Global Scientific JOURNALS

GSJ: Volume 11, Issue 2, February 2023, Online: ISSN 2320-9186
www.globalscientificjournal.com

AN EFFECTIVE PARTIAL DISCHARGE DETECTION IN A NATIONAL GRID DISTRIBUTION NETWORK USING A DEEP LEARNING APPROACH

BY

AINA, PETER KEHINDE

MATRICULATION NUMBER: AUO/12/1813

A DISSERTATION SUBMITTED TO THE DEPARTMENT OF MATHEMATICAL SCIENCES, ACHIEVERS UNIVERSITY, OWO, ONDO STATE, NIGERIA.

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF SCIENCE (M.Sc.) DEGREE IN COMPUTER SCIENCE.

MAY, 2022

DEDICATION

This project is dedicated to Almighty God and my late mother, Mrs. Aina Modupeola.

DECLARATION

I, **AINA PETER KEHINDE,** declare that;

i. This research work was undertaken by **AINA PETER KEHINDE** of the Department of Mathematical Sciences, School of Postgraduate studies, Achievers University, Owo, supervised by **DR. E. O. OYEKANMI**.

ii. This work has not been submitted in any form for the award of any other degree or diploma.

_____

AINA PETER KEHINDE

ACKNOWLEDGEMENT

I am giving all the credit and thanks to God for his mercy towards my family and me throughout the program

I also extend my heartfelt gratitude to my supervisor, Dr. OYEKANMI E.O., for his valuable advice and guidance, and I pray that God blesses him and his family.

I also want to acknowledge the support of Prof. L.O. ADETULA, who stood up for us during a sudden increase in tuition fees and other university issues. I also pay homage to the late Prof. Tunji IBIYEMI, who was loved by us students, and I wish his soul peace.

I am grateful to Prof. S.A. DARAMOLA, DR. T.O. OMOTEHINWA, and DR. E.F. AYETIRAN for their contributions to my success in my studies. In addition, I appreciate my Dean of the College of Natural and Applied Sciences and all staff and members of the School of Postgraduate Studies at Achievers University Owo.

Lastly, I want to express my appreciation for my late parents MR & MRS AINA BABATUNDE, who encouraged me to pursue academic excellence, and for my wife,

Mrs. Aina Hellen Tosin, my son and daughter (Aina Racheal and Aina Olusegun), for their prayers and support throughout my studies. I pray that God blesses all of them.

CERTIFICATION

This is to certify that the work reported in this dissertation was carried out by **AINA PETER KEHINDE** of the Department of Mathematical Sciences, Achievers University Owo, in partial fulfillment of the requirements for the award of the degree of Master of Science Computer Science.

Dr. E. O. Oyekanmi _____

Supervisor ***Signature and Date***

Prof. L.O. Adetula

_____

Head of Department ***Signature and Date***

ABSTRACT

ABSTRACT

Partial discharge (PD) is a common method for detecting faults in power equipment like generators and cables. These faults often result in power outages and costly repairs. The traditional method for detecting PD relied on field workers to identify specific pulses in the equipment using their expertise and hand-crafted features, particularly in rural areas. This research project aimed to predict partial discharge occurrences based on a ten-year manual record of field workers' observations of 200 transformers in a 33/0.415kv distribution line and 56 transformers in an 11/0.415kv distribution line in a rural area. The researchers used a novel approach that involved transforming non-image data into image maps and used parallel convolutional neural networks to classify partial discharge occurrence best. This research offers two key contributions: first, it

provides a fast and efficient way to extract features and detect PD by transforming non-image data into image maps. Second, it provides interpretability of the results for new domain experts by identifying the immediate record, which could lead to the detection of PD in an area. The performance was evaluated using binary classification metrics such as confusion matrix, accuracy score, and F1-score.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND OF STUDY

Electrical power distribution and transformation between different voltage levels are crucial to a country's power grid, which is why substations play a vital role. However, the high voltage levels during long transmissions increase the risk of partial discharges. To prevent these discharges, which cause power loss and pose a risk to workers, it is essential to provide adequate protection through healthy insulators. The need for good insulators is a critical factor for the functioning of any substation. Eroded insulation can result in partial discharges and is often caused by defects or irregularities in the insulating material. Unfortunately, this erosion can lead to the destruction of city power grid substations (Martirano, L., Chavdarian, 2015). Partial discharge (PD) detection in power distribution grids is crucial in ensuring the high-voltage equipment's reliability and preventing power outages. Traditionally, PD detection has been performed through manual inspection or conventional techniques such as Ultra-High Frequency (UHF) and Acoustic Emission (AE) analysis (Du and Wang, 2020).

Moreover, regular monitoring of power distribution grids is necessary to ensure the reliability of high-voltage equipment such as generators, motors, transformers, and power cables and to reduce maintenance costs. When a distribution grid failure occurs due to electrical insulation failure, it takes a long time and is expensive to manually locate the damaged lines. This can result in power outages, which can be caused by partial discharges - a type of electrical discharge that does not completely bridge the gap between insulation systems. Partial discharges gradually harm the power lines and, if left unchecked can lead to power outages or start fires (Wu et el., 2015). Renforth et al. (2015), measuring partial discharge can be done in two ways namely: electrical method (based on IEC 60270-2000 standard) with high-frequency

current transform (HFCT) and detecting impedance. Another way is through a non-electrical method with acoustic, optic, and ultra-high frequency (UHF).

Recently, the use of deep learning algorithms in PD detection has gained popularity due to their ability to automatically identify patterns and anomalies in data, resulting in more accurate and efficient detection of PD events (Zhou et al., 2020). In a study by Chen et al. (2021), a deep learning approach was used to detect PD in a national grid distribution network, demonstrating improved accuracy compared to conventional techniques. The use of deep learning algorithms in PD detection provides a promising solution for improving the reliability and efficiency of power distribution grids. Further research is needed to fully validate this approach's effectiveness and investigate its potential for widespread implementation in real-world power distribution networks (Wu et al., 2015).

However, it has been established by Niasar et al. (2021) that using excitation voltages at variable frequency during measurement do increase the charging power of the measuring equipment; a more efficient way is to use a power-frequency (50/60 Hz) sinusoidal voltage, which reduces the charging power while measuring. Either electrical or non-electrical, the research conducted by Khan et al. (2019) on End- to- End Partial Discharge Detection in Power Cables has shown that PD signals is a standardized diagnostic tool that can be used to monitor the state of different electrical apparatus. The method uses PD pulses as input with one-dimensional convolutional neural networks (CNNs) to automatically extract meaningful features for waveforms of PD pulses and finally detect PD. In this research, PD signals daily of affected areas within the Akoko metropolis of Owo in Ondo State, Nigeria, from 2011 until 2021, manually recorded using megger- an insulation tester. These records would be used to predict failure on the distribution grid on a new installation of electrical components.

1.2 MOTIVATION

Many researchers have worked on predicting the occurrence of partial discharge. Pereira et al. (2020) worked on the prediction of partial discharges in the stator insulation system of a hydro-generator using lambda architecture which combines real-time and batch processing techniques. The prediction accuracy was increased by developing a forecasting model which is updated by the real-time data generated from the speed layer in lambda architecture. However, it was discovered that the partial discharge monitoring system that includes sensors, data architecture, and the autoregressive forecasting model was implemented in the plant under unsupervised scenarios. The researchers do not have enough data to comprehensively analyze the hydro generator condition. Thus, this methodology gives way to the high influence of outliers in data on the forecast performance evaluation and therefore has lowered the reliability of the results.

According to Yeo et al. (2020), the complications resulting from the outliers during prediction resulted from nonstationary noise influence. A coherent multi-step PD prediction model by applying a Discrete Wavelet Transform (DWT) on the measurement to de-noise the noise from PD pulses without affecting the integrity of the waveform was proposed by Yeo et al. (2020). The result is further analyzed with a deep learning system that combines recurrent neural network (RNN) and Long Short-Term Memory (LSTM) to identify the PD accurately. The state of art on PD detection has shown that deep learning is efficient in predicting the occurrence of partial discharge. Therefore, the focus is to explore the convolutional neural network, an aspect of deep learning in detecting the occurrence of partial discharge on the distribution grid within the Akoko metropolis.

1.3 SCOPE OF STUDY

The scope of this project was to study PD within the Akoko metropolis of Owo in Ondo State, Nigerian based on manual records (dataset) of investigated partial discharge in the past ten (10) years, as well as developing a method to identify potential PD in substations before the

insulating material reaches a critical level of corrosion. This is needed to perform needed maintenance, preventing component failure and thus increasing the safety of personnel.

The study of partial discharge within the Owo Business unit of Benin Electricity Distribution Company (BEDC) Akoko has been on for the past ten years under the supervision of the researcher Aina (2022) using a manual way to keep the records associated with partial discharge.

1.4 STATEMENT OF PROBLEM

A lot of research has been done on detecting partial discharge with different methods. Each researcher mostly generates the data used in a particular domain to implement the research. In this research, the downtime rate of electricity supply to the population of Akoko land is very alarming and has resulted in a high degree of control over the condition of the electrical machines used in the area. From a financial point of view, the cost for maintenance on the part of the BEDC is too high, and this occurrence is yearly. Many costumers had lost a lot financially because of the downtime of electricity due to unceasing partial discharge at various locations with the Akoko Distribution network of Benin Electricity Distribution Company in Ondo State, Nigeria. Tracing the faulty point on the network has also been a challenge for the fieldmen who supervised the region.

Could there be a way to develop an effective classifiers using the ten (10) years of data to continuously monitor power lines for faults within the Akoko community on an installation of new equipment (for instance, a meter) within the metropolis? Hence, this research used a deep learning approach to solve this problem.

1.5 AIMS

This study aims to predict the occurrence of partial discharge on distribution network.

1.6 OBJECTIVES

The objectives of this research are to:

i. To create a dataset from manual records of partial discharge events of power distribution in the Akoko metropolis for the past ten years.

ii. To develop an optimized Convolutional Neural Network (CNN) model in TensorFlow to rightly classify the dataset resulting from the investigation in objective (i).

iii. To evaluate the model in objective (ii) with binary classification metrics such as confusion matrix, F1-Score, and an accuracy score.

1.7 METHODOLOGY

The Distribution network of the distribution line in Benin Electricity Distribution of Akoko Area has 200 transformers on 33/0.415kv distribution lines and 56 transformers on 11/0.415kv distribution lines. All records of partial discharge occurrence and all associated with it are manually recorded in a central book located in a control room. A Megger meter is mainly used, especially when the down time has exceeded 72 hours. The tool determined the condition of the insulation on the wire, generators, and motor windings. Agile method was used to convert the manual records to a soft-form (dataset) for easy computation.

CNN architectures, a method that uses the successive application of convolutional layers to an input (image) and periodically down-sampling the spatial dimensions while increasing the number of feature maps, was used to classify whether there is partial discharge or not. The dataset was transformed into an image using the DeepInsight method. The output was passed as an input to a convolution neural network architecture.

1.8 TOOL USED

Google Colab was used as a tool to implement this research. The programming language used was Python.

CHAPTER TWO

LITERATURE REVIEW AND THEORETICAL REVIEW

2.1 INTRODUCTION

Effective smart diagnosis of the problem (this time, partial discharge) on the distribution line is essential to avoid expensive electrical network system outages, especially in cases where there is no established control room, as this could result in a blackhouse for the entire region. Measurements and analyses of partial discharges (PD) are frequently employed in electrical equipment, including transformers, rotating machines, medium-voltage cables, and gas-insulated switchgear, as a diagnostic indicator of insulation damage (Montanari, 2013; Stone, 2005). With the development of smart technology, smart diagnosis is now being implemented using more automated procedures, which makes diagnosis quicker and more accurate, necessitating its consideration in the identification of partial discharge on the distribution line.

Aside from this, considering the age and size of the usage of most electrical substation-installed plants (for instance, transformer), the increasing and overall failure trend will continue to grow unless a directed maintenance program is implemented (Harry, 2022). For example, Akoko metropolis, the area of focus in this research, has 45% of their distribution network substation with an average age of failed cable of 15.6 years that were first installed 33 years ago. These utilities need a predictive maintenance tool to allow proactive maintenance before unplanned customer outages. By using a smart diagnostic measure, it is hoped to accomplish termination or splice replacement before failure and to provide scheduled cable system maintenance. Additional budget information will be gathered to reduce costs and target replacement money (Harry, 2022).

Both off-line and on-line diagnostics are commercially available to determine the condition of cable and cable accessory insulation. However, the one used mostly in the area of interest for this research is off-line. According to Harry (2022), two major types of insulation degradation occur in cable systems. One is an average or overall condition caused by chemical aging and water treeing. The diagnostics used for this type of aging include dissipation factor (loss angle), harmonic analysis, return voltage, isothermal relaxation current, dielectric response, or dc leakage current (Willem, 1999; Kraig, 1998). The second type of degradation is discrete or incremental condition assessment that utilizes dissipation factor measurements or partial discharge (PD) level measurements. This research focused on PD diagnostics applied explicitly to detecting degradation in cable accessories on 33/0.415kv and 11/0.415kv distribution lines. No matter the type of diagnostic used, it is always applied in a non-destruction manner so that the diagnostic itself does not reduce cable or accessory life (Harry, 2022). Table 1.1 shows the summary of diagnostic techniques used to determine the condition of cable and cable accessory insulation, as Harry (2022) discussed.

Table 1.1: Summary of Diagnostic Techniques for condition of cable.

Destructive		**Non-Destructive Off-line**		**Non-Destructive, On-Line**
Dissection and Microscopic Examination	Electrical Tests	Methods to Detect Singular Faults	Integral Measurement Methods	Integral & Singular Measurement Methods
Contaminants, Voids, and Water Trees	60 Hz step tests, acbd tests, radial "power factor."	PD at 60 Hz Kinectrics	Dissipation Factor at 0.1 Hz BAUR, Kinectrics	Leakage Current Sumitomo
Chemical Evaluation FTIR, DSC, DMA, OIT, DP, PIXE	DC hipot Resonant Cct 0.1 Hz Sine Oscill.Wave Impulse	PD at 0.1 Hz KEMA	Dielectric Spectroscopy ABB	DC Component Fujikura
Mechanical Evaluation Tensile, Elongation, Burst test	DIACS	PD Location System (<2U0) (IMCORP)	LIpATEST Powertech	Harmonic Current Sumitomo & NRC
		CDA & OWTS PD Lemke & Univ. Delft	Isothermal Relaxation Current SINTEF	PD, Power KEMA, DTE, Sumitomo & Eaton
			Return Voltage Hagenuk & Univ. of Siegen	

(Source: https://municipalinfonet.com/energy/magazine/44/article/Partial-Discharge-Testingof-In-Situ-Power-Cable-Accessories-An-Overview-.htm)

Furthermore, Cable accessories are treated quite differently from cables. For example, the accessory design is not always properly tested. They are man-made in the field, so the workmanship is a concern, and they are not properly tested after installation. Most cable accessory materials are more resistant to partial discharge activity than the cable and will withstand PD and treeing activity longer than the adjacent cable insulation. However, there are likely more defects in a cable accessory than in a cable, so PD detection is more applicable to cable accessory assessment (Willem, 1999).

2.2 PARTIAL DISCHARGE BACKGROUND

Partial discharge is a phenomenon caused by electrical field stress concentrated at one point. A localized electrical discharge partially bridges the insulation between conductors and can or cannot occur adjacent to a conductor (International Electrotechnical Commission, 2015). When the PD phenomenon occurs, it results in an extremely fast transient current pulse with a rise time and pulse width that depends on the discharge type or defect type, as shown in Figures 2.1a and 2.1b.

