2D Encoder-based Convolutional Neural Network and Empirical Wavelet Transform for Classification of Combined Power Quality Disturbances

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

B.Tech

in

Electrical and Electronics Engineering

By

Charmie Rajan (107118026)

G. Akash (107118033)

Rakshaa Viswanathan (107118078)

Siddharth Rao A. (107118095)

ELECTRICAL AND ELECTRONICS ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI – 620015

MAY 2022

BONAFIDE CERTIFICATE

This is to certify that the project titled **2D Encoder-based Convolutional Neural Network and Empirical Wavelet Transform for Classification of Combined Power Quality Disturbances** is a bonafide record of the work done by

Charmie Rajan (107118026)

G. Akash (107118033)

Rakshaa Viswanathan (107118078)

Siddharth Rao A. (107118095)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Electrical and Electronics Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI**, during the year 2021-22.

Dr. Karthik Thirumala	**Dr. V. Sankaranarayanan**
Guide	Head of the Department

Project Viva-voce held on ______________________

Internal Examiner	**External Examiner**

ABSTRACT

Power Quality is used to describe clean and reliable electrical power that enables a device to function efficiently. Any deviations caused in voltage or current waveforms from their ideal characteristics are termed as power quality disturbances. Hence, when a voltage loss occurs, it causes energy deficit that disturbs the working of large interconnected systems and other equipments. This is accounted by economic loss, energy loss, machine breakdowns and shutdowns. These power quality disturbances are caused due to nonlinear loads, power electronic converters, switching events or system faults. Manufacturers and users of equipment will be at risk if the system isn't monitored and will result in equipment malfunction due to the disturbance caused. A reliable identification of the disturbances by an efficient system is required, for the detection, measurement and classification of various power quality disturbances. The commonly analyzed techniques for non-stationary signals include wavelet transform, Short Time Fourier Transform, etc. But their performance is limited if the characteristics of a disturbance are not captured precisely. Empirical Wavelet Transform is used to decompose non-linear and non-stationary signals with a lesser time complexity. This method is easy to implement, customizable and appropriate to analyze time varying signals. Following which, a deep learning pipeline was established that consisted of compression of data, feature extraction and classification. The perspective of this project is to review the concepts utilized and propose an algorithm for **2D Encoder-based Convolutional Neural Network and Empirical Wavelet Transform for the classification of eight single and seven combined power quality disturbances (total 16 classes)**. Data generation and signal processing was carried out in MATLAB while the deep learning pipeline was built using PyTorch. An overall accuracy of 99.28% was obtained for model developed with an efficient testing time of 0.78ms per sample.

Keywords: *Power Quality Disturbance, Empirical Wavelet Transform, Encoder, Convolutional Neural Network*

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to the following people for guiding us through this course and without whom this project and the results achieved from it would not have reached completion.

Dr. Karthik Thirumala, Assistant Professor, Department of Electrical and Electronics Engineering, for helping us and guiding us in the course of this project. Without his guidance, we would not have been able to successfully complete this project. His patience and genial attitude is and always will be a source of inspiration to us.

Dr. V. Sankaranarayanan, the Head of the Department, Department of Electrical and Electronics Engineering, for allowing us to avail the facilities at the department.

We would also like to thank our professors, **Dr. P. Raja** and **Dr. M. Venkatakirthiga** of the Department of Electrical and Electronics Engineering, for their valuable inputs during each of the internal reviews, our parents and our friends for their constant support and help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AE	Autoencoder
AWGN	Additive White Gaussian Noise
CNN	Convolutional Neural Network
EFCNN	Encoded Feature Convolutional Neural Network
EWT	Empirical Wavelet Transform
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
MRA	Multiresolutional Analysis
NN	Neural Network
PQ	Power Quality
PQD	Power Quality Disturbances
PQM	Power Quality Monitoring
RMS	Root Mean Square
SNR	Signal to Noise Ratio
VAE	Variational Autoencoder
WGN	White Gaussian Noise

CHAPTER 1
INTRODUCTION

The term power quality (PQ) refers to a set of boundaries that allows electrical systems to function in their intended manner. Power quality is used to describe electric power that drives an electrical device and the device's ability to function properly with that electric power. As per IEEE, it is a wide variety of electromagnetic phenomena that characterize the voltage and current at a given time and at a given location on the power system. Any deviation from the ideal voltage or current is termed as power quality disturbance. Power quality disturbances are one of the main problems in electric power systems which include

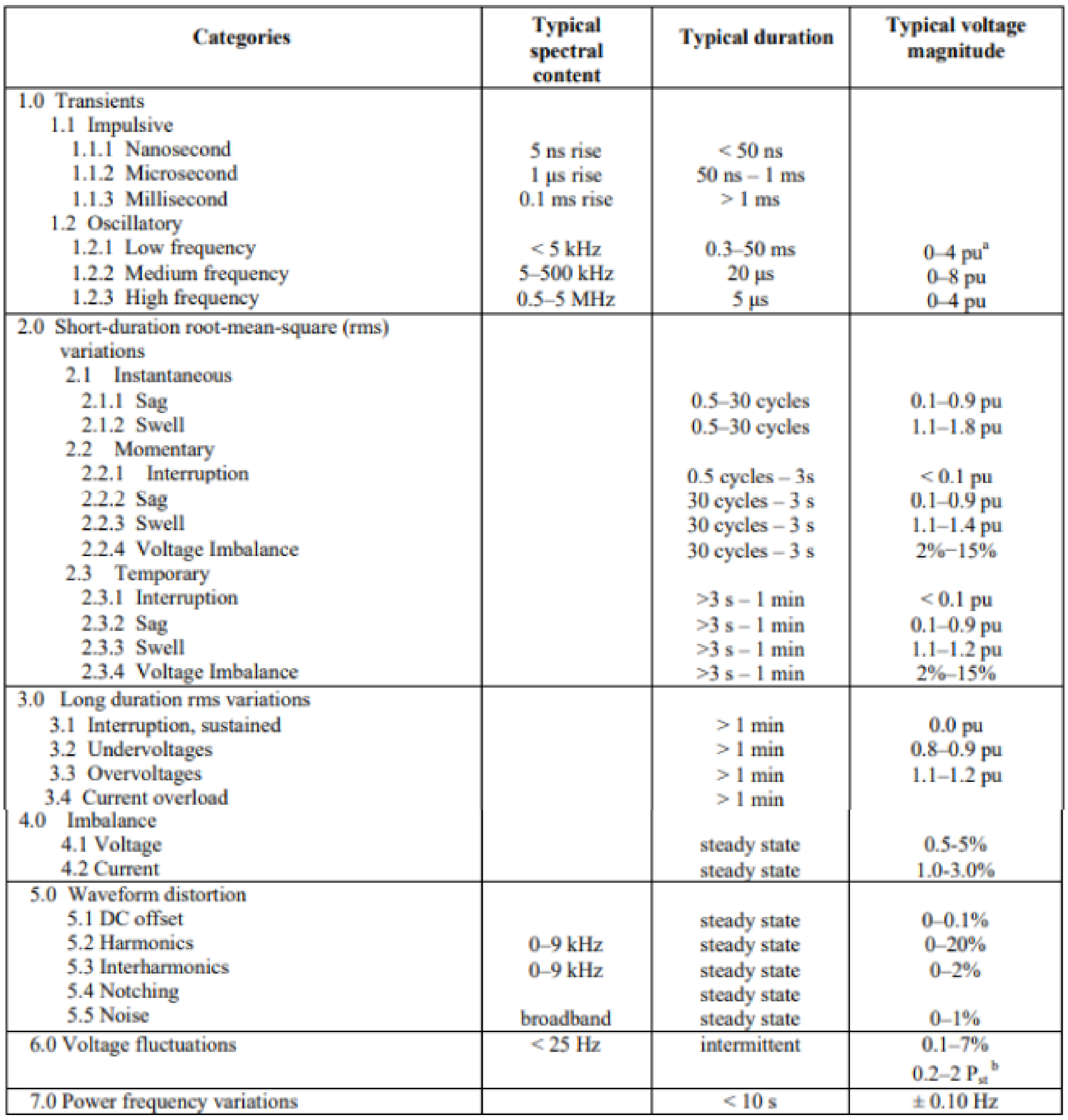
- Energy Loss
- Equipment Instability
- Financial losses to the network operators and the equipment manufacturers

The IEEE standard provides a classification of the power disturbances, that includes a detailed description for each of the variation in power quality categories. These categories evolved due to the multiple ways each of these power quality problems can be solved and the different requirements that are needed to be fulfilled to characterize each phenomenon using measurements. This standard intends to provide a unique guide to analyzing power disturbances considering the importance of power system disturbances for analysis purposes. Supply of reliable and quality power has become one of the key development objectives of the modern power sector.

1.1 Power Quality Disturbances

This section will focus on power quality, causes, effects and its types and classes.The definition of power quality (PQ) will be different in the views of utilities, consumers and equipment suppliers and other perspectives. It is applied to a wide array of electromagnetic phenomena occurring within an influence system network. Since it is a consumer-driven problem, it can be defined as, "any sudden change in the normal operation of voltage, current or frequency which causes malfunction or failure of the consumer equipment" [1, 2]. Sudden deviations in the voltage, current or frequency from the normalized rating is known as a PQ disturbance that causes breakdown or malfunctioning of the power equipment. The main components of the PQ research involve basic concepts and definitions, simulations and analysis, instrumentation and measurement, causes, effects and solutions of PQ disturbances.

In recent years, the issue of PQ has become crucial to utilities and customers owing to the increasing usage of modern power electronic devices that are sensitive to voltage disturbances. To improve power quality, one has to know about the sources of power system disturbances or the actual causes behind them. Some of the causes of PQ disturbances are the increasing application of solid-state switching devices, non-linear loads, rectifiers and inverters, lighting controls, computer and data processing equipment, protection and relaying equipment. The PQ disturbances, if not resolved correctly, may result in the overall interruption of the power transmission and distribution networks.



Categories	Typical spectral content	Typical duration	Typical voltage magnitude
1.0 Transients			
1.1 Impulsive			
1.1.1 Nanosecond	5 ns rise	< 50 ns	
1.1.2 Microsecond	1 µs rise	50 ns – 1 ms	
1.1.3 Millisecond	0.1 ms rise	> 1 ms	
1.2 Oscillatory			
1.2.1 Low frequency	< 5 kHz	0.3–50 ms	0–4 pu[a]
1.2.2 Medium frequency	5–500 kHz	20 µs	0–8 pu
1.2.3 High frequency	0.5–5 MHz	5 µs	0–4 pu
2.0 Short-duration root-mean-square (rms) variations			
2.1 Instantaneous			
2.1.1 Sag		0.5–30 cycles	0.1–0.9 pu
2.1.2 Swell		0.5–30 cycles	1.1–1.8 pu
2.2 Momentary			
2.2.1 Interruption		0.5 cycles – 3s	< 0.1 pu
2.2.2 Sag		30 cycles – 3 s	0.1–0.9 pu
2.2.3 Swell		30 cycles – 3 s	1.1–1.4 pu
2.2.4 Voltage Imbalance		30 cycles – 3 s	2%–15%
2.3 Temporary			
2.3.1 Interruption		>3 s – 1 min	< 0.1 pu
2.3.2 Sag		>3 s – 1 min	0.1–0.9 pu
2.3.3 Swell		>3 s – 1 min	1.1–1.2 pu
2.3.4 Voltage Imbalance		>3 s – 1 min	2%–15%
3.0 Long duration rms variations			
3.1 Interruption, sustained		> 1 min	0.0 pu
3.2 Undervoltages		> 1 min	0.8–0.9 pu
3.3 Overvoltages		> 1 min	1.1–1.2 pu
3.4 Current overload		> 1 min	
4.0 Imbalance			
4.1 Voltage		steady state	0.5-5%
4.2 Current		steady state	1.0-3.0%
5.0 Waveform distortion			
5.1 DC offset		steady state	0–0.1%
5.2 Harmonics	0–9 kHz	steady state	0–20%
5.3 Interharmonics	0–9 kHz	steady state	0–2%
5.4 Notching		steady state	
5.5 Noise	broadband	steady state	0–1%
6.0 Voltage fluctuations	< 25 Hz	intermittent	0.1–7% 0.2–2 P_{st}[b]
7.0 Power frequency variations		< 10 s	± 0.10 Hz

Figure 1.1: *IEEE Std 1159 Categories and typical characteristics of power system electromagnetic phenomena*

The development of new techniques for the automatic classification of power quality events is at present a major concern. While existing techniques are capable of automatically identifying and classifying various types of distribution-level power quality disturbances, they do not provide any information about the locations of the disturbance sources. A Direction Finder for Power Quality Disturbances Based Upon Disturbance Power and Energy shows that it is possible to use sampled voltage and current waveforms to determine on which side of a recording device a disturbance originates [3]. The technique of an automatic approach on adaptive filtering and multiclass SVM for classification of single and frequent combined disturbances is suitable for decomposing the signal faster [4].

