TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PASHCHIMANCHAL CAMPUS, POKHARA

A PROJECT FINAL REPORT ON

An Intelligent Deep Convolutional Neural Network Based Islanding Detection for Multi-DG Systems

by

Basant Raj Tiwari

A FINAL PROJECT REPORT

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN ELECTRICAL ENGINEERING IN DISTRIBUTED GENERATION ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING

PASCHIMANCHAL CAMPUS

POKHARA, NEPAL

SEPTEMBER, 2024

An Intelligent Deep Convolutional Neural Network Based Islanding Detection for Multi-DG Systems

By

Basant Raj Tiwari

Er. Sushmita Poudel

Supervisor

Department of Electrical Engineering

Paschimanchal Campus, Tribhuvan University

Submitted to:

DISTRIBUTED GENERATION ENGINEERING PROGRAM

DEPARTMENT OF ELECTRICAL ENGINEERING

Institute of Engineering, Paschimanchal Campus

Tribhuvan University

Pokhara, Nepal

ACKNOWLEDGEMENT

I acknowledge my deep gratitude to my project supervisor Er. Sushmita Poudel and advisor Asst. Prof. Sandeep Dhami for the insightful lessons, guidance and inspiration for the completion of project. These efforts have greatly benefited from their encouragement, suggestions, and observations.

Additionally, I would like to express my gratitude to the Pashchimanchal Campus’s faculty members of the Department of Electrical Engineering for their insightful comments and suggestions during the project's many phases.

I would like to extend my gratefulness to Arif Hussain, Postdoctoral Researcher at Iowa State University for assisting me on the model development of microgrid and giving insights required for the successful completion of the project.

I extend my thanks to my colleagues who provided constant support throughout the study. I take opportunity to thank the authors and publishers of all the sources that helped in every step of this study.

ABSTRACT

Unintentional islanding poses a significant challenge in electrical distribution networks, particularly within non-detection zones. This study presents a novel intelligent islanding detection technique with zero non-detection zone, designed for hybrid distributed generation systems. The proposed method combines short-term Fourier transform for frequency spectrum analysis and convolutional neural networks (CNNs) for pattern recognition. The approach involves monitoring three-phase voltage at the point of common coupling and collecting time-series data for various islanding and non-islanding events. These data undergo frequency computations on a scaled time-series, with complex numbers separated into magnitude and phase components. A modified CNN with forward propagation is then employed to distinguish between islanding and non-islanding occurrences.

A test system is modeled that serves as the foundation for generating diverse scenarios to train the CNN. The model's performance is evaluated using 5-fold cross-validation. Results demonstrate that the proposed methodology achieves a zero non-detection zone with the original dataset, exhibiting high accuracy and selectivity among the islanding and non-islanding conditions.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
CB	Circuit Breaker
CNN	Convolutional Neural Network
DFT	Discrete Fourier Transform
DG	Distributed Generation
FC	Fully Connected
IDT	Islanding Detection Technique
NDZ	Non-Detection Zone
PCC	Point Of Common Coupling
STFT	Short-Term Fourier Transform
WT	Wavelet Transform
TTT	Time-Time Transform

CHAPTER ONE: INTRODUCTION

1.1 Background

The swift adoption of distributed generation (DG) technologies has garnered considerable attention due to factors like energy market deregulation, investment opportunities, the demand for reliable and high-quality power, and environmental considerations. While traditional power systems rely on centralized generation and transmission, DG integration offers various benefits but also presents challenges. One major concern is unintentional islanding, which occurs when a DG unit continues to operate independently after being disconnected from the main grid [1]. This can lead to safety issues, voltage problems, synchronization difficulties during reconnection, and equipment damage.

To address this, various international and local standards outline criteria for detecting islanding within a specific timeframe. Various islanding detection schemes (IDSs) have been introduced in the literature. Each type has its pros and cons based on the non-detection zone (NDZ), detection time, and PQ. The classification is based on how the techniques are being employed. Generally, IDT are categorized under local schemes, remote schemes and Intelligent Classifier Based Schemes.

Remote IDT employs communication infrastructure for the detection of islanding. A communication link is deployed between the DG and main utility, this communication link requires additional instruments. These instruments are generally high-cost sensors, telecommunication tools, and control systems. The remote IDS have comparatively higher system and running costs, compared to active and passive techniques [2]. Example of remote schemes include power line carrier communication (PLCC), supervisory control and data acquisition system (SCADA), signal produced disconnects (SPD) etc.

Local schemes are based on monitoring various electrical parameters such as voltage, current, frequency, and power in addition to the injection of disturbances in DG-EPS for islanding detection. IDS are further categorized as active, passive, and hybrid islanding detection schemes. Active IDSs utilize external disturbances by injecting a troubling signal into DG output, this external signal injection introduces variation in system parameters. By calculating the variation in system parameters with thresholds, active methods detect islanding [2]. Active frequency drift (AFD), Sandia Frequency Shift (SFS), Sandia Voltage Shift (SVS) are the common methods of active IDTs [3].

Passive islanding is another commonly used IDT for DG-EPS, where system parameters are monitored at the point of common coupling (PCC), which shows variations when the utility is isolated from the DG-EPS. Based on these variations, the protective relay operates to detect islanding by comparing it with the defined threshold values. Passive IDTs are economical and uncomplicated schemes that pose no harm for PQ, thus considered as realistic solutions for DG-EPS. The passive IDTs used in common are over/under voltage and frequency methods (O/U V&F), the rate of change of frequency/power (ROCOF/P), total harmonic distortion method (THD), and phase jump detection (PJD) methods.

With bugs and pitfalls in local and remote schemes, intelligent classifier-based schemes are gaining more attention. The key reasons for shifting towards intelligent classifier-based IDS are the exemption from threshold settings, no noise and PQ problems, low NDZ, high speed, and no communication channel intervention, which make intelligent schemes more reliable and acceptable. The most commonly used intelligent-classifier-based schemes for islanding detection are decision trees (DTs), artificial neural networks (ANNs), support vector machines (SVMs), fuzzy logic (FL), adaptive neuro-fuzzy inference systems (ANFISs), convolution neural networks (CNNs), and deep neural networks (DNNs). [2].

1.2 Problem Statement

Researchers are continually developing new methods to detect islanding, which is when a part of the power grid becomes unintentionally isolated. These methods fall into three main types: local, remote, and intelligent. Local methods can be further categorized as active, passive, or hybrid.

Active methods work by introducing small disturbances into the system to trigger detection, but these disturbances can sometimes cause problems with noise and power quality. Passive methods, on the other hand, simply monitor the system for natural changes that indicate islanding has occurred. However, these methods may not always detect islanding and can be slow to react. Hybrid methods try to combine the best aspects of both active and passive methods, but they can be more difficult to put into practice and may take longer to detect islanding. Remote methods rely on communication between the distributed generation unit and the utility company to detect islanding. While these methods are very reliable, they are also complex and expensive, making them unsuitable for smaller power systems.

1.3 Objectives

- To study the efficacy of proposed CNN based IDT for various islanding scenarios comprising detection around non-detection zones and selectivity against non-islanding scenes.
- To study the Signal Processing Techniques like DFT, STFT and Spectrogram.

1.4 Scope and Limitations

This work stands as one of the other alternate islanding detection techniques which is highly accurate, almost has zero NDZ and quick detection time. This work can be used for IDT in PV DG power system applications. The project is more of assessment and verification of application of CNN in detection of islanding detection.
While the project successfully assesses that indeed CNN based IDT can be used for IDT, but the project has not been tested on real world systems. The real-world systems possess several kinds of noises and other disturbances included in the voltage signal which can change the performance of the trained neural network. The models for DG units and distribution networks may not fully represent the complexity of real-world systems so one has to incorporate these limitations by training the model on the real-world systems.