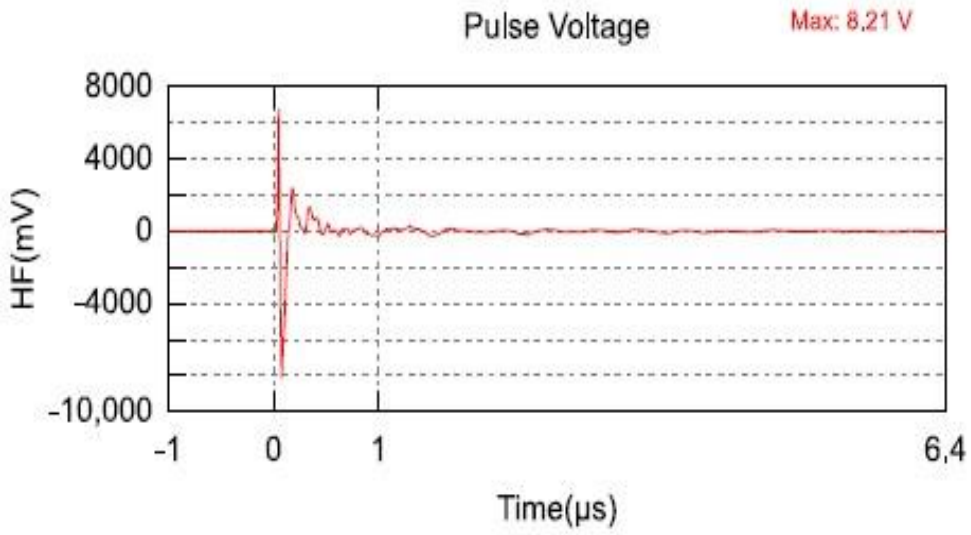


Figure 2.1a: Partial discharge waveform in time-domain(Barrios et al., 2019)

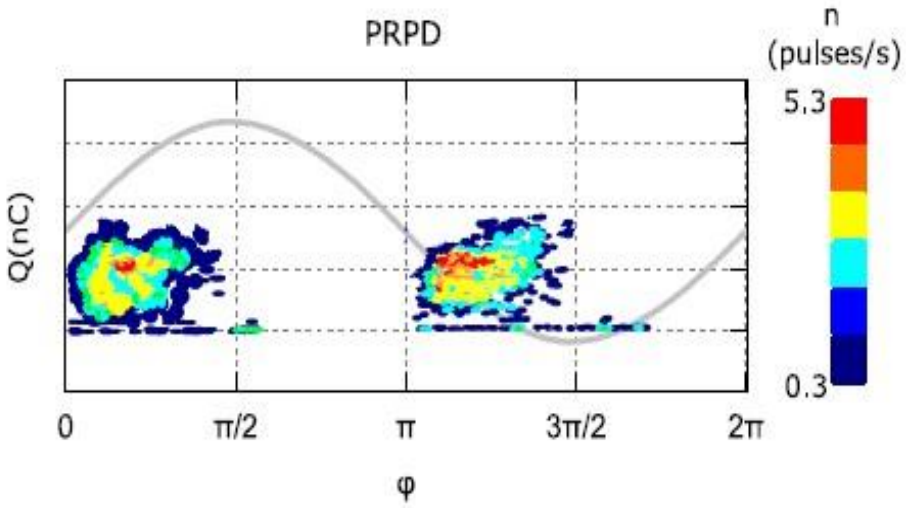
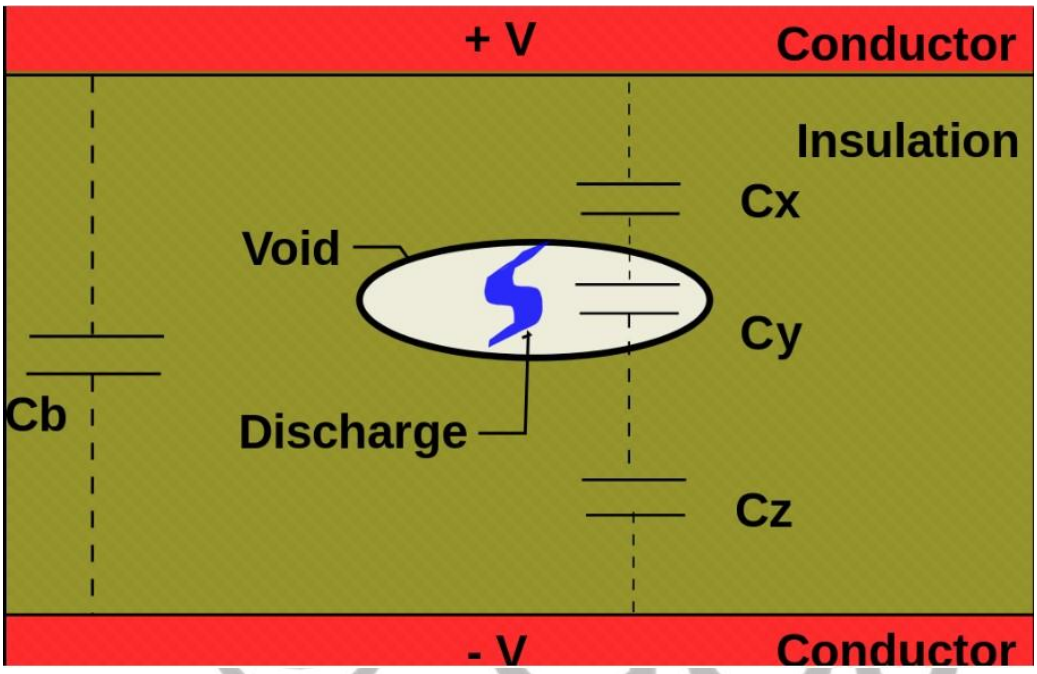


Figure 2.1b: Phase-resolved Partial Discharge (PRPD) Pattern(Barrios et al., 2019)

According to Danikas (1997), the energy inherent in a partial discharge leads to damage to the material surface surrounding the void or cavity of an insulator, as shown in Figure 2.1b. In the graph, surface erosion occurs, and electrical trees initiate and grow in the body of the insulating material. The process becomes self-perpetuating until the electrical tree bridges the insulation, and a complete breakdown occurs. Partial discharge accompanies the whole process.

2.2.1 PARTIAL DISCHARGE BASIC THEORY

Based on IEC 60270-2000 standard, partial discharge is partly (locally) electric breakdown which only connects insulation between conductors partially (Khayam, 2015). It is caused by electrical field stress concentrated at one point, either inside the insulation material or at the insulation surface, which is caused by heterogeneity in insulation shape, void or contaminant in insulation material, and imperfection of insulation shape (for example, a crack in insulator). A partial discharge within solid insulation is shown in Figure 2.2. When a spark jumps the gap within the gas-filled void, a small current flows in the conductors, attenuated by the voltage divider network Cx, Cy, and Cz in parallel with the bulk capacitance Cb (Partial Discharge, 2021).



Source: https://en.wikipedia.org/wiki/Partial_discharge

Figure 2.2: Mechanism of partial discharge caused by a void in solid insulation

The occurrence of PD usually differs in cable accessories. Firstly, it usually begins within voids, cracks, or inclusions within a solid dielectric, at conductor-dielectric interfaces within solid or liquid dielectrics, or in bubbles within liquid dielectrics. Since PDs are limited to only a portion of the insulation, the discharges only partially bridge the distance between electrodes (Partial Discharge, 2021). Secondly, PD can occur along the boundary between different insulating materials within gas-filled voids in the dielectric. When the dielectric constant of the void is considerably less than the surrounding dielectric, the electric field across the void is significantly higher than that across an equivalent distance of dielectric (Partial Discharge, 2021). Also, PD can occur along the surface of solid insulating materials if the surface tangential electric field is high enough to cause a breakdown along the insulator surface. This

phenomenon commonly manifests itself on overhead line insulators, particularly on contaminated insulators during days of high humidity (Partial Discharge, 2021).

According to Koch and Kruger (2012), partial discharge can cause primary and secondary winding damage in a transformer. 34% of transformer failure reasons lie in frequent partial discharge with no intervention. In high-voltage motors, partial discharge can cause stator winding damage which causes 37% of motor failure (Renforth et al., 2013). Figure 2.3 shows the partial discharge patterns in natural liquid insulation for High Voltage Applications.

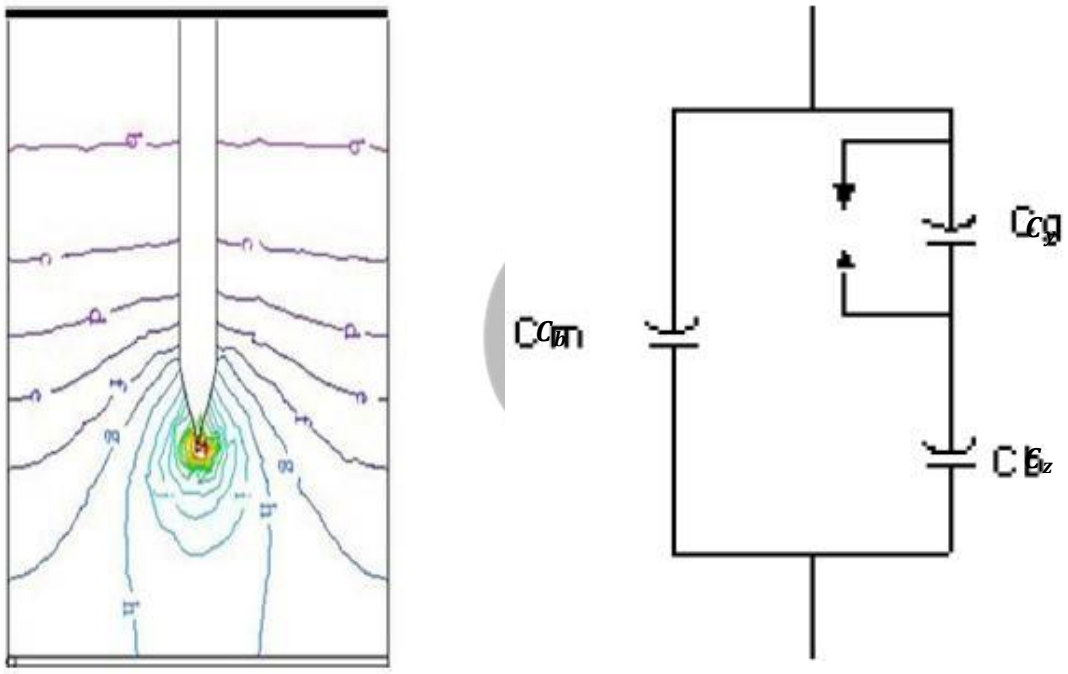


Figure 2.3: The electrode system of natural oil discharges and electric field distribution (Sipahutar et al., 2013)

According to Sipahutar et al. (2013), natural liquid (mineral oil) is still widely used as an insulating material for transformers. Many factors do affect the quality of the mineral oil. These include moisture, electrical stress, particle, thermal stress, and other factors. These stresses generate Partial Discharge (PD), which may lead to a complete breakdown of insulation while the degradation of insulation material occurs (Muhr et al., 2004; Sally et al., 2012). Therefore, continuous monitoring is needed to maintain the condition of the transformer oil. One of the

methods used to monitor oil conditions in a transformer is PD measurement. This method is an important diagnostic technique used as a non-destructive test for high-voltage insulation (Muhr et al., 2004). PD in mineral oil is strongly associated with cavity formation within the mineral oil, as shown in Figure 2.3.

In Figure 2.3, the insulating liquid is represented as capacitances C . The discharge is represented by capacitance C_y in parallel with a spark gap. The charge in the capacitance depends on the value of capacitance and the applied voltage across the capacitance. This charge strongly relates to the partial discharge magnitude. C_z represents the capacitance of the sound part of the natural oil insulation, while the rest of the sample is represented by a capacitance C_b.

For the externally applied voltage of $V(t)$, the voltage applied to the PD capacitance before any discharge take place can be expressed mathematically as shown in equation 2.1 as:

$$V_y(t) = \frac{C_z}{C_y + C_b} * V(t) \quad (2.1)$$

When a partial discharge takes place, the voltage on the discharge site falls to a very low value of residual voltage V_r. The magnitude for the first discharge will then be proportional to the PD capacitance and the different between PD voltage and residual voltage. In other words, the PD magnitude M can be expressed as

$$M = C_y k(V(t) - V_r) \quad (2.2)$$

where V_r is the residual voltage after a discharge occurs. According to (Suwarno & Sutikno, 2011), the residual voltage after the discharge V_r is much smaller than V(t). The small residual voltage is resulting in a small phase shift of PD occurrence. K is the proportionality constant

2.2.2 PARTIAL DISCHARGE DIAGNOSIS MECHANISM

According to (Khayam, 2015), the partial discharge diagnosis system has some mechanisms. These mechanisms can be detected and measured in signal processing or pattern recognition.

2.2.2.1. Partial discharge detection and measurement

Two methods are involved in this mechanism. These include the electric method (based on IEC 60270-2000 standard) with HFCT (high-frequency current transform) and detecting impedance and non-electric method with acoustic, optic, and UHF (ultra high frequency) methods.

2.2.2.2. Partial discharge signal processing

The methods used under this mechanism include the envelope method, Fast Fourier Transform (FFT)/Inverse Fast Fourier Transform (IFFT), wavelet transform, and filter (band pass filter, low pass filter, high pass filter).

A signal processing tool has been used by Josef and Martin (2010) to diagnose PD using a developed on-line application. A field-programmable gate array (FPGA) mechanism was employed to detect partial discharge in the insulator, as shown in Figure 2.4. This method extracts the PD patterns only concerning the frequency and not with respect to time. Hence, an effort is made to characterize the PD patterns/types directly in the time domain for quick removal of insulation defects (Priyanka, 2019).

Figure 2.4: Digital partial discharge detection meter

Vitor et al. (2020) applied wavelet transform in characterizing the evolution of partial discharge to identify the moment of failure and the frequency spectrum. The energy of all approximation wavelet levels is checked to see if it increases with the level of PD. To evaluate the effectiveness of the proposed wavelet-based technique in characterizing the PD evolution, an electrode with a gap of 5 mm was placed in the transformer to produce the failure activity. A piezoelectric transducer (sensor) was attached to the transformer wall to capture the acoustic emission waves produced by the failure. Wavelet transformation is applied to the result to detect partial discharge on the transformer.

2.2.2.3. Pattern recognition and judgment system

This mechanism analyzes partial discharge patterns and parameters (pulse quantity, phase angle, and electric charge) using the statistical method and artificial neural network. The output of these methods usually determines if there is a damage level and the type of partial discharge that occurs. In the case of the statistical method, statistical parameters, such as mean, variance, standard deviation, skewness, and kurtosis, have been used to identify the type of partial discharge. Kothoke et al. (2020) used a statistical method to analyze Phase-resolve partial discharge patterns to detect the type of partial discharge that occurred, such as void, surface, or corona. The data extracted for the processing involves the stage edge 'φ,' the charge extent 'q,' and the number of pulses 'n' and voltage 'v.' The phase-determined trends are obtained from the analysis, as shown in Figure 2.5

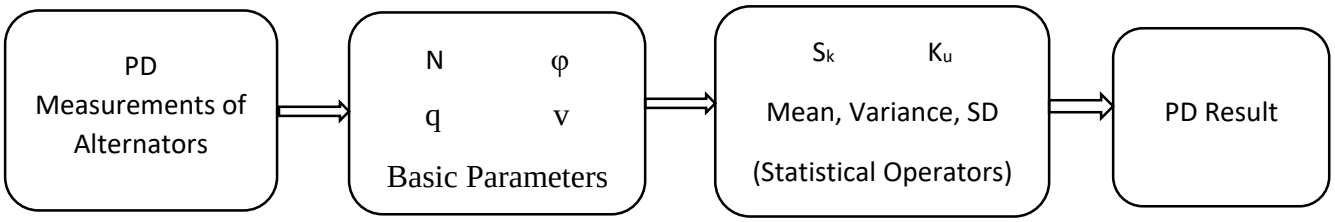


Figure 2.5: Flow diagram of partial discharge analysis for (n-q) where

SD = Standard Deviation S_k = Skewness K_u= Kurtosis

2.3 IMPROVISED METHODS USED FOR DETECTION OF TYPES OF PD

In recent years, two methods have been introduced for higher accuracy in classifying PD patterns and its speedy detection. These methods are the Support vector machine and neural network methods.

Candela et al. (2000) combined Weibull distribution and neural network to detect the type of PD. The approach was used for both time and phase-resolved classification. The former is considered the only variation of q with respect to time, which is very high for accurately noticing a chosen time interval. However, the phase-resolved partial discharge patterns (φ-q, φ-n, and n-q) consider not only q but also φ and n variations and hence obtain more accurate statistical parameters to be given as an input to ANN to detect the type of PD.

According to Kothoke et al. (2019), Neural Networks have an advantage in that once they are trained for a sample of data, they can predict the desired results for the next data. Artificial neural networks can also allow the modeling of nonlinear processes, thereby creating more ways to solve many classification problems, clustering, regression, pattern recognition, dimension reduction, structured prediction, machine translation, anomaly detection, decision-making, etc. Therefore, the researchers proposed the Back Propagation method (BPM) of ANN, which has the advantage of repeating the training process until the error in iterations between n versus q, φ versus q, and φ versus n becomes zero (Kothoke et al., 2017). It was tried on both Matlab and Python software, and a better accuracy of 89% was realized in both software.

Self-Organizing Map (SOM) of Artificial neural network (ANN) has been used on different voltage equipment, such as transformers and cables to examine and display a plotted graph of phase-resolved patterns of PD separately for each discharge (Chang & Yang, 2008). SOM is useful in a neural network for visualization and data analysis. It uses a two-dimensional space graph to create a feature map. In a nutshell, SOM is a competitive learning method that is also

a form of unsupervised learning, where constituent elements compete to produce a satisfying result, and only one gets to win the competition. However, the quality of the feature map has always been a problem of SOM because it tries to create a map of the input data in the multi-dimensional space to the less dimensional space that is usually two-dimensional (Anh Tu, 2020). The heat map in Google Colaboratory of Python has been used as an alternative by Kothoke et al. (2019) to display the visualization directly for all known and unknown data of discharges merged. This approach is adopted in this research as well.