PQ studies commonly involves disturbances that alter the sinusoidal voltage features and/or current wave shapes. In general, the power quality disturbances can be categorized into three major groups based on the waveform characteristics [5]. They are amplitude variations, frequency variations and transient phenomenon. The voltage interruption, sag, swell and voltage fluctuations fall under the first category [4]. Frequency variations cover harmonics, inter-harmonics, power frequency variations, voltage notching. This paper deals with pure sine wave, single and combined PQDs where single PQD includes sag, swell, notch, spike etc. and combined PQD includes sag with transient, swell with transient, sag with harmonics etc.

1.2 Power Quality Monitoring

A power quality monitoring system helps to gather, analyze, and interpret raw electricity measurement data into useful information. An ideal system measures voltage and electrical current. When an interruption of power occurs, the machine stops and even for a brief period might lead to significant reduction in efficiency and productivity that leads to loss in output as well. In order to monitor power, power plants use equipments like:

1. **Digital fault recorders**: It activates on fault events and records the current waveforms and voltage that caused the problem and can also capture periodic waveforms helpful in calculating harmonic distortion levels.

2. **Smart relays**: They check the power current and record disturbances.

3. **Voltage recorders**: It is used to monitor variations that are on the system and detects a trend and gives minimum, maximum, and average voltage every two seconds.

4. **In-plant power monitors**: It is usually at the service entrance, capture wave shape for evaluation of harmonic distortion levels or voltage sag conditions.

5. **Special-purpose power quality equipment**: They are able to simultaneous watch voltage and current.

There are two streams of power quality data analysis which are off-line and on-line analyses. The off-line power quality data analysis, as the term suggests, is performed off-line at the central processing locations. It is carried out separately from the monitoring instruments. A dedicated computer software is used for this. On the other hand, the on-line data analysis is performed within the instrument itself for immediate information dissemination. It analyzes data as they are captured and the analysis results are available immediately for rapid dissemination. One of the primary advantages of on-line data analysis is that it can provide instant message delivery to notify users of specific events of interest.

PQ monitoring has been used for problem solving in industrial, commercial, and residential systems. The system is installed after a problem has been identified to characterize the problem and help identify possible solutions. PQ monitoring will always be needed for such problem-solving applications, but permanent systems are starting to reduce the need for portable monitors. PQ monitoring ihas become an integral part of overall system performance assessment procedures. Permanent PQ monitoring is installed at all substations and the information from these monitors is continually available throughout the utility.

1.3 Literature Survey

This paper aims at presenting an AI based approach for the classification of PQDs. The literature focuses on PQDs, types, causes and effects, the importance of Power Quality monitoring, signal decomposition, and multiple artificial intelligence methods proposed for the detection, analysis and automatic classification of PQ disturbances. The PQDs are produced due to the nonlinear loads, power electronic converters, system faults and switching events. The equipments and consumers of electric power are expected to acquire ideal voltage and current waveforms at rated power frequency.

The wavelet transform (WT), provides an understandable transient signal representation corresponding to a time-frequency plane [6, 7]. The recent advancement in signal processing techniques has enabled in faster and accurate decomposition of non-stationary and non linear signals, such as empirical wavelet transform, variable mode decomposition and empirical mode decomposition. A comparative analysis was reviewed among the various methods and, empirical wavelet transform was chosen to mitigate PQDs faster in real time as it was found to offer lesser time complexity [8, 9].

Implementing useful representations with minimal supervision is a key challenge in artificial intelligence. Deep learning has been successfully applied in various computer vision tasks and has the potential to enhance the performance of image and data compression. The autoencoder methodology has been applied in dimensionality reduction, feature extraction and reconstructions that helps to validate latent space representation has be reviewed [10]. The abundant literature on types of PQ disturbances or events and many techniques for their detection, localization, and classification have also been reviewed.

1.4 Objective

This project aims to accurately classify multiple power quality disturbances using Empirical Wavelet Transform (EWT) and Two Dimensional Encoder-based Convolutional Neural Network (2-D-CNN). PQD waveforms for 16 classes, with and without added noise, are generated and further processed using MATLAB. EWT decomposes these signals into fundamental and harmonics components and converted into a dataset to be processed. In this thesis, we propose a novel convolutional encoder based CNN architecture to classify the disturbances. The model takes the decomposed signal as input and processes it in two stages. The first stage consists of a Convolutional Encoder derived from an Autoencoder. This portion of the network compresses the data and extracts meaningful features. The next stage of the network is a simple neural network classifier which uses the encoded input to identify the type of disturbance. These two stages of the network are trained end-to-end for better results. The proposed model classifies eight single and seven combined PQDs for multiple signal to noise ratios:
Interruption, Sag, Swell, Harmonics, Flicker, Notch, Spike, Sag + Harmonics, Swell + Harmonics, Flicker + Harmonics, Swell + Harmonics, Flicker + Harmonic, Sag + Transient, Swell + Transient, Sag + Flicker, Swell + Flicker
The final algorithm is trained, tested and validated to obtain maximum accuracy of the proposed model. The stages of workflow are explained in detail in the following chapters.

CHAPTER 2
METHODOLOGY

2.1 Introduction

The different methods of signal analysis for non-stationary signals, namely short time Fourier Transform, WT, filter bank, Gabor transform (GT), S-transform (ST), PA, KF, Cohen class and parametric methods, were reviewed before proposing the solution this project offers. Many neural network classifiers such as an Artificial Neural Network (ANN), Probabilistic Neural Network (PNN), have already been investigated for the classification of Power Quality disturbances. A two-dimensional Encoder-based Convolutional Neural Network (CNN) as used in this project, is a novel architecture and is an emerging method for PQD classification.

This method was chosen over other algorithms due to the advantages CNN offers. The computational time of the Encoder-based 2D CNN classifier is relatively less among other methods. Feature extraction and compression done by the encoder also significantly reduces the model size and number of model parameters. The process followed in the course of this project has been illustrated in the following sub-section.

2.2 Process Flow

The process followed to generate and classify the Power Quality Disturbances has been represented in the following flowchart 2.1. The concept behind each stage including the algorithms used have been discussed in the upcoming chapters.

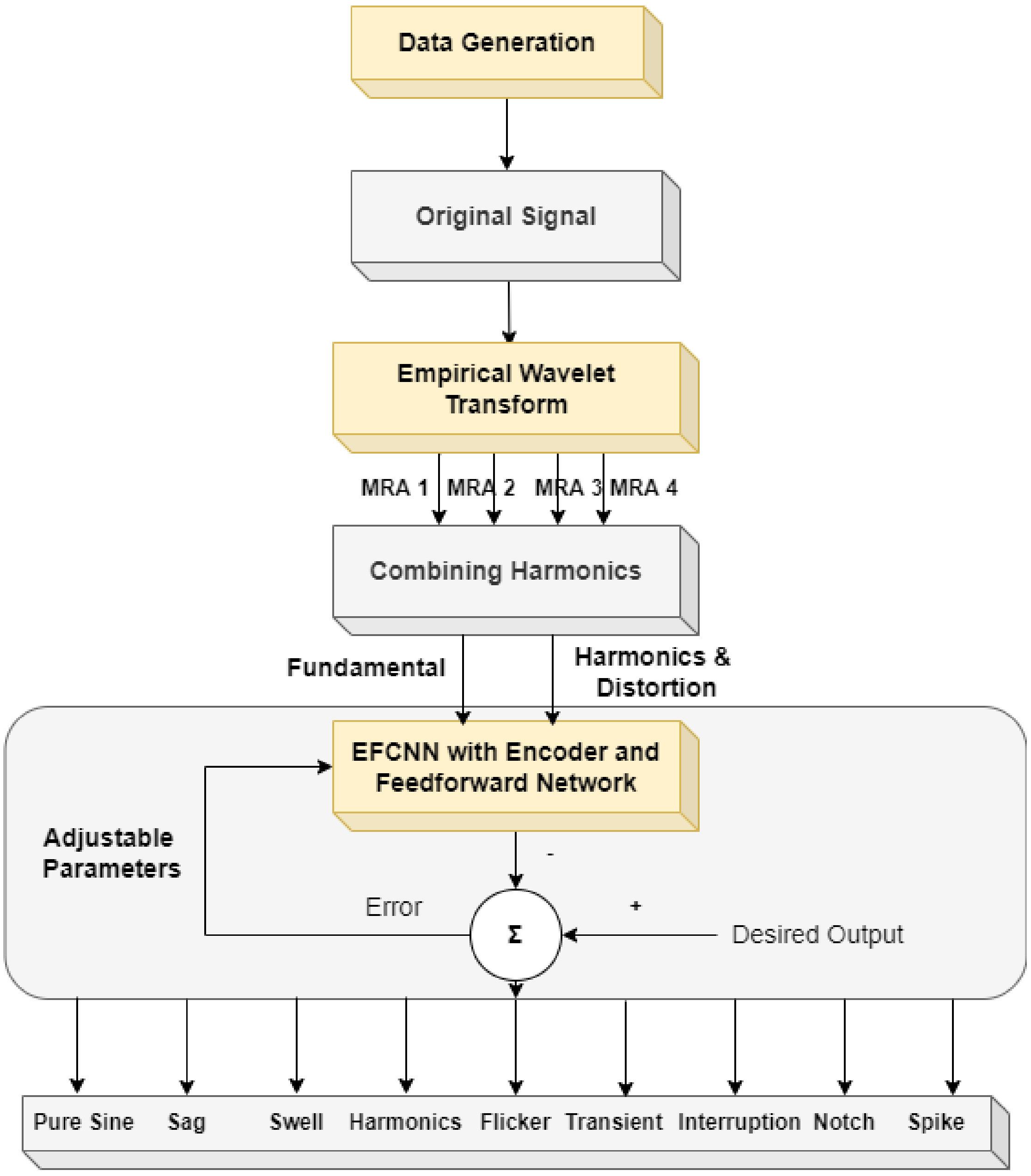


Figure 2.1: *Process flow diagram*

CHAPTER 3

DATA GENERATION AND EMPIRICAL WAVELET TRANSFORM

3.1 Data generation

The data is generated from the mathematical model which is defined for 16 classes of power quality disturbances or PQD in short using MATLAB. These 16 classes of PQD is further classified into one pure class, eight single or unique disturbances and seven combined disturbances. Each class is generated for five different Signal to Noise Ratio (SNR) of 15 dB, 20 dB, 30 dB, 40 dB and a no noise case. For each class, a total of 1080 samples per SNR were generated using parametric equations and range variations as mentioned in Table 3.1.

Class Labels	PQD Events	Equation	Parameters
C0	Normal	$y(t) = A\sin(2\pi f\text{t-}\theta)$	$49.9 \leq f \leq 50.1, 0 \leq \theta \leq 180$
C1	Sag	$y(t) = A[(1-\alpha(u(t-t_1)-u(t-t_2))\sin(2\pi ft-\theta)]$	$0.2 \leq \alpha \leq 0.9$, $0.5T \leq$ (t_2-t_1)$\leq 9T$
C2	Swell	$y(t) = A[(1+\alpha(u(t-t_1)-u(t-t_2))\sin(2\pi ft-\theta)]$	$0.1 \leq \alpha \leq 0.8$, $0.5T \leq$ (t_2-t_1)$\leq 9T$
C3	Interruption	$y(t) = A[(1-\alpha(u(t-t_1)-u(t-t_2))\sin(2\pi ft-\theta)]$	$0.9 \leq \alpha \leq 1$, $0.5T \leq$ (t_2-t_1)$\leq 9T$
C4	Harmonics	$y(t) = A[\sin(2\pi ft-\theta) + \sum a_i \sin(2\pi f_i t-\theta_i)]$	$0 \leq \theta \leq 2\pi, \alpha_1 = 1, 0.1 \leq \alpha_3 0.15, \alpha_5 = \alpha_3 * 5/3, \alpha_7 = \alpha_3 * 7/3, f_i \neq f$
C5	Transient	$y(t) = A[\sin(2\pi ft-\theta) + \alpha_t exp(-\frac{(t-t_3)}{\tau})(u(t-t_3)-u(t-t_4))\sin(2\pi f_t t)]$	$0.1 \leq \alpha_t \leq 0.8, 0.5T \leq t_4-t_3 \leq 3T, 300 \leq f_t \leq 3kHz, 8ms \leq \tau \leq 38ms$
C6	Fluctuation	$y(t) = A[[1+\alpha_{ff}\sin(2\pi\beta_{ff}t)]\sin(2\pi ft-\theta)]$	$0.1 \leq \alpha_{ff} \leq 0.9, 5 \leq \beta_{ff} \leq 10$
C7	Notch	$y(t) = A[\sin(2\pi ft-\theta) + \alpha sign(\sin(2\pi ft-\theta)) \times [\sum_{n=0}^{9} u(t-(t_1+0.02n))-u(t-(t_2+0.0205n))]]$	$0.15 \leq \alpha \leq 0.4, 0 \leq t_1 \leq 0.061, 0.01T \leq t_2-t_1 \leq 0.05T$