Also, the project trains the CNN model on 2100 set of images which may not be sufficient for classifying all islanding scenarios. More the number of datasets the model is trained on, higher is the chance of true classification on different scenarios. The project also doesn't consider the scenarios of capacitor switching as it also causes the transients in the voltage signals which can cause the misclassification.

1.5 Report Organization

The first chapter deals with a brief introduction of the project background, problem statement, objectives, scope and limitation and report organization. In the second chapter, the brief of review of different literature during the study regarding the project is presented. The third chapter provides description of the methodology followed in the course of the study in brief**.** In the fourth chapter, the results obtained are presented and discussed in detail. The fifth chapter presents conclusions of the study and recommendations for the further additions that can be done in the study.

CHAPTER TWO: LITERATURE REVIEW

2.1 Short-Term Fourier Transform Spectrogram

Signal processing techniques are crucial for islanding detection, as they can reveal hidden characteristics within measured signals that aid in the detection process. Techniques like STFT, WT, ST, HST, and TTT are particularly useful for improving passive detection methods [3], [4], [5], [6], [7] especially in reducing the non-detection zone (NDZ). While Fourier transforms are commonly used for frequency analysis, they don't capture the temporal distribution of frequencies, making them unsuitable for detecting transient changes. Therefore, Short-Time Fourier Transform (STFT) is employed for time-frequency analysis in this research. STFT breaks down a signal into smaller, potentially stationary segments and analyzes them using a moving window, showing the relationship between time and frequency. This process involves dividing discretely sampled data into chunks, performing Fourier transforms on each chunk to obtain frequency spectra, and then calculating the phase and magnitude for each spectrum. The resulting spectra are visually represented as a spectrogram, where power

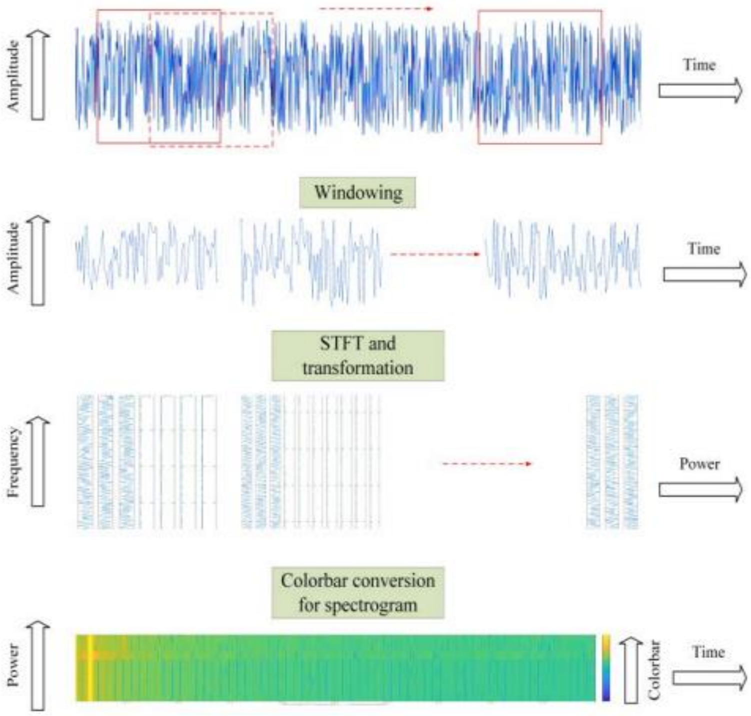


Figure 2.1: Short-Term Fourier Transform Working.[11]

spectral density is indicated by a color bar, revealing the signal's strength over time and frequency in Figure 2.1.

The Short-Time Fourier Transform (STFT) is fundamentally a series of discrete Fourier transforms (DFTs) calculated over short segments of a signal, which may or may not overlap. For a discrete-time signal x[n], this can be represented mathematically as in equation 2.1.

$$X(m, \omega) = \sum_{n=-\infty}^{\infty} x[n]\omega[n-m]e^{-j\omega n} \quad [2.1]$$

Where x[n] is the input signal to be transformed and w[n] is the window function. The squared magnitude of STFT results in a spectrogram representing the power spectral density of the function. According to the uncertainty principle, the time-frequency resolution of the resulting spectrum is determined by the window size used. A bigger size yields higher spectral, but lower temporal, resolution, whereas a smaller size yields the opposite.

2.2 Convolutional Neural Network

The CNN is a well-known deep learning architecture influenced by the automatic visual experience of living beings. CNNs, in contrast, use end-to-end learning, which means that the classifier discovers all the features and then classifies the images, resulting in a data-driven method. A typical CNN model is a combination of a few main layers. The convolutions layer constructs a feature map to predict class probabilities.[11]. The batch normalization (BN) layer standardizes the inputs to a layer to avoid internal covariate shift. After calculating a nonlinear function of the input, a rectified linear unit (ReLU), activates the specific neuron. Pooling layers decrease the amount of information by retaining the most relevant information created by the convolution layer. Fully connected (FC) layers perform the following tasks:

- The FC input layer can flatten output generated by the previous layer,
- The FC layers apply weights to simulate precise labels, and
- The FC output layer produces final probabilities.

Finally, the SoftMax layer measures the class likelihood for the input sample, which is mostly employed in combination with the cross-entropy (CE) loss function. Equation 2.2 are the mathematical description of five layers of CNN [10].

Convolutional Layer

$$g_j^l = x_j^{l-1}(s,t) \times w_{ij}^l$$
$$g = \sum_{\sigma=-n1}^{n1} \sum_{\sigma=-n2}^{n2} x_j^{l-1}(s-\sigma, t-v) w_{ij}^l(\sigma, v)$$

Activation or ReLU layer:

$$x_j^l = \max\left(0, \sum_{i \in Mj} g_j^l + b_j^l\right)$$

Max Pooling (MP) layer:

$$x_j^{l+1} = f_p\left(b_j^{l+1}\left(x_j^l\right) + b_j^{l+1}\right)$$

Fully connected (FC) layer:

$$x^{L-1} = f_c(\beta^{L-1} x^{L-2} + b^{L-1})$$

SoftMax layer:

$$z_d = \frac{e^{o_d}}{\sum_{c=1}^{C} e^{o_c}} \rightarrow \frac{e^{x_d^{L-1}}}{\sum_{c=1}^{C} e^{x_c^{L-1}}} \qquad [2.2]$$

CNN calculates the divergence between the real distribution and the distribution created by the model using cross-entropy loss (Y) [9]. Y is computed using equation 2.3

$$Y(y,z) = -\sum_d y_d \log\left(z_d\right) \qquad [2.3]$$

CHAPTER THREE: METHODOLOGY

The proposed islanding detection technique (IDT) utilizes CNN and STFT for time-frequency analysis, following a four-stage process. First, three-phase voltage data is collected at the point of common coupling (PCC) under various islanding and non-islanding conditions. Second, the three-phase data is combined into a single voltage array. Third, STFT is applied to transform the voltage data into time-frequency representations. Finally, the resulting numeric data, representing phase and magnitude of the frequency spectrum, is fed into a CNN for feature extraction and classification of islanding and non-islanding events. The proposed IDT is based on CNN and STFT computation for time-frequency analysis; the stages of the proposed method are summarized as the flowchart in Figure 3.1.

