CONTENTS

1. Introduction

1.1. Project Overview

1.2. Objectives

2. Literature Review

2.1. Faults in Electrical Systems

2.2. High Impedance Faults

2.3. Sensors for Fault Detection in Electrical Systems

2.4. Wavelets and Discrete Wavelet Transform

2.5. Implementation of Fault Detection Techniques

2.6. Time and Frequency Domain Analysis

2.7. Different Implementation of HIF Models

2.8. Deep Learning Techniques and Neural Networks

2.9. Implementation of CNN in HIFs detection

3. Present Work / Simulation / Programs

3.1. Modelling of High Impedance Faults

3.2. Fast Fourier Transform for Harmonic Content Detection

3.3. Wavelet Packet analysis for the HIF simulated data

3.4. Feature Ranking: Selecting the Most Informative Feature

3.5. Training the Convolutional Neural Network for the Best Feature

4. Results

5. Future Work

6. References

List of Images

Figure no.	Figure Title
1	Characteristics of HIF current
2	Process of Wavelet transform
3	Different types of Mother wavelets
4	Time-Frequency domain waveform and envelope spectrum
5	HIF model introduced by [10]
6	HIF model proposed by [11]
7	HIF model proposed by [12]
8	Comparison of the field HIF sample with the dummy HIF sample
9	Loss of discriminator and generator during the training process (a) IGAN (b) AGCAN
10	Developed Simulink model to Simulate HIF generation
11	High Impedance Fault current
12	High Impedance Fault Arc Voltage
13	I-V Characteristic of HIF model
14	Current during Normal operation
15	Voltage during Fault operation
16	Current during Fault operation
17	Voltage during Fault operation
18	FFT analysis for Normal current
19	FFT analysis for Fault current
20	Wavelet packet tree for a 3-level Decomposition
21	T-test ranking of statistical features
22	Bhattacharyya ranking of statistical features
23	One-Way ANOVA ranking of statistical features
24	Structure of CNN models
25	Validation Loss over each iteration of model training
26	Accuracy over each iteration of model training
27	Confusion matrix for the Test set.

Chapter-1: Introduction

Project Overview

Electrical power systems are vulnerable to a variety of faults, such as short circuits, line-to-ground faults, open circuits, and **High Impedance Faults (HIFs)**. Among these, HIFs are especially challenging to detect and diagnose. These faults typically occur when a conductor comes into contact with a high-resistance surface, such as asphalt, dry soil, or tree branches. Despite their low current, HIFs can present serious safety hazards, including the risk of fire and electrical shock, and can damage sensitive equipment.

The focus of this project is to improve the detection of High Impedance Faults using a hybrid approach that combines **Wavelet Packet Analysis (WPA)** with **Convolutional Neural Networks (CNNs)**. The goal is to design an intelligent, data-driven fault detection system that can effectively distinguish HIFs from normal operating conditions, overcoming the limitations of conventional protection schemes.

To begin, a detailed Simulink model of an electrical distribution system was developed. This model was used to simulate a range of operating conditions, including both normal behavior and various HIF scenarios. From these simulations, a comprehensive dataset was generated. The voltage and current waveforms from the model were subjected to **harmonic analysis** and **wavelet packet decomposition**, allowing for the extraction of rich time-frequency domain features that are sensitive to subtle changes caused by HIFs.

Using the wavelet packet coefficients, **16 statistical parameters** (such as mean, standard deviation, entropy, skewness, etc.) were calculated for each signal. These parameters were then analyzed to identify the one showing the **maximum variation between normal and fault conditions**, making it a suitable input feature for the machine learning model.

A **Convolutional Neural Network** was then trained on this selected feature set to classify the input signals as either normal or indicative of a high impedance fault.

CNNs are well-suited for this task due to their ability to automatically learn hierarchical patterns from input data. The model was trained using a portion of the dataset and tested on a separate validation set—both generated from the Simulink environment.

The experimental results showed that the proposed WPA-CNN-based approach is effective in accurately identifying HIFs.

Overall, this project lays a strong foundation for the development of real-time, intelligent fault detection systems. Future work can focus on hardware implementation, online learning techniques, and the integration of the model with real-world electrical systems to further enhance reliability and scalability.

Objectives

❖ Modelling and Simulation of Electrical Faults

- Develop a comprehensive Simulink model to simulate both normal and faulty conditions, with a special focus on High Impedance Faults.
- Generate realistic voltage and current waveforms representing various fault scenarios.
- Ensure the model captures the subtle characteristics of HIFs for accurate analysis.

❖ Signal Processing and Feature Extraction

- Apply Harmonic Analysis to study the distortion in the electrical signals caused by faults.
- Use Wavelet Packet Decomposition to extract time-frequency features from the simulated data.
- Calculate a set of 16 statistical parameters (e.g., mean, standard deviation, kurtosis) from the wavelet packet coefficients.
- Identify the most discriminative parameters that show maximum variation between normal and HIF conditions.

❖ Development of Machine Learning-Based Fault Classifier

- Design and train a Convolutional Neural Network (CNN) using the extracted features to classify normal and faulty conditions.
- Evaluate the model using a separate testing set generated from the Simulink model.
- Measure the performance of the CNN model using standard metrics.

Chapter-2: Literature Review

Faults in Electrical Systems

Faults in electrical power systems are abnormal conditions that disrupt the normal flow of current. These disruptions can range from minor disturbances to catastrophic failures. Faults in electrical power systems can cause significant disruptions, including power outages, equipment damage, and safety hazards. To mitigate these effects, power system operators employ various protection measures such as protective relays, circuit breakers, and grounding systems.

While electrical faults in general can manifest in various forms, High Impedance Faults (HIFs) present a unique challenge due to their specific characteristics and potential consequences.

High Impedance Faults

High impedance faults (HIFs) are a type of electrical fault that occur when a conductor comes into contact with a highly resistive medium. This can happen due to various factors such as tree branches touching lines, downed conductors landing on dry ground, or damage to the power infrastructure caused by accidents, construction activities, or natural disasters.

There are 2 types of HIFs:

- **Active HIFs**
- **Passive HIFs**

HIFs can potentially escalate into more severe faults, such as short-circuit or open-circuit faults. The intermittent arcing characteristic of HIFs can intensify, leading to sustained arcing faults that can damage equipment and further degrade the fault path. In some cases, the arcing path may expand, causing the fault to bridge the gap between conductors, resulting in a short-circuit fault.

In 2020, Aljohani, A., and Habiballah, I. [8] reviewed High-Impedance Faults (HIFs) in their paper, outlining their characteristics as follows:

- **Low Magnitude Current**
- **Intermittent Arcing**
- **Asymmetry and Randomness**
- **Nonlinear Behavior**
- **Build-up and Shoulder Effect**

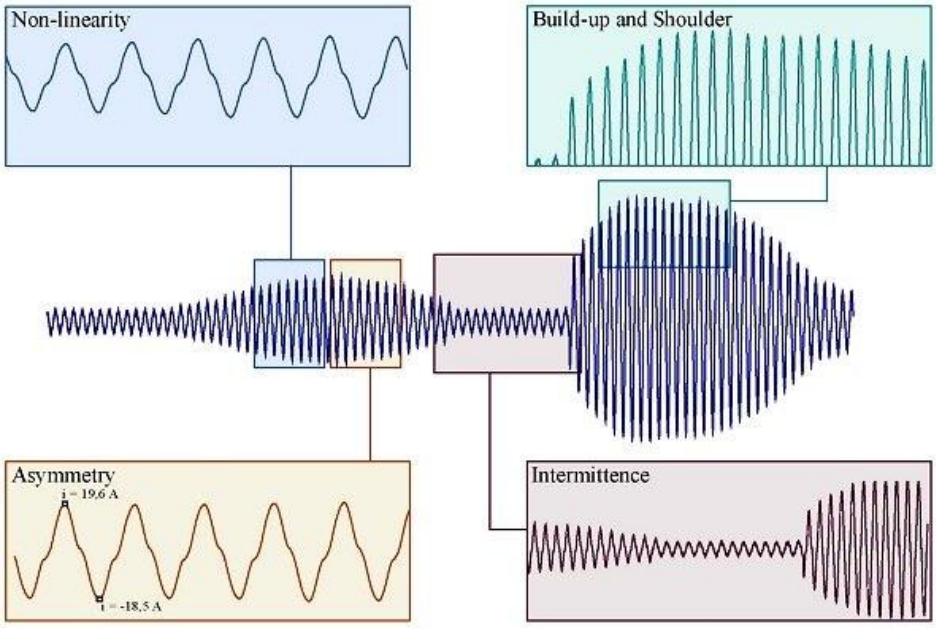


Fig. 1 Characteristics of HIF current

Despite its low current, during its undetected state, an HIF is a risk to public safety, since a downed conductor can create hazardous shock, fire, or life-threatening injuries through unintentional human contact.

Sensors for Fault Detection in Electrical Systems

Cynthia M. Furse et.al [5] mention sensors for electrical health monitoring perform three major functions. Detection determines that a fault exists or is developing and is often used to shut down an electrical system to prevent hazards or damage and trigger a maintenance call. Maintainers need to locate the fault in order to repair it.

Wavelets and Discrete Wavelet Transform

Wavelets are mathematical functions that look like small, short waves and are used to break down complex signals or data into simpler parts. They are like zoomable tools that help us study signals (like sounds, images, or data) at different levels of detail. Wavelets are tools that help to identify sudden changes (like a sharp edge in an image or a sudden spike in a sound wave). They are used in fault detection in electrical systems, signal processing, medical diagnostics, etc.

The discrete wavelet-transform deals with digitized signals. There is a sister continuous wavelet transform that deals with analog signals, but that is not discussed at all here. Wavelet transform (DWT) is a mathematical process that quantifies how much energy is contained within specific frequency bands at particular times within a signal.

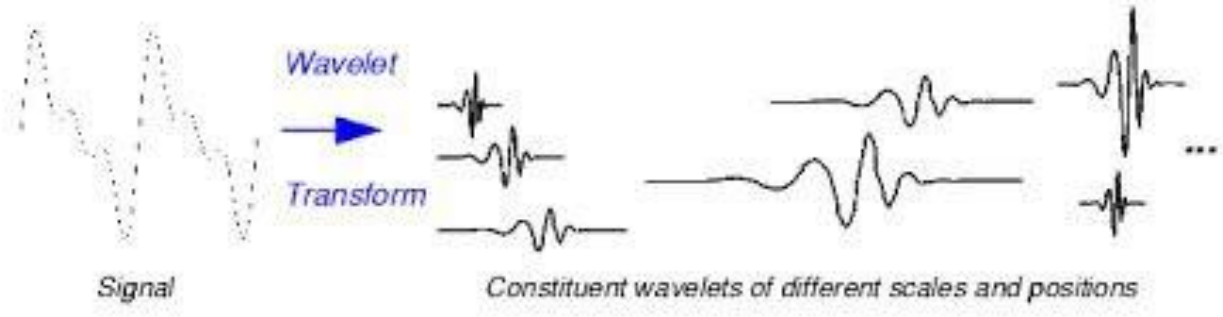


Fig. 2 Process of Wavelet transform

The big difference between the Fourier transformation and wavelet transformation (and the main reason for the usefulness of the latter) is that when we do a wavelet transformation, we only move *part* of the wave from the time domain to the frequency domain. After wavelet analysis, we gain some information about the frequency content of the original signal, but we also *retain* some information about when in the original signal particular frequencies occur.

Wavelet transform, like Fourier transform, breaks down a signal into smaller components, but instead of simple sinusoidal waves, it uses complex wavelets that resemble squiggly shapes. There are a whole series of different types of “mother

“wavelets (Daubechies, Coiflet, Symmlet, etc.) available, and each type occurs in a range of sizes (Daubechies-4, Daubechies-8, etc.).