Class Labels	PQD Events	Equation	Parameters
C8	Spike	$y(t) = A[\sin(2\pi ft - \theta) - \alpha sign(\sin(2\pi ft - \theta)) \times [\sum_{n=0}^{9} u(t - (t_1 + 0.02n)) - u(t - (t_2 + 0.0205n))]]$	$0.15 \leq \alpha \leq 0.4, 0 \leq t_1 \leq 0.061, 0.01T \leq t_2 - t_1 \leq 0.05T$
C9	Sag + Harmonics	$y(t) = A[(1 - \alpha(u(t - t_1) - u(t - t_2)) \sin(2\pi ft - \theta) + \sum a_i \sin(2\pi f_i t - \theta_i)]$	$0.1 \leq \alpha \leq 0.9$, $0.5T \leq$ (t_2-t_1)$\leq 9T, , \alpha_1 = 1, 0.1 \leq \alpha_3 \leq 0.15, \alpha_5 = \alpha_3 * 5/3, \alpha_7 = \alpha_3 * 7/3$
C10	Swell + Harmonics	$y(t) = A[(1 + \alpha(u(t - t_1) - u(t - t_2)) \sin(2\pi ft - \theta) + \sum a_i \sin(2\pi f_i t - \theta_i)]$	$0.1 \leq \alpha \leq 0.8$, $0.5T \leq$ (t_2-t_1)$\leq 9T, , \alpha_1 = 1, 0.1 \leq \alpha_3 \leq 0.15, \alpha_5 = \alpha_3 * 5/3, \alpha_7 = \alpha_3 * 7/3$
C11	Flicker + Harmonics	$y(t) = A[[1 + \alpha_{ff} \sin(2\pi\beta_{ff} t)] \sin(2\pi ft - \theta) + \sum a_i \sin(2\pi f_i t - \theta_i)]$	$0.1 \leq \alpha_{ff} \leq 0.9, 5 \leq \beta_{ff} \leq 25, \alpha_1 = 1, 0.1 \leq \alpha_3 \leq 0.15, \alpha_5 = \alpha_3 * 5/3, \alpha_7 = \alpha_3 * 7/3$
C12	Sag + Flicker	$y(t) = A[[(1 - \alpha(u(t - t_1) - u(t - t_2)) \sin(2\pi ft - \theta)] \sin(2\pi ft - \theta)[1 + \alpha_{ff} \sin(2\pi\beta_{ff} t)]]$	$0.1 \leq \alpha \leq 0.9$, $0.5T \leq$ (t_2-t_1)$\leq 9T, 0.1 \leq \alpha_{ff} \leq 0.2, 5 \leq \beta_{ff} \leq 10$
C13	Swell + Flicker	$y(t) = A[[(1 + \alpha(u(t - t_1) - u(t - t_2)) \sin(2\pi ft - \theta)] \sin(2\pi ft - \theta)[1 + \alpha_{ff} \sin(2\pi\beta_{ff} t)]]$	$0.1 \leq \alpha \leq 0.9$, $0.5T \leq$ (t_2-t_1)$\leq 9T, 0.1 \leq \alpha_{ff} \leq 0.2, 5 \leq \beta_{ff} \leq 10$
C14	Sag + Transient	$y(t) = A[[(1 - \alpha(u(t - t_1) - u(t - t_2))] \sin(2\pi ft - \theta) + \alpha_t exp(-\frac{(t-t_3)}{\tau})(u(t - t_3) - u(t - t_4)) \sin(2\pi f_t t)]$	$0.1 \leq \alpha \leq 0.8$, $0.5T \leq$ (t_2-t_1)$\leq 9T, 0.1 \leq \alpha_t \leq 0.8, 0.5T \leq t_4 - t_3 \leq 3T, 300 \leq f_t \leq 3kHz, 8ms \leq \tau \leq 38ms$
C15	Swell + Transient	$y(t) = A[[(1 + \alpha(u(t - t_1) - u(t - t_2))] \sin(2\pi ft - \theta) + \alpha_t exp(-\frac{(t-t_3)}{\tau})(u(t - t_3) - u(t - t_4)) \sin(2\pi f_t t)]$	$0.1 \leq \alpha \leq 0.8$, $0.5T \leq$ (t_2-t_1)$\leq 9T, 0.1 \leq \alpha_t \leq 0.8, 0.5T \leq t_4 - t_3 \leq 3T, 300 \leq f_t \leq 3kHz, 8ms \leq \tau \leq 38ms$

Table 3.1: Mathematical Model

The synthetic signals are generated with a sampling frequency of 3.2kHz with a time window from 0 to 0.2 s generating a total of 640 sampling points per signal. The parameter *A* represents the RMS voltage amplitude per unit of the signal which is considered to be a constant (equal to 1.414). The parameter α defines the intensity of the disturbances.The step function $u(t)$ models the duration of the disturbance on the signal. For flicker, parameters β_{ff} and α_{ff} defines the frequency and magnitude variation respectively. In the case of harmonics, only the odd harmonics i.e., the 3rd, 5th and 7th harmonic components are involved. Lastly, the parameters α_t, f_t and τ represents the magnitude, frequency variations and duration of transience respectively.

Following are the samples of waveforms generated with and without noise addition for each of the PQDs:

1. Without Noise:

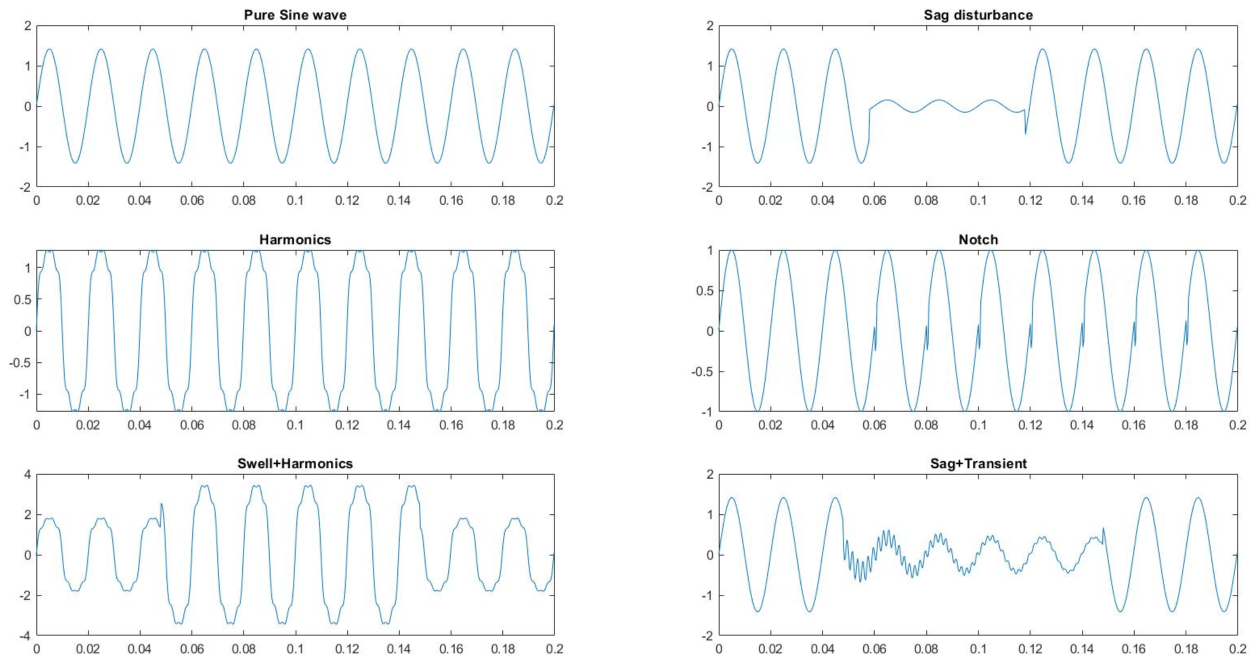


Figure 3.1: *PQD waveforms without noise a) Pure sine, b) Sag, c) Harmonics, d) Notch, e) Swell + Harmonics, f) Sag + Transient.*

2. With added white gaussian noise:

Additive White Gaussian Noise (AWGN) of SNRs of 15dB, 20dB, 30dB, 40dB were added. It is additive since it is added to any noise that might be intrinsic to the information system. Moreover, this noise is statistically independent of the signal. Just like the white colour which is composed of all frequencies in the visible spectrum, white noise refers to uniform power across the whole frequency band.

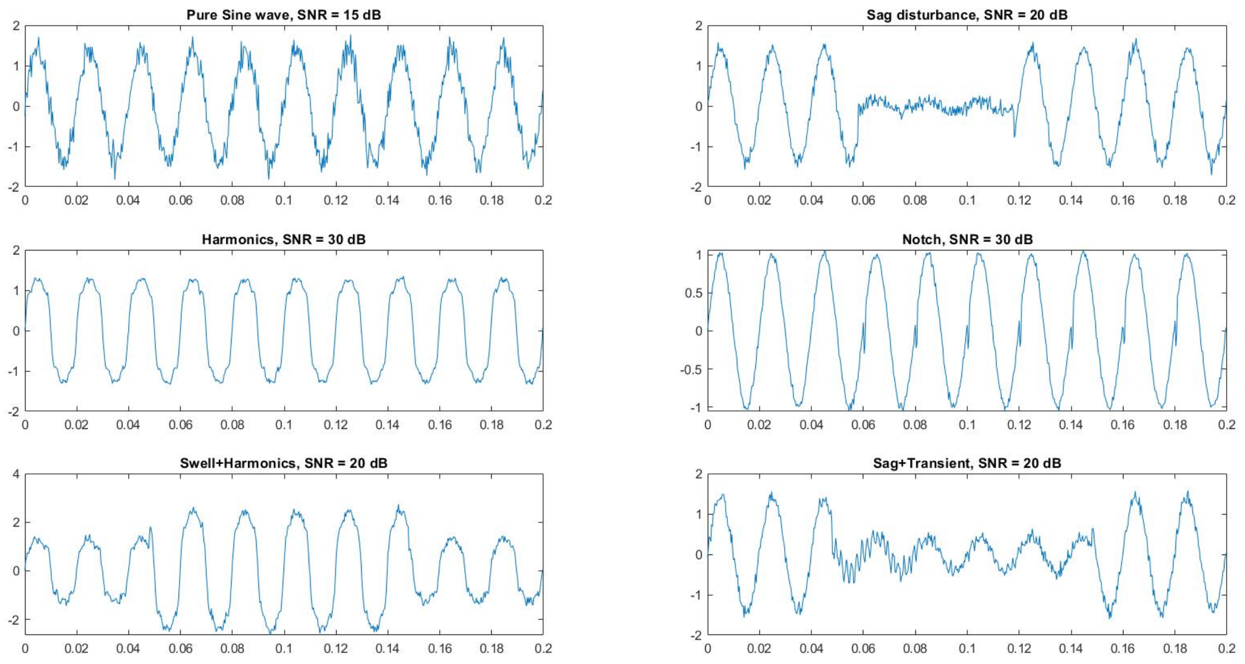


Figure 3.2: *PQD waveforms with added WGN a) Pure sine, b) Sag, c) Harmonics, d) Notch, e) Swell + Harmonics, f) Sag + Transient.*

After the generation of data for 5 different SNRs, the obtained dataset is saved as a matfile which contains 86400 samples (17280 samples per SNR and 1080 samples per class per SNR).

3.1.1 Algorithm

The algorithm followed to generate data using MATLAB has been explained in this section. Initially an array of values 15, 20, 30, 40, 100 is declared that represents SNR values. An array t of size 640 is initialised . A 'for' loop is initiated to iterate over SNR array. Nested 'for' loops were used to vary parameters like Amplitude (α), Start time (t1), Duration (t2-t1), Phase angle (ϕ), etc. to obtain over 1080 samples of each PQD waveform per SNR (from the equations given in Table 3.1) and generate the dataset to be used for the further classification process.

The two conditions for which the above process was repeated, and waveforms were simulated are given below:

i Constant frequency case with fundamental frequency of 50Hz.

ii With frequency variation (49.9, 50 and 50.1Hz). Each of the signal y generated were vertically concatenated with a matrix z that was initialised with array t at the start. The matrix z alongside a column vector class holding each of the labels was written into an excel file.

The code is given in Appendix A.1. The dataset was stored with a time step of 0.3125ms in the time window of 0 to 0.2s. The last field holds the class labels of each of the PQDs with 0 for pure sine wave, 1 for sag, 2 for swell, 3 for interruption, 4 for harmonics, 5 for oscillatory transients, 6 for fluctuations, 7 for notch, 8 for spike, 9 for Sag + Harmonics, 10 for Swell + Harmonics, 11 for Flicker + Harmonics, 12 for Sag + Flicker, 13 for Swell + Flicker, 14 for Sag + Transient, 15 for Swell + Transient.

3.2 Empirical Wavelet Transform

The matfile obtained from the data generation which constitutes of non-linear and non-stationary signals is decomposed into two components namely the fundamental and harmonic components using the signal decomposition method called Empirical Wavelet Transform (EWT). EWT creates a multiresolution analysis (MRA) of a signal using an adaptive wavelet subdivision scheme [11]. The method starts with segmentation of the Fourier spectrum of the signal which determines the performance of the EWT. Then wavelet filter banks are designed in the Fourier domain with the estimated frequencies as center frequency. The MRAs are formed sequentially from higher frequencies to lower frequencies (i.e MRA 1 corresponds to the highest frequency and the last MRA corresponds to the lowest frequency present in the original signal).