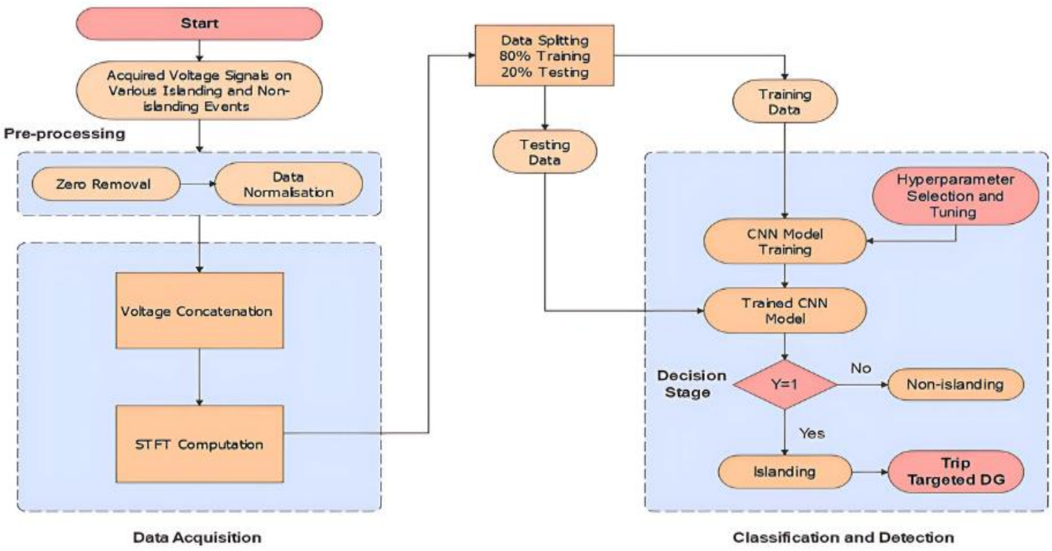


Figure 3.1: Flowchart of Proposed Scheme [11]

The CNN architecture is optimized for islanding detection with carefully chosen layers, filter sizes, and training hyperparameters to maximize accuracy and efficiency. A three-layer architecture is selected, and the Stochastic Gradient Descent with Momentum (SGDM) optimizer with a learning rate of 0.01 is used. K-fold cross-validation with k=5 is also implemented. In K-fold cross-validation we split the dataset into K equal subsets or "folds". Then we train the model K times, each time using K-1 folds for training and 1-fold for validation. Finally, we average the performance across all K iterations.

The SGDM optimizer estimates the error gradient between the current model output and the desired output, updating the model weights accordingly. The learning rate controls the amount of error used to update the weights, and it's carefully chosen to ensure optimal convergence speed and accuracy.

Also, confusion matrix is evaluated to assess the performance of a classification model. It shows the actual vs. predicted classifications, providing a comprehensive view of the model's accuracy.

3.1 System Under Study

The test system consists of several DG units with both synchronous and inverter-based DG as shown in Figure 3.2. Zone 1 wraps all DGs and when the CB1 is open the DG system goes into islanded mode when CB1 is open. The zone 2 consists of synchronous based DG only, while zone 3 has two inverter-based DGs and a synchronous DG. A zone 4 has inverter-based DG only and is islanded using CB4. The loads are realized using the RLC loads in Simulink. Various voltage measurements at PCCs have been done and logged to perform the required computation for generating spectrograms. The model was created in MATLAB/Simulink R2021a based on the SLD in Figure 3.2

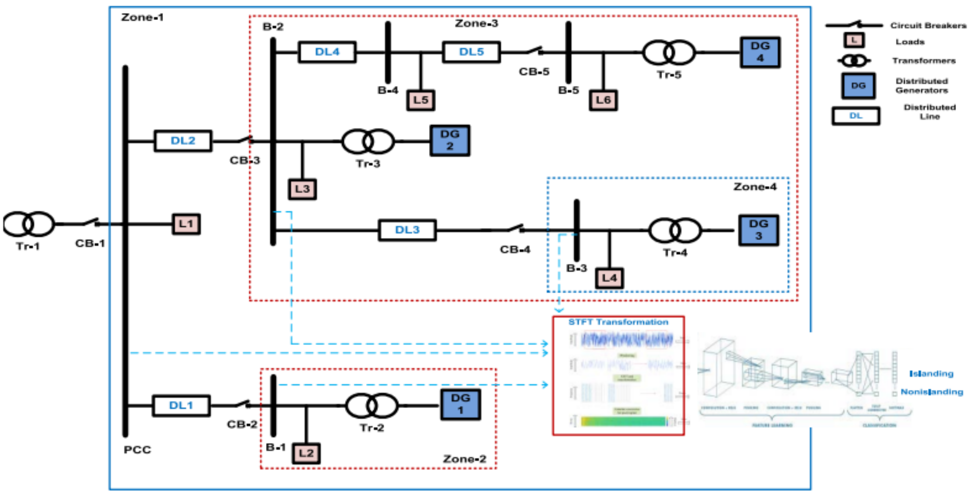


Figure 3.2: Single Line Diagram for test system [11]

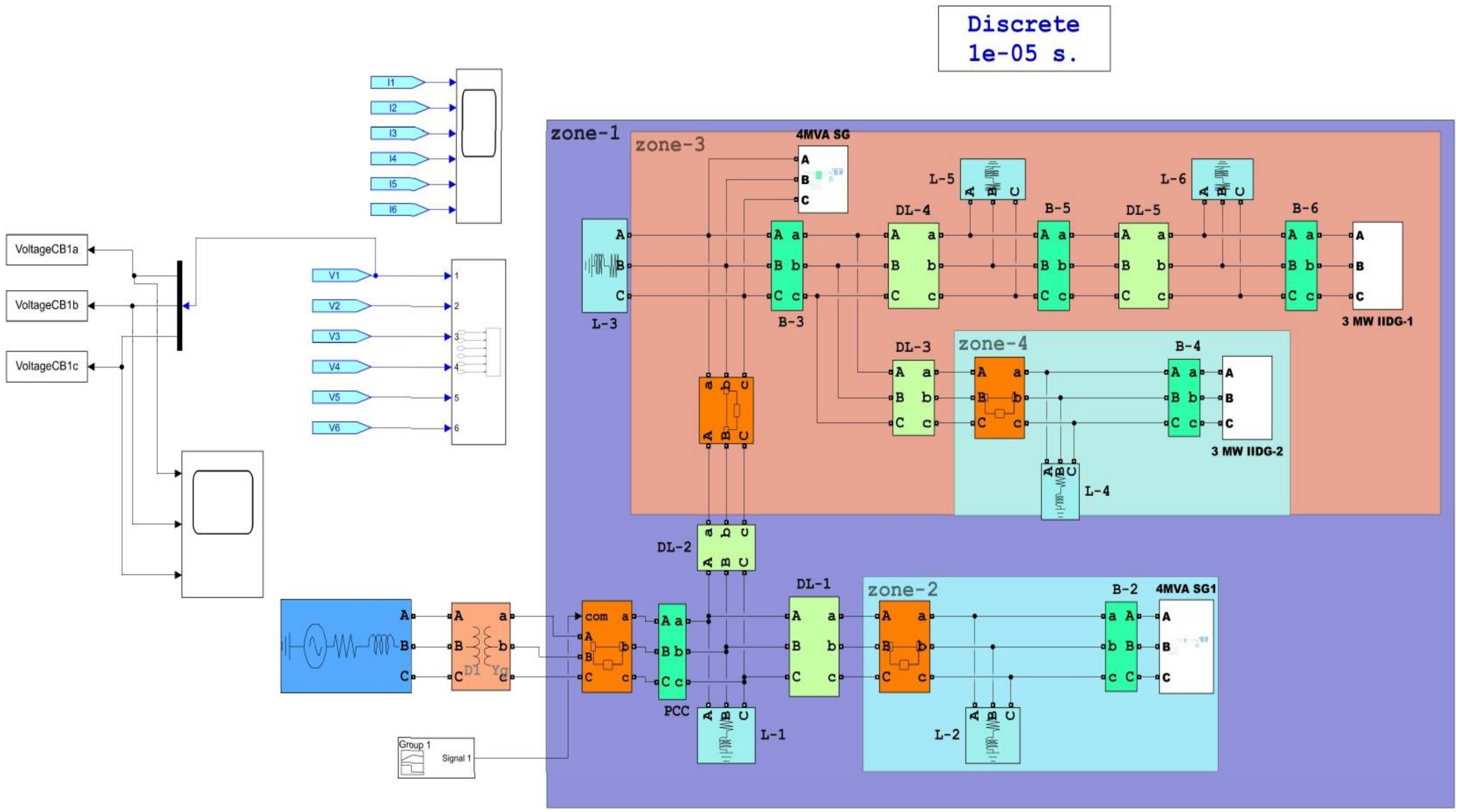


Figure 3.3: Test System for the Proposed Scheme

The power and voltage ratings for DGs and transformers used in the test system is tabulated in Table 3.1.