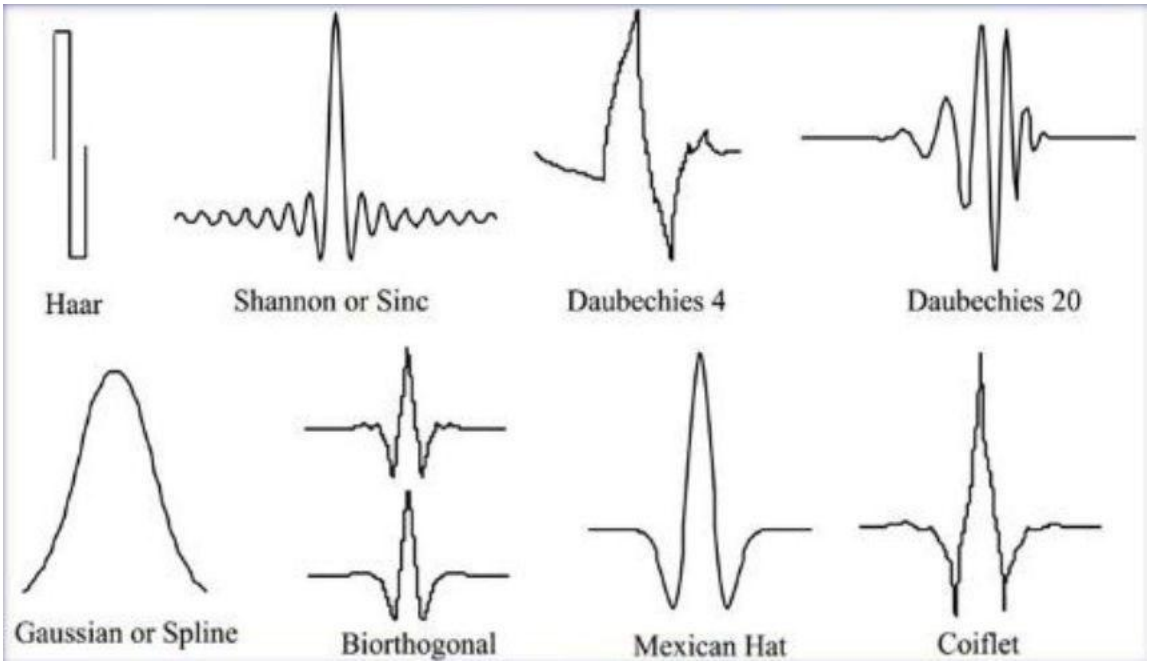


Fig. 3 Different types of Mother wavelets

When a raw data signal is transformed using a mother wavelet, the result is a series of **basis waveforms** called **daughter wavelets**. These daughter wavelets are compressed or expanded versions of the mother wavelet, representing different **scales or frequencies**, and they cover different parts of the original signal, corresponding to their locations.

Each daughter wavelet produced during a wavelet transform is associated with a specific coefficient. This coefficient indicates how much the daughter wavelet at a particular scale and location contributes to the original signal. In other words, the coefficients capture the essential information about the original input signal, while the daughter wavelets themselves are fixed patterns determined by the selected mother wavelet and do not depend on the input signal.

Implementation of Fault Detection technique

In 2019, Gadanayak, D. A., and Mallick, R. K. [6] proposed a novel technique for detecting high impedance faults (HIFs) in distribution systems, utilizing maximum overlap discrete wavelet packet transform (MODWPT) and a modified empirical mode decomposition (EMD) to analyze inter harmonic components in current signals.

The technique utilizes maximum overlap discrete wavelet packet transform (MODWPT) to decompose the current signal into various frequency components, allowing for a detailed analysis of the inter harmonics. This decomposition is crucial because inter-harmonics can provide valuable information about the presence of faults that may not be detectable through conventional means. Additionally, the authors introduce a modified empirical mode decomposition (EMD) that is specifically designed to enhance the separation of inter-harmonic components from integer harmonics, further improving the accuracy of fault detection. The algorithm is capable of detecting HIFs within 15 cycles, which is a substantial improvement over existing methods that often rely on longer analysis windows, leading to undesirable delays.

The author Kartika Dubey, et al [1], proposed a high-impedance fault (HIF) detection technique to achieve accurate detection and discrimination of HIFs from other external and non-fault transients, such as inrush currents and capacitor switching events.

The method also aims to reduce computational complexity by eliminating the need for extensive data training, unlike heuristic-based approaches that require significant memory and processing power. This feature enhances the feasibility of real-time deployment. Lastly, the proposed technique focuses on achieving a fast relay response time, identifying HIFs within three cycles of occurrence, making it faster and more responsive than many traditional methods.

The datasets used in the research paper were obtained through simulations conducted on modified IEEE-13 and IEEE-34 bus distribution test systems. These

simulations were designed to create various scenarios, including high-impedance faults (HIFs) and other non-fault events, allowing for a comprehensive evaluation of the proposed HIF detection technique.

- **Calculation of Differential Positive Sequence Current Components (PSCCs):**

The PSCCs (I1) are calculated based on the current magnitudes and angles, which differ under normal and fault conditions. The PSCCs for buses 1 and 2 are computed using the equation:

$$I(i,x) = \frac{1}{3}\left(I_A(i,x) + j \cdot I_B(i,x) + j^2 \cdot I_C(i,x)\right)$$

where i=1 or 2 for buses 1 and 2, and x=pre or post for pre- and post-fault conditions. The differential pre- and post-fault PSCCs are given as:

$$\Delta I_{\text{pre}} = I_{12,\text{pre}} - I_{11,\text{pre}} \quad \text{and} \quad \Delta I_{\text{post}} = I_{12,\text{post}} - I_{11,\text{post}}$$

- **Calculation of Differential Energy Index Using Energy Operator (Ψ):**

The nonlinear energy operator (Ψ) is used to track the signal's local energy, offering several advantages, such as lower memory usage, noise resistance, and effective transient detection. For a discrete current signal i[n], the energy operator is defined as:

$$E = \Psi[i(n)] = i^2[n] - i[n-1] \cdot i[n+1]$$

The energy for pre- and post-fault PSCCs is calculated as E1 and E2, respectively, and the differential energy ΔE is obtained as:

$$|\Delta E| = |E_1 - E_2| = \left|\Psi\left[\Delta I_{pre}(n)\right] - \Psi\left[\Delta I_{post}(n)\right]\right|$$

- **Calculation of Fault Detection Indicator:**

To ensure accurate HIF detection and avoid false relay tripping, a cumulative fault detection index (CFDI) is introduced, calculated as:

$$\text{CFDI} = \sum_{k=1}^{N} |\Delta E_k|$$

Sr. No.	*Method*	*Detection Time*
1.	Modified FFT	55 ms
2.	Short-time Fourier Transform	150.8 ms
3.	Mathematical Morphology	1 sec
4.	HHT-based	58.33 ms
5.	Proposed Technique	48 ms

The response time for pre-existing techniques and the proposed technique

Time and Frequency Domain Analysis

The time-domain analysis uses the measure of zero-sequence voltage and current for feature extraction of HIF. Time-domain takes out the temporary irregularities in the HIF waveform, making the system computationally complex. Nezamzadeh-Ejieh and Sadeghkhani [3] proposed that Kullback–Leibler divergence extracts the non-linearity and asymmetry characteristics of two half-cycles of the current waveform from the substation in a time-domain detection of HIF.

Frequency-domain analysis extracts harmonics in the current spectrum. In the current spectrum, a HIF event will produce low and high-frequency components. Low-frequency components are based on non-linearity results, whereas high-frequency components are based on sudden and random changes in a non-stationary HIF current waveform. Aucoin and Russell [4] utilized high-frequency current components to detect HIF.

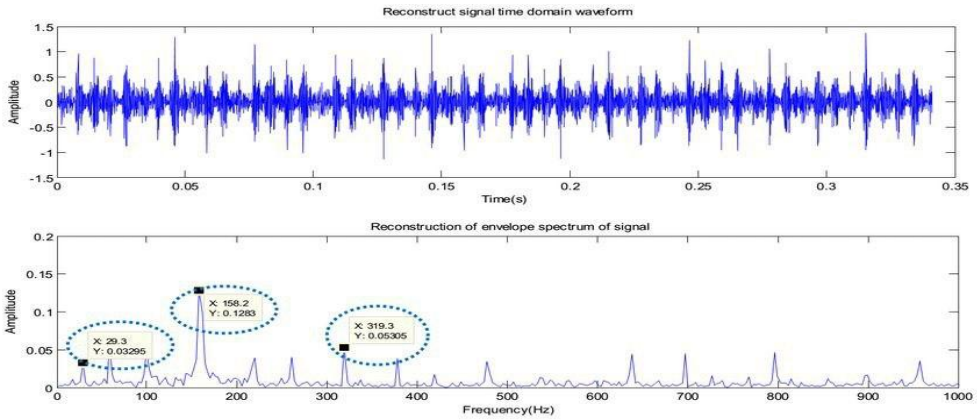


Fig. 4 Time-Frequency domain waveform and envelope spectrum

Different Implementations of HIF's Models

High impedance fault is a difficult case to model because most HIF phenomena involve arcing, which has not been perfectly modeled so far. Some previous researchers have reached a consensus that HIFs are nonlinear and asymmetric and that modeling should include random and dynamic qualities of arcing. Emanuel's model is based on laboratory measurements and theoretical components. In a paper by C. G. Wester [9], he suggests two DC sources connected anti-parallel with two diodes to simulate zero periods of arcing and asymmetry

In 2013, A. H. Eldin et.al [10] presented a model including consideration of nonlinearity in earth impedance the arcing high impedance fault was modeled as two sets (positive and negative) of diodes in series with a resistance and a DC source Figure 5 illustrates that model.

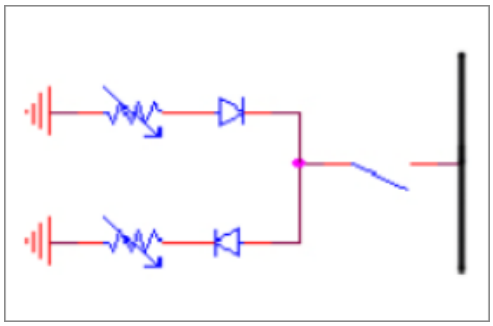


Fig. 5 HIF model introduced by *[10]*

In 2022, Hao Bai et.al [11] suggested an improved model for detecting HIFs. They used time-varying resistance and voltage sources to express the random volatility of the arc voltage and resistance. In addition, the inductance is introduced and adjusted to control the zero-off feature and offset degree. The structure of the proposed improved arc model is shown in Figure 6.

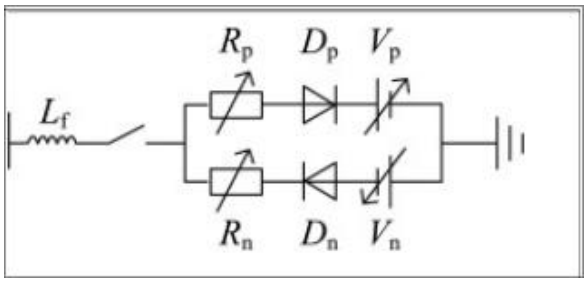


Fig. 6 HIF model proposed by [11]

In 2013, S. Gautam et.al [12] proposed a model that is structured with two anti-parallel direct currents (DC) sources connected through diodes, effectively capturing the asymmetric nature of fault currents and the intermediate arc extinction that occurs around current zero. The DC sources are of unequal magnitudes and vary randomly every 0.1 milliseconds, simulating the unpredictable behavior of fault currents and the fluctuations in arc resistance. Additionally, two variable resistances are incorporated in series with the diodes, which also change independently and randomly, further modeling the randomness of the effective impedance. The proposed model is shown in Figure 7.

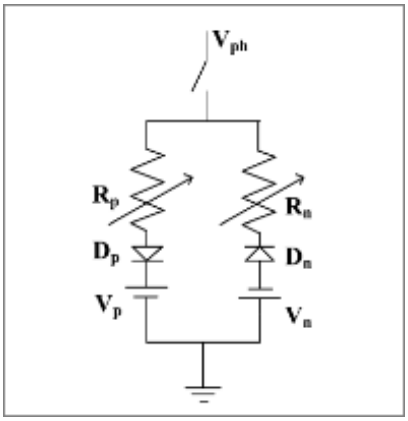


Fig. 7 HIF model proposed by [12]

Deep Learning Technique and Neural Networks

Deep learning techniques and neural networks play a pivotal role in addressing complex problems such as fault detection in electrical distribution systems. These methods are inspired by the functioning of the human brain and consist of interconnected layers of artificial neurons that process input data to extract meaningful patterns and make accurate predictions.

In the context of High Impedance Fault (HIF) detection, deep learning methods, particularly Convolutional Neural Networks (CNNs), excel in identifying intricate and non-linear relationships within the data. HIFs are characterized by unique current and voltage patterns that traditional fault detection methods struggle to identify due to their low magnitude and irregular nature. Deep learning models, with their ability to handle high-dimensional data, are well-suited for analyzing these subtle patterns.

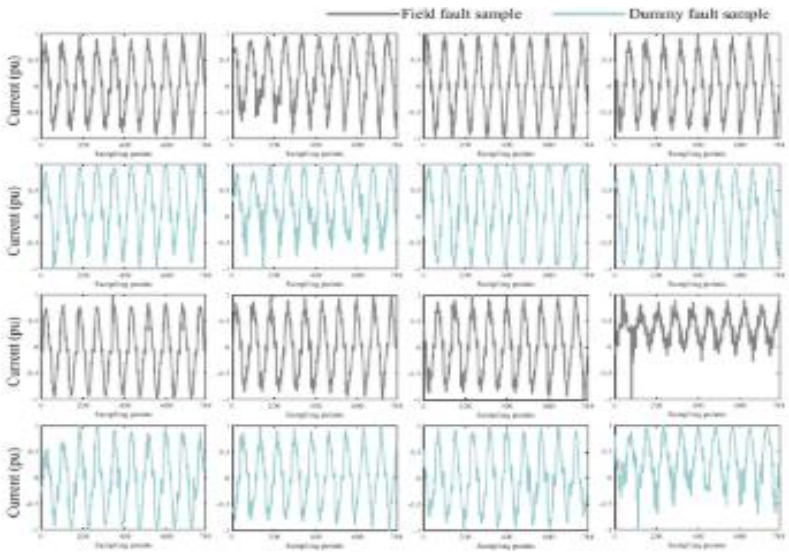


Fig. 8 Comparison of the field HIF sample with the dummy HIF sample

The network consists of multiple convolutional and pooling layers arranged in an alternating sequence, creating a feature extractor that processes raw input data into more abstract, high-level feature representations. Fully connected layers, along with activation functions, are then used to perform tasks like classification or regression on these extracted features. The components of CNN are as follows:

- **Convolutional Layer**
- **Pooling Layer**
- **Activation Function**
- **Fully Connected Layer**

Implementation of CNN in HIFs detection

In their paper, Mou-Fa et.al [2] introduced a novel methodology to address these challenges by leveraging advanced neural networks, including Generative Adversarial Networks (GANs) and Improved Generative Adversarial Networks (IGANs), for data augmentation. This approach compensates for the inherent imbalance in the dataset, creating a richer environment for training deep learning classifiers. Notably, the system demonstrated the ability to detect faults within 60 milliseconds, outperforming existing detection methods in both speed and accuracy.