The original signal's energy is partitioned by the EWT coefficients into separate passbands. Based on the frequency information, EWT decomposes the time dependent signal into distinct components or MRAs. The original signal can be retrieved by summing up the MRAs at each point in time.

3.2.1 Algorithm

i Apply Fast Fourier Transform (FFT) for the input signal to determine the dominant frequencies present in the signal and obtain the Fourier spectrum of the processed signal.

ii Identify the boundaries of the spectrum and compute the local minima between two consecutive frequencies shown in (fig).

iii Segment the Fourier Spectrum.

iv Design the empirical wavelet filter bands in the frequency domain and four MRAs are obtained.

v Inverse fast Fourier transform (IFFT) is used to obtain the approximation and detail coefficients of the filtered spectrum.

vi Using the coefficients, the energy of the original signal and the MRAs are evaluated.

vii Using these separated components, the component with the highest energy is considered as the fundamental component and every other MRAs are summed up to produce the harmonic component.

viii To verify the decomposition, the fundamental and harmonic components are summed up to reconstruct the original signal.

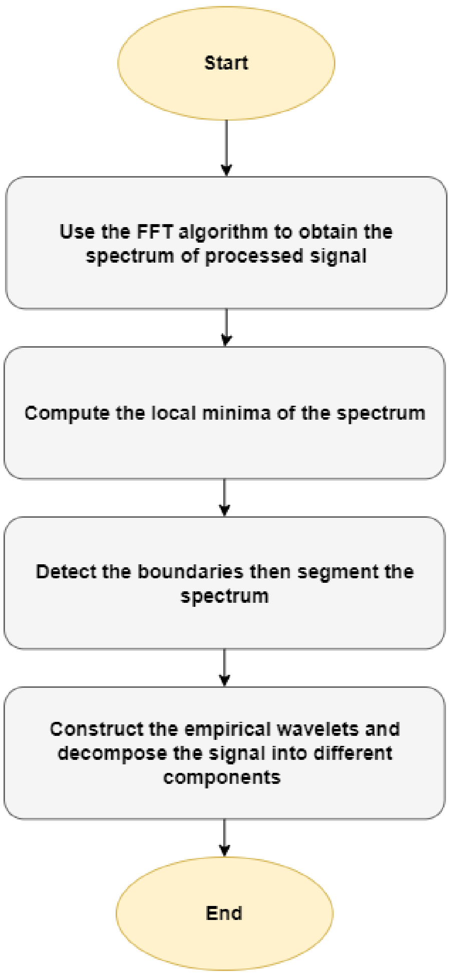

Start
Use the FFT algorithm to obtain the spectrum of processed signal
Compute the local minima of the spectrum
Detect the boundaries then segment the spectrum
Construct the empirical wavelets and decompose the signal into different components
End

Figure 3.3: *Flowchart for EWT*

MATLAB was used for the implementation of EWT method for our model and the method is explained graphically below along with the segmentation of Fourier spectrum.

1. Original Signal - Harmonics

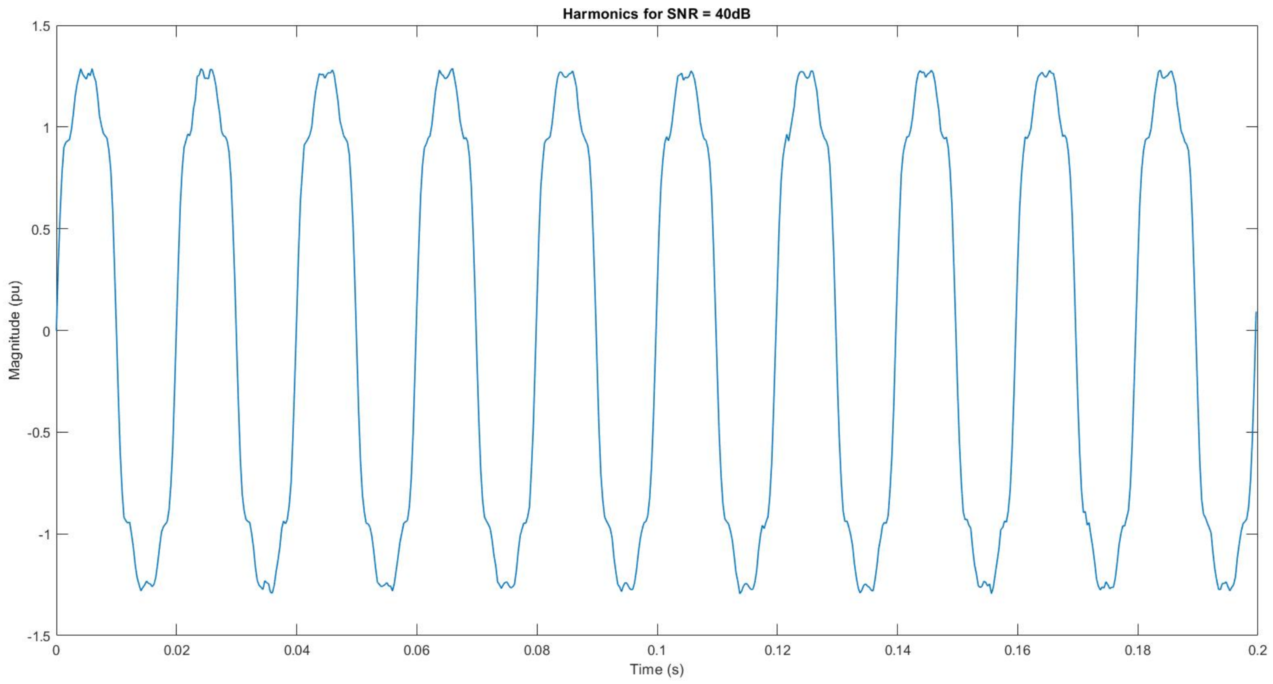


Figure 3.4: *Original Signa*l

2. Harmonics - Fourier spectrum analysis

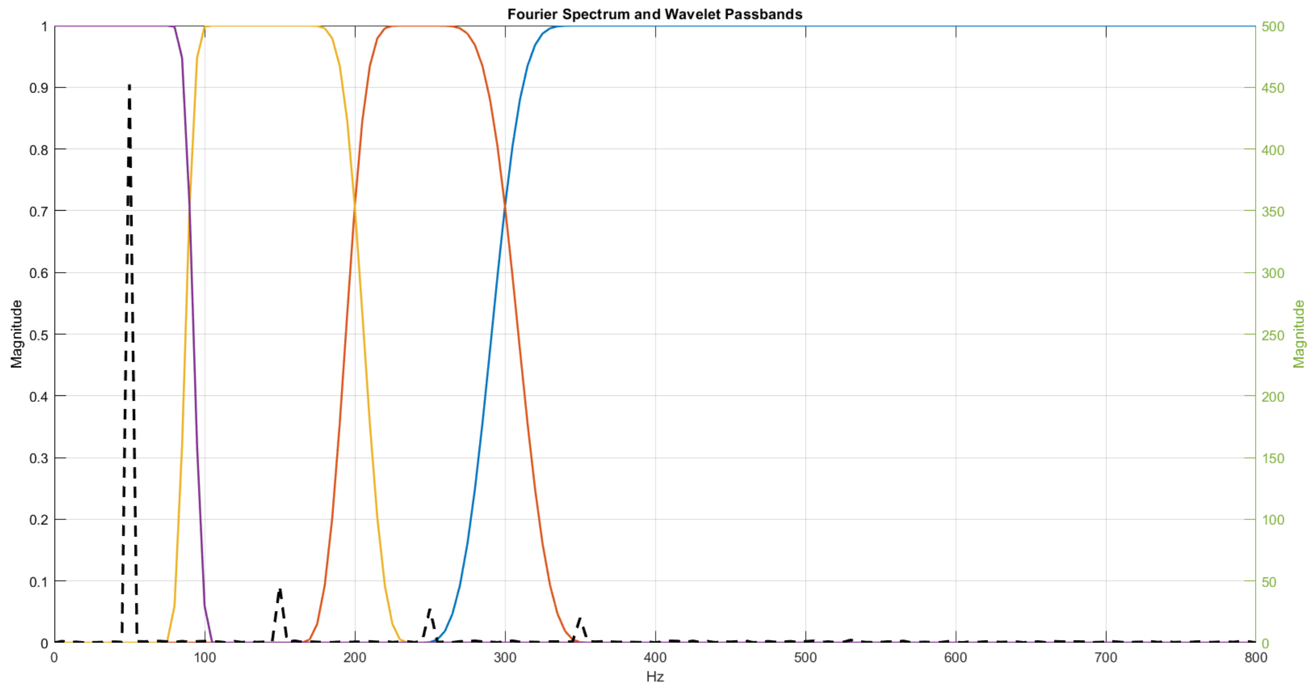


Figure 3.5: *Fourier Spectrum with designed wavelet Filter Banks*

MRA	Start Frequency (Hz)	End Frequency (Hz)	Peak Frequencies (Hz)
1	300	1600	350
2	200	300	255
3	90	200	150
4	0	90	50

Table 3.2: Peak frequencies and Passbands of filters

From the above plot, the information about filter banks and peak frequencies can be retrieved and is shown in Table 3.2. The above plot explains the segmentation, identification of dominant frequencies (peak frequencies) and the design of empirical wavelet filter banks in EWT.