Table 3.1: DGs and transformers data for test system.

DG	Power Rating	Voltage Rating
Synchronous based DGs (1&2)	4 MVA	2.4kV
Inverter based DGs (3 &4)	3 MVA	311 V
Transformer 1	20 MVA	120/25 kV
Transformer 2 &3	5 MVA	2.4/25 kV
Transformer 4 & 5	5 MVA	0.311/25 kV

Note: The choice of particular ratings is based on the main reference paper chosen.[11]

Similarly, the load parameters for the test system are tabulated in Table 3.2.

Table 3.2: Load Parameters for the test System.

Load	P (MW)	Q (MVAR)	Load	P (MW)	Q (MVAR)
L1	1.5	0.6	L4	1.5	0.3
L2	1.5	0.3	L5	0.5	0.1
L3	1	02	L6	1	0.3

3.2 Data Mining for Islanding and Non-Islanding Scenarios

To mine the data several islanding and non-islanding scenarios are needed to be realized in the test system. The data refers to the voltage data at PCC during the islanding and non-islanding scenarios. For that following procedure will be followed.

i. Obtain the three phase voltage data at PCC during the individual islanding and non-islanding scenario with a simulation time of 0.75s.
ii. Export the voltage data back to the MATLAB workspace in the timeseries dataset with a sampling frequency of 3840Hz.
iii. Perform STFT for acquired data using the in-built function *spectrogram ()* or we can divide the sampled data into non-overlapping chunks and each chunk is Fourier transformed to obtain the complex frequency spectra.
iv. After the STFT computation for each individual scenarios the image data is stored and classified under two categories islanding and non-islanding for training and testing data.

- **Checking the test system and computing the spectrogram.**

The CB at zone-4 is provided an external signal containing the logic level (0) to open at 0.5s so that the zone 4 will be islanded from the system. The three phase voltage data is logged to MATLAB workspace as dataset *"VoltageCB1a", "VoltageCB1b", "VoltageCB1c",* sampled at 3840Hz. The dataset is accessed in the script file as shown in MATLAB code appended in Appendix-I. A MATLAB script is run to produce the spectrogram from the dataset. The spectrogram is shown in Figure 3.4. Spectrogram's X-axis and Y-axis indicates the time and frequency respectively.

A SLG fault is introduced at distribution line-1 of zone-1 to realize a non-islanding scenario at 0.5s and the voltage was monitored at PCC. Spectrogram is generated as shown in Figure 3.5.

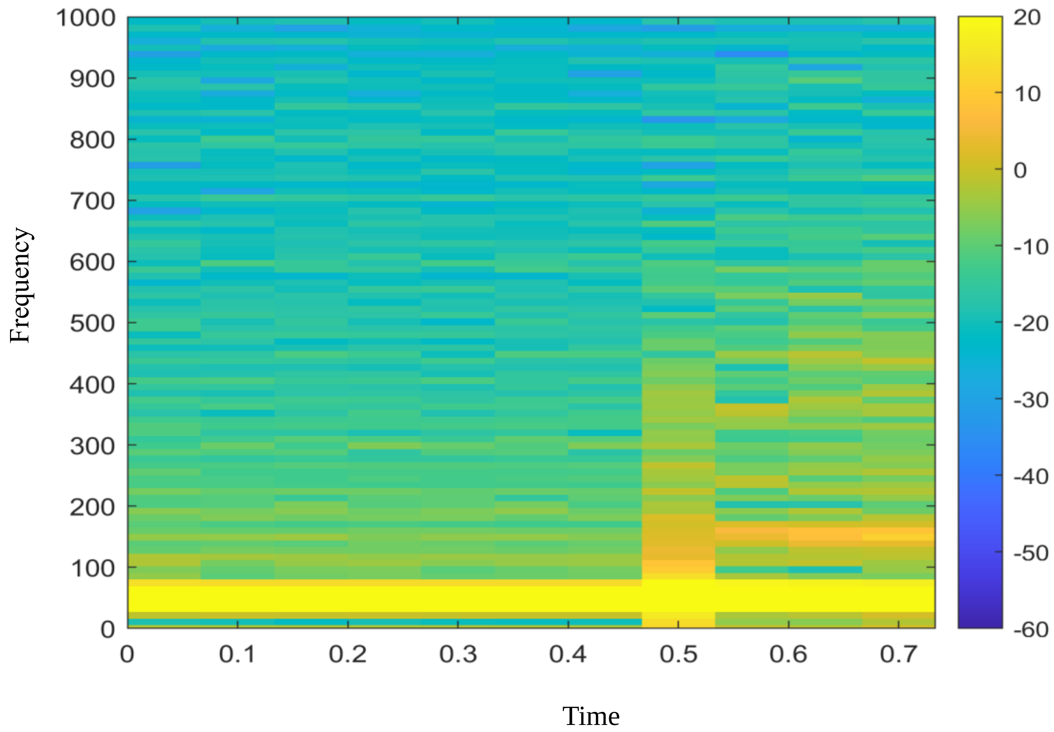


Figure 3.4: Spectrogram of voltage signal at Bus-4 during islanding

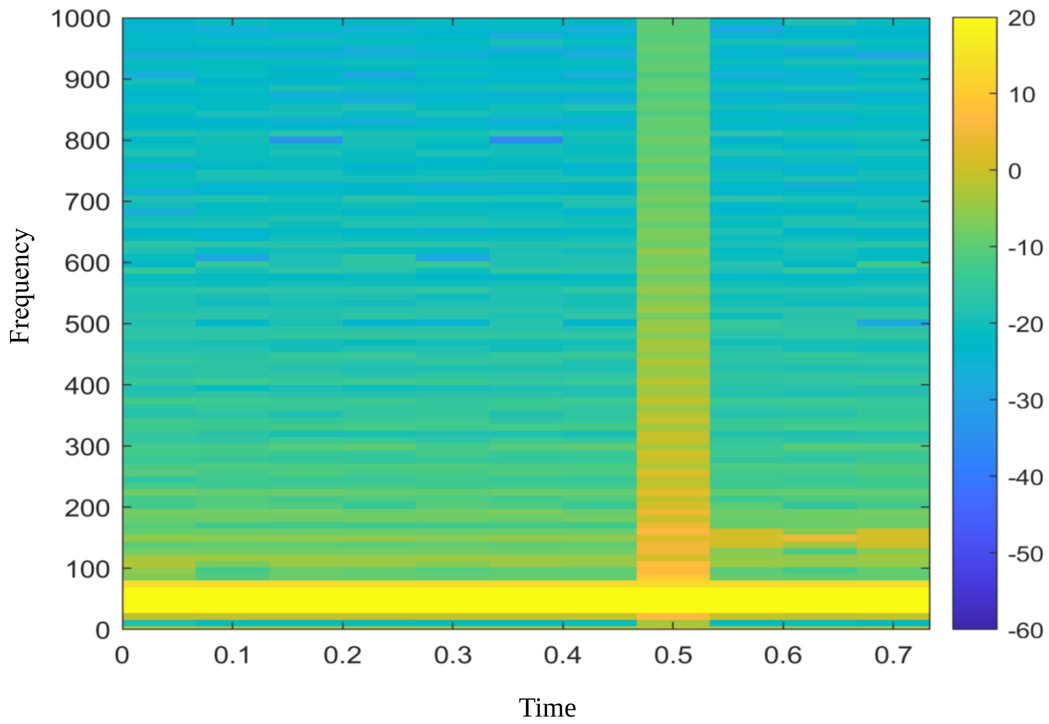


Figure 3.5: Spectrogram of voltage signal during SLG fault (non-islanding) at DL-1

A case of introduction of a RLC load of 10 MVA at 0.5s is at zone-2 is created to realize a non-islanding based on load switching scenario. The voltage is monitored at PCC and Spectrogram is generated as shown in Figure 3.6.