The dataset employed in the study was derived from a medium-voltage distribution system comprising electrical signals that represent both normal operating conditions and instances of high impedance faults (HIFs). Data acquisition was performed using the Raspberry Pi 4 Model B, which functions as a real-time data acquisition (DAQ) device. The dataset reflects real-world conditions as it was collected from a substation in China. Normal operating samples illustrate typical electrical system behavior, while fault samples document rare but critical occurrences of HIFs. To address the imbalance, the authors incorporated synthetic HIF data generated using an Improved Generative Adversarial Network (IGAN).

The proposed methodology leverages both on-site collected signals and preprocessed acquired signals, combining online and offline methods to improve the accuracy and efficiency of High Impedance Fault (HIF) detection. The process is divided into three stages:

Stage-1: Offline Training with IGAN: the **Improved Generative Adversarial Network (IGAN)** is trained using an adversarial approach.

Stage-2: Offline Training with CNN: Next, a **Convolutional Neural Network (CNN)** classifier is trained using the enhanced balanced training dataset generated in Stage-1.

Stage-3: Real-Time Detection (Online): In the final stage, real-time data is acquired using Data Acquisition (DAQ) devices installed on electrical feeders. These devices incorporate a preprocessing module to handle real-time signals.

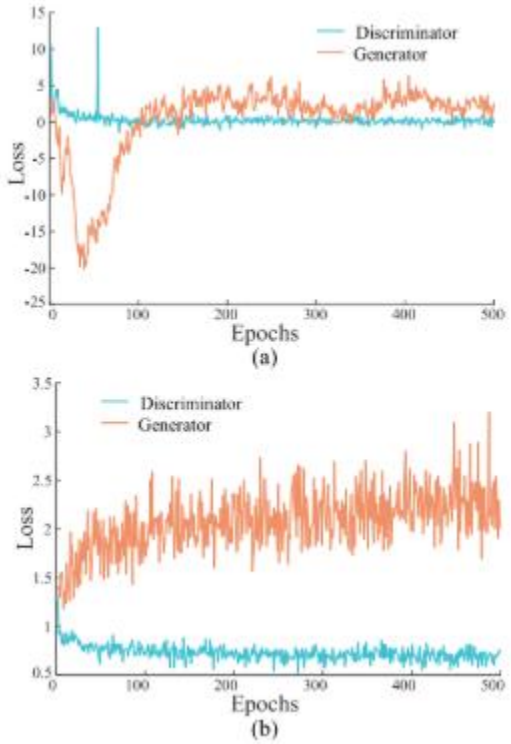


Fig. 9 Loss of discriminator and generator during the training process (a) IGAN (b) ACGAN

Method	*Accuracy*	*Sensibility*	*Safety*	*F1-score*
Raw data	0.645	100%	57.7%	0.732
M1	0.806	100%	71.4%	0.833
M2	0.938	100%	88.2%	0.938
M3	0.968	100%	93.8%	0.968
M4	0.935	86.7%	100%	0.929
M5	1	100%	100%	1

Comparison of fault Detection Results with different Data Enhancement methods

This table presents a comparison of fault detection results using different data enhancement methods. The comparison is based on four performance metrics: Accuracy, Sensibility (Recall), Safety, and F1-score. Raw data shows the lowest performance across all metrics. Methods M1 to M5 demonstrate a progressive improvement in the model's performance, where **M1**: ROS algorithm, **M2**: algorithm, **M3**: GAN, **M4**: ACGAN and **M5**: IGAN.

Chapter-3: Present Work / Simulation / Programs

Modelling of High Impedance Faults

HIF detection methods are based on either staged faults or simulation data. Staged faults are system-specific, require specialized equipment, and risk service disruption, whereas simulation-based methods offer greater adaptability and extensive testing. This study adopts a simulation approach, as the generated waveforms closely match staged HIFs, validating the model. Due to arcing, HIFs exhibit unpredictable variations in fault resistance, current magnitude, and waveform shape, making them inherently random and nonlinear.

Several models have been proposed in the past for the purpose of time domain simulation. A fixed resistance at the point of fault is the simplest model, which was later modified by including nonlinear impedances to incorporate the nonlinearity of the fault current. Inclusion of two anti-parallel DC sources connected via two diodes modelled the asymmetric nature of the fault current, as well as the intermediate arc extinction around current zero. This model was further modified by adding one or two variable resistances in series with the DC sources to model the randomness of the effective impedance and thus the randomness of the resulting fault current. Other models used for the analysis of HIF are TACS (transient analysis of control systems), controlled switch to connect and disconnect the fault or randomly vary the effective HIF resistance, and differential equation-based models.

The transmission line is used as the test system and is integrated with a **High Impedance Fault (HIF) model**, as illustrated in **Fig. 10**. The model simulates a fault between a phase and ground, where V_{ph} denotes the phasor value of the phase voltage. Structurally, the HIF model resembles the **source-diode-resistance configuration** described in the literature, with parameters adjusted to match the voltage level of the system under study.

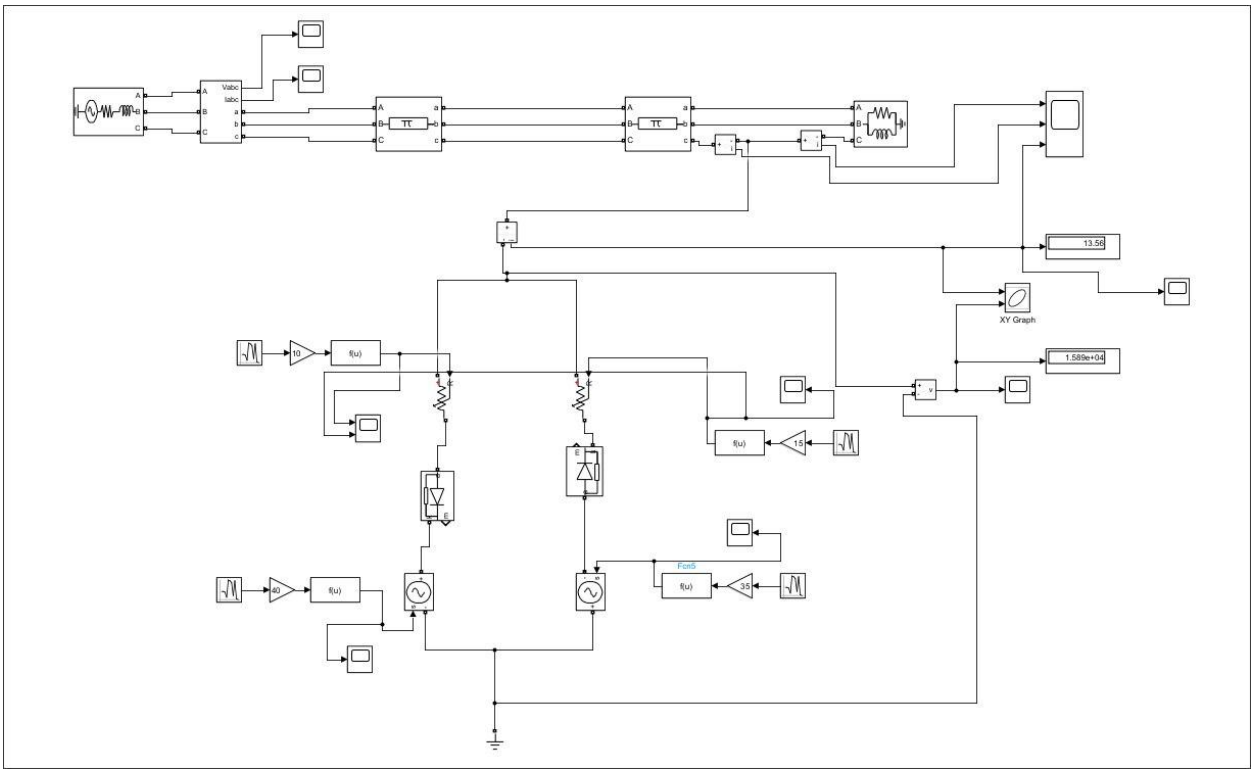


Fig. 10 Developed Simulink model to Simulate HIF generation

Despite its simplicity, the model effectively captures the essential characteristics of HIF behavior. It consists of two unequal **DC voltage sources** (V_p and V_n) connected to ground through corresponding **diodes** (D_p and D_n). These DC sources vary randomly around their base values every 0.1 ms to replicate the **asymmetric nature of HIF current** and simulate **intermittent arc extinction**. The values of V_p and V_n are selected based on the system voltage and the degree of asymmetry intended to be modelled. The fault behavior follows these conditions:

- When $V_{ph} > V_p$, Current flows from the phase to the ground.
- When $V_{ph} < V_n$, current reverses direction.
- During intervals when $V_n < V_{ph} < V_p$, **no current** flows, simulating arc extinction periods.

To further introduce randomness and emulate the **dynamic arc resistance**, two variable resistors R_p and R_n are connected in series with the respective diodes. These resistors vary **independently and randomly** every 0.1ms.

To ensure the realism of the simulation and test the robustness of the proposed algorithm, the parameters are constrained such that the resulting **fault current remains below 10% of the feeder's full load current**, in line with typical HIF behavior.

	Minimum Value	Maximum Value
R_n	7.200 x 10^2 Ω	1.100 x 10^3 Ω
R_p	7.300 x 10^2 Ω	1.070 x 10^3 Ω
V_n	3.610 x 10^3 V	4.410 x 10^3 V
V_p	3.600 x 10^3 V	4.400 x 10^3 V