3. Harmonics - 40dB

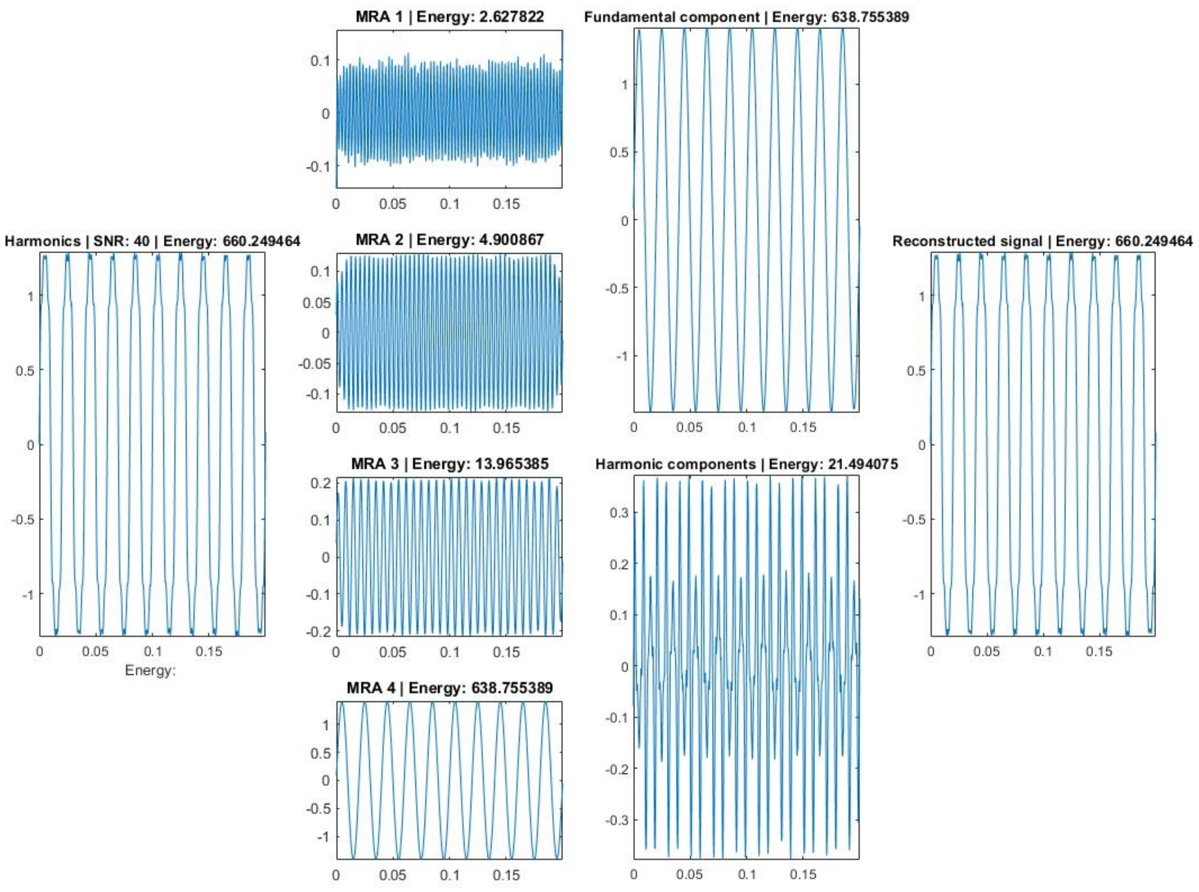


Figure 3.6: *EWT of Harmonics with SNR 40 dB*

4. Sag + Transient (30 dB)

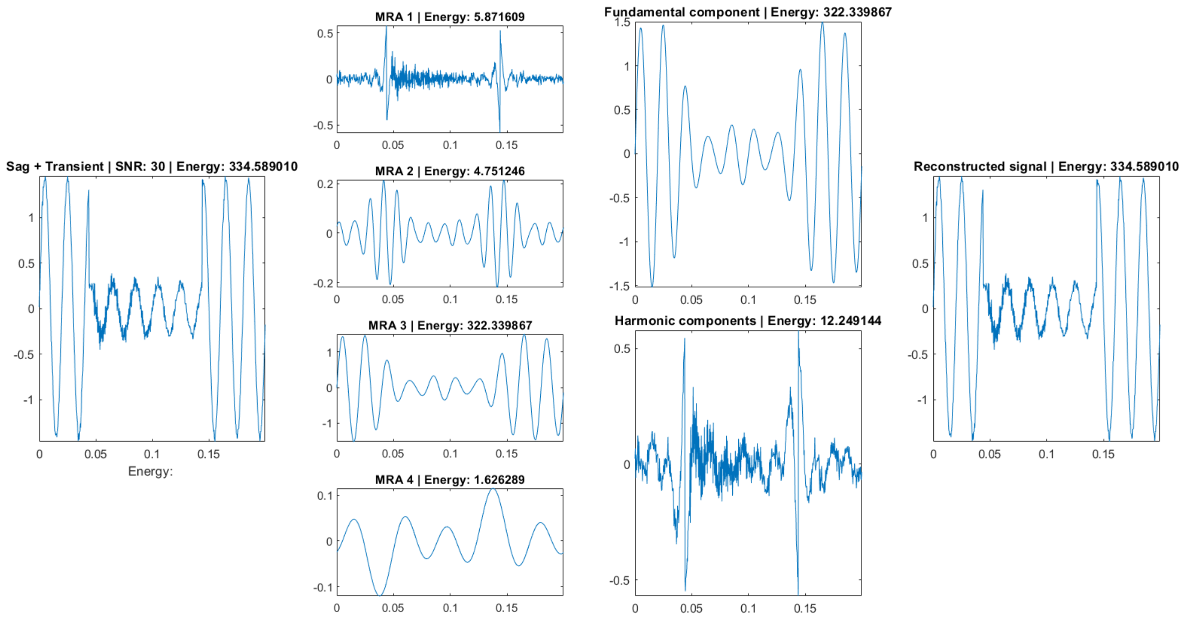


Figure 3.7: *EWT of Sag + Transient with SNR 30 dB*

5. Swell + Transient (No Noise)

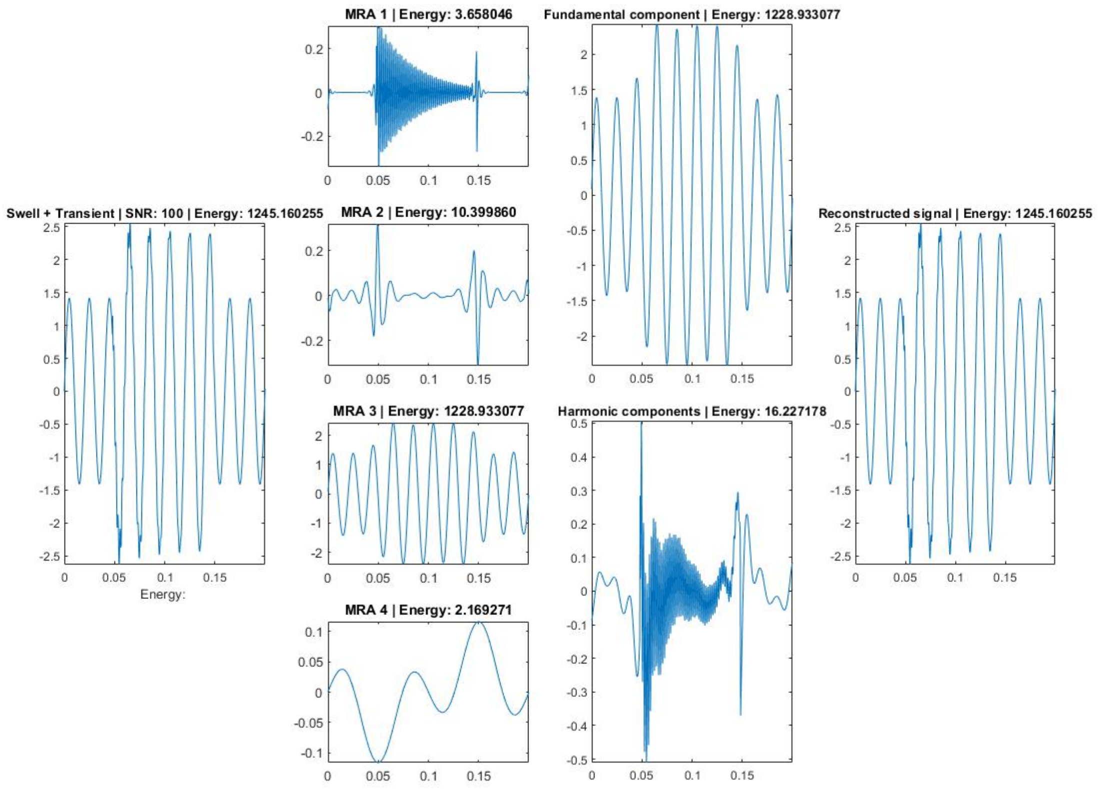


Figure 3.8: *EWT of Swell + Transient for No noise case*

Fig 3.7 and 3.8 explains about the effect of noise in the analysis of component resolution. Fig 3.7 involves a SNR of 30dB and Fig 3.8 is a case of no noise, it is observed that the harmonic component of the latter only contains the harmonics whereas the former figure contains both harmonics and noise.

6. Pure Sine wave (15dB)

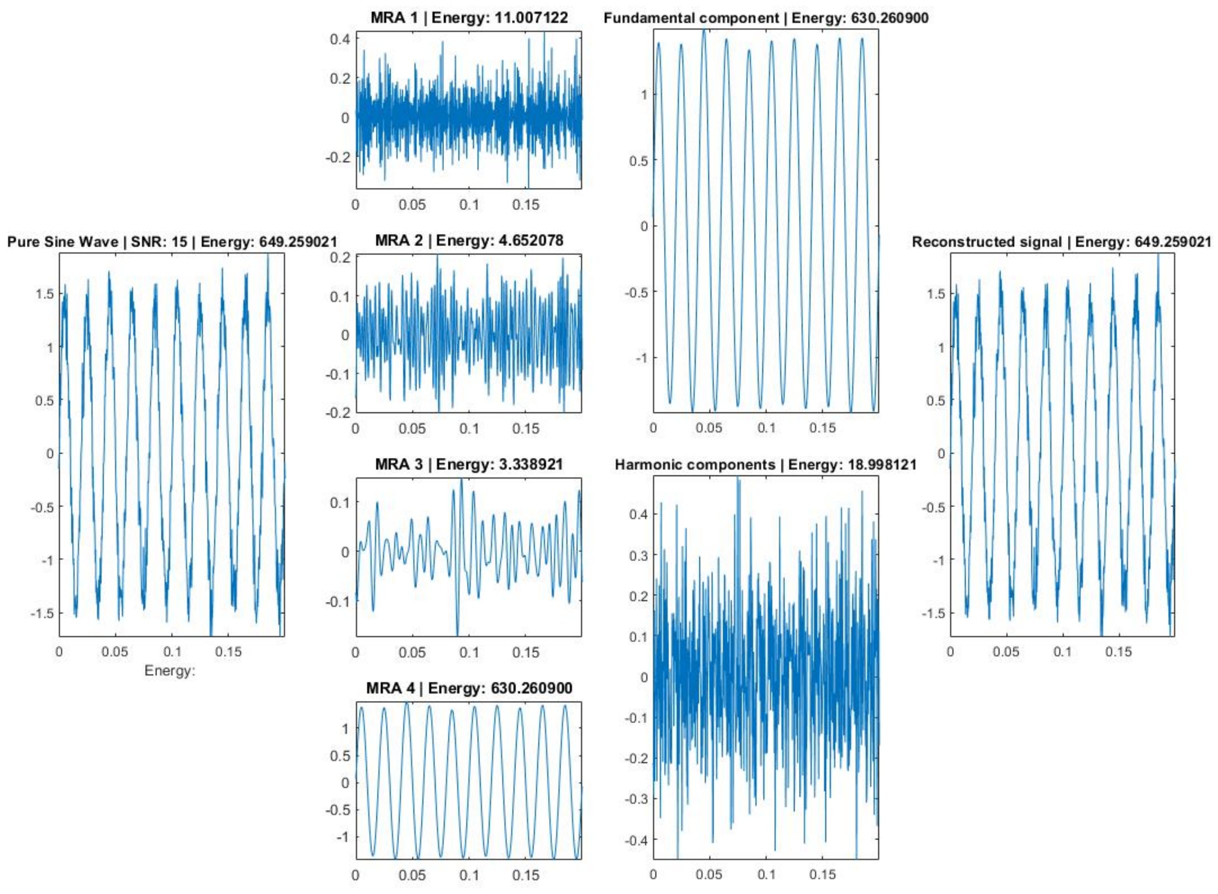


Figure 3.9: *EWT of Pure Sine with SNR 15 dB*

The above graphs gives an idea about the working of EWT and how the components are separated. Along with that, the power spectral information of harmonics is obtained to visually represent the dominant frequencies and the wavelet filters. The separation of harmonic and fundamental can be clearly observed in Fig 3.9 where MRA 4 with the maximum energy represents the fundamental and the rest of the MRAs are summed up to get the harmonic component which represents the distortion due to noise.

The processed signals output from the EWT method is of dimension $86400 \times 2 \times$ $\times 640$ and is used as the input for the deep learning model which will be discussed in the next section.

CHAPTER 4

DEEP LEARNING PIPELINE

In this thesis, a novel architecture consisting of two stages is proposed. The first stage is an encoder module which learns a latent space representation of the input data. This encoding is fed to the second stage, which consists of a simple neural network classifier to identify the type of disturbance.

Two different approaches for training this pipeline were experimented. In the first approach, we train the encoder with a auxiliary decoder network. This structure forms a Convolutional Autoencoder architecture. The neural network classifier is trained separately using the encoded feature space as input. In the second approach, we train the encoder and classifier together in an end-to-end manner to take the decomposed signal as input and classify the disturbances.

The first section gives a general description of Encoders, Variational Autoencoders, and the optimizers used. The following sections detail the two different training approaches mentioned above.

4.1 Background

Encoders are a class of neural network architectures that output a feature map/vector/tensor for each input. These feature vectors, also called latent space representations, contain high-level representative features extracted from the input. Encoders are generally used alongside decoders for training. These encoders can be of multiple types: Fully Connected Encoders, Convolutional Encoders, Recurrent Encoders. In this thesis, Convolutional Encoders are employed.

Autoencoders are a class of deep generative models that aim to learn a latent space representation of high-level features from the input [12]. They consist of an encoder and decoder neural network. The general idea of Autoencoders is to learn the best encoding-decoding scheme using an iterative optimisation process. At each iteration the Autoencoder architecture (the encoder followed by the decoder) is fed with some data, and the output is compared with the initial data. The encoder tries to produce a feature-rich representation of the input. This encoding is then validated and refined by attempting to regenerate the input from the encoding. The Autoencoder architecture creates a bottleneck for data that ensures only the main structured part of the information can go through and be reconstructed [13].

Variational autoencoders (VAEs) are a special type of Autoencoders whose training is regularised to avoid overfitting, and ensure that the latent space has good properties that enable generative process. In this case, the input is encoded as a distribution over the latent space, instead of just a point in the latent space. The reconstruction is generated by sampling a point from this distribution and using it as a point encoding. This helps the VAE produce feature-rich encodings that provide better reconstructions.