Random Forest (RF) Method, a supervised machine learning technique, has been known for classification tasks more effectively (Kothoke et al., 2019). This technique considers multiple decision trees before giving an output. So, it is an ensemble of decision trees. This technique is based on the belief that more trees would converge to the right decision. For classification, it uses a voting system and then decides the class. According to Kothoke et al. (2019), RF works well with large data sets with high dimensionality. It envisages more decision trees that can be found for the particular type of data available from PD sensors to improve the resulting accuracy for PD-type identification further.

Support Vector Machine (SVM) using Python learning has also been used for partial discharge classification, where data input is very important. Using correlation techniques, Hao and Lewin (2010) applied wavelet analysis to pre-process measurement data from PD sensors and cluster the discharges into different groups. Then they applied SVM techniques to identify the type of different PD discharges. However, they concluded that the technique over-identified the void discharge being mixed with internal cavity discharge. Hence, the exact segregation was shifted to their future scope to get the output with full clarity. According to the review by Kothoke et al. (2019), the accuracy of the BPM method of ANN is 89%. From the RF method, it is 95%, and from the SVM method, it is 100 %. Hence, SVM can be considered the best method for classification.

2.4 PARTIAL DISCHARGE CLASSIFICATION USING DEEP LEARNING METHODS

The previous section shows the effort made using neural networks and support vector machines to classify partial discharge on voltage equipment such as transformers and cables. The survey report by Barrios et al. (2019) shows that the most efficient among the methods were those with Machine Learning (ML), and that itself was only a semiautomated classification because the input data has to be previously given by the user, who must have knowledge about which features are essential for the algorithm, and as such includes a lot of bias. In other words, much effort and expertise are required to get a good result. A review of different techniques for feature extraction and PD classification methods has been shown by Raymond et al. (2015). Mas'ud et al. (2016) further investigated the application of conventional Artificial Neural Networks (ANN) for PD classification through a literature survey. Currently, with the advent of computational technology and precisely on data storage, the focus has been on automated features extraction and classification by Deep Learning (DL) algorithms with Deep Neural Networks (DNN), where the expert is not so necessary (Barrios et al.,2019).

Artificial Intelligence is a broad field that comprises many subsets, one of which is Machine Learning. Deep Learning is a subfield of Machine Learning, as shown in Figure 2.6. ML uses algorithms to parse data, learn from that data, and make informed decisions based on what they have learned, but usually, they need some manual feature engineering, as illustrated at the topright of Figure 2.6. On the other hand, the DL model is based on ANNs (precisely Deep Neural Networks) built by many layers and nodes that can learn and make intelligent decisions or predictions, including automatic feature extraction. This model is described at the bottom-right of Figure 2.6.

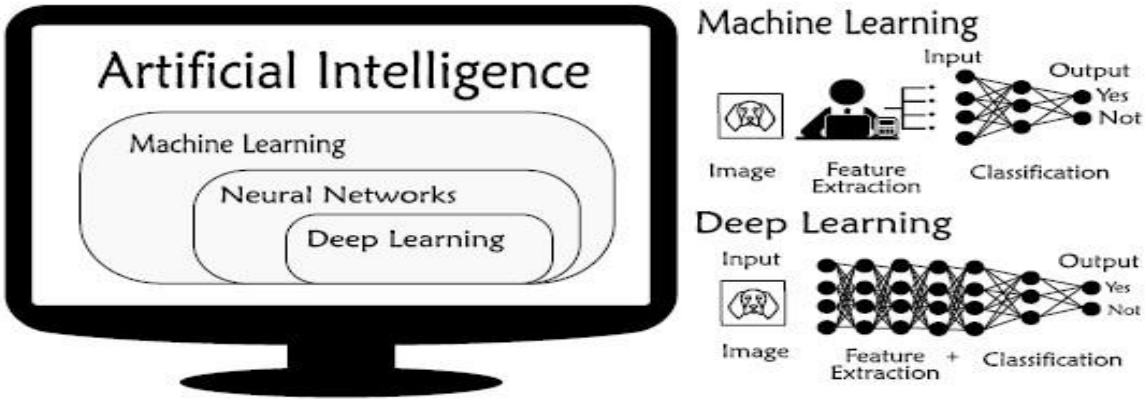


Figure 2.6: Link between Artificial Intelligence, Machine Learning, and Deep Learning (Barrios et al., 2019).

2.4.1 DEEP NEURAL NETWORK (DNN)

The use of "deep" in neural networks came from the research conducted by Hinton et al. (2006), where the procedure of training a DNN was described in detail. Any ANN with more than two hidden layers may be considered as deep. Some DNN models have been used for PD classification in recent years. Some of them are presented as follows.

2.4.1.1 RECURRENT NEURAL NETWORK (RNN)

This model predicts time series data based on analyzing the feedback connections. The architecture is shown in Figure 2.7. At the left of Figure 2.7, one hidden layer receives inputs, produces an output, and sends that output back to itself. If the RNN is expanded in a temporal frame, as shown on the right side of Figure 2.7, at each time step t, this recurrent layer receives the inputs x(t) and its outputs from the previous time step, y(t - 1).

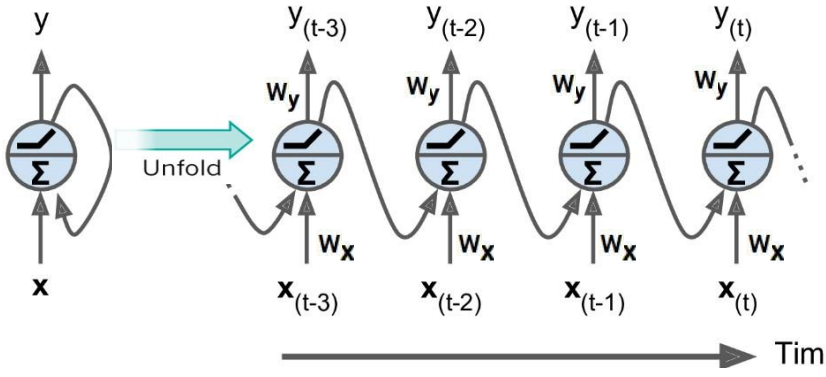


Figure 2.7 represents a simple Recurrent Neural Network (RNN) Architecture.

Each recurrent neuron has two sets of weights: one for the inputs $x(t)$ and the other for the outputs of the previous time step $y(t-1)$. Assuming these weight vectors are W_x and W_y. If we consider the whole recurrent layer instead of just one recurrent neuron, then the weight vectors can be placed in two weight matrices, W_x and W_y . The output vector of the whole recurrent layer can be computed as shown in Equation 2.1, where b is the bias vector, and $\varphi(\cdot)$ is the activation function (e.g., ReLU). Equation 2.1 is the output of a recurrent layer for a single instance.

$$y_{(t)} = \varphi(W_x^T * x_{(t)} + W_y^T * y_{(t-1)} + b) \quad (2.1)$$

Since the output of a recurrent layer's feed-forward neural networks can be computed in one shot for a whole mini-batch (Vaswani et al., 2017) by placing all the inputs at time step t in an input matrix $x_{(t)}$. Therefore, the outputs of a layer of recurrent neurons for all instances in a mini-batch can be represented mathematically as follow:

$$y_{(t)} = \varphi(x_{(t)} * W_x + y_{(t-1)} * W_y + b)$$

$$= \varphi([x_{(t)} \quad y_{(t-1)}] * W + b) \quad (2.2)$$

with $W = \begin{bmatrix} W_x \\ W_y \end{bmatrix}$

$y_{(t)}$ is an $m \times n_{neurons}$ matrix containing the layer's outputs at time step t for each instance in the mini-batch (m is the number of instances in the mini-batch and $n_{neurons}$ is the number of neurons).

$x_{(t)}$ is an $m \times n_{inputs}$ matrix containing the inputs for all instances (n_{inputs} is the number of input features).

W_x is an $n_{inputs} \times n_{neurons}$ matrix containing the connection weights for the inputs of the current time step.

W_y is an $n_{neurons} \times n_{neurons}$ matrix containing the connection weights for the outputs of the previous time step.

b is a vector of size $n_{neurons}$ containing each neuron's bias term.

The weight matrices W_x and W_y are often concatenated vertically into a single-weight matrix

W of shape $(n_{inputs} + n_{neurons}) \times n_{neurons}$

The notation $[x_{(t)} \quad y_{(t-1)}]$ represents the horizontal concatenation of the matrices $x_{(t)}$ and $y_{(t-1)}$.

From Figure 2.7, it shows that $y_{(t)}$ is a function of $x_{(t)}$ and $y_{(t-1)}$, which is a function of $x_{(t-1)}$and $y_{(t-2)}$ which is a function of $x_{(t-2)}$ and $y_{(t-3)}$, and so on. This makes $y_{(t)}$ a function of all the inputs since time t = 0 (that is $x_{(0)}$, $x_{(1)}$, …, $x_{(t)}$). At the first time step, t = 0, there are no previous outputs, so they are typically assumed to be all zeros.

The main drawback of RNN is that recurrent neurons have a short-term memory of the previous state. Moreover, the DNN structures may suffer from the vanishing gradients problem. When updates from Gradient Descent leave the layer connection, the weights are virtually unchanged for initial inputs when time series are very long. Training does not converge to a suitable solution (Barrios et al., 2019). One way to solve this problem involves a neural network layer that will preserve some state across time steps, called a memory cell. A typical memory cell example is the Long Short-Term Memory model (LSTM) (Hochreiter and Schmidhuber, 1997).

2.4.1.2 AUTOENCODERS (AES)

These kinds of models are a specific type of feed-forward neural networks that are unsupervised (or self-supervised) which generate an efficient representation of the input data (feature extraction). Autoencoders (AEs) aims to output a replication of their inputs. Therefore, the outputs are often called reconstruction, and its cost function contains a recognition loss that penalizes the model when the reconstructions differ from the inputs (Arden, 2022; Barrios et al., 2019). In Figure 2.8, the number of neurons in the output layer must equal the number of inputs. The encoding compresses the input into a lower-dimensional code called latent-space representation, and the decoding part converts this internal representation to the outputs.

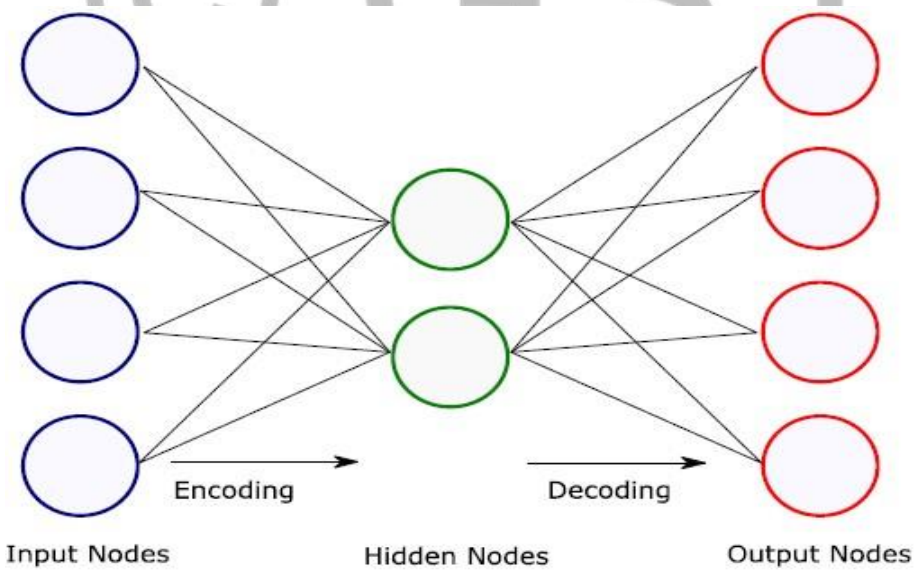


Figure 2.8: Simple autoencoder architecture with one hidden layer

An autoencoder consists of 3 components: encoder, code, and decoder. The encoder compresses the input and produces the code. The decoder then reconstructs the input only using this code

(Arden, 2022). AEs are mainly a dimensionality reduction (or compression) algorithm with a couple of important properties:

It is data-specific: Autoencoders can only compress data meaningfully similar to what they have been trained on. Since they learn specific features for the given training data, they differ from a standard data compression algorithm like gzip (Arden, 2022). So it is not possible to have an autoencoder trained for handwritten digits to compress landscape photos.

It has a lossy effect: The quality of the output of the autoencoder is not exact with the input data, it will be close, but some quality will be lost during the process. To have a lossless compression, an autoencoder should not be used at all (Arden, 2022).

It is unsupervised: Training an autoencoder system just the input of raw data at it. It is unsupervised because it does not need any explicit labels to train on. In other words, they are self-supervised as they generate their labels from the training data (Arden, 2022).

Vincent et al. (2008) have shown that AEs can be applied only when the hidden layer contains few nodes than the inputs. It can function well as a feature extractor and a data dimensionality reducer. The number of nodes in the hidden layer determines the extracted features' quality. However, this could be a problem when training the AE if the number of hidden neurons is larger than the optimum number of features (Barrios et al., 2019). A typical solution to this problem is a Sparse AE, which can manage more nodes in the hidden layer than inputs by forcing the generation of a sparse encoding during the training phase (Ng, 2022). Sparse feature learning algorithms range from sparse coding approaches (Olshausen & Field, 1997) to training neural networks with sparsity penalties (Nair & Hinton, 2009; Lee et al., 2007). According to Barrios et al. (2019), when AEs have multiple hidden layers, they are called Stacked AE or deep autoencoders. The autoencoder also behaves as a generative model; it can generate new data from the input data that are very similar to the original set.

2.4.1.3. GENERATIVE ADVERSARIAL NETWORKS (GANS)

Goodfellow et al. (2014) was the researcher who proposed Generative Adversarial Networks (GAN) to synthesize data through deep generative modeling. The framework, as shown in Figure 2.9, has two stages, each having neural networks. These stages are the generator network and the discriminator network. The generator tries to emulate random data as input called latent space (representation) in a probability distribution form. The discriminator then estimates the sampled probability, whether from the real data or the generator. The training ends when the discriminator can no longer differentiate between the real and the fake data, and the generator network can be used to generate new simulated data, with the hope that it has been trained to identify correctly. GAN is mostly used in computer and communication networks, including mobile networks, network analysis, the internet of things, physical layer, and cyber-security (Navidan et al., 2021).

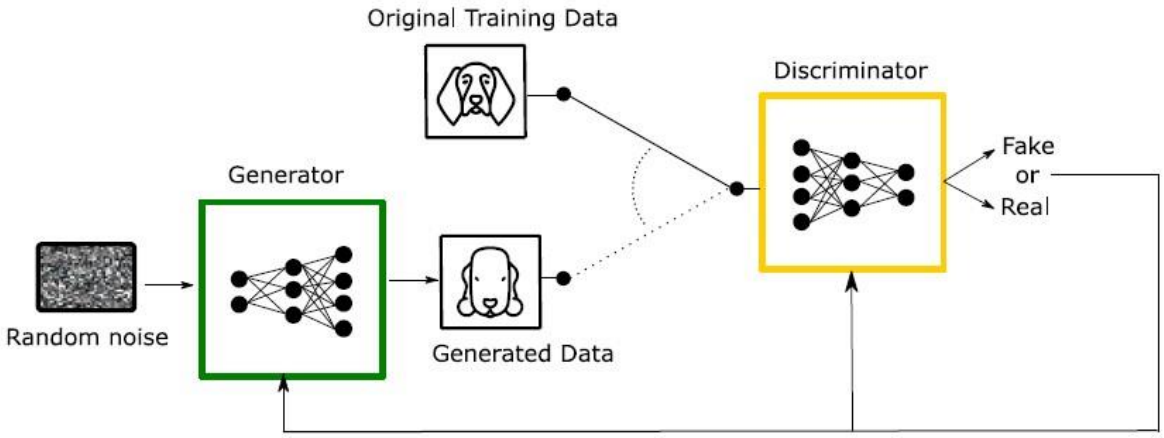


Figure 2.9: Generative Adversarial Network Architecture (Barrios et al., 2019)

GAN has a generative design with a broader history and scope beyond the ML space (Krish, 2011). Also, an optimally trained GAN recreates the training distribution, allowing people to explore. However, as an ML system, it has a “black-box” quality which is the fundamental

difference between ML and heuristics-driven systems. The outputs of these systems are often not easily explained, mainly when errors occur (Hughes et al., 2021).

2.4.1.4. CONVOLUTIONAL NEURAL NETWORK (CNN)

The concept behind Convolutional Neural Networks (CNNs) was based on the brain′s visual cortex but are not restricted to visual perception; they can also be used with signal processing and recognition. CNN typically has convolutional layers, pooling (or sub-sampling) layers, and then is usually followed by fully connected (FC) layers (Khan et al., 2019). The principal task of CNN is to take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. It is for image classification to obtain a class or a probability of classes that best describes an input image(Barrios et al., 2019). The CNN algorithm was designed to recognize features (edges, curves, ridges) and their compositions by itself (LeCun et al., 2010). A basic CNN architecture is shown in Figure 2.10, whose basic layers are:

(i). Convolutional layer: each filter (also called Kernel) is applied to the image in successive positions along the image and generates a features map through convolution operations.