Table for Minimum and Maximum Values of various resistances

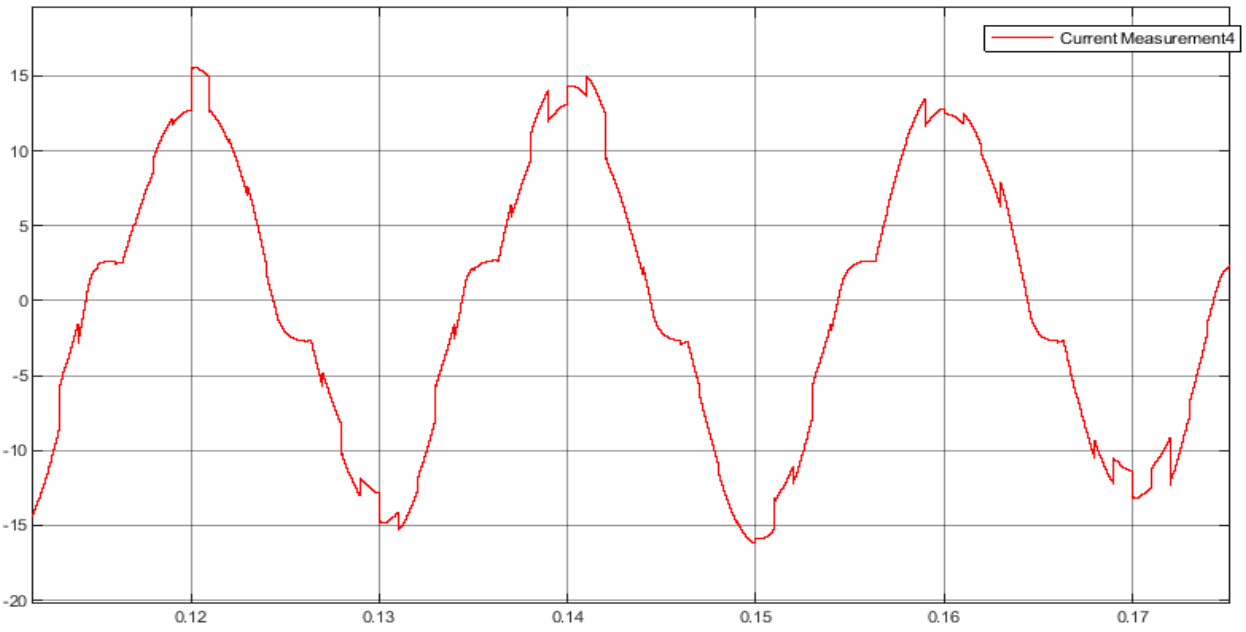


Fig. 11 High Impedance Fault current

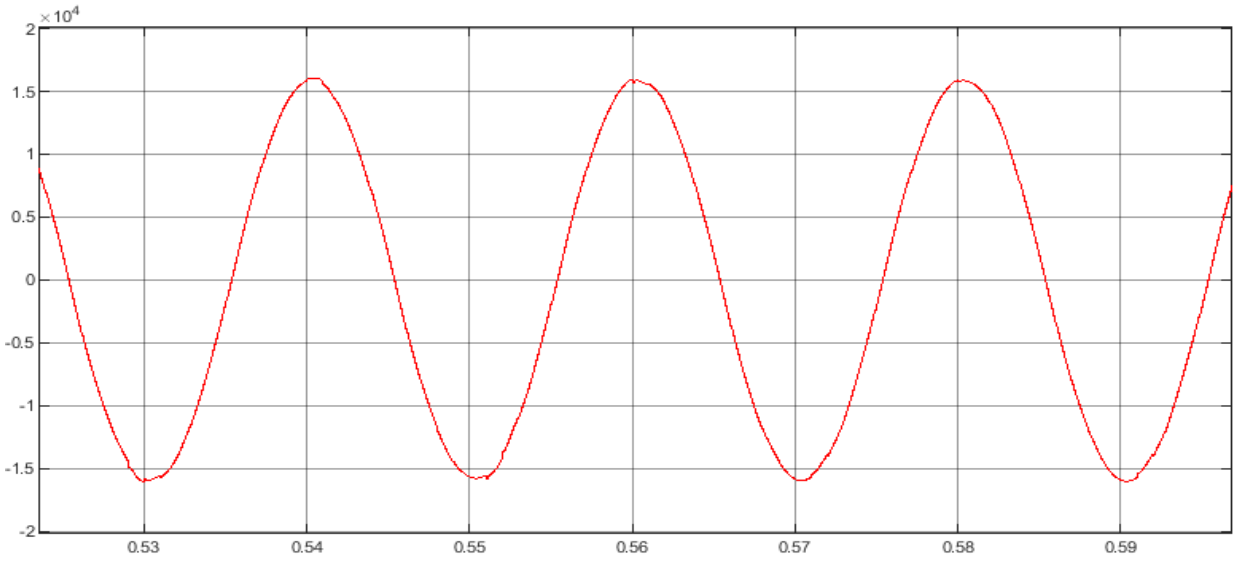


Fig. 12 High Impedance Fault Arc Voltage

Fig. 11 illustrates the voltage and current waveforms observed during a high impedance fault (HIF) simulated on the test feeder using the HIF model described in the previous section. The **current waveform** in **Fig. 2** exhibits a **random and asymmetric pattern**, characterized by unequal positive and negative peaks. The brief interruptions near the zero-crossing points of the waveform indicate **temporary arc extinctions**, a typical phenomenon in high impedance faults.

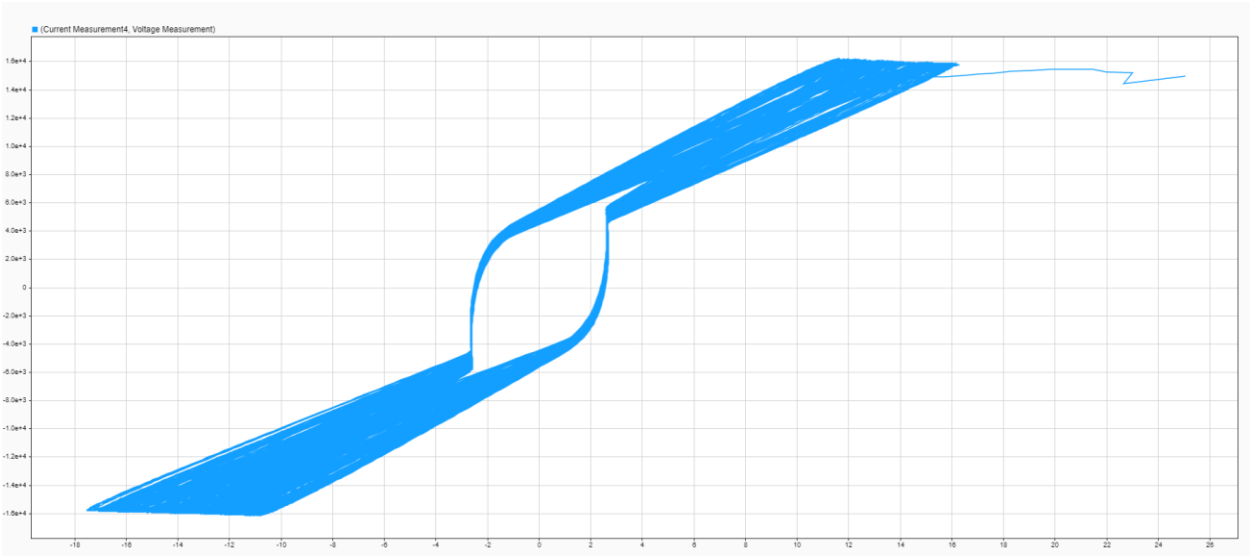


Fig. 13 I-V Characteristics of HIF model

Fig. 13 further presents the distinct characteristics of the modelled HIF. The current waveform in **Fig. 12** and the features highlighted in **Fig. 13** closely resemble those obtained from **laboratory experiments, staged faults**, and other **established arc-based HIF models**. This consistency reinforces the accuracy and realism of the proposed HIF model.

Additionally, the **Fast Fourier Transform (FFT)** analysis of the HIF current waveform reveals the presence of **harmonic components**, underscoring the nonlinear nature of the fault. Thus, the model depicted in the figures effectively captures the **nonlinearity, asymmetry, randomness**, and **arc dynamics**, including intermediate **arc extinction phenomena**, inherent to high impedance faults.

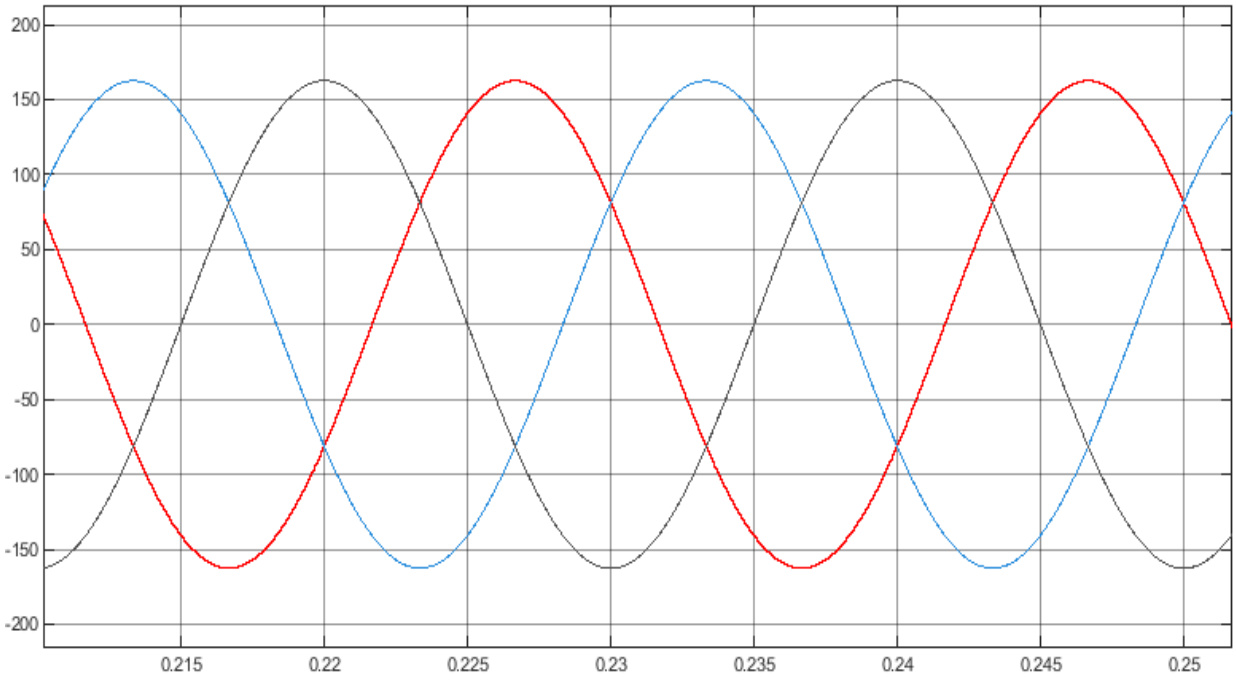


Fig. 14 Current during Normal operation

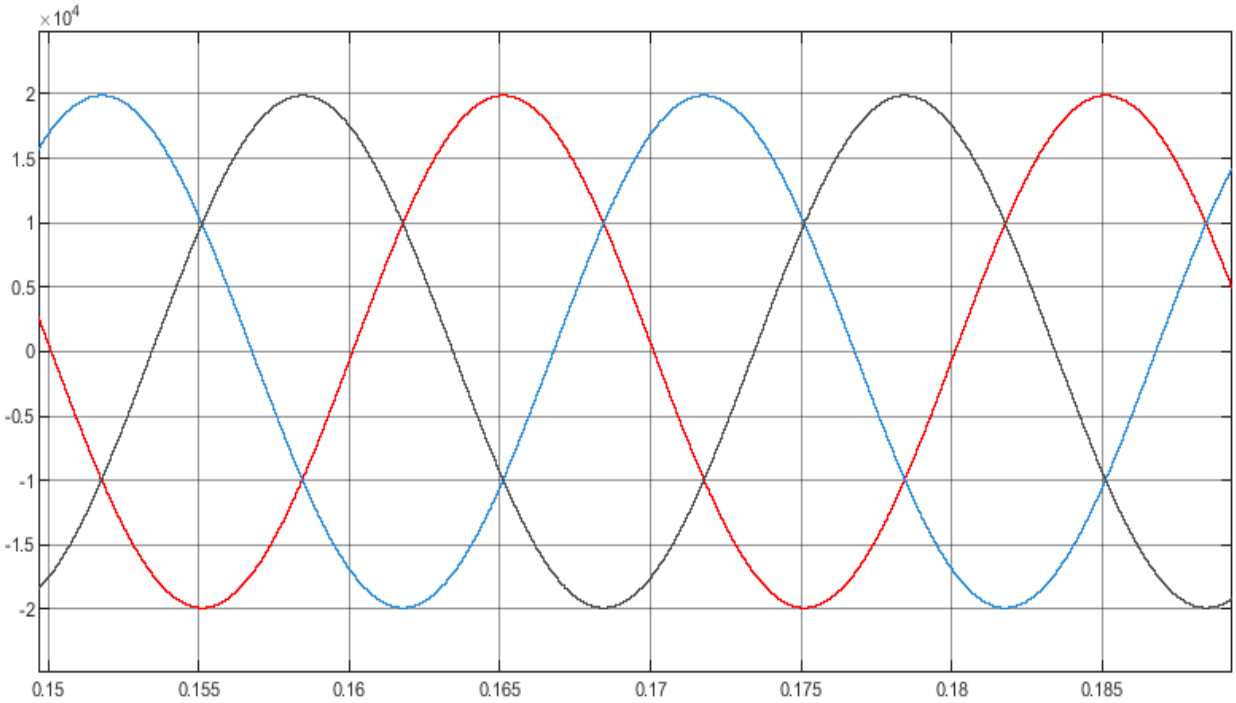


Fig. 15 Voltage during Fault operation

Fig. 15 depicts the system under **stable operating conditions**, where both the voltage and current waveforms exhibit clean, undistorted sinusoidal patterns. The **absence of harmonic content** further confirms that the system maintains high **power quality** during normal operation. Additionally, the **proportional relationship** between voltage and current waveforms aligns with theoretical expectations, thereby validating the **accuracy of the simulation model** and ensuring proper synchronization within the system.

Various methods for **High Impedance Fault (HIF) detection** either utilize current waveforms alone or a combination of voltage and current signals to extract distinguishing fault signatures. However, it is important to note that the waveform shown in **Fig. 12** represents the **actual fault current at the fault location**, not the current measured at the **substation source**. The current at the substation typically remains less distorted due to the predominance of sinusoidal **load current**, which tends to mask the subtle irregularities introduced by HIFs.