4.2 VAE Based Training - (Autoencoder + Neural Network Model)

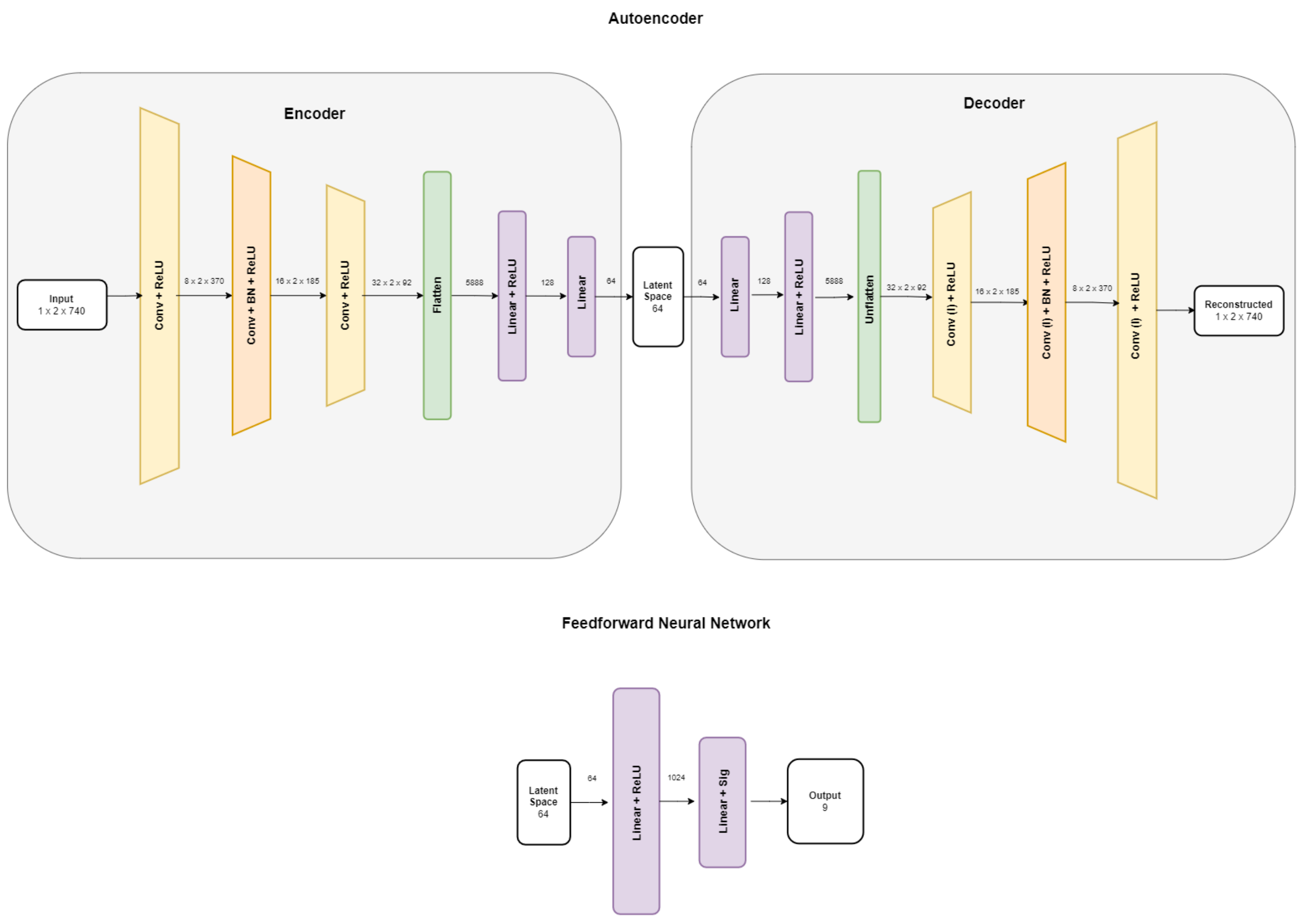


Figure 4.1: *Structure of AE + NN Classifier*

This approach seeks to decouple the training-time and inference-time architectures for the model. The training-time architecture employs an auxiliary decoder to train the encoder. The auxiliary decoder takes the feature encoding and tries to reconstruct the input signal, similar to a VAE. At the end of the VAE training, the decoder is able to reconstruct the input signal only using the encoding formed by the encoder. This shows that the encoding has all necessary features of the input signal in the form of a compressed representation. Hence, the encoding can be used in place of the original signal, to classify the disturbance. Generating a compressed encoding in this fashion allows for faster processing and better classification. The VAE is trained in an unsupervised manner.

The neural network classifier is then trained separately, on the encodings learnt by the encoder. It uses the high-level features in the encoder to classify the disturbance, as opposed to using the low level features from the raw data. The former approach provides better classification accuracy as well as improved run-times.

The inference-time architecture consists of the Encoder part of the trained VAE, and the trained Classifier. The Encoder has been trained to compress the input and extract features. The Neural Network Classifier uses this encoding to identify the type of disturbance.

4.3 End-to-end Training - (Encoded Feature CNN Model)

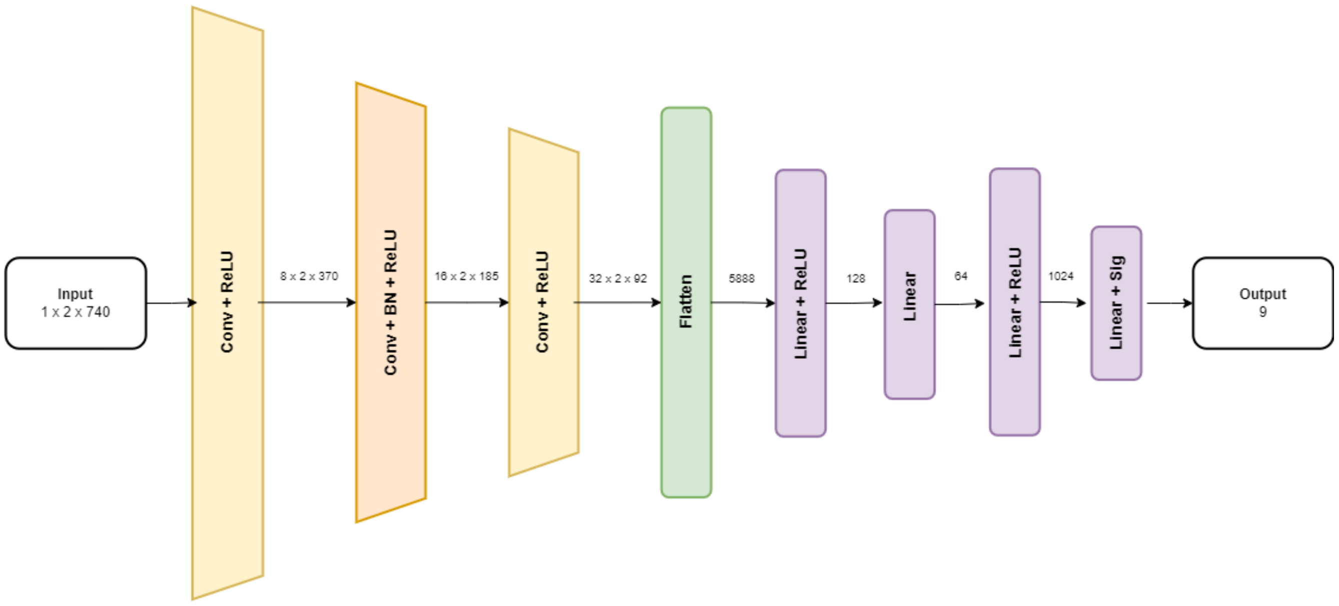


Figure 4.2: *Structure of EFCNN Classifier*

In this approach, the entire pipeline is trained end-to-end. Hence, the auxiliary decoder is not required in this case. The Encoder is directly trained with the Neural Network Classifier to classify disturbances. This functions as a regular 2D CNN due to the Convolutional layers in the encoder.

4.4 Conventional 2D CNN

In order to compare our model, we choose a standard 2D CNN architecture and train it to classify the disturbances. The model consists of four convolutional blocks. Each block contains a Convolutional layer followed by ReLu, MaxPool and BatchNorm layers. The Convolutional layers are followed by two linear layers to classify the signal.

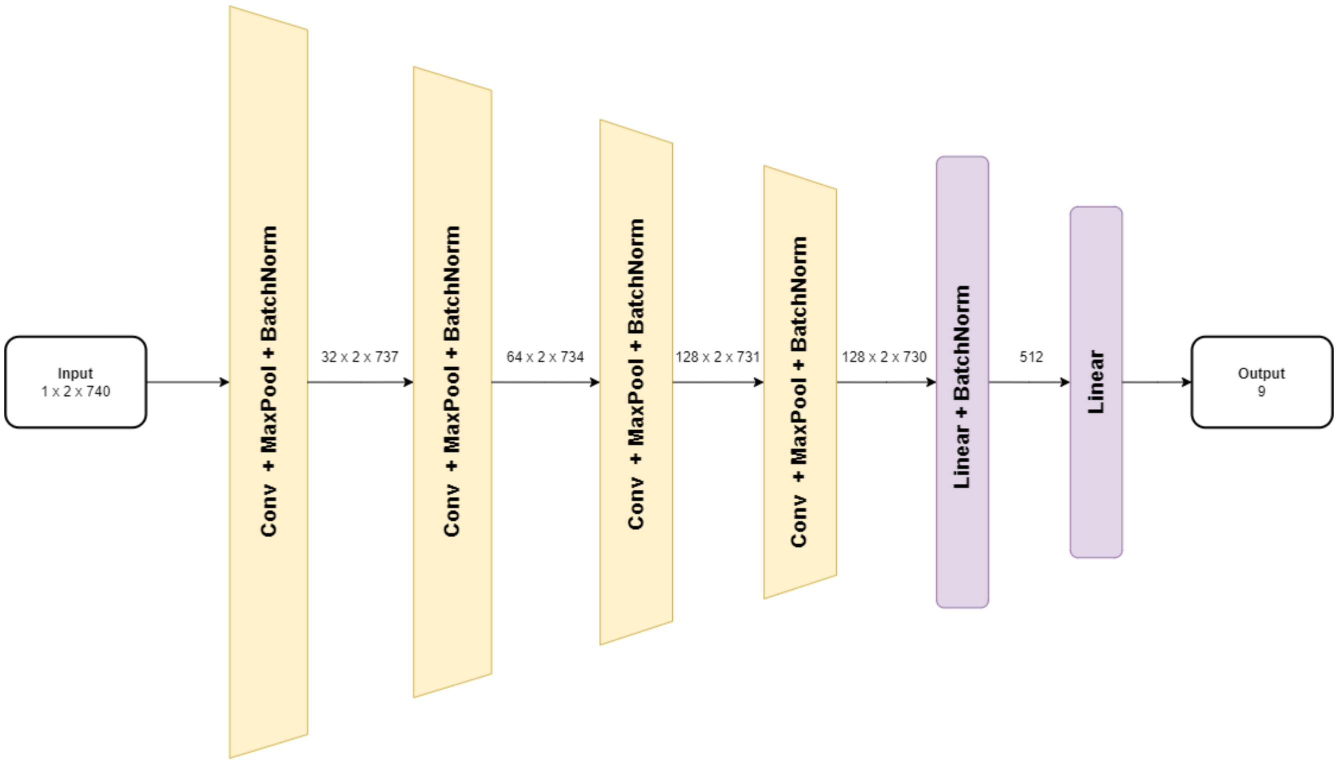

Input
1 x 2 x 740
Conv + MaxPool + BatchNorm
32 x 2 x 737
Conv + MaxPool + BatchNorm
64 x 2 x 734
Conv + MaxPool + BatchNorm
128 x 2 x 731
Conv + MaxPool + BatchNorm
128 x 2 x 730
Linear + BatchNorm
512
Linear
Output
9

Figure 4.3: *Structure of 2D CNN Classifier*

CHAPTER 5
RESULTS AND DISCUSSIONS

5.1 EFCNN Results

The results are described with the use of accuracy values, graphs, and confusion matrices for different cases. Two main cases were considered, namely:

i Single and Combined power quality signals without noise

ii Single and Combined power quality signals with noise variation

In each of these cases, 1080 samples were taken for each of the 16 categories of signals. 10% of these samples were taken as testing samples, 10% as validation samples, and the rest 80% samples were used to train the model (576 training samples).

A confusion matrix is a table that is often used to describe the performance of a classification model (or ”classifier”) on a set of test data for which the true values are known. The two axes of the matrix represent the true class and predicted class. The rows represents the true values belonging to the samples and the columns represents the values predicted by the classifier. For example, considering the 4th element in the 4th row of the confusion matrix, it shows that out of 527 Interruption signals, the model predicted all of the 527 samples correctly. However, considering the 7th element in the 7th row, it shows that out of 527 samples belonging to the ’Notch’ class, 521 of them were correctly classified by our model and 6 of them were mistaken for the ’Spike’ class. Hence, the diagonal elements represent the number of samples correctly classified by our model. Accuracy, precision, and prevalence can be obtained from the confusion matrix for each class.

$$Accuracy = \frac{\textit{Number of correct samples}}{\textit{Number of testing samples}} \times 100 \tag{5.1}$$

$$ClassAccuracy = \frac{\textit{Number of correctly classified samples in the specific class}}{527} \times 100 \tag{5.2}$$

$$TotalAccuracy = \frac{\textit{Sum of diagonal elements}}{\textit{Number of testing samples}} \times 100 \tag{5.3}$$

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0	533	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2
1	0	550	0	9	0	0	0	0	0	0	0	0	5	0	0	0	2
2	0	0	544	0	0	0	0	0	0	0	0	0	0	0	0	1	0
3	0	3	0	527	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	539	0	0	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	517	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	567	0	0	0	0	1	1	0	0	0	0
7	0	0	0	0	0	0	0	521	6	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	13	532	0	0	0	0	0	0	0	4
9	0	0	0	0	0	0	0	0	0	534	0	0	0	0	0	0	0
10	0	0	1	0	1	0	1	0	0	0	547	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	551	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	529	0	0	0	0
13	0	0	1	0	0	0	0	0	0	0	0	0	0	496	0	0	0
14	0	1	0	0	0	1	1	0	0	0	0	0	0	0	541	0	0
15	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	556	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.1: EFCNN Confusion Matrix for No noise case

Number	**Class**
0	Pure Sine Wave
1	Sag
2	Swell
3	Interruption
4	Harmonics
5	Oscillatiory Transients
6	Fluctuations
7	Notch
8	Spike
9	Sag + Harmonics
10	Swell + Harmonics
11	Flicker + Harmonics
12	Sag + Flicker
13	Swell + Flicker
14	Sag + Transient
15	Swell + Transient
16	Others

Table 5.2: Class Number and its corresponding Class Name

Table 5.3 shows the performance of the EFCNN model for different signal-to-noise ratios. As expected, the performance of the model decreases with increase in the amount of noise added, i.e., lower SNR values have lower classification accuracy. As seen from the table, the model achieves near perfect classification accuracy if no noise is added. With increase in the noise level, the random distortions overshadow actual features in the input. This results in a noisy encoding and thereby a drop in classification accuracy.