(ii). Nonlinear layer: a non-linear activation function, such as ReLu (Rectified Linear Unit) function, is used to avoid linearity in the system.

(iii). Pooling (down-sampling) layer: This aims to reduce the computational load by reducing the size of the feature maps and introducing positional invariance.

(iv). Fully connected layers: This is an ANN that takes the convolutional features (previously flattened) generated by the last convolutional layer and makes a prediction (e.g., softmax function). The loss function establishes the output error to inform how accurate the network is and uses an optimizer to increase its effectiveness.

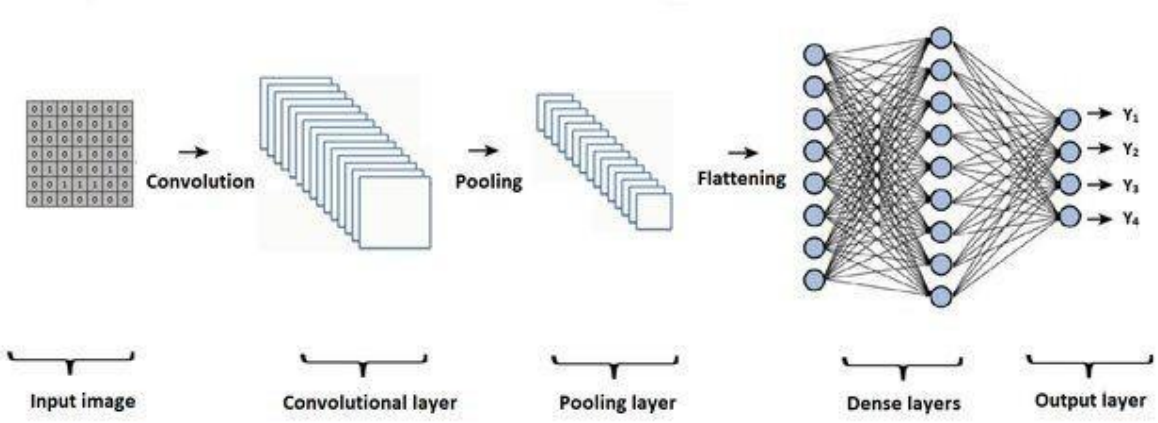


Figure 2.10: Architectural design of Convolutional Neural Network

Many authors have recently reported using convolutional neural networks (CNN) to classify PD sources (Tuyet-Doan et al., 2020; Puspitasari et al., 2019; Barrios et al., 2019). Different formats have also been employed for the input of PD source identification; some formats include waveform spectrogram, time-domain waveform signal, and phase-resolved partial discharge (PRPD) patterns. In the case of waveform spectrogram data, Lu et al. (2016) used CNN to detect PD signals with varying noise and interference signals. The input to the network was an image showing the time-frequency spectrum of sound clips, which were measurements recorded from a switch gear using the transient earth voltage method (TEV). CNN showed superior detection accuracy and time performance compared to other methods prevalent in the industry. Che et al. (2019) used 2D- CNN to classify three PDs sources in Cross-linked polyethylene (XLPE) cable: internal PDs, corona PDs, surface PDs, and noise.

Wang et al. (2019) use 64 by 64 images that were down-sampled originally from a 600 by 438 time-resolved partial discharge (TRPD) image as input in CNN for partial discharge detection in GIS. Song et al. (2018) passed PRPD patterns for six different sources of partial discharge represented as a 72 by 50 matrix into CNN for classification of partial discharge. The accuracy achieved was 89.7%. The application of convolutional neural networks to a sequence of partial

discharge images was presented by Florkowski (2020). The aim is to recognize the stages of aging of high-voltage electrical insulation based on PD images. Partial discharge images are phase-resolved patterns revealing various discharge stages and forms. The four distinguishable classes of the electrical insulation degradation process are a start, middle, end, and noise/disturbance. The process is for monitoring electrical insulation deterioration.

According to Ganguly et al. (2020). All prior methods reported above depend on the availability of training data from multi-source or single-source PD inputs. There are several drawbacks associated with this choice. Such training data is difficult to collect in practice and time-consuming. However, the accuracy is still noteworthy since it involves the usage of CNN. Convolution Neural Network (CNN) architecture has a lot of advantages. For instance, deep neural networks accept a sample as an image (i.e., a matrix of size m × n) and perform feature extraction and classification via hidden layers (such as convolutional layers, RELU layer, max-pooling layers) (Sharma et al., 2019). CNN does not require additional feature extraction techniques as it automatically derives features from the raw elements. The second advantage is that it finds higher-order statistics of image and nonlinear correlations. Third, convolution neurons process data for their receptive fields or restricted subarea, relaxing the need to have a very high number of neurons for large input sizes and enabling the network to be much more profound with fewer parameters (Habibi et al., 2017). Therefore, another distinguishing attribute of CNN is weight sharing; i.e., many receptive fields share the same weights and biases (or filters), reducing the memory footprint compared to conventional neural networks.

2.5 MACHINE LEARNING AND PARTIAL DISCHARGE IMAGE RECOGNITION

Machine learning (ML), a subdomain of artificial intelligence, refers to the ability of algorithms with tunable parameters that are adjusted automatically and adapted accordingly to previously seen data (Florkowski, 2020). ML is the brain behind computer learning. It learns based on

experience, leading systems to behave more intelligently by generalizing rather than only operating on data elements, in contrast to a conventional dataset system. ML can be classified into unsupervised learning and supervised learning. The unsupervised learning approach aims to group and interpret data based only on input data, whereas the supervised learning method relies on predictive models based on input and output data (IBM Cloud Education, 2020).

According to Florkowski (2020), unsupervised learning implies that the algorithm will find patterns and relationships among different data clusters. The dataset in machine learning usually consists of a multi-dimensional entry associated with several attributes or features. Unsupervised learning can be further divided into clustering and association. Clustering refers to the automatic grouping of similar objects into sets. At the same time, the association is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable (IBM Cloud Education, 2020). Supervised learning implies an algorithm's ability to recognize elements based on provided samples to recognize new data based on training data. Its algorithms include decision trees, support vector machines (SVM), naive Bayes classifiers, k-nearest neighbors, and linear regressions (Florkowski, 1995; Dai, 2019; Tuyet-Doan, 2020).

Supervised learning can be further divided into classification and regression: classification means that samples belong to two or more classes to predict the class of unlabeled data from the already-labeled data and thus identifying to which category an object belongs; regression is understood as predicting an attribute associated with an object (IBM Cloud Education, 2020).

Florkowski (2020) states that machine-learning workflow consists of several steps. The first step is preprocessing, i.e., data preparation in a form on which the network can train. This involves collecting images and properly resizing and labeling them up to the normalization stage (for example). The second step refers to the definition of the neural network in terms of the number of layers to be used in a model, the size of the input and output layers, the type of

the implemented activation functions, whether or not dropout will be used, the number of epochs, the sizes of the training sets and many other factors. Setting up all of the hyperparameters of a neural network framework is an art and relies greatly on experience.

According to Candela et al. (2000), PD recognition is complicated. Only experts with extensive personal experience can discriminate between various discharge phenomena and assess the risk of potential insulation breakdown. Initially, experts mainly relied on the apparent charge magnitude; later, the pulse density, phase, and amplitude distributions were considered after the introduction of acquisition systems. In the early 1990s, the introduction of PD phase-resolved acquisition and 3D representation, as well as ultra-wideband detection and pulse registration, created new possibilities and tools for automatic pattern recognition and expert systems. A phase-resolved PD pattern is treated as an image; therefore, image-processing algorithms are used to extract and distinguish an image's features (Florkowski, 2020).

Partial discharge images have also introduced a new category in PD evaluation, referring to qualitative analysis and defect discrimination. A kind of system that requires no calibration in absolute units (pC, mV, mA, etc.) and in which qualitative discrimination could be performed by analyzing the shapes of statistically accumulated images which would be of most importance, especially on-site or monitoring measurements. This direction has been an area of interest in PD expert systems over the last few decades (Florkowski, 2020).

2.6 RECAP

Recent research has been done in detecting partial discharge with trending technology using the deep learning approach with the best accuracy. This research adopts a Convolutional Neural Network (CNN) based on our review of partial discharge detection. To achieve the best performance, the data used is converted to 2d-image using the deep insight method, which is explained in the next chapter.

CHAPTER THREE

METHODOLOGY

3.1 INTRODUCTION

In this chapter, a binary classification of partial discharge (PD) was carried out as to whether PD would occur at an installation of a new meter in a location within the Akoko metropolis in Ondo State, Nigeria. A non-image dataset was used in this research with about 965 records. A DeepInsight model was adopted to transform the non-image dataset to an image-based one. The classification was processed using this same model's parallel Convolutional Neural Network. Some of the metrics used in evaluating the performance of the model were adopted from Durand et al. (2019), which include: accuracy score, confusion matrix, and F1-Score

3.2 GATHERING OF DATASET

The distribution line network in Benin Electricity Distribution of Akoko Area has a total of 256 transformers comprising 200 transformers on 33/0.415kv distribution line and 56 transformers on 11/0.415kv distribution line. All records of partial discharge occurrence and all associated with it for the eleven (11) years have always been recorded in a central book located in a control room manually. A Megger meter is mainly used, especially when the down time has exceeded 24 hours to take some readings. This tool can be used to determine the condition of the insulation on the wire, generators, and motor windings. 965 records of partial and non-partial discharge occurrence in the recent 11 years were gathered and fine-tuned (pre-processed) using the agile method. This method was used to convert the manual records to a soft form used as a dataset for this research.

The agile method for dataset conversion involves three (3) processes: Data mapping, Data Integration, and Data Transformation, as shown in Figure 3.1. The data is recorded from the central station anytime there is patrol to troubleshoot distribution lines, usually including year, month, pole-number, phase, location, the total number of the pole in that area, the last signal

values on each phase (indicated as Red, Yellow, and Blue) and class the incidence belongs (partial or non-partial discharge). From the list of attributes, the mapped-out data used in this research were all except the year. The data were captured from the daily field tracing faults on distribution lines from 2011 to 2021 in the Akoko metropolis. The attributes are described in Table 3.1

Table 3.1: Description of dataset attribute

Attribute Name	Description
Month	The month that field tracing was done
Pole-Number	The label number inscribed on the pole that the fault was detected on
Phase	The phase (number) where the partial discharge occurred
Location	The area or territory where tracing/troubleshooting was done
Total Pole Number	This is the total number of poles in the location where troubleshooting was done.
PhasePD_Red	This is the phase1 on the three lines that comes from the distribution station
PhasePD_Yellow	This is phase 2 on the three lines from the distribution station.
PhasePD_Blue	This is the phase3 on the three lines that comes from the distribution station
Signal	The signal value where the partial discharge occurred.

A DeepInsight model is adopted to transform these captured data into an image-based format for Convolutional Neural Network (CNN) classification.

3.3 DEEPINSIGHT MODEL FOR NON-IMAGE TRANSFORMATION

Artificial Intelligent Model is a field that enables machines to use cognitive functions like humans to perceive, learn, reason, and solve problems. The model can efficiently classify objects and guide well (as in the case of driverless cars) using CNN due to its high performance. CNN is used in AI for high performance when the input is image-based and gives a highly accurate result during processing. Similarly, DeepInsight Model is used for high performance when dealing with nonimage input samples. The procedure used is so that samples (impossible to envisage as an image) can be transformed into images and processed by CNNs, as shown in **figure 3.1.**

The main function of DeepInsight is to transform a feature vector x to an image matrix M such that CNN can now be used to perform the target classification. A profound transform is then used on the features $g1$ to gd. Assuming from Figure 3.1 that the first feature $g1$ is closely related to g3 and g1 is also related to $g6$, and $g6$ is related to gd, so basically, the features which are closely related are mapped together (as the case of $g3$, $g1$, $g6$, and gd) while the one which is not closely related are located far apart (as the case of $g7$). The location of features in the Cartesian coordinates depends on the similarity of features. Once the locations of each feature are determined in a feature matrix, the expression values or feature values are mapped. This will generate a unique image for each sample (or feature vector).

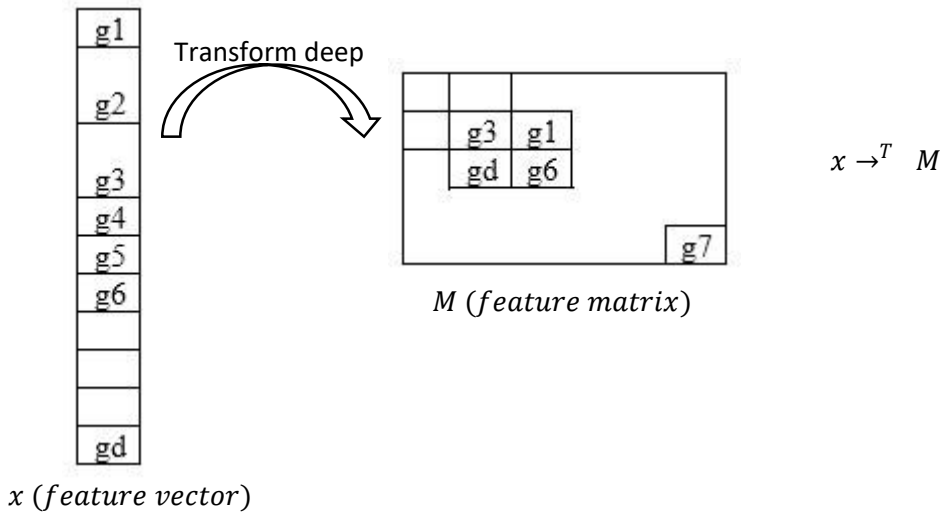


Figure 3.1: An Illustration of DeepInsight transformation

During the transformation T of feature x to M from Figure 3.1, the following activities were performed.

1. Defining a d-dimensional feature vector to a 2-dimensional space
2. Find a rectangle that encompasses all the data (for data framing)
3. Adjust the rectangle and data horizontally/vertically.
4. Find the number of rows and columns (A, B)
5. Transform from Cartesian coordinates to pixel frames

3.3.1 DIMENSIONALITY REDUCTION TO 2D-SPACE (GENE-WISE, NOT SAMPLE-WISE)

Figure 3.2 shows how the input data (dataset) is converted to a 2D-space using a gene-wise approach. g represents the value in each cell of the tabula data. d represents the dimension or number of features in the dataset, x is the sample, and n represents the number of samples in that dataset.

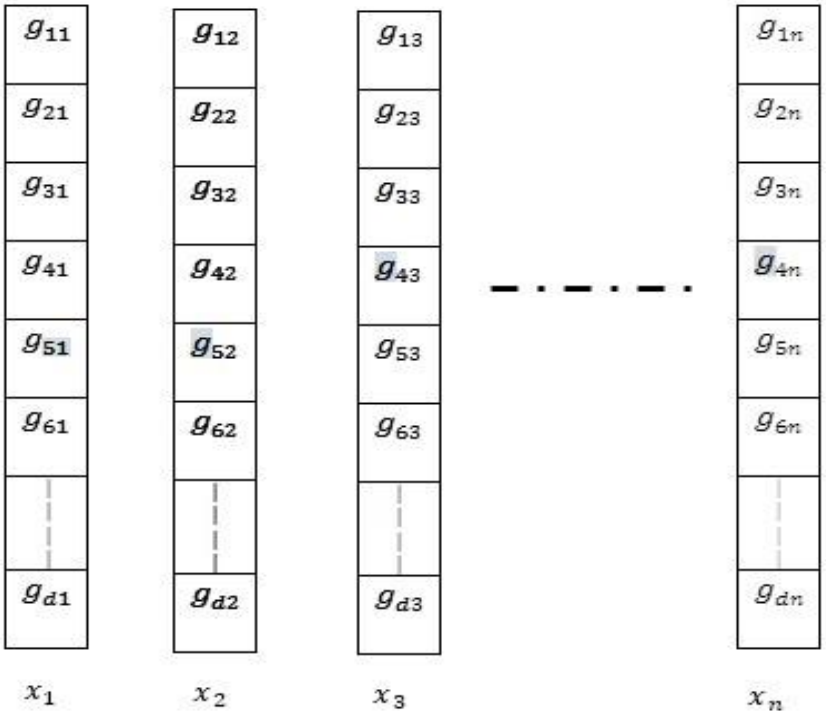
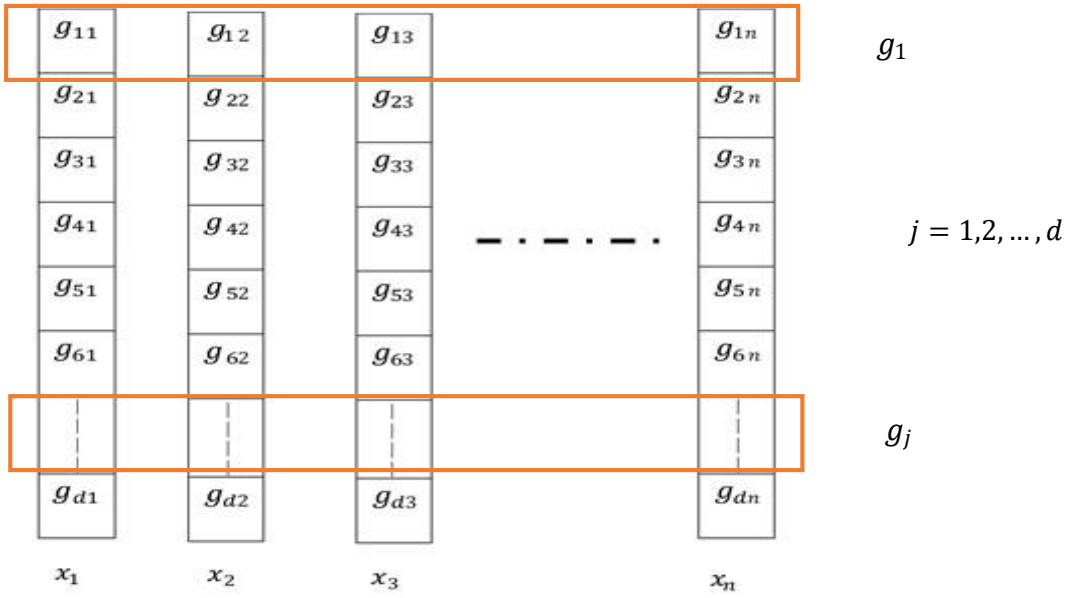


Figure 3.2: A 2D-space dimensional reduction process for dataset



There are some dimension reduction techniques built-in functions in DeepInsight model which include the kernel Principal Component Analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) (Maaten and Hinton, 2008). The latter is taken as a default in the DeepInsight model. From the d dimension feature vector, by applying t-SNE, a 2D matrix form having all the d features mapped into g_j as shown in Figure 3.3 and Figure 3.4 where $1 \geq j \leq d$.