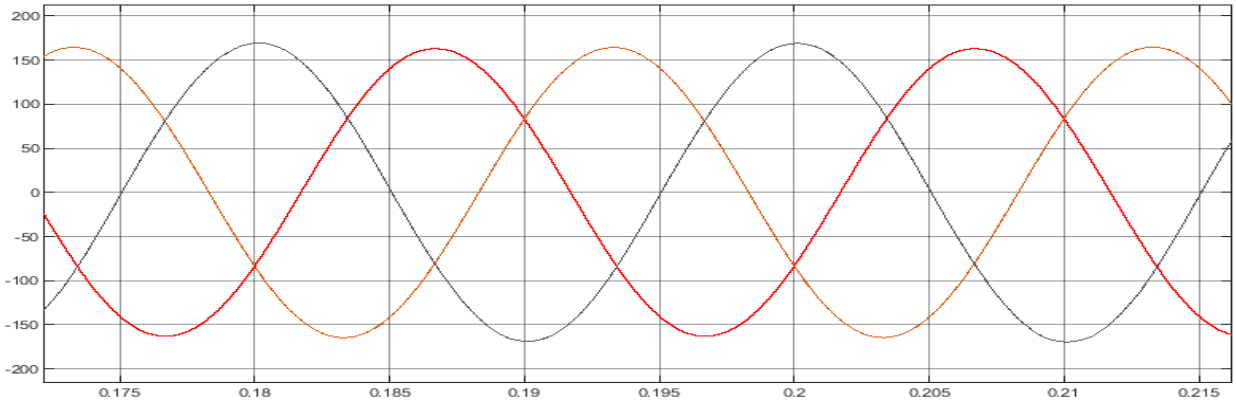


Fig. 16 Current during Fault operation

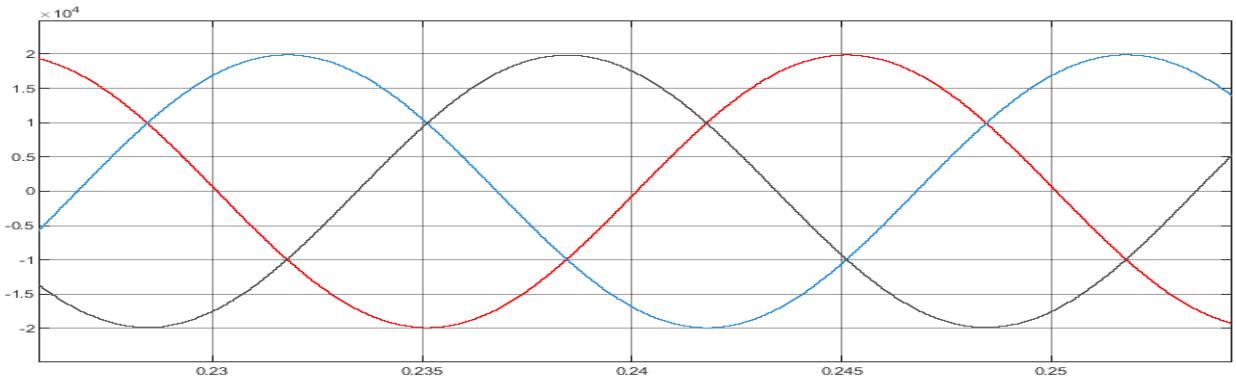


Fig. 17 Voltage during Fault operation

To simulate realistic and **challenging detection conditions**, our study restricts the HIF current to **less than 10% of the feeder's full load current**. Under such conditions, the **disturbances in the HIF current** may become almost **imperceptible** at the substation. This hypothesis is illustrated in **Fig. 17**, where the current measured at the substation during an HIF event shows minimal signs of distortion. Despite this, the strength of **mathematical morphology (MM)-based detection tools** lies in their ability to identify and differentiate even **subtle waveform anomalies**.

Furthermore, **Fig. 13** presents the **voltage waveform across the HIF**, which reveals slight but consistent distortion. As shown in **Fig. 16**, this distortion is also **reflected in the substation voltage**, highlighting its potential as a more **reliable and consistent fault signature**. Unlike current distortions, which are influenced by **pre-fault load conditions**, the distortion in the **voltage waveform** is less dependent on these factors and can therefore serve as a more **robust indicator** for HIF detection.

Fast Fourier Transform for Harmonic Content Detection

The **Fast Fourier Transform (FFT)** is a computational algorithm used to determine the **Discrete Fourier Transform (DFT)** of a sequence or its inverse (IDFT). The **Fourier Transform** enables the conversion of a signal from the **time domain** (its original domain) into the **frequency domain**, facilitating the analysis of frequency components within the signal. The DFT works by decomposing a signal into its constituent sinusoidal components at various frequencies.

In the context of this study, the FFT is employed to analyze the **normal current** and **fault current** waveforms in the frequency domain. This analysis is essential as a precursor to **wavelet packet decomposition**, which involves localized signal processing by breaking the signal into various frequency bands and sub-bands. Studying the fault current in the frequency domain helps identify the **frequency range** that contains the most informative characteristics of the signal. Consequently, this range can be targeted for more **effective feature extraction** in the wavelet domain.

In the **MATLAB environment**, the **FFT Analyzer App** is utilized to perform the frequency domain transformation of the signal data. A **20-second signal** is analyzed with a **sampling frequency of 20 kHz** and a **fundamental frequency of 50 Hz**. The FFT window is configured to divide the signal into multiple overlapping or non-overlapping **sections**, each spanning approximately **10 to 20 cycles** of the current waveform. This segmentation reduces computational complexity while allowing a consistent frequency pattern to be observed across sections. As a result, a **generalized frequency profile** is established for both normal and fault current signals, guiding subsequent wavelet-based feature extraction and analysis.

After successfully developing our model for simulating **High Impedance Faults (HIFs)** in the transmission system, the next step involves extracting more detailed information from the generated dataset to serve as input for the deep learning model.

To prepare the data for **FFT analysis in MATLAB**, the time-series signals are divided into smaller segments, each representing a 10ms interval. These segments,

referred to as chunks, are created for both normal current and fault current waveforms. Each chunk contains 2000 samples, and a **total of 200 chunks** are generated for each class, normal and faulty. These structured data chunks allow for efficient and consistent frequency analysis, enabling the extraction of meaningful features for training the deep learning model.

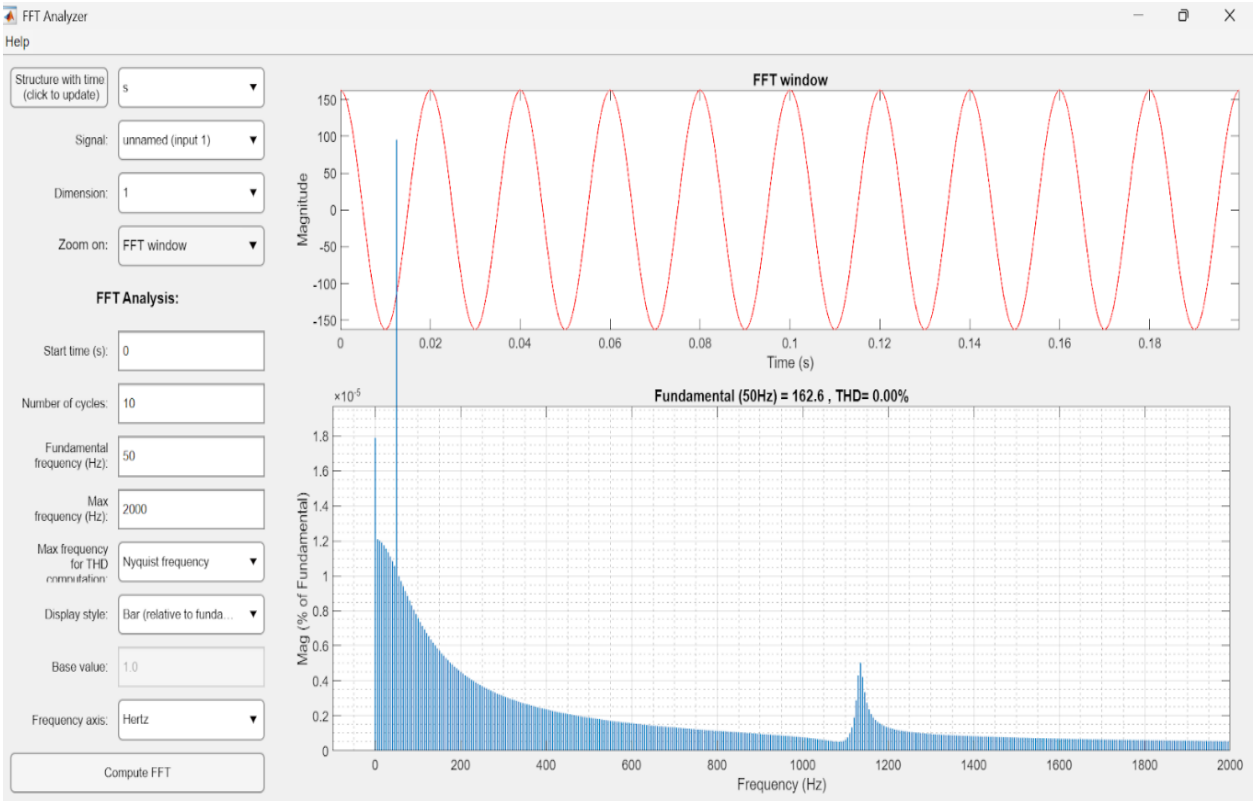


Fig. 18 **FFT** analysis for Normal current

The **spectral analysis** of the **normal current waveform**, as illustrated in **Fig. 18**, reveals the presence of **harmonic components** with magnitudes in the range of **5–10%** of the **fundamental frequency component**. Given the relatively low amplitude of these harmonics, they can be considered **negligible** for practical purposes. This observation indicates that the **normal current** predominantly consists of the **fundamental frequency (50 Hz)** component, thereby confirming that the system is operating under **stable and undistorted conditions**.

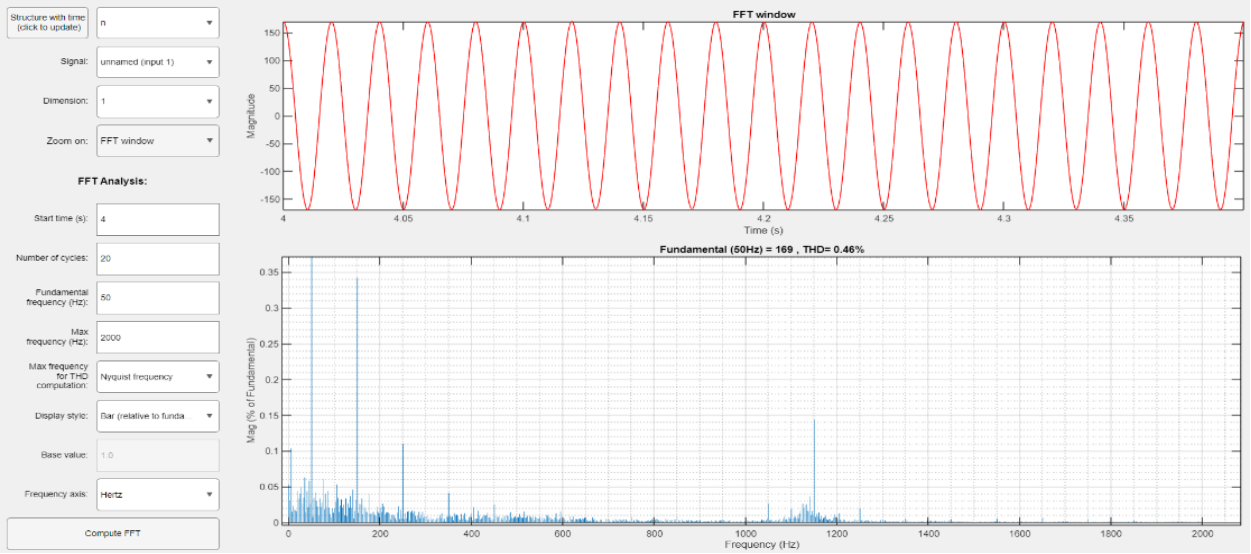


Fig. 19 FFT analysis for the Fault current

The **FFT analysis** of the **fault current dataset**, in **Fig. 19,** performed across different time intervals, revealed **consistent spectral variations** indicative of high impedance fault behavior. In each analyzed segment, distinct **harmonic components**, particularly the **3rd (150 Hz)**, **5th (250 Hz)**, and **23rd** harmonics, were found to **dominate the frequency spectrum**. This analysis ensures that the **most significant spectral features** of the fault current, maximizing the extraction of meaningful information and enhancing the effectiveness of subsequent feature-based fault classification, are selected.

Wavelet Packet analysis for the HIF simulated data

With the objective of extracting maximum information from the generated High Impedance Fault (HIF) data, the data is grouped into bundles of 2000 samples each. Each of these bundles effectively captures **100ms** of signal data, ensuring that a significant amount of information is retained within each segment. A total of 200 such bundles are created for both normal and fault condition data. These bundles are referred to as "**chunks**," and each chunk contains the corresponding current values under both normal and fault conditions for every time instant within that window.

Each chunk encapsulates valuable signal information, and to extract relevant current features and identify harmonic peaks from these signals, wavelet packet

analysis proves to be a highly effective technique. **Wavelet packets** represent an advanced **signal processing method,** particularly suited for feature extraction in applications involving time-series or spatial data. Unlike the standard wavelet transform, which only decomposes the approximation (low-frequency) coefficients at each level, wavelet packet decomposition (WPD) also decomposes the detail (high-frequency) coefficients. This results in a more comprehensive and versatile breakdown of the signal across various frequency bands.