This effect is more prominently seen in the case of disturbances like transients, notch, and spike, where the noise completely masks the disturbance. Disturbances like Sag, Swell, etc, are less affected by the addition of noise.

SNR (dB)	**EFCNN Accuracy**	**Average**
15	97.32	99.28
20	99.14	
30	99.77	
40	99.58	
No Noise	99.88	

Table 5.3: Accuracy of EFCNN for different SNR

5.2 Comparison with other models

Metric	**2D CNN**	**AE + NN**	**EFCNN**
Train time per epoch	48s	20.1 + 0.9 = 21s	15s
Inference time per sample	1.63ms	0.78ms	0.78ms
No. of parameters	48.59M	839.8k	839.8k
Model size	213.6MB	3.58MB	3.58MB

Table 5.4: Comparison of Performance Metrics between the different models proposed

From Table 5.4, it is seen that the EFCNN model outperforms both the AE + NN model, and the 2D CNN model. This improvement is especially large for noisy inputs. The EFCNN model sees a 5.7% improvement in accuracy compared to the AE + NN model, and a 5% improvement in accuracy compared to the 2D CNN model. As the noise level reduces, the accuracy of all three networks increase, and hence the relative improvement of the EFCNN model decreases.

Both, the AE + NN model and the EFCNN model employ an encoder to extract features from the input. As described later in this section, both these models are significantly faster and computationally lighter than the 2D CNN. In spite of this, these models obtain accuracies comparable to the 2D CNN, due to the fact that the encoders learn high-level features, which enable easier classification than the lower level features in the raw data.

The EFCNN considerably outperforms the AE + NN even though both models employ an encoder. In the former model, the latent space representation is directly refined to ensure better classification of the signal. This is in contrast to the AE + NN where the encoding was refined to enable better reconstructions of the input signal. A feature space that can be used for reconstruction (the AE + NN model) can also be used for classification (the EFCNN model), while the converse is not necessarily true. This is because if the feature space contains enough information to reconstruct the signal, it has enough features to classify it. Based on this, it can be said that the encoder in the former model learns a more representative feature space. However, the feature space in that case is not customized to the task at hand, while the one in the EFCNN is less representative but more customized to the task at hand.

SNR (dB)	**2D CNN**	**AE + NN**	**EFCNN**
15	92.7	92.08	97.32
20	95.4	94.85	99.14
30	98.1	97.35	99.77
40	98.7	98.13	99.59
No Noise	99.1	98.05	99.88
Overall	96.9	96.15	99.28

Table 5.5: Accuracy comparison between the different models proposed

The EFCNN and AE + NN model, both provide significant improvements in terms of run-time, compared to the 2D CNN model. Both these models have the same inference time due to the fact that their inference-time architectures are the same. They only differ in their training time architectures. Due to the same reason, they also have the same model size and number of parameters. However, the EFCNN model trains 28.6% faster than the AE + NN model, owing to its simple training time architecture. On the other hand, the AE + NN model employs an auxiliary decoder which slows down the training process.

Both these modes are however significantly faster than the 2D CNN architecture. They can run inference nearly 2.1x faster than the 2D CNN, while also achieving 60x reduction in number of parameters and model size. The later two metrics indicate that the EFCNN model is computationally lighter than the 2D CNN. This may be useful for real-time deployment on embedded devices.

CHAPTER 6
CONCLUSION

6.1 Conclusion

In the context of modern grids, it can be suggested that there is a need to monitor the nature of PQ events on these sites and prepare detection / classification schemes accordingly.

This project presented a novel approach based on Empirical Wavelet Transform and 2D Convolutional Encoders, to classifying power quality disturbances. The proposed approach provides considerable improvements over previous models, in terms of both accuracy and run-time. The proposed approach can identify and classify both single and combined power quality disturbances.

The Encoder-based 2D CNN classifier was used for the classification of PQ disturbances. 2D CNN performs convolution operations on input data with each kernel and creates the features for classification. The encoding layers help in compressing the data samples and reduces the model size significantly. The various stages involved such as, feature extraction and classification already increase the computational burden. Therefore, optimal feature selection is also indispensable to reduce the memory usage and computational time.

The proposed method was evaluated for different noise cases and a simulated data set. It was observed that in cases with discrepancies in the signal, the model worked better when the simulated data signals were processed with EWT. The comparative results depict that the proposed method is most suited for simulated data sets and noisy environments.

6.2 Future Scope

The modern improvement in artificial intelligence-based algorithms and Deep-learning based algorithms have been added to the PQ analysis technology. Hence, it could be a significant focused area for power quality disturbances recognition. As long as there is dependence on the conventional modes of electrical generation and transmission, we need to put in efforts to improve its efficiency by identifying and reducing power quality disturbances in order to cut down costs, reduce losses, and build a sustainable future. The project depicts a system of classification using an Encoder-based 2D CNN with the help of EWT and can serve as a learning platform. The model is to be further developed to categorize more combined classes, such as three or more classes of power quality disturbances combined and the real time use would require alterations and to tailor-made structure to fit the system.

APPENDIX A

CODE ATTACHMENTS

A.1 Data Generation and Signal Decomposition code on MATLAB

```
%% Data Generation

noise=[15, 20, 30, 40, 100];
t=[0 :0.0003125:0.2-0.0003125];
z=t;
x=0;
cl=x;

for i =1:5
    fprintf('SNR: %i dB \n', noise(i))
% Sine wave
x=0;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f= 49.9:0.0001852:50.1
y=awgn(1.4142*sin(2*pi*f*t),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
end
figure(1)
plot(t,y)
title('Pure Sine wave')

%Sag  1080 samples*5
x=1;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1
for alpha=0.2:0.063:0.9
 for t1=0.04:0.002:0.058
y=awgn(1.4142*(1-alpha*((heaviside(t-t1)-heaviside(t-(t1+0.14))))). ...
    *sin(2*pi*f*t),noise(i)); %7cycles
z=vertcat(z,y);
cl=vertcat(cl,x);
y1=awgn(1.4142*(1-alpha*((heaviside(t-t1)-heaviside(t-(t1+0.1))))). ...
    *sin(2*pi*f*t),noise(i));%5cycles
z=vertcat(z,y);
cl=vertcat(cl,x);
y=awgn(1.4142*(1-alpha*((heaviside(t-t1)-heaviside(t-(t1+0.06))))). ...
    *sin(2*pi*f*t),noise(i));%3cycles
z=vertcat(z,y);
cl=vertcat(cl,x);
end
end
end
figure(2)
plot(t,y);
title('Sag disturbance');

%Swell  1080 samples*5
x=2;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1
for alpha=0.1:0.063:0.8
 for t1=0.04:0.002:0.058
y=awgn(1.4142*(1+ alpha*((heaviside(t-t1)-heaviside(t-(t1+0.14))))). ...
    *sin(2*pi*f*t),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
y=awgn(1.4142*(1+ alpha*((heaviside(t-t1)-heaviside(t-(t1+0.1))))). ...
    *sin(2*pi*f*t),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
y=awgn(1.4142*(1+ alpha*((heaviside(t-t1)-heaviside(t-(t1+0.06))))). ...
    *sin(2*pi*f*t),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
end
end
end
figure(3)
plot(t,y);
title('Swell disturbance');

%Interruption  1080 samples*5
x=3;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1
for alpha=0.9:0.009:1
 for t1=0.04:0.002:0.058
y=awgn(1.4142*(1-alpha*((heaviside(t-t1)-heaviside(t-(t1+0.14))))). ...
    *sin(2*pi*f*t),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
y=awgn(1.4142*(1-alpha*((heaviside(t-t1)-heaviside(t-(t1+0.1))))). ...
    *sin(2*pi*f*t),noise(i));
z=vertcat(z,y);
```

```
cl=vertcat(cl,x);
y=awgn(1.4142*(1-alpha*((heaviside(t-t1)-heaviside(t-(t1+0.06))))). ...
    *sin(2*pi*f*t),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
end
end
end
figure(4)
plot(t,y);
title('Interruption');

%harmonics  1080 samples*5
x=4;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1
for alpha3=0.1:0.0014:0.15
for theta=0.628:0.628:6.28
alpha5=alpha3*3/5;
alpha7=alpha3*3/7;
alpha1=1;
y=awgn( 1.4142*alpha1* sin(2*pi*f*t+theta)+1.4142*alpha3* ...
    sin(3*2*pi*f*t+theta)+1.4142*alpha5*sin(5*2*pi*f*t+theta)+1.4142 ...
    *alpha7*sin(7*2*pi*f*t+theta),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
 end
end
end
figure(5)
plot(t,y)
title('Harmonics');

%Transient  1080 samples*5
x=5;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1     %3
for amp=0.2:0.2:0.8
for fn=300:600:3000
for t1=0.04:0.004:0.048
for tau=0.008:0.015:0.038
t2=t1+0.02; %duration = 1cycle
y=awgn( 1.4142*amp*sin(2*pi*50*t)+ 1.4142*amp*(heaviside(t-t2)- ...
    heaviside(t-t1)).*exp(t1-t/tau).*sin(2*3.14*fn*t),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
t2=t1+0.04; %duration = 2cycles
y=awgn( 1.4142*amp*sin(2*pi*50*t)+ 1.4142*amp*(heaviside(t-t2)- ...
    heaviside(t-t1)).*exp(t1-t/tau).*sin(2*3.14*fn*t),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
end
end
end
end
end
figure(6)
plot(t,y)
title('Transient');

%Fluctuation  1080 samples*5
x=6;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1 %3
for alpha=0.1:0.2:0.9 %5
for beta=5:0.4167:10 %12
for theta=0:0.628:3.14 %6
y=awgn(1.4142*(1+alpha*sin(beta*2*pi*t)).*sin(2*pi*f*t+theta),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
 end
 end
end
end
figure(7)
plot(t,y);
title('Fluctuation');

% Notch 1080 samples
x = 7;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1 %3
for alpha=0.15:0.05:0.4 %6
for t1=0:0.0067:0.061 %10
for theta=0:1.254:6.27 %6
    sum=0;
    for n = 0:1:9
      sum = sum+heaviside(t-(t1+0.02.*n))-heaviside(t-(t1+0.0005+0.02.*n));
    end
    y=awgn(sin(2*pi*f*t-theta)-alpha.*sign(sin(2*pi*f*t-theta)). ...
        *sum,noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
 end
 end
end
```

```
end
figure(8)
plot(t,y);
title('Notch');

% Spike 1080 samples
x = 8;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1 %3
for alpha=0.15:0.05:0.4 %6
for t1=0:0.0067:0.061 %10
for theta=0:1.254:6.27 %6
    sum=0;
    for n = 0:1:9
     sum = sum+heaviside(t-(t1+0.02.*n))-heaviside(t-(t1+0.0005+0.02.*n));
    end
y=awgn(sin(2*pi*f*t-theta)+alpha.*sign(sin(2*pi*f*t-theta)).*sum,noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
 end
 end
end
end
figure(9)
plot(t,y);
title('Spike');

%Sag&Har  1080 samples*5
x=9;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1 %3
for alpha=0.1:0.2:0.9 %5
for t1=0.04:0.004:0.048 %3
for alpha3=0.1:0.015:0.15 %4
 for theta=0:1.254:6.27 %6
alpha5=alpha3*3/5;
alpha7=alpha3*3/7;
alpha1=1;
y=awgn((1-alpha*((heaviside(t-t1)-heaviside(t-(t1+0.1))))). ...
    *(1.4142*alpha1* sin(2*pi*f*t+theta)+1.4142*alpha3* ...
    sin(3*2*pi*f*t+theta)+1.4142*alpha5*sin(5*2*pi*f*t+theta)+1.4142* ...
    alpha7*sin(7*2*pi*f*t+theta)),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
 end
 end
 end
end
```

```
end
figure(10)
plot(t,y)
title('Sag+Harmonics');

%Swell&Har  1080 samples*5
x=10;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1 %3
for alpha=0.1:0.2:0.9 %5
for t1=0.04:0.004:0.048 %3
for alpha3=0.1:0.015:0.15 %4
 for theta=0:1.254:6.27 %6
alpha5=alpha3*3/5;
alpha7=alpha3*3/7;
alpha1=1;
y=awgn(1.414*(1+alpha*((heaviside(t-t1)-heaviside(t-(t1+0.1))))). ...
    *(1.4142*alpha1* sin(2*pi*f*t+theta)+1.4142*alpha3* ...
    sin(3*2*pi*f*t+theta)+1.4142*alpha5*sin(5*2*pi*f*t+theta)+1.4142* ...
    alpha7*sin(7*2*pi*f*t+theta)),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
end
end
end
end
end
figure(11)
plot(t,y);
title('Swell+Harmonics');

%Flicker&Har  1080 samples*5
x=11;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1 %3
for alpha=0.1:0.2:0.8 %4
for beta=5:4:25 %5
for alpha3=0.1:0.0125:0.15 %5
 for theta=0:3.135:6.27
alpha5=alpha3*3/5;
alpha7=alpha3*3/7;
alpha1=1;
y=awgn(1.414*(1+alpha*sin(2*pi*beta*t)).*(1.4142*alpha1* ...
    sin(2*pi*f*t+theta)+ 1.4142*alpha3*sin(3*2*pi*f*t+theta)+ 1.4142* ...
    alpha5*sin(5*2*pi*f*t+theta)+ 1.4142*alpha7*sin(7*2*pi*f*t+theta)), ...
    noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
```

```
 end
 end
 end
end
end
figure(12)
plot(t,y);
title('Flicker+Harmonics');

%Sag+flicker 1080 samples
x=12;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1 %3
for alpha=0.1:0.267:0.9 %3
for t1=0.04:0.004:0.048 %3
for alpha2=0.1:0.1:0.2 %2
 for theta=0:2.09:6.27 %4
     for beta=5:5:25 %5
y=awgn(1.414*(1-alpha*((heaviside(t-t1)-heaviside(t-(t1+0.1))))). ...
    *sin(2*pi*f*t).*(1+alpha2*sin(2*pi*beta*t)),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
     end
 end
 end
 end
end
end
figure(13)
plot(t,y);
title('Sag+Flicker');

%Swell+flicker 1080 samples

x=13;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1 %3
for alpha=0.1:0.267:0.9 %3
for t1=0.04:0.004:0.048 %3
for alpha2=0.1:0.1:0.2 %2
 for theta=0:2.09:6.27 %4
     for beta=5:5:25 %5
y=awgn(1.414*(1+alpha*((heaviside(t-t1)-heaviside(t-(t1+0.1))))). ...
    *sin(2*pi*f*t).*(1+alpha2*sin(2*pi*beta*t)),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
     end
 end
 end
 end
end
end
figure(14)
plot(t,y);
title('Swell+Flicker');

%Sag+Transient 1080 samples
x=14;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1 %3
for alpha=0.1:0.2:0.8 %4
for t1=0.04:0.004:0.048 %3
for theta=0:3.135:6.27 %3
for fn=300:600:3000 %5
for t3=0.04:0.008:0.048 %2
y=awgn(1.414*(1-alpha*((heaviside(t-t1)-heaviside(t-(t1+0.1))))). ...
    *sin(2*pi*f*t)+ 1.4142*amp*(heaviside(t-t3)-heaviside(t-t3-0.1)). ...
    *exp(t3-t/tau).*sin(2*pi*fn*t),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
end
 end
 end
 end
end
end
figure(15)
plot(t,y);
title('Sag+Transient');

% Swell + Transient 1080 samples
x=15;
disp(x)
t=[0 :0.0003125:0.2-0.0003125];
for f=49.9:0.1:50.1 %3
for alpha=0.1:0.2:0.8 %4
for t1=0.04:0.004:0.048 %3
for theta=0:3.135:6.27 %3
for fn=300:600:3000 %5
for t3=0.04:0.008:0.048 %2
y=awgn(1.414*(1+alpha*((heaviside(t-t1)-heaviside(t-(t1+0.1))))). ...
    *sin(2*pi*f*t)+ 1.4142*amp*(heaviside(t-t3)-heaviside(t-t3-0.1)). ...
    *exp(t3-t/tau).*sin(2*pi*fn*t),noise(i));
z=vertcat(z,y);
cl=vertcat(cl,x);
end
 end
 end
 end
end
end
figure(16)
plot(t,y);
title('Swell+Transient');
end

Output = horzcat(z,cl);
mat_file = matfile('Dataset1.mat','Writable',true);
save('Dataset1.mat','Output')
%% Empirical Wavelet Transform
mp = 4;
t=[0 :0.0003125:0.2];
l = size(Output,2);
ewtdata = horzcat(t,t,0);

for i=2:size(Output,1)

fprintf(i+" / "+size(Output,1)+'\n')
y = Output(i, 1:l-1);
mp = 4;
[mra_tsp,cfs,wfb,info] = ewt(y,'MaxNumPeaks',mp);
mra = transpose(mra_tsp);

cfsenergy = sum(abs(cfs).^2);
[sum(cfsenergy) norm(y,2)^2];

[fmra, fi] = max(cfsenergy);
hmra = sum(cfsenergy) - fmra;
h = y - mra(fi,:);
size(mra(fi,:));



recon = max((y-sum(mra,2)));

cl = Output(i, l);
sample = horzcat(mra(fi,:), h, cl);
ewtdata = vertcat(ewtdata, sample);

end

mat_file = matfile('EWT_data.mat','Writable',true);
save('EWT_data.mat','ewtdata')

size(ewtdata)

%% % EWT Output for 1 PQD signal
```

```

t=[0 :0.0003125:0.2-0.0003125];
mp = 4;
[mra_tsp,cfs,wfb,info] = ewt(y,'MaxNumPeaks',mp);
mra = transpose(mra_tsp);

cfsenergy = sum(abs(cfs).^2);
[sum(cfsenergy) norm(y,2)^2];

[fmra, fi] = max(cfsenergy);
hmra = sum(cfsenergy) - fmra;
h = y - mra(fi,:);
size(mra(fi,:));

subplot(4,4,[5 9])
plot(t,y)
str = sprintf('Sag + Transient | SNR: 30 | Energy: %f', norm(y,2)^2);
title(str);
xlabel(['Energy: ',])
axis tight

for k=1:mp
    subplot(4,4,4*k-2)
    plot(t,mra(k,:))
    str = sprintf('MRA %d | Energy: %f',k,cfsenergy(k))
    title(str)
    axis tight
end

subplot(4,4,[3 7])
plot(t,mra(fi,:))
str = sprintf('Fundamental component | Energy: %f',fmra);
title(str);
axis tight
subplot(4,4,[11 15])
plot(t,h)
str = sprintf('Harmonic components | Energy: %f',hmra);
title(str);
axis tight

recon = sum(mra);
subplot(4,4,[8 12])
plot(t,recon)
str = sprintf('Reconstructed signal | Energy: %f',sum(cfsenergy))
title(str)
axis tight

%% Passbands and Filter Banks

Fs= 3.2*1e3;
```

```
f = 0:Fs/length(y):Fs-1/length(y);
size(f)
plot(f,wfb,'linewidth',1.5)
ylabel('Magnitude')
grid on
yyaxis right
plot(f,abs(fft(y)),'k--','linewidth',2)
ylabel('Magnitude')
xlabel('Hz')
```

