] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International Conference on Machine Learning*, pp. 448–456, PMLR, 2015.

[37] H. Ide and T. Kurita, "Improvement of Learning for CNN with ReLU Activation by Sparse Regularization," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2684–2691, IEEE, 2017.

[38] C. F. G. D. Santos and J. P. Papa, "Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks," *ACM Computing Surveys (CSUR)*, vol. 54, no. 10s, pp. 1–25, 2022.

[39] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How Does Batch Normalization Help Optimization?," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[40] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light Gated Recurrent Units for Speech Recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 92–102, 2018.

[41] S. Sharma, S. Sharma, and A. Athaiya, "Activation Functions in Neural Networks," *International Journal of Engineering Applied Sciences and Technology*, vol. 4, no. 12, pp. 310–316, 2020.

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[43] P. Rai, N. D. Londhe, and R. Raj, "Fault classification in power system distribution network integrated with distributed generators using CNN," *Electric Power Systems Research*, vol. 192, p. 106914, 2021.

[44] Srivastava, Nitish and Hinton, Geoffrey and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research*, vol. 15, p. 1929–1958, jan 2014.

[45] W. Sun, A. R. Paiva, P. Xu, A. Sundaram, and R. D. Braatz, "Fault Detection and Identification using Bayesian Recurrent Neural Networks," *Computers Chemical Engineering*, vol. 141, p. 106991, 2020.

[46] A. D. Rajapakse, D. Muthumuni, N. Perera, and K. Strunz, "Electromagnetic Transients Simulation for Renewable Energy Integration Studies," in *2007 IEEE Power Engineering Society General Meeting*, pp. 1–6, IEEE, 2007.

[47] K. P. Schneider, B. A. Mather, B. C. Pal, C.-W. Ten, G. J. Shirek, H. Zhu, J. C. Fuller, J. L. R. Pereira, L. F. Ochoa, L. R. de Araujo, R. C. Dugan, S. Matthias, S. Paudyal, T. E. McDermott, and W. Kersting, "Analytic Considerations and Design Basis for the IEEE Distribution Test Feeders," *IEEE Transactions on Power Systems*, vol. 33, no. 3, pp. 3181–3188, 2018.

[48] J. Sola and J. Sevilla, "Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems," *IEEE Transactions on Nuclear Science*, vol. 44, no. 3, pp. 1464–1468, 1997.

[49] M. Ghazavi, P. Mancarella, T. Saha, and R. Yan, "System Strength and Weak Grids: Fundamentals, Challenges, and Mitigation Strategies," in *2018 Aus-*

tralasian Universities Power Engineering Conference (AUPEC), pp. 1–7, IEEE, 2018.

[50] N. Nimpitiwan, G. T. Heydt, R. Ayyanar, and S. Suryanarayanan, "Fault Current Contribution From Synchronous Machine and Inverter Based Distributed Generators," *IEEE Transactions on Power Delivery*, vol. 22, no. 1, pp. 634–641, 2007.

[51] J. E. Caicedo, D. Agudelo-Martínez, E. Rivas-Trujillo, and J. Meyer, "A Systematic Review of Real-Time Detection and Classification of Power Quality Disturbances," *Protection and Control of Modern Power Systems*, vol. 8, no. 1, pp. 1–37, 2023.

[52] Schweitzer Engineering Laboratories, Inc., *SEL-751 Feeder Protection Relay: Comprehensive Feeder Protection and Control With Arc-Flash Detection*, 2023. PDS751-01, https://selinc.com/api/download/10734/.

[53] S. K. Bairwa and S. P. Singh, "Phasor Measurement Unit Application-Based Fault Allocation and Fault Classification," *International Journal of Advances in Applied Sciences*, vol. 12, no. 1, pp. 15–26, 2023.

[54] S. Shankar, K. B. Yadav, A. Priyadarshi, and V. Rathore, "Study of phasor measurement unit and its applications," in *Recent Advances in Power Systems* (O. H. Gupta and V. K. Sood, eds.), (Singapore), pp. 247–257, Springer Singapore, 2021.

[55] "IEEE Standard for Synchrophasor Measurements for Power Systems – Amendment 1: Modification of Selected Performance Requirements," *IEEE Std C37.118.1a-2014 (Amendment to IEEE Std C37.118.1-2011)*, pp. 1–25, 2014.

[56] O. A. Youssef, "Combined Fuzzy-Logic Wavelet-Based Fault Classification Technique for Power System Relaying," *IEEE Transactions on Power Delivery*, vol. 19, no. 2, pp. 582–589, 2004.

[57] Z. Zhang, Y. Cheng, and N. C. Liu, “Comparison of the effect of mean-based method and z-score for field normalization of citations at the level of Web of Science subject categories,” *Scientometrics*, vol. 101, no. 3, pp. 1679–1693, 2014.

[58] X. Glorot and Y. Bengio, “Understanding the Difficulty of Training Deep Feedforward Neural Networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.

[59] Q. Sun, Y. Wang, and Y. Jiang, “A Novel Fault Diagnostic Approach for DC-DC Converters Based on CSA-DBN,” *IEEE Access*, vol. 6, pp. 6273–6285, 2018.

[60] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” 2015.

[61] B. Zhang, Q. Zhao, W. Feng, and S. Lyu, “AlphaMEX: A Smarter Global Pooling Method for Convolutional Neural Networks,” *Neurocomputing*, vol. 321, pp. 36–48, 2018.

[62] M. Lin, Q. Chen, and S. Yan, “Network In Network,” 2014.

[63] C. Huang, F. Li, D. Zhou, J. Guo, Z. Pan, Y. Liu, and Y. Liu, “Data Quality Issues for Synchrophasor Applications Part I: A Review,” *Journal of Modern Power Systems and Clean Energy*, vol. 4, no. 3, pp. 342–352, 2016.

[64] A. Tharwat, “Classification Assessment Methods,” *Applied Computing and Informatics*, vol. 17, no. 1, pp. 168–192, 2020.

[65] T. Saito and M. Rehmsmeier, “The Precision-Recall Plot is More Informative than the ROC Plot when Evaluating Binary Classifiers on Imbalanced Datasets,” *PloS one*, vol. 10, no. 3, p. e0118432, 2015.

[66] Ž. Vujović *et al.*, “Classification Model Evaluation Metrics,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 599–606, 2021.

[67] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A Survey of Quantization Methods for Efficient Neural Network Inference," in *Low-Power Computer Vision*, pp. 291–326, Chapman and Hall/CRC, 2022.

[68] X. XIAO, Z. Wang, and S. Rajasekaran, "AutoPrune: Automatic Network Pruning by Regularizing Auxiliary Parameters," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[69] A. Gordon, E. Eban, O. Nachum, B. Chen, H. Wu, T.-J. Yang, and E. Choi, "Morphnet: Fast & simple resource-constrained structure learning of deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1586–1595, IEEE, 2018.

[70] A. Sato and K. Yamada, "Generalized Learning Vector Quantization," *Advances in Neural Information Processing Systems*, vol. 8, 1995.

[71] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.