These wavelet packet decompositions yield arrays of coefficients corresponding to specific frequency ranges, which can be easily utilized for further analysis. Referring to the FFT analysis conducted earlier, it was determined that a **3-level decomposition** is most appropriate for this application, as it is capable of capturing the most informative components of the current signals. The structure of a **3-level wavelet packet tree**, along with its nodes, is illustrated below:

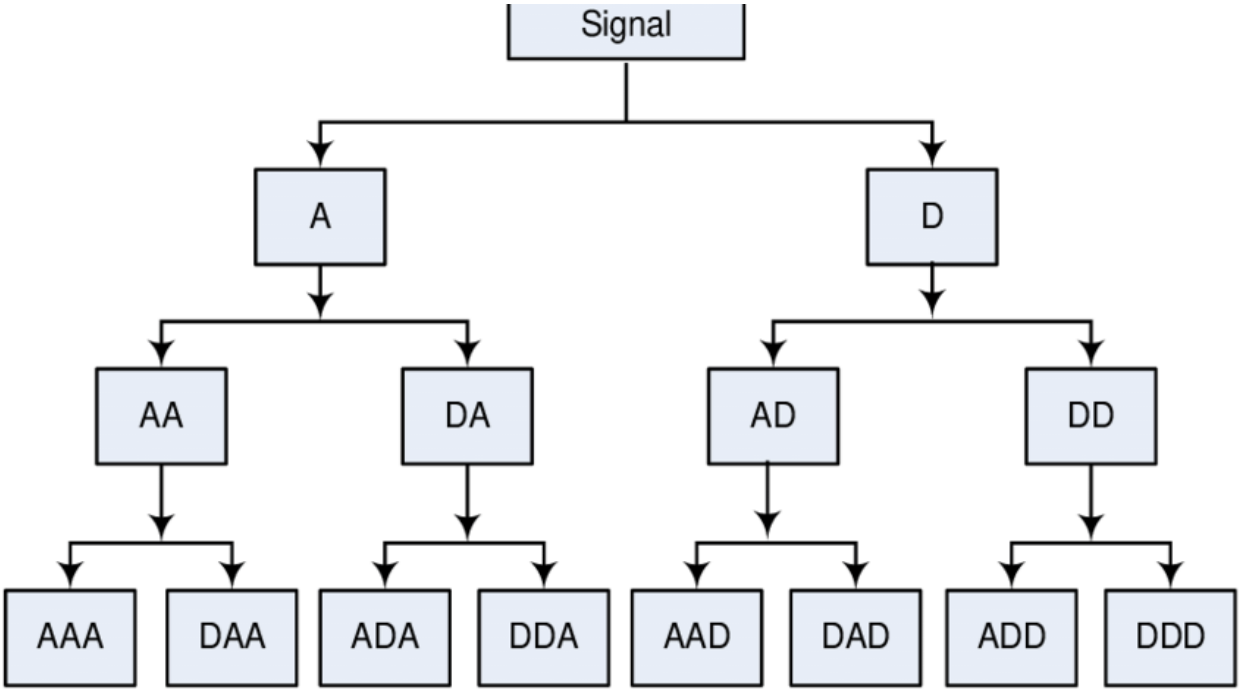


Fig. 20 Wavelet packet tree for a 3-level Decomposition

Dedicated code blocks were implemented to generate the wavelet packets corresponding to both normal and fault condition signals. The resulting outputs from these processes were subsequently stored in separate Excel files for further analysis.

```
for chunk_idx, chunk in enumerate(chunks):
    wavelet_packet = pywt.WaveletPacket(data=chunk, wavelet='db4',
mode='symmetric', maxlevel=3)
    nodes = wavelet_packet.get_level(3, 'freq')

    for node_idx, node in enumerate(nodes):
        excel_data.append({
            "Chunk": chunk_idx + 1,
            "Node": node.path,
            "Data": node.data.tolist()
        })
```

Chunk	Node	Data
1	aaa	-459.6034465
1	aaa	-459.6034362
2	aaa	-459.6034362
2	aaa	-459.603436

The generated dataset after Wavelet Packets decomposition.

Such **tabular results** are generated for both the *normal* and *fault* datasets. For each node, there are **254 rows**, and a total of **8 such nodes** correspond to a single chunk. In total, there are **200 chunks** for both the *normal* and *faulty* conditions, respectively.

This dataset now forms a **rich collection of numerical information**. To derive meaningful insights from this high-volume numerical data, we employ **descriptive statistical analysis**. Specifically, we compute **16 statistical measures** for each node within each chunk for both operating conditions.

The outcome is a **high-dimensional data-frame** that captures extensive statistical characteristics of the signal. The code blocks implemented to perform this task are provided below:

```
def give_current_stats(data_frame, no_of_chunks, patterns):
    current_stats = []
    # this loop will traverse through all of the chunk 1 -> 200
    for i in range(1, int(no_of_chunks) + 1):
        data_of_each_chunk = data_frame[data_frame["Chunk"] == i]
        # this loop will traverse through all the levels of a particular
chunk
        for j in range(0, total_levels_in_a_chunk):
            data_for_each_pattern =
data_of_each_chunk[data_of_each_chunk["Node"] == patterns[j]]

            min_val = data_for_each_pattern["Data"].min()
            max_val = data_for_each_pattern["Data"].max()
            range_val = max_val - min_val
            mean_val = data_for_each_pattern["Data"].mean()
            median_val = data_for_each_pattern["Data"].median()

            # Handle mode correctly (take the first if multiple exist)
            mode_val = data_for_each_pattern["Data"].mode()
            if not mode_val.empty:  # Check if mode is not empty.
                mode_val = mode_val.iloc[0] # Take the first mode if it
exists
            else:
                mode_val = None # Or some other appropriate value for "no
mode"

            variance_val = data_for_each_pattern["Data"].var()
            standar_deviation_val = data_for_each_pattern["Data"].std()
            q25 = data_for_each_pattern["Data"].quantile(0.25)
            q75 = data_for_each_pattern["Data"].quantile(0.75)
            iqr_val = q75 - q25
            mad_val = np.mean(np.abs(data_for_each_pattern["Data"] -
mean_val))  # Correct MAD calculation

            percentile_25 = data_for_each_pattern["Data"].quantile(0.25)
            percentile_50 = data_for_each_pattern["Data"].quantile(0.50)
            percentile_75 = data_for_each_pattern["Data"].quantile(0.75)
            percentile_90 = data_for_each_pattern["Data"].quantile(0.90)
            skewness_val = data_for_each_pattern["Data"].skew()
            kurtosis_val = data_for_each_pattern["Data"].kurt()

            summary_stats = {
                "Chunk": i,
                "Node": patterns[j],
                "min": min_val,
```

```
                "max": max_val,
                "range": range_val,
                "mean": mean_val,
                "median": median_val,
                "mode": mode_val,
                "variance": variance_val,
                "std_dev": standar_deviation_val,
                "iqr": iqr_val,
                "mad": mad_val,
                "percentile_25": percentile_25,
                "percentile_50": percentile_50,
                "percentile_75": percentile_75,
                "percentile_90": percentile_90,
                "skewness": skewness_val,
                "kurtosis": kurtosis_val,
            }

            current_stats.append(summary_stats)

    return current_stats
```

Chunk	min_a	max_d	range_a	mean_a	median_d	mode_a	variance_aa
1	-229.8811973	0.001978770436	459.7626531	0.686518407	0.0000000424677 5304	-229.8811973	53462.12486
2	-229.8811858	0.001977492712	459.7626415	0.6864874795	0.0000000389312 9671	-229.8811858	53462.09753
3	-229.8811 86	0.001977476224	459.7626414	0.6864871866	0.0000000389740 7133	-229.8811 86	53462.09754

The flattened dataset with statistical values for every node

This is how the **flattened dataset** appears — it encapsulates the **descriptive statistical information** for each **node** of every **chunk**. This enriched dataset serves as the foundation for the **subsequent stages of analysis and modeling**.

Feature Ranking: Selecting the Most Informative Feature

The resulting dataset generated from the **statistical analysis** phase is **not purely one-dimensional**; instead, it encapsulates **multi-dimensional information** derived from both **faulty and non-faulty current signals**. Each data entry in the dataset is labeled according to its corresponding class, **fault** or **normal**, and is characterized by a rich set of **features extracted through wavelet packet decomposition**.

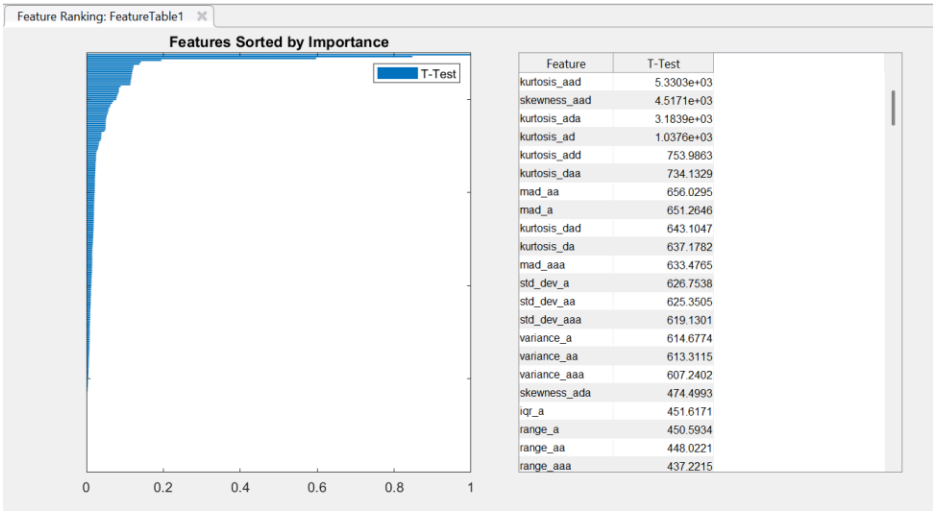
For each node in the **wavelet packet tree** (14 nodes in total), a set of **16 statistical attributes** is computed. These attributes include commonly used descriptors such as **minimum, maximum, mean, standard deviation, variance, skewness, kurtosis**, and other statistical measures. Consequently, each data entry is represented by a **feature vector of 224 attributes** (**16 attributes × 14 nodes**), capturing localized time-frequency characteristics of the signal across multiple resolutions.

In the **MATLAB environment**, the **Diagnostic Feature Designer App** is employed to **rank the 224 feature combinations** derived from the statistical and wavelet packet analysis. The dataset used in this process is classified into two distinct classes: **Class 0** for **normal current data** and **Class 1** for **fault current data**. Feature ranking is carried out using multiple statistical methods, each providing a **feature ranking score** that indicates the relevance of each feature (i.e., a statistical attribute-node combination) in distinguishing between the two classes.

The features are then **sorted in descending order of importance**, where a **higher score implies a stronger discriminative capability** for fault detection. The feature with the **highest ranking score** is considered the most effective in training a model to identify fault current patterns.

The following feature ranking techniques are applied:

1. T-Test Ranking:

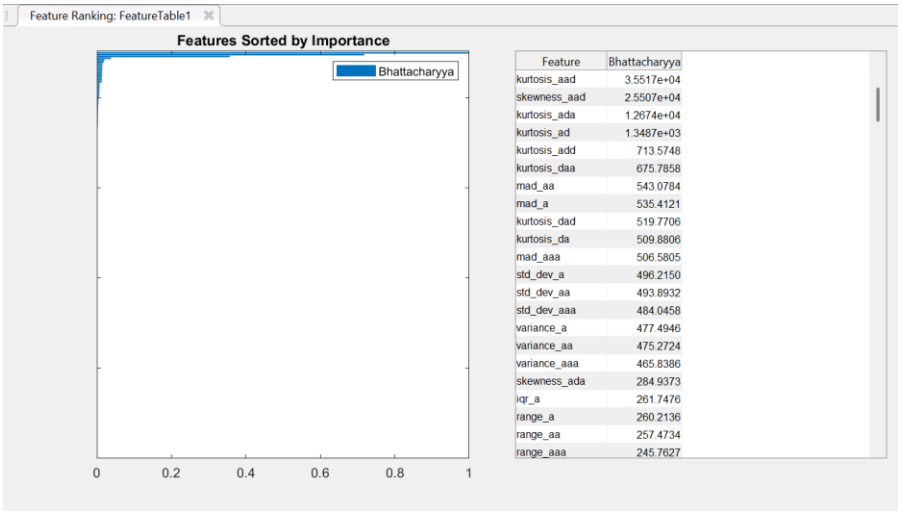


Feature	T-Test
kurtosis_aad	5.3303e+03
skewness_aad	4.5171e+03
kurtosis_ada	3.1839e+03
kurtosis_ad	1.0376e+03
kurtosis_add	753.9863
kurtosis_daa	734.1329
mad_aa	656.0295
mad_a	651.2646
kurtosis_dad	643.1047
kurtosis_da	637.1782
mad_aaa	633.4765
std_dev_a	626.7538
std_dev_aa	625.3505
std_dev_aaa	619.1301
variance_a	614.6774
variance_aa	613.3115
variance_aaa	607.2402
skewness_ada	474.4993
iqr_a	451.6171
range_a	450.5934
range_aa	448.0221
range_aaa	437.2215

Fig. 21 T-test ranking of statistical features

a. The T-test evaluates whether there is a statistically significant difference in the distribution of a feature between the two classes.
b. This method assumes the data is approximately normally distributed and is useful for binary classification.
c. The top-ranked feature using this method is *kurtosis_aad*, with an importance score of **5.3303×10^3**.