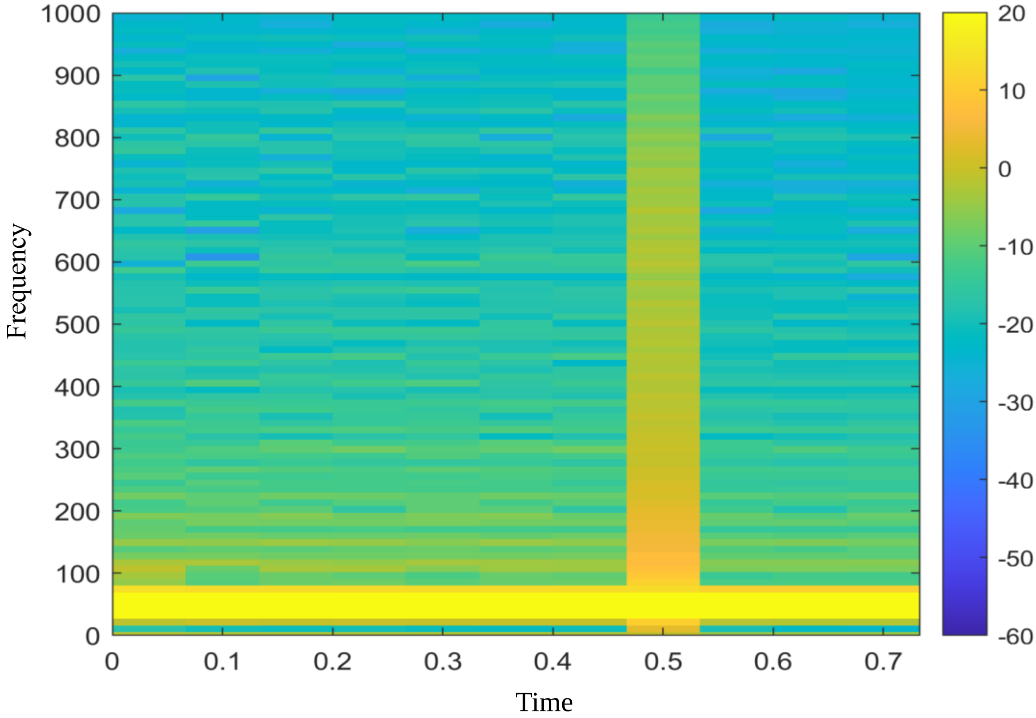


Figure 3.6: Spectrogram of voltage signal during load switching (non-islanding) at zone 2

Similarly, several other scenarios are created for various islanding and non-islanding conditions. Respective spectrogram of each scenario is generated as mentioned in the Table 3.3.

Table 3.3: Data Mining for the proposed IDT

Islanding and non-islanding cases on various disturbances	No. of Cases
Islanding • Islanding by changing ΔP and ΔQ from (-50% to +50%) and (-25% to +25%) respectively. • Non detection Zones cases	**1050** 800 250
Non-islanding • Faults on DL-1 and DL-2 (SLG, DLG, TLG) by changing fault resistance from (0.1-100 Ω). • Load Switching on L2, L4, L5 in the range of (0.5-50 MVA), (5-55 MVA), (10-70 MVA).	**1050** 980 70
Total Islanding and non-islanding cases	**2100**

3.3 Modelling and Training of CNN

To create a model of CNN, the CNN architecture presented in Table 3.4 is used. To realize the architecture in code, MATLAB Deep Learning Toolbox environment is used. The code required is presented in the Appendix -I.

Training Options: -

Optimizer: SGDM (Stochastic Gradient Descent with Momentum)
Initial Learn Rate: 0.01
Max Epochs: 10
Mini Batch Size: 32
Shuffle: Every epoch
Validation Frequency: 30
Execution Environment: CPU
Validation Method: k-fold; k=5

Table 3.4: CNN Architecture

Layer	Type	Output Size	Kernel Size	Filters	Stride	Padding
Input	Image Input	224 x 224 x 3	-	-	-	-
1	Convolution 2D	224 x 224 x 16	3 x 3	16	1	same
2	Batch Normalization	224 x 224 x 16	-	-	-	-
3	ReLU	224 x 224 x 16	-	-	-	-
4	Max Pooling 2D	112 x 112 x 16	2 x 2	-	2	-
5	Convolution 2D	112 x 112 x 32	3 x 3	32	1	same
6	Batch Normalization	112 x 112 x 32	-	-	-	-
7	ReLU	112 x 112 x 32	-	-	-	-
8	Max Pooling 2D	56 x 56 x 32	2 x 2	-	2	-
9	Convolution 2D	56 x 56 x 64	3 x 3	64	1	same
10	Batch Normalization	56 x 56 x 64	-	-	-	-
11	ReLU	56 x 56 x 64	-	-	-	-
12	Global Average Pooling 2D	1 x 1 x 64	-	-	-	-
13	Fully Connected	1 x 1 x 2	-	2	-	-
14	Softmax Layer	1 x 1 x 2	-	-	-	-
15	Classification Output	-	-	-	-	-

Based on architecture presented in Table 3.4 a CNN model is developed in MATLAB and trained and tested using the training options mentioned earlier.

CHAPTER FOUR: RESULTS AND DISCUSSIONS

4.1 Results of CNN Training

After the training is completed, the confusion matrix is published as shown in Figure 4.1. The confusion matrix represents for all the folds of validation with 2 cases mis-classified. After the training session the mean accuracy is calculated which equals to 99.9%.