Figure 3.3: 2D matrix formed after applying t-SNE or Kernel PCA

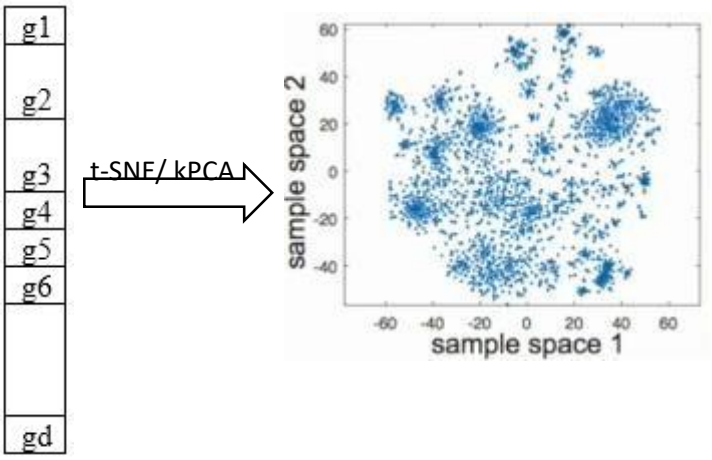


Figure 3.4: 2D image space formed after applying t-SNE or Kernel PCA

3.3.2 DATA FRAMING USING CONVEX HULL ALGORITHM

The DeepInsight model applies a Convex Hull algorithm on a 2D-space from Figure 3.4. The Convex Hull technique carved out the smallest convex polygon that contains all the points in the set of points in the 2D-space plane. The result, after the convex hull algorithm is applied, is shown in Figure 3.5.

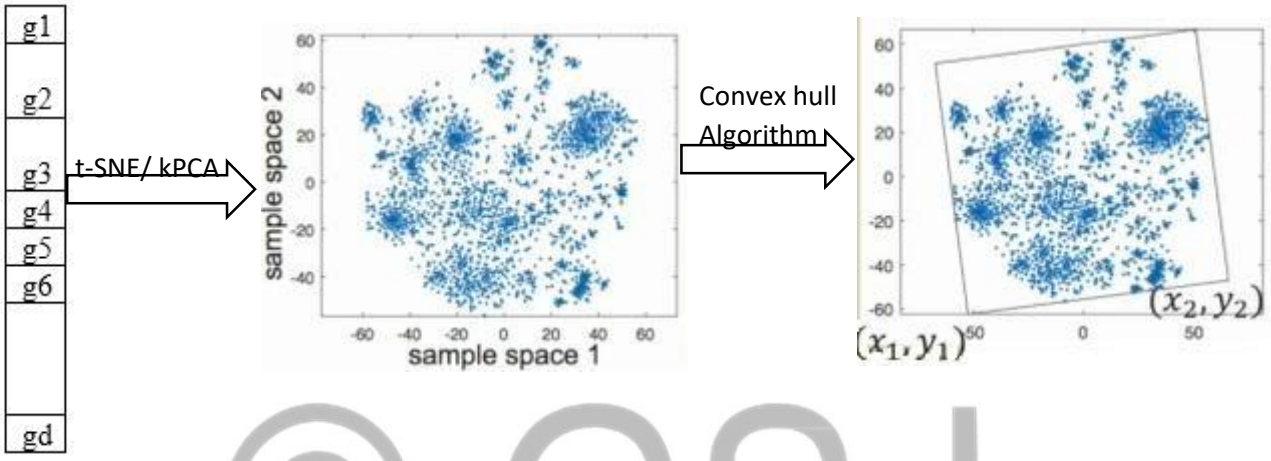


Figure 3.5: The smallest rectangle consisting of all the data

3.3.3 ADJUSTING RECTANGLE AND DATA HORIZONTALLY AND VERTICALLY

In Figure 3.5, the image formed is not in a stable condition. So there is a need for rotation of the image to align it both vertically and horizontally. To rotate the image, the gradient of the two corner points of the tilted rectangle was found, as shown in equation 3.1. The angle for the rotation was calculated using equation 3.2

$$G = \frac{y_2 - y_1}{x_2 - x_1} \quad (3.1)$$

$$\theta = \tan^{-1}(G) \quad (3.2)$$

The rotation matrix $$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \quad (3.3)$$

The output of the image after rotation is shown in figure 3.6

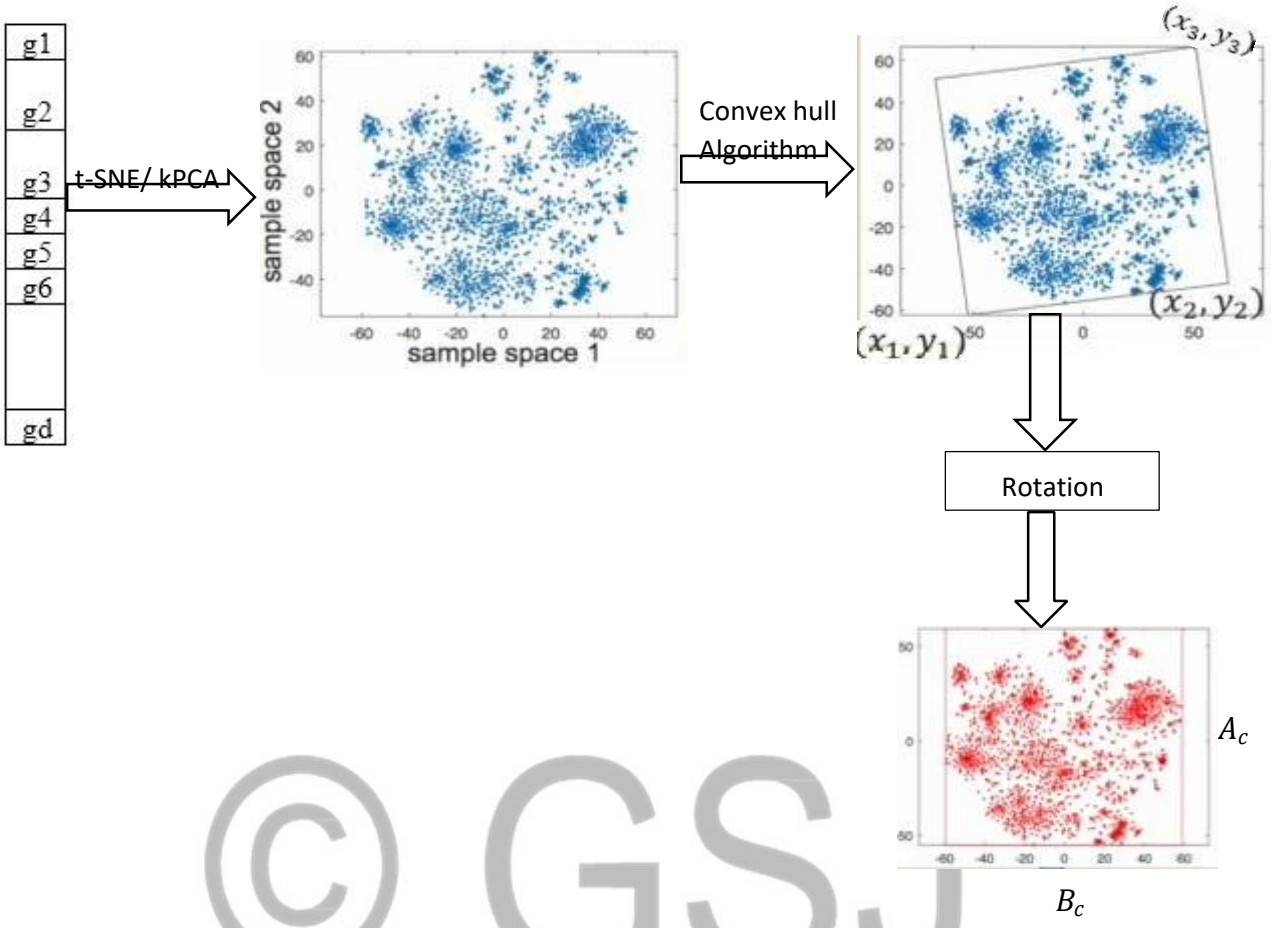


Figure 3.6: Plot showing the rotated data points and the smallest bounding rectangle

3.3.4 IMAGE FRAME HORIZONTAL AND VERTICAL LENGTH

The horizontal and vertical length of the image frame provided by the rotation matrix from Figure 3.6 is calculated using equations 3.4 and 3.5

Cartesian Horizontal axis $A_c = |x_2 - x_1|$ (3.4)

Cartesian Vertical axis $B_c = |y_3 - x_2|$ (3.5)

3.3.5 TRANSFORMATION OF IMAGE FRAME TO PIXEL FRAME

It is required to convert the Cartesian coordinates to pixel forms for fast processing. This was done by determining the minimum distance between the two closest pointsd_{min} of the mapped genes. The pixel coordinate is represented in equation 3.6 and 3.7 as

$$A_p = ceil(A_c * \frac{Precision}{d_{min}}) \quad (3.6)$$

$$B_p = ceil(B_c * \frac{Precision}{d_{min}}) \quad (3.7)$$

Where () is the product's ceiling value, precision will define the resolution. The A_p and B_p values will help to convert Cartesian coordinates to pixel coordinates using equations 3.8 and 3.9 as

$$x_p = round(1 + \frac{(x_c - x_{min}) * A_p}{x_{max} - x_{min}}) \quad (3.8)$$

$$y_p = round(1 - B_p \frac{(y_c - y_{min})}{y_{max} - y_{min}}) \quad (3.9)$$

Where (x_c, y_c)x-axis and y-axis are coordinates in the Cartesian plane and (x_p, y_p) are coordinates in the pixel frame.

3.3.6 FEATURE MAPPING ON PIXEL LOCATION

The mapping of genes/features on pixel locations could be overlapped, as shown in Figure 3.7. The feature locations are defined using the training set and map feature values to these locations. If two or more features occupy the same location, then their averaged values are used, leading to lossy compression of features; otherwise, the feature will maintain its location if no overlap occurs.

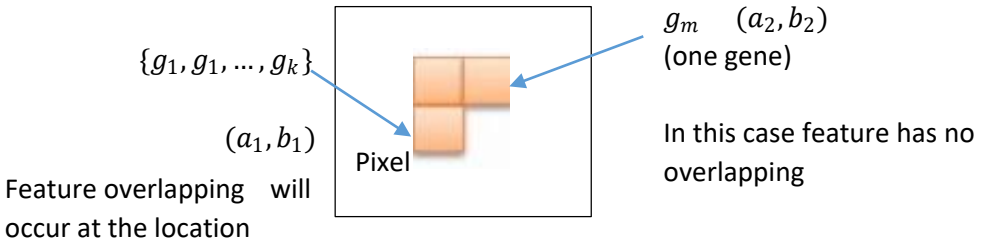


Figure 3.7: Features mapping using pixel coordinates.

The mapped features from pixel coordinates, as shown in **Figure 3.8,** are then normalized using norm-1 and norm-2 techniques.

3.3.6 .1 NORMALIZATION OF MAPPED FEATURES

Two types of normalization are embedded in DeepInsight: norm-1 and norm-2. For norm-1, each feature is normalized by its minimum and maximum. This will bring a feature between 0 and 1. This normalization will assume that features are mutually independent as a feature is normalized by its extrema values. The minimum and maximum values for norm-1 can be computed as shown in equation 3.10 to 3.14 as follow:

$$Max_j = max_{samples}^{X_{tr}(j,:)} \quad (3.10)$$

$$Min_j = min_{samples}^{X_{tr}(j,:)} \quad (3.11)$$

Where X_{tr} is the training set and $(j, :)$ refers to all the samples of the jth feature or attribute. Therefore, Max_j and Min_j are the maximum and minimum of the jth attribute. These extrema values are used to normalize training, validation, and test sets as

$$X_{tr}(j,:) = \frac{X_{tr}(j,:) - Min_j}{Max_j - Min_j} \quad (3.12)$$

$$X_{val}(j,:) = \frac{X_{val}(j,:) - Min_j}{Max_j - Min_j} \quad (3.13)$$

$$X_{test}(j,:) = \frac{X_{test}(j,:) - Min_j}{Max_j - Min_j} \quad (3.14)$$

Where $j = 1, 2, \ldots, d$, and d is the dimension of the samples in the dataset, if after normalization any feature value of the validation set or test set is less than 0 or greater than 1, then such feature values are clamped between 0 and 1 to maintain the consistency.

The norm-2 normalization method adjusts the minimum value for each feature or attribute. Then a global maximum is used in the logarithmic scale to place the feature values between 0 and 1. The norm-2 uses equation 3.10 to calculate the minimum of the jth attribute, and further process is done using equation 3.15 to get the logarithmic scaling as follow:

$$Min_j = min_{samples}^{X_{tr}(j,:)} \quad \text{(from equation 3.10)}$$

$$X_{tr}(j,:) = \log\left(X_{tr}(j,:) + |Min_j| + 1\right) \quad (3.15)$$

$$Max = max_{samples}^{X_{tr}(j,:)} \quad (3.16)$$

$$X_{tr}(j,:) = \frac{X_{tr}(j,:)}{Max} \quad (3.17)$$

The validation and test sets are adjusted using the training extrema values for normalization. In case, after adjusting by the minimum values (Min_j), any element of validation or test set is less than 0, then it is clamped at 0. Similarly, if after normalizing by the maximum value (Max), any feature from the validation and test sets is above 1, then it is clamped to 1.

DeepInsight method employs these two types of normalizations (norm-1 and norm-2), and the validation error is evaluated on both norms. The norm, which gives the lowest validation error, is used for further processing. The pixel frame size is fixed at 120 × 120.

The normalized output is passed into two parallel CNN architectures consisting of four convolutional layers and tuned using the Bayesian Optimization technique for classification.

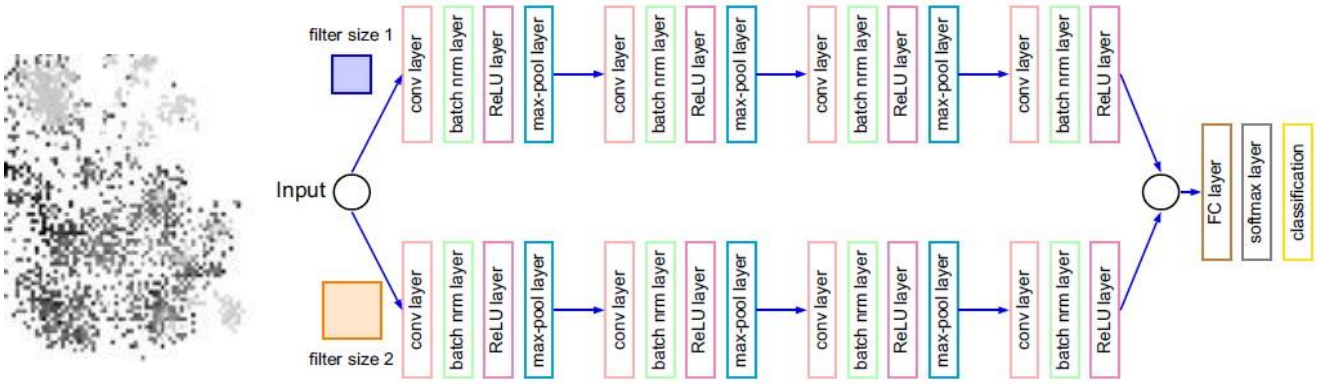


Figure 3.8: Parallel CNN for classification of mapped features from pixel coordinates

Basically, the adopted DeepInsight method used in this research can perform three main actions: element arrangement via mapping, feature extraction, and classification via CNN. During mapping, Dimensionality Reduction Techniques (DRTs), like t-SNE and Kernel PCA, mapped the dataset from high-dimensional to 2D space in a non-linear fashion. The samples with similarity were mapped close to each other, and the ones with dissimilarity were mapped apart. Many linear DRTs map data to a 2D plane, however, mapped samples are highly convoluted, and it becomes very challenging for clustering algorithms to find a reasonable level of groupings. However, t-SNE has the potential which maps very high-dimensional data to a 2D plane while keeping the data topology. The in-built t-SNE algorithm uses Euclidean distance to compute probabilities. However, in DeepInsight, cosine distance was used. The processing of t-SNE can be prolonged. Therefore, for faster processing, the Barneshut algorithm has been incorporated into DeepInsight, which is then used to approximate joint distributions instead of the exact distribution.