A.2 Deep Learning Pipeline Code

```
# Encoder-based 2D CNN Model Class

class EncAndNN(nn.Module):
    def __init__(self, encoded_space_dim,fc2_input_dim):
        super().__init__()
        # Convolutional section
        self.encoder_cnn = nn.Sequential(
            nn.Conv2d(1, 8, (1, 3), stride=(1, 2), padding=(0, 1)),
            nn.ReLU(True),
            nn.Conv2d(8, 16, (1, 3), stride=(1, 2), padding=(0, 1)),
            nn.BatchNorm2d(16),
            nn.ReLU(True),
            nn.Conv2d(16, 32, (1, 3), stride=(1, 2), padding=0),
            nn.ReLU(True)
        )

        # Flatten layer
        self.flatten = nn.Flatten(start_dim=1)
        # Linear section
        self.encoder_lin = nn.Sequential(
            nn.Linear(92 * 2 * 32, 128),
            nn.ReLU(True),
            nn.Linear(128, encoded_space_dim)
        )
        self.layer1 = torch.nn.Linear(64, 1024)
        self.layer2 = torch.nn.Linear(1024, 9)

    def forward(self, x):
        x = self.encoder_cnn(x)
        x = self.flatten(x)
        x = self.encoder_lin(x)
        x = F.relu(self.layer1(x))
        x = torch.sigmoid(self.layer2(x))
        return x

```

```

# Training model
batches = math.ceil(len(train_data)/batch_size)
epochs = 10
for epoch in range(epochs):

    correct = 0
    total = 0
    running_loss = 0.0
    iter = 0
    for data in train_loader:

        iter += 1
        input, label = data[0].cuda(), data[1].type(torch.LongTensor).cuda()

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        output = model(input)
        loss = criterion(output, label)
        loss.backward()
        optimizer.step()

        # print statistics
        running_loss += loss.item()

        predicted = torch.round(output)
        for i in range(0, label.size(0)):
            if ((predicted[i] == label[i]).all()):
                correct += 1

        total += label.size(0)
        accuracy = 100 * correct/total

        running_loss = 0.0

        if iter % 100 == 1:
            print(f"[{epoch+1}, {iter}/{batches}] Train - Loss: ...
                {loss.item():.4f} Accuracy: {accuracy:.4f}\n")


    correct = 0
    total = 0
    for i, val_data in enumerate(valid_loader, 0):

        input, label = val_data[0].cuda(), val_data[1].cuda()

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        output = model(input)

        val_loss = criterion(output, label)

        predicted = torch.round(output)
        for i in range(0, label.size(0)):
            if ((predicted[i] == label[i]).all()):
                correct += 1

        total += label.size(0)
        val_accuracy = 100 * correct/total

        # print(f"{correct} / {total}")

    print(f"Val - Loss: {val_loss.item():.4f} Accuracy: ...
       {val_accuracy:.4f}\n")

print('Finished Training')
```

REFERENCES

[1] Mp Manjula, Sohan Mishra, and A.V.R.S. Sarma. "Empirical mode decomposition with Hilbert transform for classification of voltage sag causes using probabilistic neural network". In: *International Journal of Electrical Power Energy Systems* 44 (Jan. 2013), 597–603. DOI: 10.1016/j.ijepes.2012.07.040.

[2] Ebrahim Balouji and Ozgul Salor. "Classification of power quality events using deep learning on event images". In: Apr. 2017, pp. 216–221. DOI: 10.1109/PRIA.2017.7983049.

[3] A.C. Parsons, W.M. Grady, E.J. Powers, and J.C. Soward. "A direction finder for power quality disturbances based upon disturbance power and energy". In: *8th International Conference on Harmonics and Quality of Power. Proceedings (Cat. No.98EX227)*. Vol. 2. 1998, 693–699 vol.2. DOI: 10.1109/ICHQP.1998.760129.

[4] Karthik Thirumala, Sushmita Pal, Trapti Jain, and Amod C. Umarikar. "A classification method for multiple power quality disturbances using EWT based adaptive filtering and multiclass SVM". In: *Neurocomputing* 334 (2019), pp. 265–274. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.01.038. URL: https://www.sciencedirect.com/science/article/pii/S092523121930058X.

[5] Ananta Agarwalla, Diya Dileep, P. Jyothsana, Purnima Unnikrishnan, and Karthik Thirumala. "Principal Component Analysis and CNN for Classification of Power Quality Disturbances". In: *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*. 2021, pp. 1–5. DOI: 10.1109/UPCON52273.2021.9667629.

[6] Sharmeela Chenniappan, M.R. Mohan, Godson Uma, and Jothi Baskaran. "A Novel Detection and Classification Algorithm for Power Quality Disturbances using Wavelets". In: *American Journal of Applied Sciences* 3 (Oct. 2006), pp. 2049–2053. DOI: 10.3844/ajassp.2006.2049.2053.

[7] Suresh Gawre, V. Kumar, and Tushar Kumar. "Power Quality Analysis Using Wavelet Transform: A Review". In: *International Journal of Innovative Research in Science, Engineering and Technology* 3 (Mar. 2014), pp. 130–136.

[8] C. Aneesh, Sachin Kumar, P.M. Hisham, and K.P. Soman. "Performance Comparison of Variational Mode Decomposition over Empirical Wavelet Transform for the Classification of Power Quality Disturbances Using Support Vector Machine". In: *Procedia Computer Science* 46 (2015). Proceedings of the International Conference on Information and Communication Technologies, ICICT

2014, 3-5 December 2014 at Bolgatty Palace Island Resort, Kochi, India, pp. 372–380. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.02.033. URL: https://www.sciencedirect.com/science/article/pii/S1877050915000976.

[9] Sihan Chen, Ziche Li, Guobing Pan, and Fang Xu. "Power Quality Disturbance Recognition Using Empirical Wavelet Transform and Feature Selection". In: *Electronics* 11.2 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11020174. URL: https://www.mdpi.com/2079-9292/11/2/174.

[10] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. "Extracting and Composing Robust Features with Denoising Autoencoders". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: Association for Computing Machinery, 2008, 1096–1103. ISBN: 9781605582054. DOI: 10.1145/1390156.1390294. URL: https://doi.org/10.1145/1390156.1390294.

[11] Jerome Gilles. "Empirical Wavelet Transform". In: *IEEE Transactions on Signal Processing* 61 (Aug. 2013), pp. 3999–. DOI: 10.1109/TSP.2013.2265222.

[12] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. "Deep Convolutional AutoEncoder-based Lossy Image Compression". In: *CoRR* abs/1804.09535 (2018). arXiv: 1804.09535. URL: http://arxiv.org/abs/1804.09535.

[13] Qinxue Meng, Daniel Catchpoole, David Skillicom, and Paul J. Kennedy. "Relational autoencoder for feature extraction". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017, pp. 364–371. DOI: 10.1109/IJCNN.2017.7965877.