2. Bhattacharyya Distance Ranking:

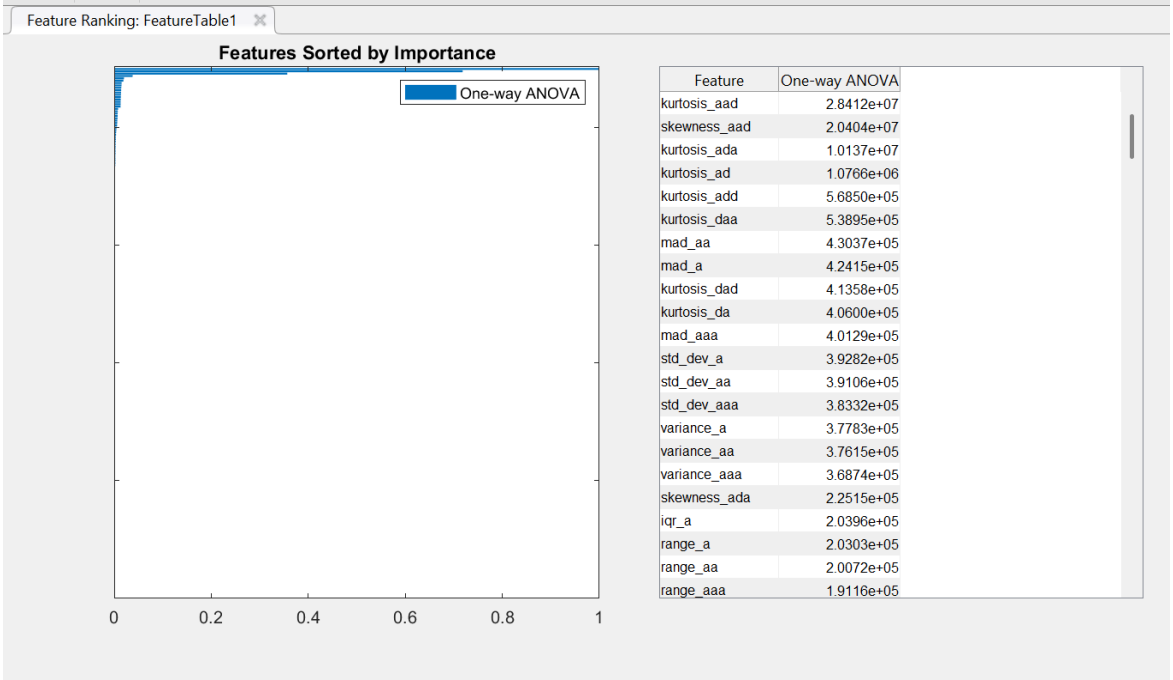


Feature	Bhattacharyya
kurtosis_aad	3.5517e+04
skewness_aad	2.5507e+04
kurtosis_ada	1.2674e+04
kurtosis_ad	1.3487e+03
kurtosis_add	713.5748
kurtosis_daa	675.7858
mad_aa	543.0784
mad_a	535.4121
kurtosis_dad	519.7706
kurtosis_da	509.8806
mad_aaa	506.5805
std_dev_a	496.2150
std_dev_aa	493.8932
std_dev_aaa	484.0458
variance_a	477.4946
variance_aa	475.2724
variance_aaa	465.8386
skewness_ada	284.9373
iqr_a	261.7476
range_a	260.2136
range_aa	257.4734
range_aaa	245.7627

Fig. 22 Bhattacharyya ranking of statistical features

a. The Bhattacharyya Distance measures the separability between two probability distributions, making it highly effective for evaluating the discriminatory power of features.
b. A higher value implies better separation between the normal and fault classes.
c. The feature with the highest score using this method is again *kurtosis_aad*, with a score of **3.5517×10^4**.

3. One-Way ANOVA Ranking:



Feature	One-way ANOVA
kurtosis_aad	2.8412e+07
skewness_aad	2.0404e+07
kurtosis_ada	1.0137e+07
kurtosis_ad	1.0766e+06
kurtosis_add	5.6850e+05
kurtosis_daa	5.3895e+05
mad_aa	4.3037e+05
mad_a	4.2415e+05
kurtosis_dad	4.1358e+05
kurtosis_da	4.0600e+05
mad_aaa	4.0129e+05
std_dev_a	3.9282e+05
std_dev_aa	3.9106e+05
std_dev_aaa	3.8332e+05
variance_a	3.7783e+05
variance_aa	3.7615e+05
variance_aaa	3.6874e+05
skewness_ada	2.2515e+05
iqr_a	2.0396e+05
range_a	2.0303e+05
range_aa	2.0072e+05
range_aaa	1.9116e+05

Fig. 23 One-Way ANOVA ranking of statistical features

a. One-Way ANOVA (Analysis of Variance) compares the mean feature values across multiple groups to determine whether significant differences exist.
b. While typically used for multi-class problems, it is also effective in identifying features with strong inter-class variance in binary classification tasks.

c. According to this method, the most dominant feature is *kurtosis_aad*, with a notably high importance score of **2.8412×10^7**, confirming its strong discriminatory potential.

The table represents a comparative overview of the top-ranked features using different ranking methods:

	Top-Ranked Feature	Importance Score
T-test	kurtosis_aad	$5.3303e^{+03}$
Bhattacharyya Distance	kurtosis_aad	$3.5517e^{+04}$
One-Way ANOVA	kurtosis_aad	$2.8412e^{+07}$

Table Showing the best-ranked feature for respective tests and their score

These methods were chosen as they offer the most effective comparison between features, particularly highlighting those with the highest importance scores. Among them, **One-Way ANOVA** provided the most distinct separation, identifying **kurtosis_aad** as the top-ranked feature with a score of **2.8412×10^7**. Since kurtosis_aad consistently ranked highest across multiple methods, it is selected as the most defining feature for training the deep learning model on fault current data.

Training the Convolutional Neural Network for the Best Feature

The most critical task at this stage is to **train a Convolutional Neural Network (CNN)** on the **one-dimensional dataset** derived from our feature extraction process. In machine learning, the standard workflow involves training a model on a labeled training set, followed by evaluating its performance on **unseen data** to assess its predictive accuracy.

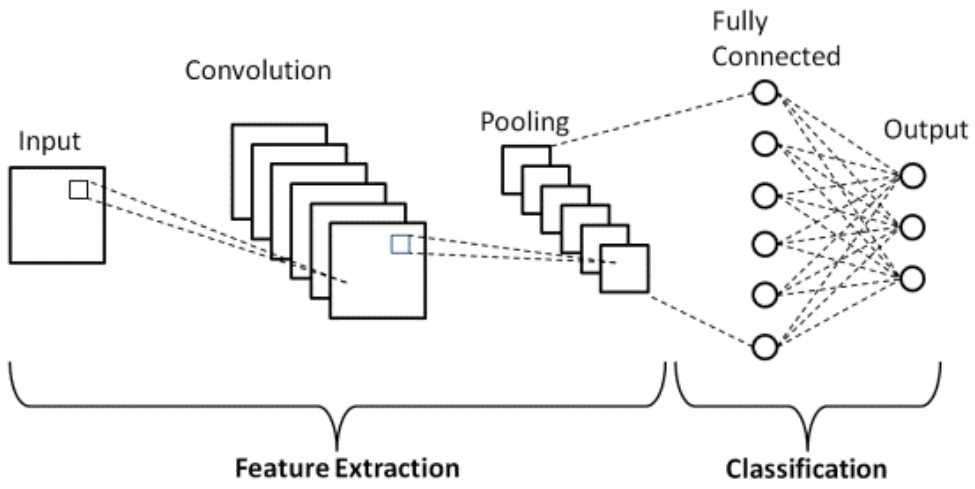


Fig. 24 Structure of CNN models

CNN, or *Convolutional Neural Network*, is a type of deep learning model specifically designed for pattern recognition. Traditionally used for image analysis, CNNs consist of several key layers, like:

- **Convolutional Layer**: Acts like a feature detector, scanning the input data for local patterns (edges, textures, trends).
- **Pooling Layer**: Simplifies the output from the convolutional layer by reducing dimensionality while retaining the most important information.
- **Fully Connected Layer**: Interprets the extracted features and produces a final classification or prediction.

While typical CNNs are used for two-dimensional image data, a **1D CNN** is adapted for sequential or time-series data. In our case, the model is trained on **one-dimensional arrays** of *kurtosis values* extracted from the '**aad**' node of each chunk via wavelet packet decomposition.

This sequence of kurtosis values forms the **input** to the model, and the associated class labels (normal or fault) act as the **output**. Once trained, the model will be able to classify any new, unseen data based on the learned features, predicting whether a signal corresponds to a **normal** or **faulty** condition. The code blocks for building, training, and evaluating the CNN model, along with the performance results, are provided below.

```
# Necessary libraries and imports
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, GlobalAveragePooling1D, Dense,
Dropout, BatchNormalization, Input
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

# Extract features and labels
features = df['feature'].values
labels = df['label'].values

# Normalize the feature values
scaler = StandardScaler()
features = scaler.fit_transform(features.reshape(-1, 1)).flatten()

# Function to create sliding window sequences
def create_sequences(features, labels, window_size):
    sequences = []
    targets = []
    for i in range(len(features) - window_size):
        sequences.append(features[i:i + window_size])
        targets.append(labels[i + window_size])
    return np.array(sequences), np.array(targets)

# Set the window size to 3
window_size = 3

# Create sequences and corresponding labels
X, y = create_sequences(features, labels, window_size)

# Split df into training and validation sets (70% train, 30% validation)
train_size = int(0.7 * len(X))
X_train, X_val = X[:train_size], X[train_size:]
y_train, y_val = y[:train_size], y[train_size:]

# Reshape df for 1D-CNN input: (samples, window_size, 1)
X_train = X_train.reshape(-1, window_size, 1)
X_val = X_val.reshape(-1, window_size, 1)

# Define the 1D-CNN model
```

```
model = Sequential([
    Input(shape=(window_size, 1)),  # Use Input layer as recommended
    Conv1D(filters=64, kernel_size=3, activation='relu'),
    BatchNormalization(),
    GlobalAveragePooling1D(),
    Dense(32, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Define early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the model
history = model.fit(X_train, y_train,
                    epochs=100,
                    batch_size=32,
                    validation_data=(X_val, y_val),
                    callbacks=[early_stopping],
                    verbose=1)
```

Based on the trained model, the results on the validation set are as follows:

- **Time for each iteration**: 15ms/step
- **Validation Accuracy**: 1.0000
- **Validation Loss**: 8.8710e-08