CHAPTER FOUR

RESULT AND DISCUSSION

4.1 INTRODUCTION

This research was implemented using Colab, a free notebook environment that runs entirely in the cloud. It lets one and team members edit documents anywhere online, just like how Google Docs works. Colab supports many popular machine learning libraries which can be easily loaded in a notebook. However, it has limited space and Time: Google Colab platform stores files in Google Drive with a free space of 15GB; any bigger datasets will require more space, making it difficult to execute. This, in turn, can hold most of the complex functions to execute. The result of the implementation of Colab on the effective prediction of partial discharge is discussed in the next section.

4.2 DATASET

The dataset has seven headings: month, poleno, phase, pdarea, total pole, sigalval, and discharge. This was explained in chapter 3, Table 3.1. Figure 4.1 shows a part view of the dataset. Since this research used a supervised learning approach in prediction, the dataset was partitioned into two. The first part contains the value of the attribute used in the training and testing process: month, poleno, phase, pdarea, total pole, and sigalval, as shown in Figure 4.1 with 965 records. The second part contains an attribute with the predicted value, pdischarge. Each part is divided into two again for training and testing using 80% and 20%, respectively.

	month	poleno	phase	pdarea	totalpole	signalval
0	1	4	1	1	30	109
1	1	10	2	2	12	219
2	1	5	1	3	15	220
3	1	7	3	4	35	108
4	1	10	3	5	50	83
...	...	...	...	...	...	...
960	12	12	1	1	27	81
961	12	5	3	6	15	114
962	12	12	2	5	10	102
963	12	5	3	2	11	218
964	12	17	1	3	10	103

965 rows × 6 columns

Figure 4.1: Cross-section of the dataset with the headings

4.3 APPLICATION OF T-SNE ON DATASET

t-SNE (t-distributed Stochastic Neighbour Embedding) is a dimensionality reduction algorithm, as discussed in chapter three. The real training of the dataset occurred at this stage. The learning rate for t-SNE is usually in the range [10.0, 1000.0]. If the learning rate is too high, the data may look like a 'ball' with any point approximately equidistant from its nearest neighbors. If the learning rate is too low, most points may look compressed in a dense cloud with few outliers. If the cost function gets stuck in a bad local minimum, increasing the learning rate may help. The number of parallel jobs (n_jobs) to run for neighbors search is -1, as indicated in Figure 4.2. This means all processors in the virtual machine are used for parallel computation. Further explanation of parameters used in the t-SNE is shown in Table 4.1

```
tsne = TSNE(
    n_components=2,
    random_state=3000, learning_rate= 1000, perplexity = 50, metric='cosine', verbose = 1,
    n_jobs=-1)  #1701  1515
```

Figure 4.2: Implemented code for t-SNE and its parameters

Table 4.1: Detail description of t-SNE parameters used in dimensionality reduction.

t-SNE Parameters	Description
n_components	Dimension of the embedded space. It is a 2D plane, so 2 was used during the implementation
random_state	Number of reproducible results passed across multiple function calls during training and dimensionality reduction. In this research, 1515 is used for better performance. It checks whether the results are stable across several different distinct random seeds. This also controls the rotation of the features.
metric	The metric to use when calculating the distance between instances in a feature array. It is used by t-SNE to compute probabilities of the distance between instances in a feature array. It can either be Euclidean distance or Cosine. In this research, Cosine is used for high performance.
verbose	A verbose parameter is usually available to either display logs or not. By default, it is set to False (no loggings). Verbose logs are usually printed to Standard Output, as shown in Figure 4.3.
learning rate	The learning rate is a critical parameter. It should be between 100 and 1000. However, for t-SNE, it is between 10 and 1000. The default value (200) is used in this research for better performance.
perplexity	The perplexity is related to the number of nearest neighbors used in other manifold learning algorithms. The range is [5, 50]. Larger datasets usually require a larger perplexity. The choice is not extremely critical since t-SNE is insensitive to this parameter.

The t-SNE computation computed one hundred and fifty-one (151) nearest neighbors and indexed seven hundred and seventy two (772) samples within 0.001 seconds. The computed neighbors for 772 samples were done within 0.135 seconds on a mean sigma value of 0.155. The sigma value describes how far a sample or data point is away from its mean. The Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data is minimized after 1000 iterations to 0.492, as shown in Figure 4.3. The computation of t-SNE lapsed for 5.72 seconds

```
[t-SNE] Computing 151 nearest neighbors...
[t-SNE] Indexed 772 samples in 0.001s...
[t-SNE] Computed neighbors for 772 samples in 0.135s...
[t-SNE] Computed conditional probabilities for sample 772 / 772
[t-SNE] Mean sigma: 0.155325
[t-SNE] KL divergence after 250 iterations with early exaggeration: 66.683929
[t-SNE] KL divergence after 1000 iterations: 0.492324
t-SNE done! Time elapsed: 5.721296548843384 seconds
```

Figure 4.3: Logs of t-SNE computation

The graph of a created probability distribution that represents similarities between neighbors is shown in Figure 4.4

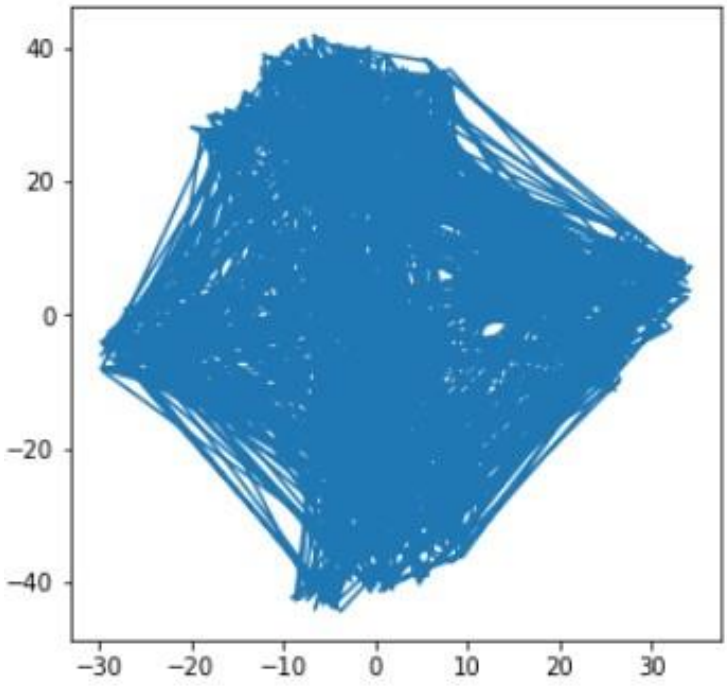


Figure 4.4: A 2D probability distribution of computed neighbors

4.4 DATA FRAMING

Data framing is a process to carve out the smallest convex polygon that contains all the points in the set of points in the 2D-space plane of the train image transformer on the training dataset. Figure 4.3 is the result of the plot showing the reduced features (blue points), convex Hull (red), and minimum bounding rectangle (green) prior to rotation. The edges of the polygon indicate the density area of the reduced features. Figure 4.5 shows the convex polygon at random_state of 1515.

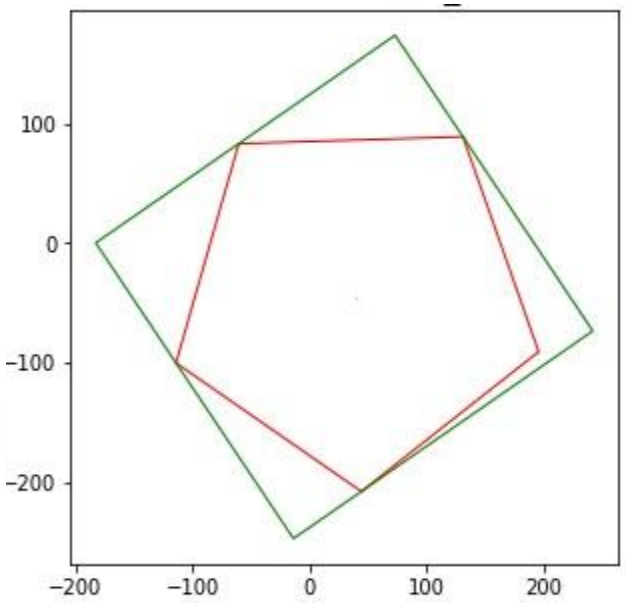


Figure 4.5: Convex Hull transformation of reduced features

4.5 FEATURE DENSITY

The feature density matrix extracted from the trained transformer to view the overall feature overlap is shown in Figure 4.6.

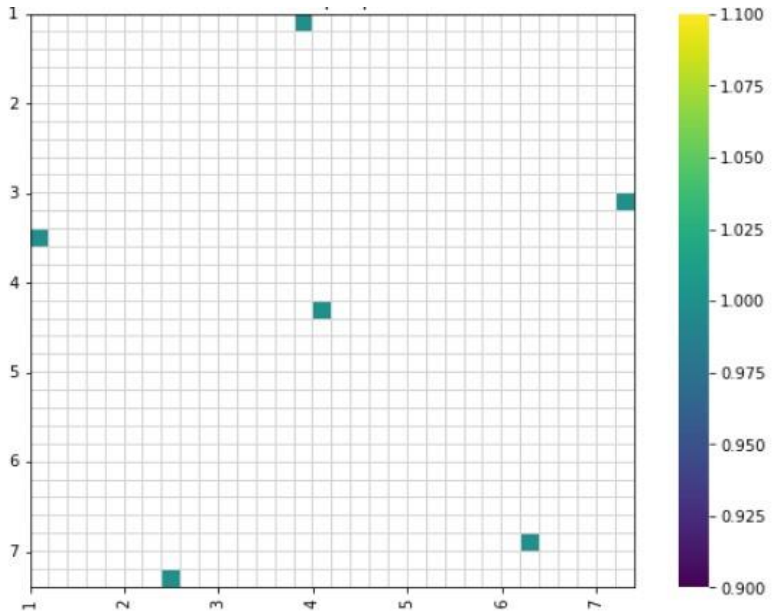


Figure 4.6: Overall pixel frame of feature density matrix

4.6 TRAIN SET IMAGE MATRICES

The following plots in Figure 4.8 show the image matrices of the first three samples of the training set. Figure 4.7 shows the indexed trained samples. Figure 4.9 and 4.10 shows the indexed test samples and the image matrices of the first three samples of the test set.


```
print (pd.DataFrame(y_train),"\n",pd.DataFrame(X_train))
```

y_train value

	-
0	0
1	1
2	1
3	0
4	0
..	..
767	1
768	1
769	0
770	0
771	0

[772 rows x 1 columns]

X_train value

	month	poleno	phase	pdarea	totalpole	signalval
611	7	5	1	12	14	220
370	6	4	1	1	26	101
229	1	4	2	3	15	103
905	5	4	2	14	61	220
522	5	5	2	6	10	218
..	...	...	...	...	...	...
822	3	4	2	6	17	118
577	2	8	1	8	20	115
826	3	4	2	7	11	218
557	12	6	1	10	10	219
721	12	4	2	17	10	218

[772 rows x 6 columns]

Figure 4.7: The cross-section of trained samples

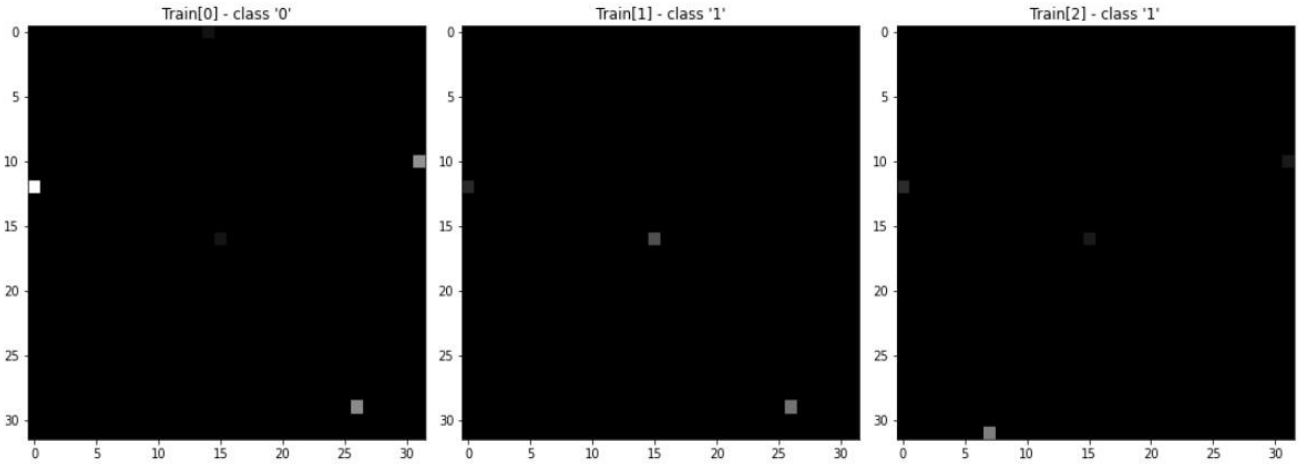


Figure 4.8: Image matrices of first three trained samples

```
print (pd.DataFrame(y_test),"\n",pd.DataFrame(X_test))
```

y_test values

0	1
1	1
2	0
3	1
4	1
..	..
188	0
189	1
190	1
191	1
192	0

[193 rows x 1 columns]

X_test values

	month	poleno	phase	pdarea	totalpole	signalval
103	12	7	2	5	50	88
533	8	8	2	20	10	115
763	7	12	1	15	10	217
585	3	12	1	18	10	118
948	10	5	1	14	10	102
..	...	...	...	...	...	...
710	11	17	1	2	11	217
93	12	6	1	10	10	86
807	1	6	2	8	10	80
879	12	4	2	10	10	89
944	10	6	2	9	14	219

[193 rows x 6 columns]

Figure 4.9: The cross-section of test samples

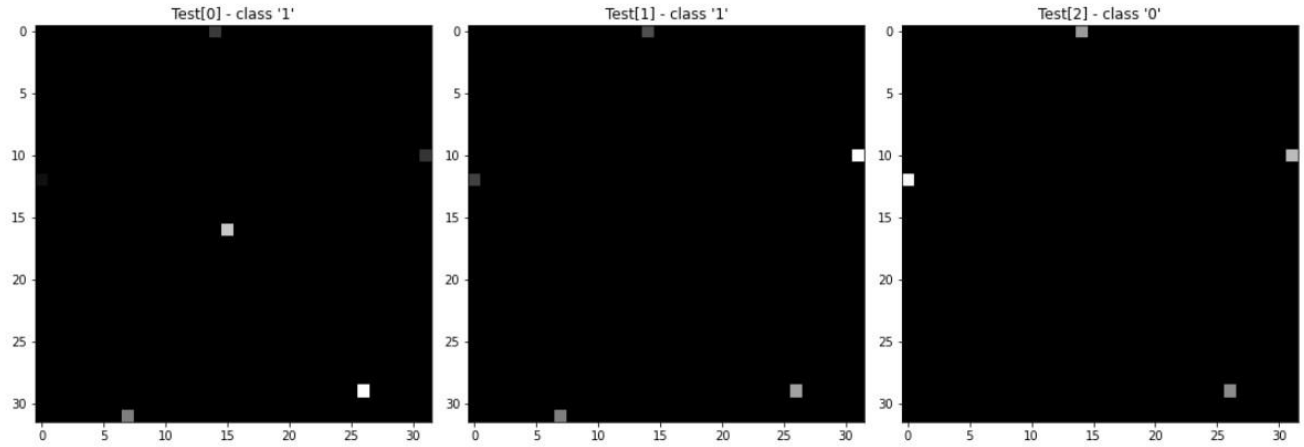


Figure 4.10: Image matrices of first three test samples

4.7 PROCESSING IMAGE MATRICES WITH CNN

The image matrices produced from Figures 4.9 and 4.10 are then used as input for the CNN model for effective classification. Conv2d pytorch with 512 channels both for input and output. It contains two classes (1, 0), indicating whether there is partial discharge. The size of the convolving kernel and the stride that controls the cross-correlation are represented in a tuple of two integers (1, 1). The first integer is used for the height dimension and the second integer for the width dimension. The transform numpy image format is then converted to a PyTorch tensor using the untrained network (image matrices) in other to generate pyTorch datasets and data loaders for training and testing sets. The dataset stores the samples and their corresponding labels, and Dataloader wraps an iterable around the Dataset to enable easy access to the samples.