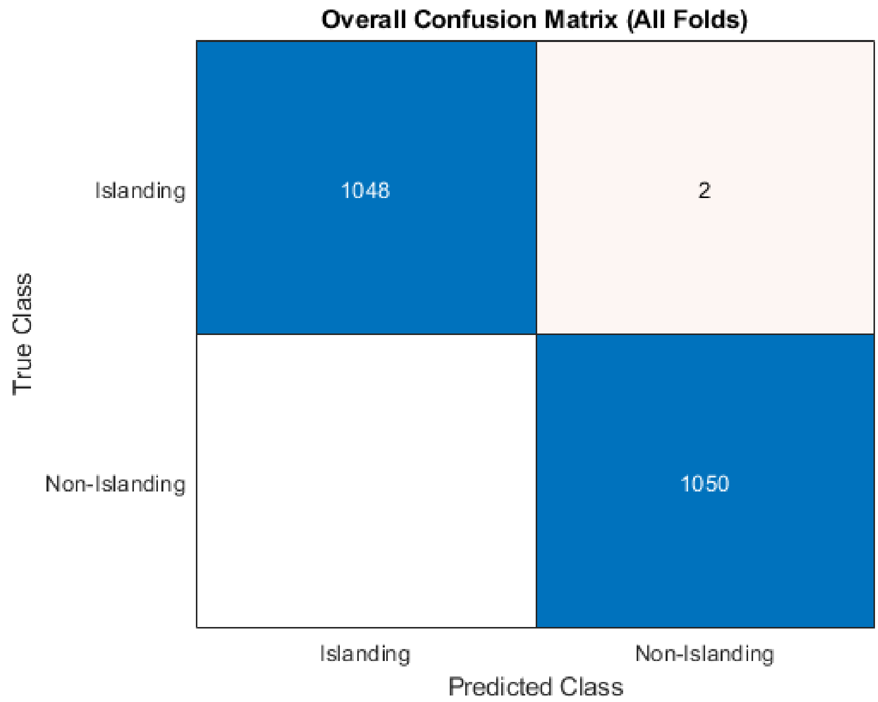
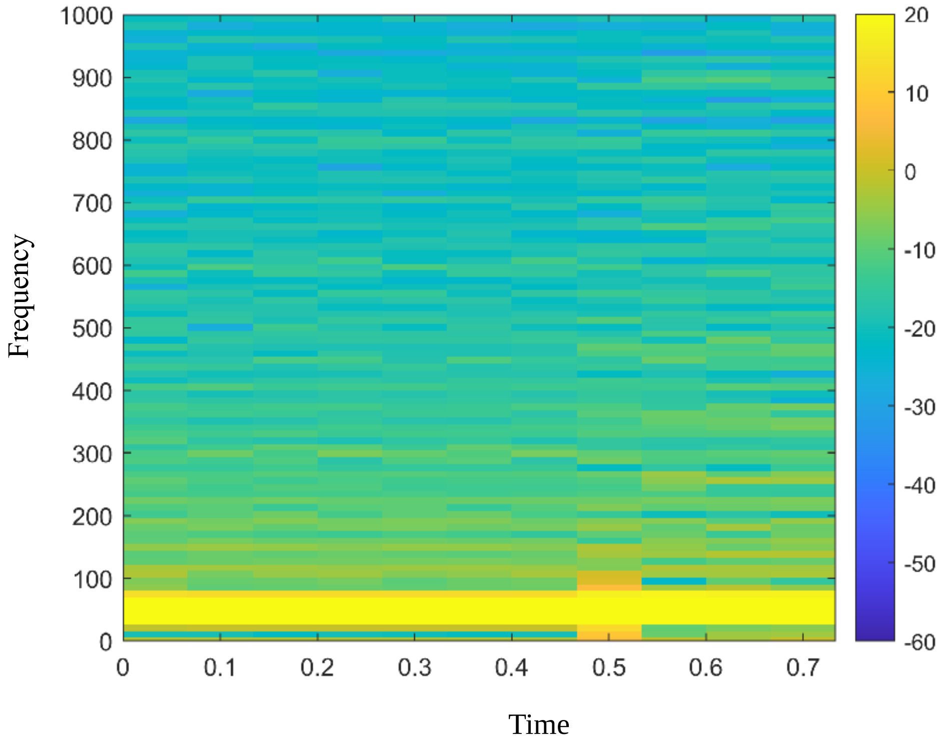


Figure 4.1: Overall Confusion Matrix

4.2 Results in Terms of Validation on Unseen Images

As the scope of this project doesn't involve on testing the model at real word test-system, some sets of unseen images with islanding and non-islanding at different instances of time were created. Then they were fed to the model to classify those images containing the information of either islanding or non-islanding. The categorical classification results of those images with confidence of prediction are shown below.

Case 1: Islanding created at 0.5s with generation and load being almost similar (a case of NDZ) at the zone of interest.

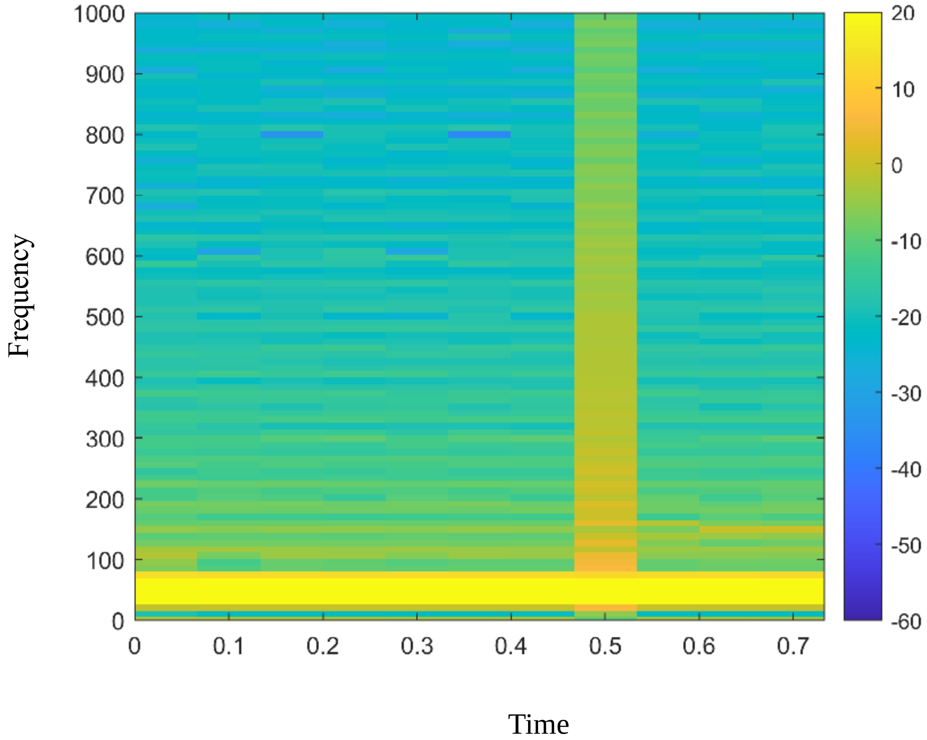



```
message = "Islanding Condition Detected"
message1 = " The confidence level of classification is
99.9029                                                              %"
```

Figure 4.2: Islanding scenario at 0.5s

The spectrogram in Figure 4.2 is generated to realize the case of non-detection zone for islanding by closely matching the local generation and demand at zone-2 by opening the CB at 0.5s. The spectrogram shows the low amount harmonics portrayed by the yellowish band in vertical direction at 0.5s. The low number of harmonics in voltage signal is due to the almost power balance during the islanding condition. The image is fed to CNN model and the model classified it as islanding condition with confidence level of 99.9029%. The classification holds true for the islanding scenario.

Case 2: Non-islanding created at 0.5s.

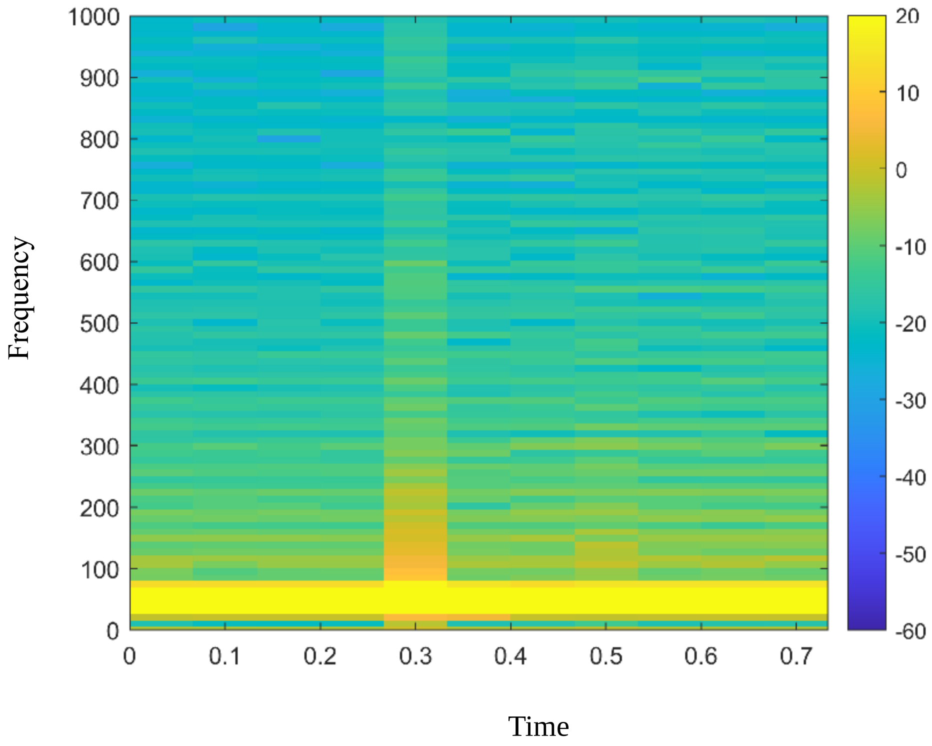



```
message = " The abnormality in voltage waveform does not
correspond to islanding condition"
message1 =" The confidence level of classification is
99.2865 %"
```

Figure 4.3: Non-islanding scenario at 0.5s

The spectrogram in Figure 4.3 is generated to realize the case of non-islanding condition. For this SLG fault at 0.5s is introduced at DL-1 with the fault resistance of 5 ohm and the spectrogram is produced. The spectrogram shows the harmonics distribution at 0.5s event. The yellowish band at lower frequencies indicates the presence of lower harmonics with greater amplitude while greenish yellow at higher portion indicates the decreasing amplitude of higher harmonics. The image is fed to CNN model and the model classified it as non-islanding condition with confidence level of 99.2865%. The classification holds true for the non-islanding scenario.