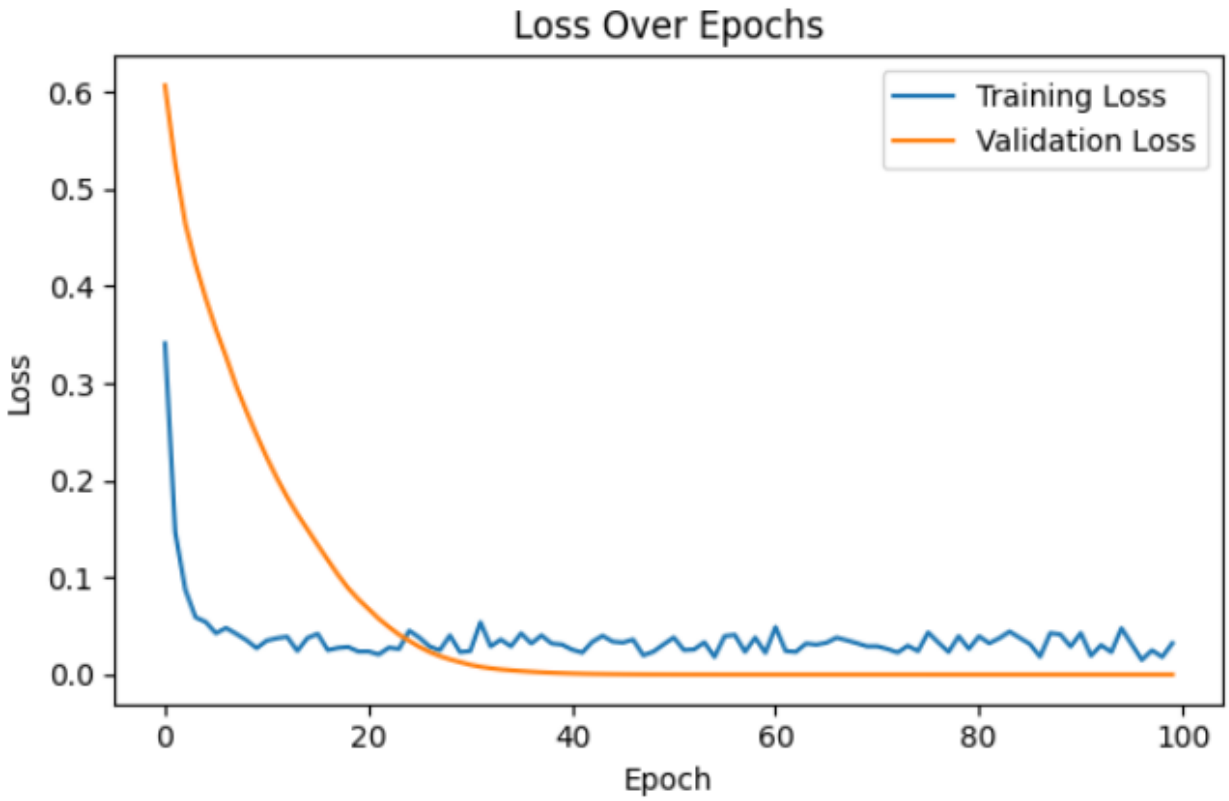


Fig. 25 Validation Loss over each iteration of model training

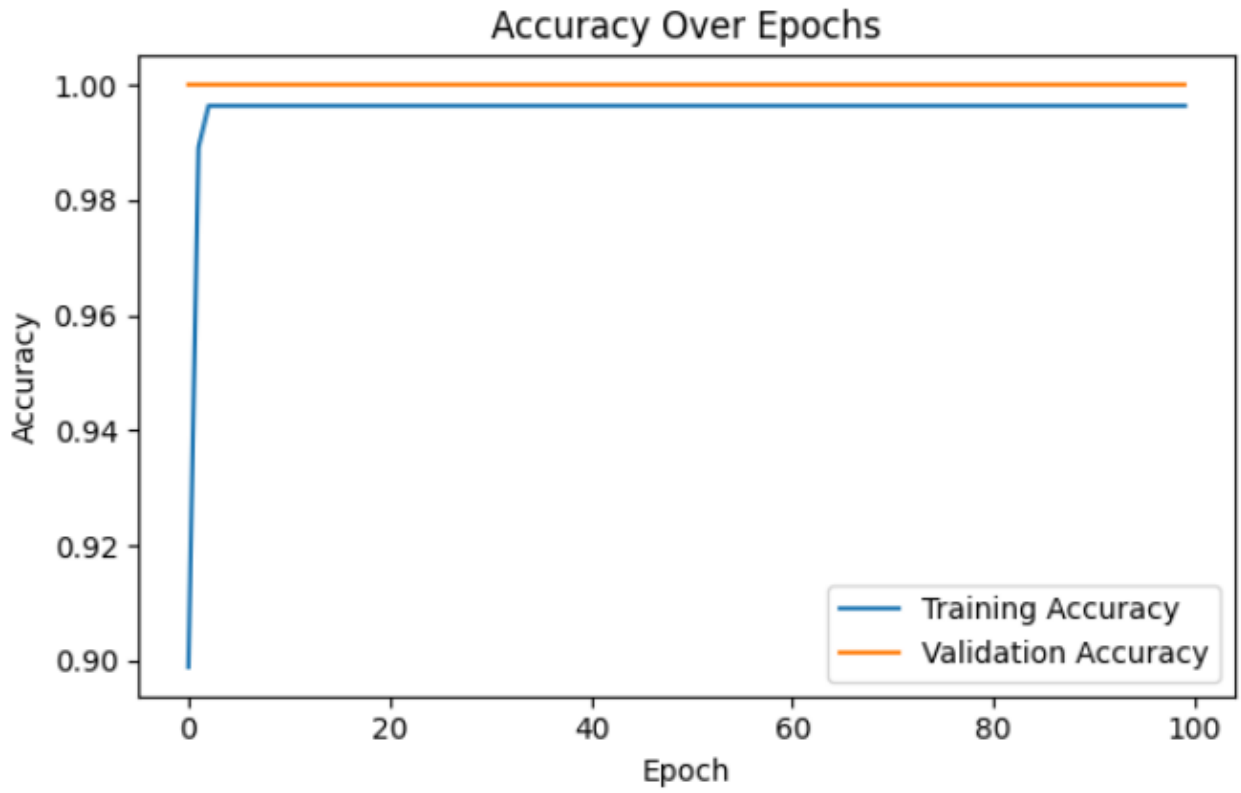


Fig. 26 Accuracy over each iteration of model training

Chapter-4: Results

After successfully training the model, the next **crucial step** is to evaluate its performance on a **new testing dataset**. This dataset is provided as a *1-dimensional array* consisting of **kurtosis values** for the *'aad'* node. The objective is to compare the model's predictions with the **actual class labels**, using various **standard classification evaluation metrics**.

To achieve this, a **dedicated code block** is used, which performs the necessary **pre-processing** on the test data and then passes it through the **trained CNN model** to generate predictions. This block accepts the *kurtosis array* as input and returns the corresponding **predicted class labels**, allowing us to assess the **accuracy and reliability** of the model. The complete code block is included below:

```
# 1. Extract kurtosis_aad values from X_test
test_features = X_test['kurtosis_aad'].values

# 2. Normalize the test data using the same scaler used during training
test_features = scaler.transform(test_features.reshape(-1, 1)).flatten()

# 3. Create sliding window sequences for test data
def create_test_sequences(features, window_size):
    sequences = []
    for i in range(len(features) - window_size):
        sequences.append(features[i:i + window_size])
    return np.array(sequences)

X_test_seq = create_test_sequences(test_features, window_size)

# 4. Reshape to match CNN input shape (samples, window_size, 1)
X_test_seq = X_test_seq.reshape(-1, window_size, 1)

# 5. Make predictions
predictions = model.predict(X_test_seq)

# 6. Convert probabilities to class labels (0 or 1), if needed
predicted_classes = (predictions > 0.5).astype(int)

# Output the predictions
print("Predicted Classes:")
```

The various **metrics** used to evaluate the performance of a classification model are as follows:

- **Confusion Matrix**: A confusion matrix is a table that provides a **comprehensive summary of prediction results** on a classification problem. It shows the number of **true positives (TP)**, **true negatives (TN)**, **false positives (FP)**, and **false negatives (FN)**, which together help in understanding how well the model is distinguishing between classes.

- **Accuracy**: Accuracy is the most intuitive metric and represents the **ratio of correctly predicted observations to the total observations**. It gives a general idea of how often the model is correct, but it may not always be reliable in the case of **imbalanced datasets**.

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

- **Precision**: Precision indicates the **proportion of true positive predictions out of all positive predictions made by the model**. It is a critical metric when the **cost of false positives is high**, such as in fault detection systems where incorrect alerts should be minimized.

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

- **Recall (Sensitivity)**: Recall is the **ratio of correctly predicted positive observations to all actual positives**. It becomes especially important in scenarios where **missing a positive instance (false negative)** is more critical, such as failing to detect a fault condition.

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score**: The F1-score is the **harmonic mean of precision and recall**, offering a balanced measure when both false positives and false negatives are crucial. It is especially useful when we need to find an **optimal balance between precision and recall**.

$$f1 - score = \frac{1}{Precision} + \frac{1}{Recall}$$

For our testing dataset, the results we obtained are as follows:

- Confusion Matrix

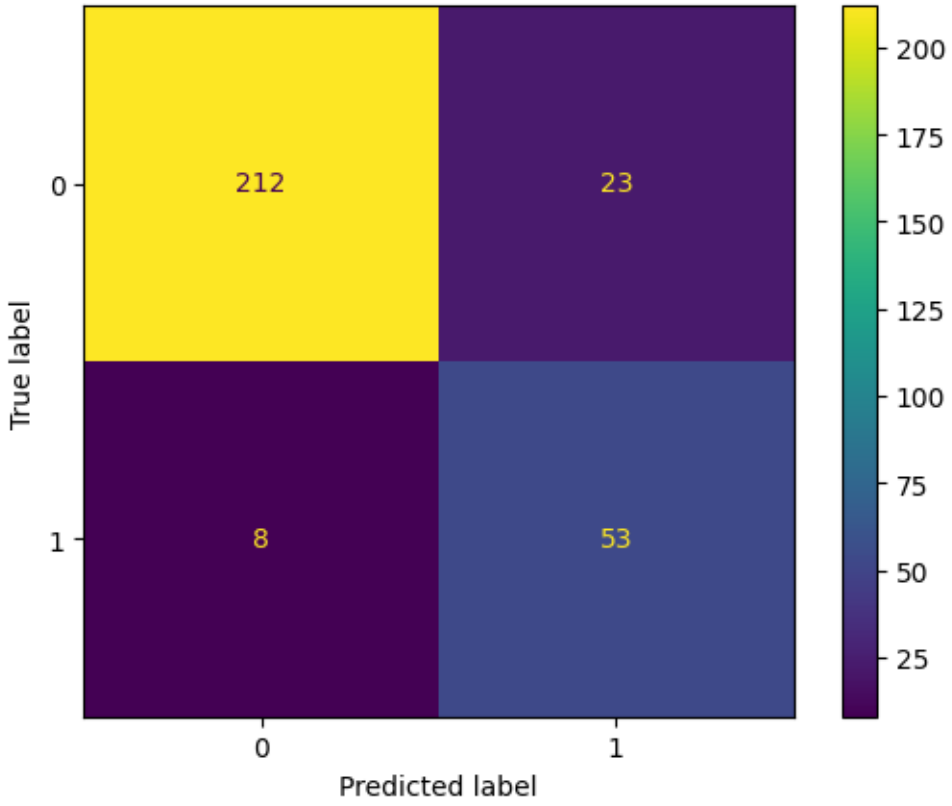


Fig. 27 Confusion matrix for the Test set.

- Accuracy on the test set = 89.527%
- Precision on the test set = 69.736%
- Recall on the test set = 86.885%
- f-1 Score for the test set = 77.372%

Chapter-5: Future Work

To extend the applicability and robustness of the proposed system, potential areas for future work have been explored. These include:

1. Real-Time Fault Detection System

The next immediate step is to transform the existing offline fault detection model into a real-time system. This would involve integrating the Simulink model with live data inputs, allowing the trained CNN to process and classify signals in real-time. Implementing such a system would significantly improve its practical value, enabling proactive monitoring and faster response to faults.

2. Hardware Implementation

To further extend the system's utility, a hardware implementation using embedded platforms such as Arduino or other microcontrollers is planned. This would allow the deployment of the fault detection model in real-world environments where continuous monitoring is essential, such as industrial or residential electrical systems. However, due to the computational limitations of microcontrollers, optimizations or model compression techniques may be required.

3. Continuous Learning and Model Update

Since real-world conditions are often more variable than simulated environments, the base model may not perform accurately under all circumstances. Therefore, it is crucial to develop a mechanism for continuous training or online learning. This would ensure that the model adapts to new patterns and fault types over time, thereby improving its reliability and accuracy in dynamic conditions.

4. Comparative Study with Other Machine Learning Models

Although CNNs have shown promising results, future studies can include a comparison with other machine learning and deep learning models like Recurrent Neural Networks (RNNs) and traditional classifiers like Support Vector Machines (SVMs) or Random Forests to identify the best-performing approach under different scenarios.

Chapter-6: References

1. K. Dubey and P. Jena, "A novel high-impedance fault detection technique in smart active distribution systems," 2024 IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 71, NO. 5, MAY 2024.
2. M. Guo and D. Chen, "A Data-Enhanced High Impedance Fault Detection Method Under Imbalanced Sample Scenarios in Distribution Networks," IEEE Transactions on Power Delivery, vol. 59, no. 4, 2023.
3. Nezamzadeh-Ejieh, S., and Sadeghkhani, 1. (2020). HIF detection in distribution networks based on Kullback-Leibler divergence. IET Gener. Transm. Distrib. 14 (1), 29-36. doi:10.1049/iet-gtd.2019.0001
4. Aucoin, B. M., and Russell, B. D. (1982). Distribution high impedance fault detection utilizing high frequency current components. IEEE Trans. Power Appar. Syst. (6), 1596-1606. PAS-101. doi:10.1109/TPAS, 1982.317209
5. Furse, C. M., Kafal, M., Razzaghi, R., & Shin, Y.-J. (2021). “Fault diagnosis for electrical systems and power networks: A review”. IEEE SENSORS JOURNAL, VOL. 21, NO. 2.
6. Gadanayak, D. A., & Mallick, R. K. (2019). “Interharmonics based high impedance fault detection in distribution systems using maximum overlap wavelet packet transform and a modified empirical mode decomposition”. *IEEE Access*, 7, 142741-142751. International Journal of Electrical Power & Energy Systems.
7. X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," 2024. Artificial Intelligence Review (2024).
8. Aljohani A, Habiballah I. High-Impedance Fault Diagnosis: A Review. *Energies*. 2020; 13(23):6447. https://doi.org/10.3390/en13236447
9. C. G. Wester, "High impedance fault detection on distribution systems," 1998 Rural Electric Power Conference Presented at 42nd Annual Conference, St. Louis, MO, USA, 1998, pp. c5-1, doi: 10.1109/REPCON.1998.666955.

10. A. H. Eldin, E. Abdallah and N. Mohamed, "Detection of high impedance faults in medium voltage distribution networks using discrete wavelet transform," *22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013)*, Stockholm, 2013, pp. 1-4, doi: 10.1049/cp.2013.0608.
11. Hao Bai, Bingnan Tang, Tianyu Cheng, Hongwen Liu,High impedance fault detection method in distribution network based on improved Emanuel model and DenseNet,Energy Reports,Volume 8, Supplement 10, 2022.
12. S. Gautam and S. M. Brahma, "Detection of High Impedance Fault in Power Distribution Systems Using Mathematical Morphology," in *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1226-1234, May 2013, doi: 10.1109/TPWRS.2012.2215630.