The training is conducted for 20 epochs. The loss for training, its valid accuracy, and valid losses are shown in Figures 4.11, 4.12, and 4.13, respectively.

```
Epoch : 1 Train Loss: 0.585 | Accuracy: 81.736
Epoch : 2 Train Loss: 0.061 | Accuracy: 99.741
Epoch : 3 Train Loss: 0.003 | Accuracy: 100.000
Epoch : 4 Train Loss: 0.002 | Accuracy: 100.000
Epoch : 5 Train Loss: 0.001 | Accuracy: 100.000
Epoch : 6 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 7 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 8 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 9 Train Loss: 0.001 | Accuracy: 100.000
Epoch : 10 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 11 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 12 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 13 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 14 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 15 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 16 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 17 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 18 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 19 Train Loss: 0.000 | Accuracy: 100.000
Epoch : 20 Train Loss: 0.000 | Accuracy: 100.000
```

Figure 4.11: CNN pytorch training loss and accuracy at each epoch

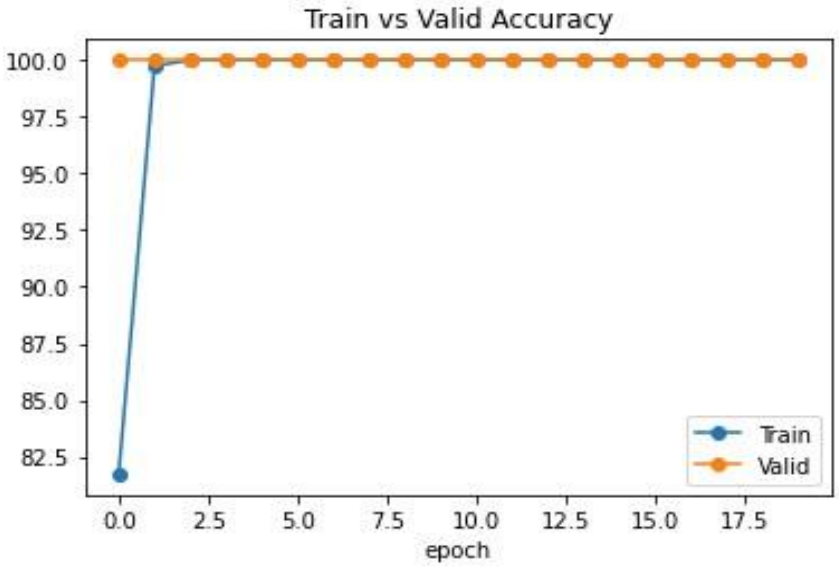


Figure 4.12: Graph of CNN pytorch valid accuracy

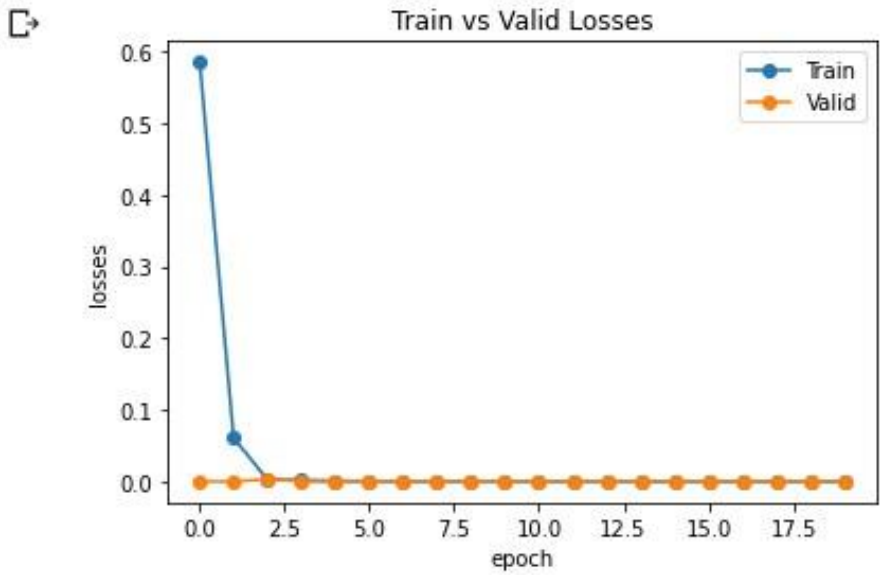


Figure 4.13: Graph of CNN pytorch valid losses

The overall accuracy and f1_score for training and testing are shown in Figure 4.14. F-score (F1Score) measures a model's accuracy on a dataset.

```
The train accuracy was 1.000
The test accuracy was 1.000
The f1_Score was [1. 1.]
```

Figure 4.14: Screen-shot of train accuracy and F1-Score

The Confusion matrix during training was also recorded, as shown in Figure 4.15. Confusion matrix is a very popular measure used while solving classification problems. It can be applied to binary classification as well as to multiclass classification problems.

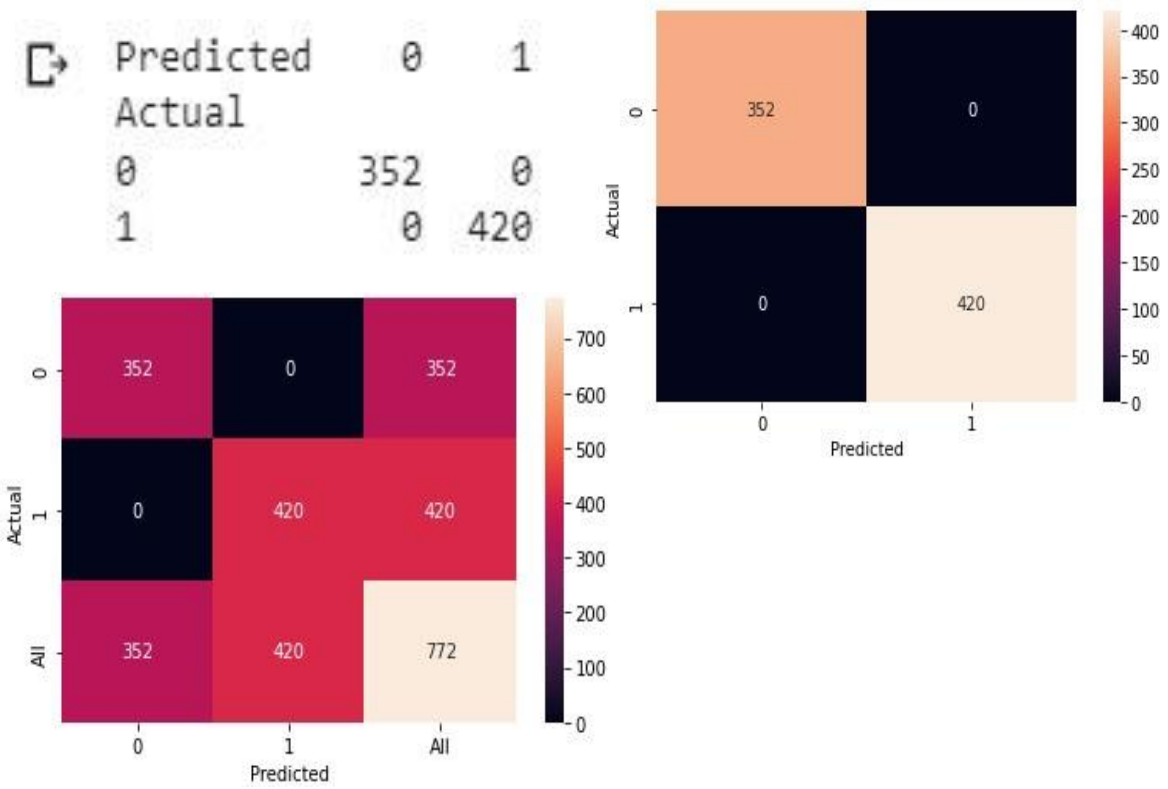


Figure 4.15: Confusion matrix on the target class

4.8 ENVIRONMENT FOR IMPLEMENTATION

This research was implemented on Google Colab (short for Collaboratory), as shown in Figure 4.14. It is a product offered by Google Research that allows machine learning researchers to work on projects in the browser. It is similar to Google Docs, allowing one to share projects between many people, and best of all, it gives free access to GPUs to train models quickly without any signup.

When working on projects, testing code on a portion of a dataset on one's computer or Google Colab initially is the best. The tools can be used as a quick way to see if code has bugs or if the output of a model is reasonable. After testing, the code can then be copied with the full dataset to a group's Amazon Web Services (AWS) instance to run the full version of the model. Testing a model locally / on Colab is free, whereas, on AWS, it will require using credits for running on AWS).

Figure 4.16: Colab IDE for writing python language

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.0 CONCLUSION

Application of a convolutional neural network has been explored on non-image data using DeepInsight as an intermediary to convert the non-image data to partial discharge and nonpartial discharge images to recognize when a partial occurred or not. The presented dataset refers to the monitored electrical insulation deterioration that occurs on different substations within Akoko Metropolis in Ondo State. The performance of the applied architecture was assessed based on the recognition score, confusion matrix, and accuracy metric.

The generated PD images from the DeepInsight algorithm represent a new category of a diagnostic evaluation, referring to qualitative analysis and with no bias. The system performed by analyzing the shapes of statistically accumulated images, which would be very desirable, especially in on-site diagnostics or monitoring situations.

5.1 RECOMMENDATION

One way to leverage the stress on field workers in BEDC is by predicting a likely situation in a particular area with some predefined data. This research has focused on this area, which will be highly recommended for rural or urban locations, especially where new field managers are posted. Based on the research. The study provides the below policy recommendation:

i. Implement deep learning algorithms for partial discharge detection in the national grid distribution network in Edo, Benin City.
ii. Provide training for technical personnel to effectively use deep learning-based partial discharge detection systems.

iii. Develop a monitoring and maintenance program for partial discharge detection systems, including regular system updates and data analysis.

iv. Collaborate with relevant stakeholders such as power companies and universities to gather data for deep learning algorithms to improve the accuracy of partial discharge detection.

v. Encourage power companies to adopt partial discharge detection systems as a standard practice in their operations.

vi. Provide incentives for power companies to invest in partial discharge detection systems, such as tax breaks or subsidies.

vii. Implement regulations requiring power companies to regularly report partial discharge detection results to relevant government agencies.

viii. Establish a database for partial discharge detection data, allowing for centralized data analysis and improved decision-making.

ix. Encourage research and development in the area of deep learning-based partial discharge detection, with a focus on improving accuracy and reducing costs.

x. Work with international organizations and experts to share knowledge and best practices for partial discharge detection in the national grid distribution network.

5.2 CONTRIBUTION TO KNOWLEDGE

The effort put into this research will reduce or eliminate harmful environmental impacts, help to reduce maintenance costs, and prevent power outages. The field workers would also have what to leverage instead of taking many days to solve partial discharge issues.

5.3 FUTURE RESEARCH

This research direction is a visible trend in future autonomous PD expert systems.

Although the most challenging aspects of today's partial discharge image recognition are related to multi-source and multi-labeled PD classification, the separation between real internal PDs, and both noise and disturbances, thus, future work will focus on adjusting the CNN architecture and hyperparameters for PD recognition for diagnostic applications in this areas.

REFERENCES

Anh Tu, L. (2020). Improving Feature Map Quality of SOM Based on Adjusting the Neighborhood Function. *Sustainability in Urban Planning and Design*, 15–18. https://doi.org/10.5772/intechopen.89233

Arden D. (2022). Applied Deep Learning -Part 3: Autoencoders.URL: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders 1c083af4d798. Accessed: 2022-03-19

Barrios S., Buldain D., Comech M.P., Gilbert I., Orue I. (2019). Partial discharge classification using deep learning methods—Survey of recent progress. Energies 2019, 12, 2485.

Barrios, S., Buldain, D., Gilbert, I., & Orue, I. (2019). *Methods — Survey of Recent Progress*. *Dl*.

Candela, R., Mirelli, G., & Schifani, R. (2000). PD recognition by means of statistical and fractal parameters and a neural network. IEEE Transactions on Dielectrics and Electrical Insulation, 7(1), 87–94.

Chang, W. Y., & Yang, H. T. (2008). Application of self organizing map approach to partial discharge pattern recognition of cast-resin current transformers. *WSEAS Transactions on Computer Research*, *3*(3), 142–151.

Che Q., Wen H., Li X., Peng Z., Chen K.P. (2019). Partial discharge recognition based on optical fiber distributed acoustic sensing and a convolutional neural network. IEEE Access 2019, 7, 101758–101764.

Dai, J.; Teng, Y.; Zhang, Z.; Yu, Z.; Sheng, G.; Jiang, X. Partial discharge data matching method for GIS case-based reasoning. Energies 2019, 12, 3677.

Danikas M.G. (1997). "Small Partial Discharges and their Role in Insulation", IEEE DEIS Transactions 1997, Vol. 4, pp 863-867. URL: https://municipalinfonet.com/energy/magazine/44/article/Partial-Discharge-Testing of-In-Situ-Power-Cable-Accessories-An-Overview-.htm, Retrieved March 7, 2022

Durand, T.; Mehrasa, N.; Mori, G. (2019). Learning a deep convnet for multi-label classification with partial labels. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 647–657.

Florkowski, M. (2020) Classification of partial discharge images using deep convolutional neural networks. Energies 2020, 13, 5496.

Florkowski, M. Application of image processing techniques to partial discharge patterns. In Proceedings of the International Symposium on High Voltage Engineering, Graz, Austria, 28 August–1 September 1995; p. 5649.

Ganguly, B.; Chaudhury, S.; Biswas, S.; Dey, D.; Munshi, S.; Chatterjee, B.; Dalai, S.; Chakravorti, S. (2020). Wavelet Kernel based Convolutional Neural Network for Localization of Partial Discharge Sources within a Power Apparatus. IEEE Trans. Ind. Inform. 2020, 17, 1831–1841

Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y.(2014). Generative Adversarial Nets. In Advances in Neural Information Processing Systems 27; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D.,Weinberger, K.Q., Eds.; Curran Associates, Inc.: Dutchess County, NY, USA, 2014; pp. 2672–2680.

Habibi, A. H. & Jahani, H. E. (2017). Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification. (Springer International Publishing, 2017).

Hao L. and Lewin P. L. (2010), "Partial Discharge Source Discrimination using a Support Vector Machine, IEEE Transactions on Dielectrics and Electrical Insulation Vol. 17, No. 1; February 2010.

Harry E. (2022). Partial Discharge Testing of In-Situ Power Cable Accessories – An Overview, Orton OCEI. URL: https://municipalinfonet.com/energy/magazine/44/article/Partial-Discharge-Testing-of-In-Situ-Power-Cable-Accessories-An-Overview-.htm, Retrieved March 7, 2022

Hinton G.E., Osindero S., Teh Y.W (2006). A Fast Learning Algorithm for Deep Belief Nets. Neural Comput. 2006, 18, 1527–1554.

Hochreiter S. and Schmidhuber J. (1997). Long Short-Term Memory. Neural Comput. 1997, 9, 1735–1780.

Hughes R. T., Zhu L., Bednarz T. (2021).Generative Adversarial Networks–Enabled Human–Artificial Intelligence Collaborative Applications for Creative and Design Industries:

A Systematic Review of Current Approaches and Trends,Frontiers in Artificial Intelligence, Vol. 4, pg.1-17, 2021. URL=https://www.frontiersin.org/article/10.3389/frai.2021.604234

IBM Cloud Education (2020). Supervised Learning. URL: https://www.ibm.com/cloud/learn/supervised-learning#toc-unsupervis-Fo3jDcmY. Accessed on 21-03-2022

International Electrotechnical Commission (2015). High-Voltage Test Techniques—Partial Discharge Measurements =: Techniques des essais à haute tension—mesures des décharges partielles; International Electrotechnical Commission Central O ce: Geneva, Switzerland, 2015; ISBN 978-2-8322-3053-4.

Johansson, E., & Lind, P. (2019). *Measurements of partial discharge in insulators*.

Khan, M. A., Choo, J., & Kim, Y. H. (2019). End-to-End Partial Discharge Detection in Power Cables via Time-Domain Convolutional Neural Networks. *Journal of Electrical Engineering and Technology*, *14*(3), 1299–1309. https://doi.org/10.1007/s42835-01900115-y

Khayam, U. (2015). *Design , Implem mentation , and Testing of Partial Discharge Signal Pattern Recognition an nd Judgment System Applic cation Using Statistical l Method*. 314–318.