Case 3: Islanding created at 0.3s.




```
message = " Islanding Condition Detected"
message1 = " The confidence level of classification is 99.5984 %"
```

Figure 4.4: Islanding scenario at 0.3s

The spectrogram in Figure 4.4 is generated to realize the case of islanding condition but at different time. For this the CB is opened at 0.3s at zone-1 at and the spectrogram is produced. The spectrogram shows the harmonics distribution at 0.3s event. The yellowish band at lower frequencies indicates the presence of lower harmonics with greater amplitude while greenish bands at medium and higher portion indicates the decreasing amplitude of higher harmonics. The image is fed to CNN model and the model classified it as islanding condition with confidence level of 99.5984%. The classification holds true for the islanding scenario.

CHAPTER FIVE: CONCLUSION AND RECOMMENDATION

The training and testing of the CNN model led to mean accuracy of 99.9% for categorical classification. The model's confusion matrix shows two instances of misclassification of the image data for islanding scenario. The accurate validation of unseen images (Case1, Case2, Case3) further assures the confidence of model in predicting the outcomes. So, once the model is trained, the model parameters is further used for classifying the voltage signals measured at PCC. The classifier classifies the signal and sends a corresponding signal to controller to make further decisions on opening or avoiding the opening of CBs.

Though the detection time could not be assessed in hands in this project but one can realize it with using the trained model on real world test systems. During the project several realizations were done regarding how to continuously monitor the voltage signal at PCC along with the several noises that might impact the data passed to the model. This can mislead the model and false classification could happen. So, the CNN model should be trained with data that incorporates the noises present in the real power systems.

REFERENCES

[1] A. Khamis, H. Shareef, E. Bizkevelci, and T. Khatib, ‘‘A review of islanding detection techniques for renewable distributed generation systems,’’ Renew. Sustain. Energy Rev., vol. 28, pp. 483–493, Dec. 2013, doi: 10.1016/j.rser.2013.08.025

[2] A. Hussain, C.-H. Kim, and A. Mehdi, ‘‘A comprehensive review of intelligent islanding schemes and feature selection techniques for distributed generation system,’’ IEEE Access, vol. 9, pp. 146603–146624, 2021.

[3] S. R. Mohanty, N. Kishor, P. K. Ray, and J. P. S. Catalo, ‘‘Comparative study of advanced signal processing techniques for islanding detection in a hybrid distributed generation system,’’ IEEE Trans. Sustain. Energy, vol. 6, no. 1, pp. 122–131, Jan. 2015, doi: 10.1109/TSTE.2014.2362797.

[4] C.-T. Hsieh, J.-M. Lin, and S.-J. Huang, ‘‘Enhancement of islanding detection of distributed generation systems via wavelet transform-based approaches,’’ Int. J. Electr. Power Energy Syst., vol. 30, no. 10, pp. 575–580, Dec. 2008, doi: 10.1016/j.ijepes.2008.08.006.

[5] S. Raza, H. Mokhlis, H. Arof, J. A. Laghari, and L. Wang, ‘‘Application of signal processing techniques for islanding detection of distributed generation in distribution network: A review,’’ Energy Convers. Manage., vol. 96, pp. 613–624, May 2015, doi: 10.1016/j.enconman.2015.03.029.

[6] M. Hanif, M. Basu, and K. Gaughan, ‘‘Development of EN50438 compliant wavelet-based islanding detection technique for three-phase static distributed generation systems,’’ IET Renew. Power Gener., vol. 6, no. 4, p. 289, 2012, doi: 10.1049/iet-rpg.2011.0290.

[7] P. K. Ray, S. R. Mohanty, and N. Kishor, ‘‘Disturbance detection in grid connected distributed generation system using wavelet and S-transform,’’ Electr. Power Syst. Res., vol. 81, no. 3, pp. 805–819, Mar. 2011, doi: 10.1016/j.epsr.2010.11.011.

[8] Y. Sun, B. Wang, J. Jin, and X. Wang, ‘‘Deep convolutional network method for automatic sleep stage classification based on neurophysiological signals,’’ in Proc. 11th Int. Congr. Image Signal Process., Biomed. Eng. Informat. (CISP-BMEI), Oct. 2018, pp. 1–5, doi: 10.1109/CISP-BMEI.2018.8633058.

[9] T. Chakravorti, R. K. Patnaik, and P. K. Dash, ‘‘Detection and classification of islanding and power quality disturbances in microgrid using hybrid signal processing and data mining

techniques,'' IET Signal Process., vol. 12, no. 1, pp. 82–94, Feb. 2018, doi: 10.1049/iet-spr.2016.0352.

[10] S. K. G. Manikonda and D. N. Gaonkar, ''IDM based on image classification with CNN,'' J. Eng., vol. 2019, no. 10, pp. 7256–7262, Oct. 2019, doi:10.1049/joe.2019.0025.

[11] A. Hussain, C.-H. Kim, and M. S. Jabbar, ''An intelligent deep convolutional neural networks-based islanding detection for multi-DG systems,'' IEEE Access, vol. 10, pp. 131920–131931, 2022, doi:10.1109/ACCESS.2022.3229698

APPENDIX -I

- The MATLAB code referenced in Section 3.2 is written below.