Koch M. and Kruger M. (2012). "A New Method for On-Line Monitoring of Bushings and Partial Discharges of Power Transformer," pp. 1205–1208, 2012.

Kothoke P., Anupama D., and Chaudhari Y. (2020). Partial Discharge Type Detection utilizing Statistical Techniques (n-q) and Random Forest Method, ICASISET 2020, May 16-17, Chennai, India.

Kothoke, P., Deshpande, A., & Chaudhari, Y. (2019). *A Literature Review of methods used for Partial Discharge Type Detection and its Improvised Techniques*. 9030–9036. https://doi.org/10.15680/IJIRSET.2019.0808056

Kothoke P., Kumar S., and Chaudhari Y. (2017). "Analysis of Partial discharge source using Artificial Neural Network", International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), Vol. 6, Issue 6, June 2017.

Kraig Bader (1998) “Partial Discharge Site Location Experience at Colorado Springs Utilities”, Kraig Bader, IEEE, ICC Minutes Spring 1998, Appendix 10J. Retrieved March 7, 2022 (https://municipalinfonet.com/energy/magazine/44/article/Partial Discharge-Testing-of-In-Situ-Power-Cable-Accessories-An-Overview-.htm).

Krish, S. (2011). A practical generative design method. Comput. Aided Des. Vol.43, pg.88–100., 2011. doi:10.1016/j.cad.2010.09.009.

LeCun Y., Kavukcuoglu K., Farabet C. (2010). Convolutional networks and applications in vision. In Proceedings of the 2010 IEEE International Symposium on Circuits and Systems, Paris, France, 30 May–2 June 2010; pp. 253–256.

Lee, Honglak, Ekanadham, Chaitanya, and Ng A. (2007). Sparse deep belief net model for visual area v2. In Advances in neural information processing systems, pp. 873–880, 2007.

Lu Y., Wei R., Chen J., Yuan J. (2016). Convolutional neural network based transient earth voltage detection. In Proceedings of the 2016 15th International Symposium on Parallel and Distributed Computing (ISPDC), Fuzhou, China, 8–10 July 2016; pp. 386–389.

Maaten L. Van der and Hinton G. (2008). Visualizing Data using t-SNE. Journal of Machine Learning Research, Vol 9 (86), pp. 2579-2605, 2008.

Mas’ud A., Albarracín R., Ardila-Rey J., Muhammad-Sukki F., Illias H., Bani N., Munir A. (2016). Artificial Neural Network Application for Partial Discharge Recognition: Survey and Future Directions. Energies 2016, 9, 574.

Montanari, G.C.; Cavallini, A (2013). Partial discharge diagnostics: from apparatus monitoring to smart grid assessment. IEEE Electr. Insul. Mag. 2013, 29, 8–17.

Muhr M., Schwarz R., Pack S., Koerbler B. (2004). Unconventional Partial Discharge Measurement, proceedings of The Conference on Electrical Insulation and Dielectric Phenomena; 2004. p 430-433.

Nair, V. and Hinton, Geoffrey E. (2009). 3d object recognition with deep belief nets. In Advances in Neural Information Processing Systems, pp. 1339–1347, 2009

Navidan, H., Moshiri, P. F., Nabati, M., Shahbazian, R., Ghorashi, S. A., Shah-Mansouri, V.,

& Windridge, D. (2021). Generative Adversarial Networks (GANs) in networking: A comprehensive survey & evaluation. *Computer Networks*, *194*(May). https://doi.org/10.1016/j.comnet.2021.108149

Ng A. (2022). Sparse autoencoder, CS294A Lecture notes. URL: https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf. Accessed 18-03-2022

Niasar, M.G.;Wang, X.; Kiiza, R.C. Review of Partial Discharge Activity Considering Very-Low Frequency and Damped Applied Voltage. Energies 2021, 14, 440. https://doi.org/10.3390/ en14020440

Olshausen, Bruno A and Field, David J (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? Vision research, 37(23):3311–3325, 1997.Partial Discharge (2021).

Pereira, F. H., Bezerra, F. E., Oliva, D., de Souza, G. F. M., Chabu, I. E., Santos, J. C., Junior, S. N., & Nabeta, S. I. (2020). Forecast model update based on a real-time data processing lambda architecture for estimating partial discharges in hydrogenerator. *Sensors (Switzerland)*, *20*(24), 1–23. https://doi.org/10.3390/s20247242

Puspitasari N., Khayam U., Kakimoto Y., Yoshikawa H., Kozako M., Hikita M. (2019). Partial DischargeWaveform Identification using Image with Convolutional Neural Network. In Proceedings of the 54th International Universities Power Engineering Conference (UPEC), Bucharest, Romania, 3–6 September 2019; pp. 1–4.

Raymond W.J.K, Illias H.A., Bakar A.H.A., Mokhlis H. (2015). Partial discharge classifications: Review of recent progress. Measurement 2015, 68, 164–181.

Renforth L., Hamer P. S., Clark D., Goodfellow S., and Tower R. (2013). "Continuous , Remote On-Line Partial Discharge (Olpd) Monitoring Of Hv Ex / Atex Motors In The Oil And Gas Industry," 2013.

Renforth, L. A., Hamer, P. S., Clark, D., Goodfellow, S., & Tower, R. (2015). Continuous Remote Online Partial Discharge Monitoring of HV Ex/ATEX Motors in the Oil and Gas Industry. *IEEE Transactions on Industry Applications*, *51*(2), 1326–1332. https://doi.org/10.1109/TIA.2014.2357576

Sally N.R., Ayman H.E,Salama. M.A., Bartnikas R. (2012). Partial Discharge Detection and

Location Inside The Winding of Power Transformer", proceedings of Electrical Insulation and Dielectric Phenomena (CEIDP);2012. p. 84 - 87.

Sharma, A., Vans, E. and Shigemizu, D. (2019). DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture. Sci Rep 9, 11399 (2019). https://doi.org/10.1038/s41598-019-47765-6

Sipahutar, T. F., Kemma, A. A., Pattanadech, N., Pratomosiwi, F., Suwarno, & Muhr, M. (2013). Effect of Test Method and Needle Plane Configuration on Partial Discharge Inception Voltage Measurement of Mineral Oil based on Weibull Analysis. *Procedia Technology*, *11*(Iceei), 411–418. https://doi.org/10.1016/j.protcy.2013.12.210

Song, H.; Dai, J.; Sheng, G.; Jiang, X. (2018). GIS partial discharge pattern recognition via deep convolutional neural network under complex data source. IEEE Trans. Dielectr. Electr. Insul. 2018, 25, 678–685.

Stone, G.C (2005). Partial discharge diagnostics and electrical equipment insulation condition assessment. IEEE Trans. Dielectr. Electr. Insul. 2005, 12, 891–904.

Suwarno S. (2012), "Phase-resolved Properties and Simulation of Partial Discharges in High Voltage Liquid Insulation," no. 1, pp. 2–5, 2012.

Suwarno, & Sutikno, H. (2011). Model and computer simulation of partial discharge patterns in natural liquid insulation for high voltage application. *International Journal of Mathematical Models and Methods in Applied Sciences*, *5*(5), 966–973.

Tuyet-Doan V.N., Tran-Thi N.D., Youn Y.W., Kim Y.H. (2020). One-Shot Learning for Partial Discharge Diagnosis Using Ultra-High-Frequency Sensor in Gas-Insulated Switchgear. Sensors 2020, 20, 5562.

Tuyet-Doan, V.-N.; Nguyen, T.-T.; Nguyen, M.-T.; Lee, J.-H.; Kim, Y.-H. (2020). Self attention network for partial-discharge diagnosis in gas-insulated switchgear. Energies 2020, 13, 2102.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999–6009.

Vincent P., Larochelle H., Bengio Y., Manzagol P. A. (2008). Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning—ICML '08, Helsinki, Finland, 5–9 July 2008; ACM Press: Helsinki, Finland, 2008; pp. 1096–1103.

Vitor V., Bruno A.,Amanda B., Jorge A.,Guilherme B., and André L. (2020). An Application of Wavelet Analysis to Assess Partial Discharge Evolution by Acoustic Emission Sensor,Eng. Proc. 2020, Vol 2, issue 33; doi:10.3390/ecsa-7-08244

Wang, Y.; Yan, J.; Yang, Z.; Liu, T.; Zhao, Y.; Li, J. (2019). Partial Discharge Pattern Recognition of Gas-Insulated Switchgear via a Light-Scale Convolutional Neural Network. Energies 2019, 12, 4674.

Willem Boone (1999). "Non-destructive Diagnostic Testing of Distribution Cable Accessories", Willem Boone, IEEE, ICC Minutes, Spring 1999, Page 129. Retrieved March 7, 2022 (https://municipalinfonet.com/energy/magazine/44/article/Partial Discharge-Testing-of-In-Situ-Power-Cable-Accessories-An-Overview-.htm).

Wu, M., Cao, H., Cao, J., Nguyen, H. L., Gomes, J. B., & Krishnaswamy, S. P. (2015). An overview of state-of-the-art partial discharge analysis techniques for condition monitoring. *IEEE Electrical Insulation Magazine*, *31*(6), 22–35. https://doi.org/10.1109/MEI.2015.7303259

Yeo, J., Jin, H., Yuen, C., & Ng, C. S. (2020). Predicting of Partial Discharge in MediumVoltage Cables Using Recurrent Neural Network with Long Short-Term Memory Cells after Wavelet Denoising. *Proceeding - 8th International Conference on Condition Monitoring and Diagnosis, CMD 2020, July 2021*, 194–197. https://doi.org/10.1109/CMD48350.2020.9287241

APPENDIX I

```
from sklearn.model_selection import train_test_split
import pandas as pd import numpy as np

from matplotlib import pyplot as plt
import seaborn as sns from
google.colab import drive

from sklearn.preprocessing import MinMaxScaler from
sklearn.manifold import TSNE
import matplotlib.ticker as ticker
```

```
drive.mount("/content/gdrive")
expr_file = "/content/gdrive/My Drive/Colab Notebooks/dataset2.csv"

expr = pd.read_csv(expr_file) y =
expr['pdischarge'].values
X=expr.drop('pdischarge',axis=1)
#divide the dataset into 80% train and 20% test
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.2, random_state=23, stratify=y)
#from 80% train, divide the traindataset into 60% train and 20% validation
X_train, X_val, y_train, y_val = train_test_split(
   X_train, y_train, test_size=0.25, random_state=23)
#In this way, train, val, test set will be 60%, 20%, 20% of the dataset respectively.
```

Normalize data using StandardScaler and create tsne class

```
mmsc = MinMaxScaler()
X_train_norm = mmsc.fit_transform(X_train)
X_test_norm = mmsc.transform(X_test).clip(0,1)

tsne = TSNE(
n_components=2,
   random_state=1515,metric='cosine',    n_jobs=-
1)  #1701  1515
#tsne = TSNE(
#    n_components=2,
#    random_state=3000, learning_rate= 1000, perplexity = 50, metric='cosine', verbose = 1, #
n_jobs=-1)  #1701  1515

#tsne = TSNE(n_components=2, perplexity=30, metric='cosine',
 #          random_state=1515, n_jobs=-1)

#it = ImageTransformer(feature_extractor=tsne, pixels=50)

from sklearn import metrics from
__future__ import print_function
import time
time_start = time.time() #1701
1515  metric='cosine',
#tsne = TSNE(n_components=2,random_state=1515,  #
n_jobs=-1, verbose=1, perplexity=30, n_iter=300)
tsne_results = tsne.fit_transform(X_train_norm)
print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))

tsne_results
```

```
plt.figure(figsize=(5, 5))
plt.plot(tsne_results[:,0],tsne_results[:,1])
```

Initialize image transformer.

```
it = ImageTransformer(
feature_extractor=tsne,      pixels=32)
```

Train image transformer on training data. Setting plot=True results in

at a plot showing the reduced features (blue points), convex full (red)

, and minimum bounding rectagle (green) prior to rotation.

```
time_start = time.time()
plt.figure(figsize=(5, 5)) it.fit(X_train_norm,
plot=True)
print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))
```

Train image transformer on training data and transform training and testing sets. Values should be between 0 and 1.


```
X_train_img = it.fit_transform(X_train_norm)
X_test_img = it.transform(X_test_norm)
X_val_img = it.transform(X_val_norm)
```

The feature density matrix can be extracted from the trained transformer in order to view overall feature overlap.

X_train_imgThe following are showing plots for the image matrices first four samples of the training set.

```
fig, ax = plt.subplots(1, 3, figsize=(15, 5)) for
i in range(0,3):
   ax[i].imshow(X_train_img[i])
   ax[i].title.set_text("Train[{}] - class '{}'".format(i, y_train[i])) plt.tight_layout()

print (pd.DataFrame(y_train),"\n",pd.DataFrame(X_train))
```

Transforming the testing data is done the same as transforming the training data.

```
X_test_img = it.transform(X_test_norm)

fig, ax = plt.subplots(1, 3, figsize=(15, 5)) for
i in range(0,3):
   ax[i].imshow(X_test_img[i])
   ax[i].title.set_text("Test[{}] - class '{}'".format(i, y_test[i])) plt.tight_layout()
```

```
print (pd.DataFrame(y_test),"\n",pd.DataFrame(X_test))

X_val_img = it.transform(X_val_norm)

fig, ax = plt.subplots(1, 3, figsize=(15, 5)) for
i in range(0,3):
   ax[i].imshow(X_val_img[i])
   ax[i].title.set_text("Test[{}] - class '{}'".format(i, y_val[i])) plt.tight_layout()

print (pd.DataFrame(y_val),"\n",pd.DataFrame(X_val))
import torch device = torch.device("cpu") import
torchvision.transforms as transforms from
torch.utils.data import TensorDataset, DataLoader
import torch.nn as nn
import torch.optim as optim

from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score

import warnings;
warnings.simplefilter('ignore')

Encode labels as integers.

le = LabelEncoder() y_train_enc =
le.fit_transform(y_train)
y_test_enc = le.transform(y_test)

net = torch.hub.load(
   'pytorch/vision:v0.6.0',                    'squeezenet1_1',
pretrained=False, verbose=False).double()
net.classifier[1] = nn.Conv2d(512, num_classes, kernel_size=(1,1),
                  stride=(1,1)).double()
Transform numpy image format to
PyTorch tensor. Using an untrained
network, so normalization as specificed
in SqueezeNet documentation is not
required.

preprocess              =              transforms.Compose([
transforms.ToTensor()
])

X_train_tensor = torch.stack([preprocess(img) for img in X_train_img]) y_train_tensor
= torch.from_numpy(le.fit_transform(y_train))

X_test_tensor = torch.stack([preprocess(img) for img in X_test_img]) y_test_tensor
= torch.from_numpy(le.transform(y_test))
```

```
X_val_tensor = torch.stack([preprocess(img) for img in X_val_img]) y_val_tensor
= torch.from_numpy(le.transform(y_val))
```

Generate pyTorch datasets and dataloaders for training and testing sets.

```
batch_size = 1

trainset = TensorDataset(X_train_tensor, y_train_tensor)
trainloader = DataLoader(trainset, batch_size=batch_size, shuffle=True
```

Specify loss function and optimization algorithm

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=1e-4, momentum=0.9)
```

Train SqueezeNet for 20 epochs

```
xaxis = [] yaxis
=[]
zaxis =[] num_epochs = 20 for epoch in
range(num_epochs):     total=0.0
correct=0.0     running_loss = 0.0     for i,
data in enumerate(trainloader, 0):
      # get the inputs; data is a list of [inputs, labels]
inputs, labels = data

      # zero the parameter gradients
      optimizer.zero_grad()

      # forward + backward + optimize
outputs = net(inputs)        loss =
criterion(outputs, labels)
loss.backward()        optimizer.step()

      running_loss  +=  loss.item()
# print epoch statistics
   print('[%d] train_loss: %.5f  Accuracy : %.3f' %
        (epoch + 1, running_loss / len(X_train_tensor) * batch_size),)
   #print('[%d] train_loss: %.5f ' %
   #      (epoch + 1, running_loss / len(X_train_tensor) * batch_size))

   xaxis.append(epoch + 1)
   yaxis.append(running_loss          /          len(X_train_tensor)          *          batch_size)
zaxis.append(running_loss / len(X_test_tensor) * batch_size)

plt.plot(xaxis,yaxis, color='y', label='train') plt.plot(xaxis,zaxis,
color='g', label='test')
```

```
# naming the x axis
plt.xlabel('epoch') #
naming the y axis
plt.ylabel('loss')

# giving a title to my graph plt.title('Model
loss Graph!')
# Adding legend, which helps us recognize the curve according to it's color plt.legend()
# function to show the plot plt.show()
```

Calculate accuracy of prediction

```
train_outputs = net(X_train_tensor)
_, train_predicted = torch.max(train_outputs, 1)

test_outputs = net(X_test_tensor)
_, test_predicted = torch.max(test_outputs, 1)

print("The train accuracy was {:.3f}".format(accuracy_score(train_predicted, y_train_tensor)))
print("The test accuracy was {:.3f}".format(accuracy_score(test_predicted, y_test_tensor)))
```