#

```
clear all;
% This bit of code is for variation in Active/Reactive Power
in Zone-1
basw=[];
perc=0.75;
while perc<=1.25
    basw(1,end+1)=(((14*perc)^2-49)^0.5)-0.6;
    perc=perc+0.008;
end
% % This bit of code is for variation in Active /Reactive
Power in Zone-2
perc=0.75;
basw=[];
while perc<=1.25
    basw(1,end+1)=((4*perc)^2-2.25)^0.5;
    perc=perc+0.01;
end
% % This bit of code is for variation in Active /Reactive
Power in Zone-3
perc=0.75;
basw=[];
while perc<=1.25
    basw(1,end+1)=((10*perc)^2-16)^0.5;
    perc=perc+0.01;
end
% % This bit of code is for variation in Active /Reactive
Power in Zone-4
perc=0.75;
basw=[];
while perc<=1.25
    basw(1,end+1)=((3*perc)^2-2.25)^0.5;
    perc=perc+0.01;
end
%This code segment is for variation in Fault resistance at DL-
1
res=[0.1 0.5];
for s=2:200
    res(1,end+1)=res(1,s)+0.5;
end
%This code segment is for variation in Fault resistance at DL-
2
res=[0.1 0.5];
```

```
for s=2:200
    res(1,end+1)=res(1,s)+0.5;
end
% This bit of code to realize various load switching scenarios
load1=[5 7];
for s=2:10
    load1(1,end+1)=load1(1,s)+2;
end
warning('off', 'all');
for d= 1:length(load1)
bas=basw(d)*1e6
loadsim=load1(d)*1e6
% bas=res(d)
sim('IDTmodel');
Va= VoltageCB1a.Data(1:end);
Vb= VoltageCB1b.Data(1:end);
Vc= VoltageCB1c.Data(1:end);
voltage_data=[Va'; Vb'; Vc'];

% Parameters from the paper
Fs = 3840;  % Sampling frequency (Hz)
T = 0.75;   % Total simulation time (s)

% Generate time vector
t = 0:1/Fs:T-1/Fs;
% Parameters for STFT
window = hamming(256);  % Window function
noverlap = 0;          % Number of overlapped samples
nfft = 360;             % Number of FFT points

% Compute STFT for each phase and combine
S_phases = zeros(nfft/2+1, floor((length(t)-
noverlap)/(length(window)-noverlap)), 3);
for phase = 1:3
    [S, F, T] = spectrogram(voltage_data(phase,:), window,
noverlap, nfft, Fs, 'yaxis');
    S_phases(:,:,phase) = abs(S);
end
S_max = max(S_phases, [], 3); % Method 2: Maximum


% Plot spectrogram
figure;
imagesc(T, F, 20*log10(S_max));  % Convert to dB scale
axis xy;  % Put low frequencies at the bottom
% xlabel('Time (s)');
% ylabel('Frequency (Hz)');
ylim([0 1000]);
```

```
% title('Combined Three-Phase Voltage Spectrogram');
colorbar;
caxis([-60 20]);  % Adjust color scale as needed
% Save the spectrogram image
folderPath = 'C:\Users\Dell\Desktop\Project MSC Third
Sem\MATLAB Files\Scripts\Datasets\Islanding';
a=4
fileName= [ num2str(a) '.png']
fullPath = fullfile(folderPath, fileName);
exportgraphics(gcf,fullPath,'Resolution',300);
% end
```

- The MATLAB code referenced in Section 3.3 is written below.

#

```
clc;
clear all;
% Define the base directory
baseDir = 'C:\Users\Dell\Desktop\Project MSC Third Sem\MATLAB
Files\Scripts\Datasets';
% Define the subdirectories for Islanding and Non-Islanding
images
islandingDir = fullfile(baseDir, 'Islanding');
nonIslandingDir = fullfile(baseDir, 'Non-Islanding');
% Create a custom datastore for preprocessing
imds = imageDatastore({islandingDir, nonIslandingDir}, ...
    'IncludeSubfolders', true, ...
    'LabelSource', 'foldernames', ...
    'ReadFcn', @(filename) preprocessImage(imread(filename)));
% Display some information about the datastore
fprintf('Total number of images: %d\n', numel(imds.Files));
fprintf('Labels: %s\n', strjoin(string(unique(imds.Labels)),
', '));
% Define a simpler CNN architecture
layers = [
    imageInputLayer([224 224 3])

    convolution2dLayer(3, 16, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 32, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)
```

```
    convolution2dLayer(3, 64, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer

    globalAveragePooling2dLayer
    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer
];
% Defining training options (using CPU)
options = trainingOptions('sgdm', ...
    'InitialLearnRate', 0.01, ...
    'MaxEpochs', 10, ...
    'MiniBatchSize', 32, ...
    'Shuffle', 'every-epoch', ...
    'ValidationFrequency', 30, ...
    'Verbose', false, ...
    'Plots', 'training-progress', ...
    'ExecutionEnvironment', 'cpu');
% Setting up k-fold cross validation
k = 5; % Number of folds
cv = cvpartition(imds.Labels, 'KFold', k);
% Initializing variables to store results
accuracies = zeros(k, 1);
confusionMats = cell(k, 1);
aucs = zeros(k, 1);
% Loop through each fold
for i = 1:k
    fprintf('Processing fold %d/%d\n', i, k);

    % Get training and validation sets for this fold
    trainingIdx = cv.training(i);
    validationIdx = cv.test(i);
    trainingSet = subset(imds, trainingIdx);
    validationSet = subset(imds, validationIdx);

    % Train the network
    net = trainNetwork(trainingSet, layers, options);

    % Evaluate on validation set
    [YPred, probs] = classify(net, validationSet);
    accuracies(i) = mean(YPred == validationSet.Labels);
    confusionMats{i} = confusionmat(validationSet.Labels,
YPred);
    [~, ~, ~, aucs(i)] = perfcurve(validationSet.Labels,
probs(:, 1), 'Islanding');
```

```
    fprintf('Fold %d Accuracy: %.2f%%\n', i, accuracies(i) *
100);
end
% Calculate and display overall results
meanAccuracy = mean(accuracies);
stdAccuracy = std(accuracies);
fprintf('Mean Accuracy: %.2f%% (±%.2f%%)\n', meanAccuracy *
100, stdAccuracy * 100);
% Aggregate confusion matrices
totalConfusionMat = sum(cat(3, confusionMats{:}), 3)
% Plot overall confusion matrix
figure;
confusionchart(totalConfusionMat, unique(imds.Labels));
title('Overall Confusion Matrix (All Folds)');
% Plot ROC curve (using average AUC)
figure;
plot([0, 1], [0, 1], 'r--');
hold on;
plot(linspace(0, 1, 100), linspace(0, 1, 100).^(1/mean(aucs)),
'b-');
xlabel('False Positive Rate');
ylabel('True Positive Rate');
title(sprintf('Approximate ROC Curve (Mean AUC = %.2f)',
mean(aucs)));
legend('Random Classifier', 'Model Performance', 'Location',
'southeast');
hold off;
% Save the best model (you might want to modify this criteria)
[~, bestFoldIndex] = max(accuracies);
bestNet = net; % Assuming the last trained net is the best;
modify if storing each fold's net
save('bestModel.mat', 'bestNet');

load('bestModel.mat','bestNet');
% Example usage of classifyNewImage:
[label,prob]= classifyNewImage(bestNet,
'C:\Users\Dell\Desktop\Project MSC Third Sem\MATLAB
Files\Scripts\testimages\4.png');
char(label)

function preprocessedImg = preprocessImage(img)
    % Resize image to a smaller dimension
    img = imresize(img, [224 224]);
    % Convert to single precision and scale to [0, 1]
    img = im2single(img);
    % Normalize to zero mean and unit variance
    img = (img - mean(img(:))) / std(img(:));
    preprocessedImg = img;
end
```

```
function [label, probability] = classifyNewImage(net,
imagePath)
    img = imread(imagePath);
    img = preprocessImage(img);

    % Classify the image
    [label, scores] = classify(net, img);
    probability = max(scores);
    end
```

#

- The MATLAB code referenced in Section 4.2 is written below.

  #

```
clc
clear all;
load('bestModel.mat','bestNet');
% Example usage of classifyNewImage:


[label,prob]= classifyNewImage(bestNet,
'C:\Users\Dell\Desktop\Project MSC Third Sem\MATLAB
Files\Scripts\testimages\4is.png');
label= char(label);
prob=prob*100;
if label=="Islanding"
    message= " Islanding Condition Detected"
    message1="The confidence level of classification is" +
prob
else
    message=" The abnormality in voltage waveform doesnot
correspond to islanding condition"
    message1="The confidence level of classification is " +
prob
end

function preprocessedImg = preprocessImage(img)
    % Resize image to a smaller dimension
    img = imresize(img, [224 224]);
    % Convert to single precision and scale to [0, 1]
    img = im2single(img);
    % Normalize to zero mean and unit variance
    img = (img - mean(img(:))) / std(img(:));
    preprocessedImg = img;
end
```

```
function [label, probability] = classifyNewImage(net,
imagePath)
    img = imread(imagePath);
    imshow(img)
    img = preprocessImage(img);

    % Classify the image
    [label, scores] = classify(net, img);
    probability = max(scores);
    end
```
#

Note: The repository containing the whole image datasets, required MATLAB code, Simulink models and further documents are available pubic on the GitHub repository mentioned in the link below.

https://github.com/basantceline/Islanding-Detection-using-CNN.git